Reachability in Continuous Pushdown VASS

A. R. BALASUBRAMANIAN*, MPI-SWS, Germany
RUPAK MAJUMDAR, MPI-SWS, Germany
RAMANATHAN S. THINNIYAM†, Uppsala University, Sweden
GEORG ZETZSCHE, MPI-SWS, Germany

Pushdown Vector Addition Systems with States (PVASS) consist of finitely many control states, a pushdown stack, and a set of counters that can be incremented and decremented, but not tested for zero. Whether the reachability problem is decidable for PVASS is a long-standing open problem.

We consider *continuous PVASS*, which are PVASS with a continuous semantics. This means, the counter values are rational numbers and whenever a vector is added to the current counter values, this vector is first scaled with an arbitrarily chosen rational factor between zero and one.

We show that reachability in continuous PVASS is NEXPTIME-complete. Our result is unusually robust: Reachability can be decided in NEXPTIME even if all numbers are specified in binary. On the other hand, NEXPTIME-hardness already holds for coverability, in fixed dimension, for bounded stack, and even if all numbers are specified in unary.

CCS Concepts: • **Theory of computation** → **Models of computation**.

Additional Key Words and Phrases: Vector addition systems, Pushdown automata, Reachability, Decidability, Complexity

ACM Reference Format:
A. R. Balasubramanian, Rupak Majumdar, Ramanathan S. Thinniyam, and Georg Zetzsche. 2024. Reachability in Continuous Pushdown VASS. *Proc. ACM Program. Lang.* 8, POPL, Article 4 (January 2024), 25 pages. https://doi.org/10.1145/3633279

1 INTRODUCTION

Pushdown VASS (PVASS) is a model of computation which combines a Pushdown Automaton (PDA) and a Vector Addition System with States (VASS) by using both a stack and counters. Since PDAs naturally model sequential computation with recursion [Alur et al. 2005; Reps et al. 1995] and VASSs naturally model concurrency [Karp and Miller 1969], the combination of the two is an expressive modelling paradigm. For instance, PVASS can be used to model recursive programs with unbounded data domains [Atig and Ganty 2011], beyond the capability of PDA alone. They can also model context-bounded analysis of multi-threaded programs [Atig et al. 2009], even when one thread can have arbitrarily many context switches. In program analysis, one-dimensional PVASS models certain pointer analysis problems [Kjelstrøm and Pavlogiannis 2022; Li et al. 2021].

*A part of the work was done when this author was at Technical University of Munich (TUM).
†A part of the work was done when this author was at MPI-SWS.

Authors' addresses: A. R. Balasubramanian, MPI-SWS, Germany, bayikudi@mpi-sws.org; Rupak Majumdar, MPI-SWS, Germany, rupak@mpi-sws.org; Ramanathan S. Thinniyam, Uppsala University, Sweden, ramanathan.s.thinniyam@it.uu.se; Georg Zetzsche, MPI-SWS, Germany, georg@mpi-sws.org.

ACM 2475-1421/2024/1-ART4
https://doi.org/10.1145/3633279

Many verification problems can then be reduced to the *reachability* problem [Hack 1976], where one asks, given a PVASS $\mathcal{M}$ and two of its configurations c_0 and c_f, whether there is a run of $\mathcal{M}$ that starts at c_0 and ends at c_f. Unfortunately, in spite of our understanding of the reachability problem in the context of PDA [Alur et al. 2005; Chaudhuri 2008; Reps et al. 1995; Yannakakis 1990] and VASS models [Czerwiński and Orlikowski 2022; Leroux 2022; Leroux and Schmitz 2019], the decidability of PVASS reachability remains open [Englert et al. 2021; Ganardi et al. 2022; Schmitz and Zetzsche 2019], even with just one stack and one VASS counter. Hence a natural approach to take is to consider *overapproximations* of the system behaviour, both to build up theoretical understanding and to approximate verification questions in practice. The approach we take in this paper is to *approximate* the model via continuous semantics. A continuous semantics gives an over-approximation of the possible behaviors, so if the relaxed program cannot reach a location, neither can the original one.

By *continuous* semantics of VASS, we mean that a transition is allowed to be fired *fractionally*, allowing the addition or removal of a *rational fraction* of tokens from a counter. This model was first introduced by David and Alla in the context of Petri nets [David 1987]. Continuous VASS ($\mathbb{Q}_+$-VASS) were studied by Blondin and Haase [2017], who showed reachability and coverability are NP-complete. Approximation via continuous semantics has allowed the application of SMT solvers and the development of state-of-the-art solvers from an empirical perspective [Blondin et al. 2016] to the coverability problem for VASS. More generally, relaxing integer-valued programs to continuous-valued programs is a common approximation in invariant synthesis [Colón et al. 2003; Srivastava et al. 2010].

In addition to its use as an overapproximation, the continuous semantics captures the exact behavior in systems where update vectors represent change rates (to real variables such as temperature, energy consumption) per time unit. Here, fractional addition corresponds to executing steps that take at most one time unit. For example, $\mathbb{Q}_+$-VASS are constant-rate multi-mode systems [Alur et al. 2012] where each action takes at most one time unit. Continuous PVASS can then be seen as recursive programs with such constant-rate dynamics.

Contribution. We study PVASS with continuous semantics (denoted by $\mathbb{Q}_+$-PVASS), where we allow fractional transitions on counters, but retain the discrete nature of the stack. Hence, a *configuration* is a tuple $(q, \gamma, \mathbf{v})$ where q is the control-state, γ is the stack content and $\mathbf{v}$ represents the counter values. We show that reachability is decidable, and we provide a comprehensive complexity landscape for reachability, coverability, and state reachability. The *reachability problem* asks for given configurations $(q, \gamma, \mathbf{v})$ and $(q', \gamma', \mathbf{v}')$, whether from $(q, \gamma, \mathbf{v})$, the system can reach $(q', \gamma', \mathbf{v}')$. The *coverability problem* asks for given configurations $(q, \gamma, \mathbf{v})$ and $(q', \gamma', \mathbf{v}')$, whether from $(q, \gamma, \mathbf{v})$, the system can reach a configuration of the form $(q', \gamma', \mathbf{v}'')$ where $\mathbf{v}'' \geq \mathbf{v}'$). Moreover, *state reachability* asks for a given configuration $(q, \gamma, \mathbf{v})$ and a state q', whether from $(q, \gamma, \mathbf{v})$, the system can reach a configuration of the form $(q', \gamma', \mathbf{v}')$ for some γ' and $\mathbf{v}'$.

Our main result is the following:

Theorem 1.1. *Reachability in* $\mathbb{Q}_+$-PVASS *is* NEXPTIME*-complete.*

The NEXPTIME-completeness is unusually robust. Specifically, the complexity is not affected by (i) whether we consider reachability or coverability, or (ii) whether the number of counters is part of the input or fixed, or (iii) whether counter values (in updates and configurations) are encoded in unary or binary. This is summarized in the following stronger version:

Theorem 1.2. *Reachability in* $\mathbb{Q}_+$-PVASS*, with binary encoded numbers, is in* NEXPTIME*. Moreover,* NEXPTIME*-hardness already holds for coverability in* 85*-dimensional* $\mathbb{Q}_+$-PVASS*, with unary encoded numbers, and bounded stack-height.*

Further, if we allow the configurations to be encoded in binary, then hardness already holds for coverability in 13-dimensional $\mathbb{Q}_+$-PVASS.

Our result is in stark contrast to reachability problems in classical VASS: It is well-known that there, coverability is EXPSPACE-complete [Rackoff 1978], whereas general reachability is Ackermann-complete [Czerwiński and Orlikowski 2022; Leroux 2022]. Furthermore, fixing the dimension brings the complexity down to primitive-recursive [Leroux and Schmitz 2019] (or from EXPSPACE to PSPACE in the case of coverability [Rosier and Yen 1986]).

Another surprising aspect is that for continuous PVASS, the coverability problem and the state reachability problem do not have the same complexity. We also show:

Theorem 1.3. *The state reachability problem for* $\mathbb{Q}_+$-PVASS *is* NP*-complete.*

This is also in contrast to the situation in PVASS: There is a simple reduction from the reachability problem to the coverability problem [Leroux et al. 2015], and from there to state reachability. Thus, the three problems are polynomial-time inter-reducible for PVASS.

Our results are based on a number of novel technical ingredients. Especially for our lower bounds, we show a number of subtle constructions that enable us to encode discrete computations of bounded runs of counter machines in the continuous semantics.

Ingredient I: Upper bound via rational arithmetic. We prove the NEXPTIME upper bound by observing that a characterization of runs in a cyclic $\mathbb{Q}_+$-VASS (meaning: the initial state is also the only final one) by Blondin and Haase [2017] still holds in a more general setting of cyclic $\mathbb{Q}_+$-PVASS. We apply this observation by combining several (known) techniques. As is standard in the analysis of PVASS [Englert et al. 2021; Leroux et al. 2015], we view runs as derivations in a suitable grammar. As usual, one can then decompose each derivation tree into an acyclic part and "pump derivations" of the form $A \overset{*}{\Rightarrow} uAv$ for some non-terminal A. Such pumps, in turn, can by simulated by a cyclic $\mathbb{Q}_+$-PVASS. Here, to simulate $A \overset{*}{\Rightarrow} uAv$, one applies u as is and one applies v in reverse on a separate set of counters. This idea of simulating grammar derivations by applying "the left part forward" and "the right part backward" is a recurring theme in the literature on context-free grammars (see, e.g. [Baumann et al. 2023; Berstel 1979; Lohrey et al. 2022; Reps et al. 2016; Rosenberg 1967]) and has been applied to PVASS by Leroux et al. [2015, Section 5].

As a consequence, reachability can be decided by guessing an exponential-size formula of Existential Linear Rational Arithmetic (ELRA). Since satisfiability for ELRA is in NP, this yields an NEXPTIME upper bound.

Ingredient II: High precision and zero tests. For our lower bound, we reduce from reachability in machines with two counters, which can only be doubled and incremented. A run in these machines is accepted if it is of a particular length (given in binary) and has both counters equal in the end. We call such machines $2\mathsf{CM}_{\mathsf{RL}}^{\cdot 2,+1}$, the RL stands for "run length". This problem is NEXPTIME-hard by a reduction from a variant of the Post Correspondence Problem where the word pair is restricted to have a specified length (given in binary) [Aiswarya et al. 2022].

We give the desired reduction from $2\mathsf{CM}_{\mathsf{RL}}^{\cdot 2,+1}$, through a series of intricate constructions that "control" the fractional firings. We go through an intermediate machine model called $[0,1]\text{-VASS}_{\mathsf{RL}}^{0?}$. An $[0,1]\text{-VASS}_{\mathsf{RL}}^{0?}$ is just like a $\mathbb{Q}_+$-VASS, except that the counter values are constrained to be in the interval $[0,1]$ and we allow zero tests. Further, we only consider runs up to a particular length (given in binary), as indicated by the RL subscript. When reducing from $2\mathsf{CM}_{\mathsf{RL}}^{\cdot 2,+1}$, we are confronted with two challenges: First, $[0,1]\text{-VASS}_{\mathsf{RL}}^{0?}$ cannot store numbers beyond 1 and second, $[0,1]\text{-VASS}_{\mathsf{RL}}^{0?}$ cannot natively double numbers. The key idea here is that, since we only consider runs of a $2\mathsf{CM}_{\mathsf{RL}}^{\cdot 2,+1}$ up to a given length m, the counter values of a $2\mathsf{CM}_{\mathsf{RL}}^{\cdot 2,+1}$ are bounded by 2^m along any run. Hence,

instead of storing the counter values of a $2\mathsf{CM}^{\cdot 2,+1}_{\mathsf{RL}}$ exactly, we instead use *exponential precision*. We encode a number $n \in \mathbb{N}$ by $\frac{n}{2^m} \in [0,1]$. Since then all the values are in $[0,1]$, we can double the counter values in a $[0,1]$-$\mathsf{VASS}^{0?}_{\mathsf{RL}}$ by forcing the firing fraction of the $[0,1]$-$\mathsf{VASS}^{0?}_{\mathsf{RL}}$ to be a particular value. The firing fraction is controlled, in turn, by means of the zero tests.

Ingredient III: Constructing precise numbers. In order to simulate increments of the $2\mathsf{CM}^{\cdot 2,+1}_{\mathsf{RL}}$ in our $[0,1]$-$\mathsf{VASS}^{0?}_{\mathsf{RL}}$, we need to be able to add $\frac{1}{2^m}$ to a counter. To this end, we present a $[0,1]$-$\mathsf{VASS}^{0?}_{\mathsf{RL}}$ gadget of polynomial size that produces the (exponentially precise) number $\frac{1}{2^m}$ in a given counter. The idea is to start with 1 in the counter and halve it m times. The trick is to use an additional counter that goes up by $1/m$ for each halving step. Checking this counter to be 1 in the end ensures that exactly m halvings have been performed.

Ingredient IV: Zero tests via run length. We then reduce $[0,1]$-$\mathsf{VASS}^{0?}_{\mathsf{RL}}$ to $\mathbb{Q}_+$-$\mathsf{VASS}_{\mathsf{RL}}$, which are $\mathbb{Q}_+$-VASS with a run-length constraint. Here, we need to (i) make sure that the counter values remain in $[0,1]$ and (ii) simulate zero tests. We achieve both by introducing a *complement counter* $\bar{x}$ for each counter x, where it is guaranteed that $x + \bar{x} = 1$ at all times. This means that instead of checking $x = 0$, we can check $\bar{x} = 1$ by subtracting 1 from $\bar{x}$. However, this does not suffice—we need to ensure that the firing fraction is exactly 1 in these steps. Here, the key idea is, whenever we check $\bar{x} = 1$, we also increment at the same time (and thus with the same firing fraction), a separate counter called the *controlling counter*, which in the end must equal Z, the total number of zero tests. This exploits the fact that every run attempts the same pre-determined number of zero tests due to the run-length constraint. If the controlling counter reaches the value Z at the very end, then we are assured that every zero-test along the run was indeed performed correctly.

Finally, we reduce from $\mathbb{Q}_+$-$\mathsf{VASS}_{\mathsf{RL}}$ to $\mathbb{Q}_+$-PVASS by using the pushdown stack to count from zero up to a number specified in binary. This employs a standard trick for encoding a binary number on the stack, where the least significant bit is on top. We further show that the final $\mathbb{Q}_+$-PVASS that we construct has bounded stack-height, 13 counters, and also that the target configuration can be reached from the source configuration if and only if the target can be covered from the source. This proves that coverability is hard even for a constant number of counters.

Ingredient V: Unary encodings. The above reduction produces instances of $\mathbb{Q}_+$-PVASS where the configurations are encoded in binary. Proving hardness for unary encodings requires more ideas. First, by using a trick akin to exponential precision from above, we show that hardness of coverability in $\mathbb{Q}_+$-PVASS holds already when all the values of the given configurations are less than 1. Next, by reusing the doubling and the halving gadgets from Ingredients II and III, we show that for any fraction $p/2^k$ where p is given in binary, there exists an *amplifier*, i.e., there is a $\mathbb{Q}_+$-VASS of polynomial size in $\log(p)$ and k, which starting from an unary-encoded configuration is able to put the value $p/2^k$ in a given counter. We then simply plug in a collection of these amplifiers before and after our original $\mathbb{Q}_+$-PVASS to get the desired result.

Related work. There have been several attempts to study restrictions or relaxations of the PVASS reachability problem. For example, reachability is decidable when one is allowed to simultaneously test the first i counters of a VASS for zero for any i [Reinhardt 2008]; this model can be seen as a special case of PVASS [Atig and Ganty 2011]. Furthermore, the coverability problem in one-dimensional PVASS is decidable [Leroux et al. 2015] and PSPACE-hard [Englert et al. 2021]. Reachability is decidable for *bidirected* PVASS [Ganardi et al. 2022], although the best known upper bound is Ackermann time (primitive recursive time in fixed dimension). Our work is in the same spirit. The continuous semantics reduces the complexity of reachability from Ackermann-complete for VASS to NP-complete [Blondin and Haase 2017] (and even to P for Petri nets [Fraca and Haddad

2015]). Our results show that the presence of a stack retains decidability, but allows exponentially more computational power.

For space reasons, detailed proofs can be found in the full version of this paper [Balasubramanian et al. 2023].

2 PRELIMINARIES

We write $\mathbb{Q}$ for the set of rationals and $\mathbb{Q}_+$ for the set of nonnegative rationals. Vectors over $\mathbb{Q}$ (or $\mathbb{Q}_+$) are written in bold ($\mathbf{u}, \mathbf{v}$ etc.) and are represented as a pair of natural numbers (numerator and denominator) for each rational. Throughout this paper, all numbers will be encoded in binary, unless stated otherwise. Note that, this means that, each rational number is a pair of natural numbers, with both of them encoded in binary.

Machine models. A d-dimensional *Continuous Vector Addition System with States* (d-$\mathbb{Q}_+$-VASS or simply $\mathbb{Q}_+$-VASS) $\mathcal{M} = (Q, T, \Delta)$ consists of a finite set Q of states, a finite set $T \subseteq \mathbb{Z}^d$ of transitions, and a finite set $\Delta \subseteq Q \times T \times Q$ of rules. We will on occasion consider an infinite $\mathbb{Q}_+$-VASS where T continues to be finite, but Q and Δ are infinite.

A *configuration* of $\mathcal{M}$ is a tuple $C = (q, \mathbf{v})$ where q is a state and $\mathbf{v} \in \mathbb{Q}_+^d$ is a valuation of the counters. We use the notations $\mathrm{state}(C), \mathrm{val}(C), C(i)$ to denote $q, \mathbf{v}, \mathbf{v}(i)$ respectively. Let $I = (q, t, q') \in \Delta$ be a rule and let $\alpha \in (0, 1]$ be the *firing fraction*. A *step* from a configuration C to another configuration C' by means of the pair (α, I) (denoted by $C \xrightarrow{\alpha I} C'$) is possible iff $\mathrm{state}(C) = q, \mathrm{state}(C') = q'$ and $\mathrm{val}(C') = \mathrm{val}(C) + \alpha t$. A run of $\mathcal{M}$ is a finite sequence of steps $C_0 \xrightarrow{\alpha_1 I_1} C_1 \xrightarrow{\alpha_2 I_2} \dots \xrightarrow{\alpha_n I_n} C_n$, where $\alpha_1 I_1 \dots \alpha_n I_n$ is called a *firing sequence*, and we say C_n is *reachable* from C_0 (written $C_0 \xrightarrow{\alpha_1 I_1, \alpha_2 I_2, \dots, \alpha_n I_n} C_n$ or $C_0 \xrightarrow{*} C_n$).

We assume the reader is familiar with context-free grammars and give basic definitions and notation (see, e.g., [Sipser 2012]). A *context-free grammar* $\mathcal{G} = (S, N, \Sigma, P)$ consists of a finite set of nonterminals N, a starting nonterminal S, a finite alphabet Σ and a finite set of production rules $P \subseteq N \times (N \cup \Sigma)^*$. We will assume that $\mathcal{G}$ is in Chomsky Normal Form. As usual $\overset{*}{\Rightarrow}$ is the reflexive, transitive closure of $\Rightarrow$. A word $w \in \Sigma^*$ belongs to the language $L(\mathcal{G})$ of the grammar iff $S \overset{*}{\Rightarrow} w$.

A *Continuous Pushdown* VASS ($\mathbb{Q}_+$-PVASS) is a $\mathbb{Q}_+$-VASS additionally equipped with a stack. Formally, it is a tuple $\mathcal{M} = (Q, \Gamma, T, \Delta)$ where Q is a finite set of states, Γ is a finite stack alphabet, $T \subseteq \mathbb{Z}^d \times (\Gamma \cup \bar{\Gamma} \cup \varepsilon)$ is a finite set of transitions, and $\Delta \subseteq Q \times T \times Q$ is a finite set of rules. A configuration $C = (q, w, \mathbf{v})$ of $\mathcal{M}$ contains additionally the stack $w \in \Gamma^*$ and we write $w = \mathrm{stack}(C)$. A *step* $C \xrightarrow{\alpha I} C'$ using rule $I = (q, a, t, q')$ is possible iff $\mathrm{state}(C) = q$, $\mathrm{state}(C') = q'$, $\mathrm{val}(C') = \mathrm{val}(C) + \alpha t$, and one of the following holds: (1) $a \in \Gamma$ and $\mathrm{stack}(C') = a\ \mathrm{stack}(C)$, (2) $a \in \bar{\Gamma}$ and $\mathrm{stack}(C) = a\ \mathrm{stack}(C')$, or (3) $a = \varepsilon$ and $\mathrm{stack}(C) = \mathrm{stack}(C')$. The notion of run, firing sequence, and reachability is defined as for $\mathbb{Q}_+$-VASS. In some cases, we will need to extend the notion of step to allow vectors in $\mathbb{Q}^d$ in a configuration rather than just $\mathbb{Q}_+^d$. We then explicitly specify this in the form of an underscript: $\rightarrow_{\mathbb{Q}_+}$ or $\rightarrow_{\mathbb{Q}}$ to make it clear.

Decision problems. The reachability problem for $\mathbb{Q}_+$-PVASS is defined as follows:

> Given a $\mathbb{Q}_+$-PVASS $\mathcal{M}$ and two of its configurations C_0, C_1, is C_1 reachable from C_0?

The coverability problem for $\mathbb{Q}_+$-PVASS is defined as follows:

> Given a $\mathbb{Q}_+$-PVASS $\mathcal{M}$ and two of its configurations $C_0, C_1 = (q_1, w_1, \mathbf{v}_1)$, does there exist a configuration $C' = (q_1, w_1, \mathbf{v}_1')$ with $\mathbf{v}_1' \geq \mathbf{v}_1$ such that C' reachable from C_0?

The state reachability problem is defined as follows:

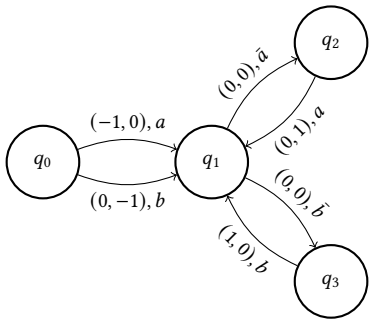


Fig. 1. An example $\mathbb{Q}_+$-PVASS with 2 counters. Here a and b are the stack symbols.

Given a $\mathbb{Q}_+$-PVASS $\mathcal{M}$, a configuration C_0 and a state q, does there exist a configuration C_1 with $\text{state}(C_1) = q$ that is reachable from C_0?

Example 2.1. Let us consider the $\mathbb{Q}_+$-PVASS from Figure 1, which we shall denote by $\mathcal{M}$. It has 2 counters and stack symbols a and b. Recall that a label a represents a push of a and $\bar{a}$ represents a pop of a. There are only two outgoing rules from the state q_0: the first rule r_1 decrements the first counter by 1, does not modify the second counter and pushes a onto the stack and the second rule r_2 decrements the second counter by 1, does not modify the first counter and pushes b onto the stack. Hence, starting from the configuration $(q_0, \varepsilon, (0,0))$ it is not possible to reach a configuration whose state is q_1. This implies that the input $(\mathcal{M}, (q_0, \varepsilon, (0,0))), q_1)$ is a negative instance of the state reachability problem.

On the other hand, starting from $(q_0, \varepsilon, (1,1))$, by firing r_1 with fraction 0.5, we can reach $(q_1, a, (0.5, 1))$. This means that $(\mathcal{M}, (q_0, \varepsilon, (1,1)), q_1)$ is a positive instance of the state reachability problem. Moreover, this also means that $(\mathcal{M}, (q_0, \varepsilon, (1,1)), (q_1, a, (0.5, 0.5)))$ is a positive instance of the coverability problem.

However, the input $(\mathcal{M}, (q_0, \varepsilon, (1,1)), (q_1, a, (1,1)))$ is a negative instance of the coverability problem. To see this, suppose for the sake of contradiction, a run exists between $(q_0, \varepsilon, (1,1))$ and $(q_1, a, (n_1, n_2))$ for some $n_1 \geq 1, n_2 \geq 1$. The first step of this run has to fire either r_1 or r_2 by some non-zero fraction α. Suppose r_1 is fired. (The argument is similar for the other case). Then a gets pushed onto the stack and the value of the first counter becomes $1 - \alpha$. From that point onwards, the only rules that can be fired are the ones going in and out of the state q_2, both of which do not increment the first counter. Hence, the first counter will have $1 - \alpha$ as its value throughout the run, which leads to a contradiction.

Finally, note that starting from $(q_0, \varepsilon, (1.1, 0.6))$, it is possible to reach $(q_1, a, (1,1))$: first, fire r_1 with fraction 0.1, then fire the incoming rule to q_2 (which pops a) with fraction 1 and then fire the outgoing rule from q_2 (which pushes a) with fraction 0.4. Hence, $(\mathcal{M}, (q_0, \varepsilon, (1.1, 0.6)), (q_1, a, (1,1)))$ is a positive instance of the reachability problem.

3 UPPER BOUND FOR REACHABILITY

We first prove the NEXPTIME upper bound in Theorem 1.2. To this end, we first use a standard language-theoretic translation to slightly rephrase the reachability problem in $\mathbb{Q}_+$-PVASS (this slight change in viewpoint is also taken in other work on PVASS [Englert et al. 2021; Leroux et al. 2015]). Observe that when we are given two configurations C_0 and C_1 of a $\mathbb{Q}_+$-PVASS, we want to know whether there exists a sequence $w \in (\mathbb{Z}^d)^*$ of update vectors such that (i) there exists a sequence σ of transitions that applies w, such that σ is a valid run from C_0 to C_1 in the pushdown

automaton underlying the $\mathbb{Q}_+$-PVASS (thus ignoring the counter updates) and (ii) there exist firing fractions for each vector in w such that adding the resulting update vectors will lead from $\mathbf{u}$ to $\mathbf{v}$, where $\mathbf{u}, \mathbf{v}$ are the vectors in the configurations C_0 and C_1. Now observe that the set of words w as in condition (i) are a context-free language. Therefore, we can phrase the reachability problem in $\mathbb{Q}_+$-PVASS by asking for a word in a context-free language that satisfies condition (ii).

Let us make condition (ii) precise. Let $\Sigma \subseteq \mathbb{Z}^d$ be the finite set of vectors that appear as transition labels in our $\mathbb{Q}_+$-PVASS. Given two configurations $\mathbf{u}, \mathbf{v} \in \mathbb{Q}_+^d$ and a word $w = w_1 w_2 \dots w_n \in \Sigma^*$ with each $w_i \in \Sigma$, we say that $\mathbf{u} \xrightarrow{w}_{\mathbb{Q}_+} \mathbf{v}$ iff there exist $\alpha_1, \dots, \alpha_n$ such that $\mathbf{u} \xrightarrow{\alpha_1 w_1, \alpha_2 w_2, \dots, \alpha_n w_n}_{\mathbb{Q}_+} \mathbf{v}$. Similarly, given a language $L \subseteq \Sigma^*$, we say that $\mathbf{u} \xrightarrow{L}_{\mathbb{Q}_+} \mathbf{v}$ iff $\mathbf{u} \xrightarrow{w}_{\mathbb{Q}_+} \mathbf{v}$ for some $w \in L$. By our observation above, the reachability problem in $\mathbb{Q}_+$-PVASS is equivalent to the following problem:

Given: A set of vectors $\Sigma \subseteq \mathbb{Z}^d$, a context-free language $L \subseteq \Sigma^*$ and $\mathbf{u}, \mathbf{v} \in \mathbb{Q}_+^d$.

Question: Does $\mathbf{u} \xrightarrow{L}_{\mathbb{Q}_+} \mathbf{v}$?

We solve this problem using results about the existential fragment of the first-order theory of $(\mathbb{Q}, +, <)$, which we call Existential Linear Rational Arithmetic (ELRA). Our algorithm constructs an ELRA formula for the following relation R_L.

Definition 3.1. The *reachability relation* R_L corresponding to a context-free language $L \subseteq (\mathbb{Z}^d)^*$ is given by $R_L = \{(\mathbf{u}, \mathbf{v}) \in \mathbb{Q}_+^d \times \mathbb{Q}_+^d \mid \mathbf{u} \xrightarrow{L}_{\mathbb{Q}_+} \mathbf{v}\}$.

The following definition of computing a formula using a non-deterministic algorithm is inspired by the definition of *leaf language* from complexity theory [Papadimitriou 2007]. We say that one can *construct an ELRA formula in* NEXPTIME *(resp.* NP*) for a relation* $R \subseteq \mathbb{Q}_+^n$ if there is a non-deterministic exponential (resp. polynomial) time-bounded Turing machine such that every accepting path of the machine computes an ELRA formula such that if $\varphi_1, \dots, \varphi_m$ are the produced formulae, then their disjunction $\bigvee_{i=1}^m \varphi_i$ defines the relation R. Here, a formula ϕ is said to define a relation $R \subseteq \mathbb{Q}_+^n$ if for every n-tuple $\boldsymbol{u} \in \mathbb{Q}_+^n$, we have $R(\boldsymbol{u})$ holds iff $\phi(\boldsymbol{u})$ is true of the rational numbers.

Proposition 3.2. *Given a context-free language* $L \subseteq (\mathbb{Z}^d)^*$ *one can construct in* NEXPTIME *an ELRA formula for the relation* R_L.

Since the truth problem for ELRA formulae can be solved in NP [Sontag 1985], the NEXPTIME upper bound follows from Proposition 3.2: Our algorithm would first non-deterministically compute a disjunct φ of the ELRA formula for R_L and then check the truth of φ in NP in the size of φ. This is a non-deterministic algorithm that runs in exponential time.

Therefore, the remainder of this section is devoted to proving Proposition 3.2. The key difficulty lies in understanding the reachability relation along *pumps*, which are derivations of the form $A \overset{*}{\Rightarrow} wAw'$ for some non-terminal A.

Definition 3.3. Let $\mathcal{G}$ be a context-free grammar over $\mathbb{Z}^d$ and A a non-terminal in $\mathcal{G}$. The *pump reachability relation* is defined as

$$P_A = \left\{ (\mathbf{u}, \mathbf{v}, \mathbf{u}', \mathbf{v}') \in \mathbb{Q}_+^d \times \mathbb{Q}_+^d \times \mathbb{Q}_+^d \times \mathbb{Q}_+^d \mid \exists w, w' \colon A \overset{*}{\Rightarrow} wAw',\ \mathbf{u} \xrightarrow{w}_{\mathbb{Q}_+} \mathbf{v},\ \mathbf{u}' \xrightarrow{w'}_{\mathbb{Q}_+} \mathbf{v}' \right\}.$$

Theorem 3.4. *Given a context-free grammar* $\mathcal{G}$ *over* $\mathbb{Z}^d$ *and a non-terminal* A *in* $\mathcal{G}$*, one can compute in* NEXPTIME *an ELRA formula for the relation* P_A.

Before proving Theorem 3.4, we first show how Proposition 3.2 follows from Theorem 3.4.

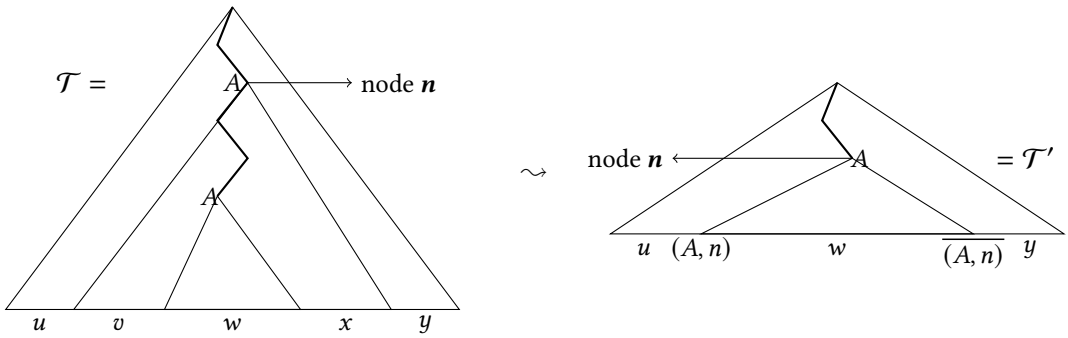


Fig. 2. Removal of a single cycle in an arbitrary derivation tree $\mathcal{T}$ to get a tree $\mathcal{T}'$ with additional leaves of the form (A, n) and $\overline{(A, n)}$.

Let $\mathcal{G}$ be a grammar for the language L in Proposition 3.2. Consider an arbitrary derivation tree $\mathcal{T}$ of $\mathcal{G}$. We say a derivation tree is *pumpfree* if along every path of the tree, every nonterminal occurs at most once. Clearly, an arbitrary derivation tree $\mathcal{T}$ can be obtained from a pumpfree tree by inserting "pumping" derivations of the form $A \stackrel{*}{\Rightarrow} wAw'$. Since every pumpfree tree is exponentially bounded in size (since its depth is bounded by the number of nonterminals $|N|$), there can only be exponentially many such pumps that need to be inserted for any given nonterminal A.

A pump $A \stackrel{*}{\Rightarrow} vAx$ on a nonterminal A in an arbitrary derivation tree $\mathcal{T}$ can be replaced by additional terminal letters called *pump letters* $(A, \mathbf{n})$ and $\overline{(A, \mathbf{n})}$ as shown in Fig. 2. Let the two occurrences of A be the first and last occurences of A along a path. Here $\mathbf{n} \in \{0, 1\}^*$ is a vector denoting the node which is labelled by the first A. Note that we assume that the grammar is in Chomsky Normal Form and hence nodes in the derivation tree can be identified in this manner since the trees are binary trees. The tree $\mathcal{T}'$ contains four children at $\mathbf{n}$, with the first and fourth being pump letters and the second and third being labelled by the nonterminals B, C occurring in the production $A \Rightarrow BC$. It could also be the case that the rule used is of the form $A \Rightarrow a$, in which case there are only three letters: the middle letter being a and the other two pump letters. Repeating this replacement procedure along each path, we finally obtain a pumpfree tree $\tilde{\mathcal{T}}$ which does not have a repeated nonterminal along any path. Since $\tilde{\mathcal{T}}$ is exponentially bounded in size, the number of pump terminals introduced is also exponentially bounded. In particular, every vector $\mathbf{n} \in \{0, 1\}^h$ for $h \leq |N|$ where N is the set of nonterminals of $\mathcal{G}$.

The algorithm guesses an exponential sized tree $\tilde{\mathcal{T}}$ and verifies the consistency of node labels between parent and children nodes in the tree using the rule set P of the grammar. It then constructs a formula $\phi_{\tilde{\mathcal{T}}}$ as follows. The formula $\phi_{\tilde{\mathcal{T}}}$ contains variables for a sequence of fractions and vectors $\mathbf{x}_0, \alpha_1, \mathbf{x}_1 \ldots \alpha_l, \mathbf{x}_l$ where l is the number of leaf nodes in $\tilde{\mathcal{T}}$. Let γ_i be the label of the i^{th} leaf node. The constructed formula is the conjunction of the following formulae ϕ_i for each leaf i:

- if γ_i is a nonpump letter then $\phi_i := (\mathbf{x}_i + \alpha_{i+1}\gamma_i = \mathbf{x}_{i+1})$, else
- γ_i is a pump letter $(A, \mathbf{n})$, then $\mathbf{x}_i, \mathbf{x}_{i+1}$ are plugged into an instantiation of the formula obtained from Theorem 3.4 for A, along with the corresponding vectors $\mathbf{x}_j, \mathbf{x}_{j+1}$ for the dual letter $\overline{(A, \mathbf{n})} = \gamma_j$ to give the formula $\phi_{i,j}$. In this case, $\phi_i = \phi_j = \phi_{i,j}$.

The formula $\phi_{\tilde{\mathcal{T}}}$ existentially quantifies over the variables $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{l-1}$ as well as the firing fractions $\alpha_1, \ldots, \alpha_l$, while $\boldsymbol{x}_0$ and $\boldsymbol{x}_l$ are free variables corresponding to $\boldsymbol{u}$ and $\boldsymbol{v}$ respectively. The final formula we want is $\bigvee_{\tilde{\mathcal{T}}} \phi_{\tilde{\mathcal{T}}}$.

3.1 Capturing Pump Reachability Relations

It remains to prove Theorem 3.4. The key observation is that a characterization of Blondin and Haase [2017] of the existence of "cyclic runs" (i.e. ones that start and end in the same control state) in $\mathbb{Q}_+$-VASS actually also applies to $\mathbb{Q}_+$-VASS with infinitely many control states. Thus, the first step is to translate the setting of pumps into that of cyclic $\mathbb{Q}_+$-VASS with infinitely many control states. It is more convenient for us to use algebraic terminology, so we will phrase this as a translation to the case of semigroups. We say a language $K \subseteq \Sigma^*$ is a *semigroup* if it is closed under concatenation, i.e., for any $u, v \in K$, we have $uv \in K$. We will show that for our particular cyclic $\mathbb{Q}_+$-VASS, the characterization of Blondin and Haase allows us to build an exponential-sized ELRA formula.

Reduction to semigroups. Let us first show that the pump reachability relations P_A can be captured using context-free semigroups. In this section, we often write letters a in normal font, even though they are vectors in $\mathbb{Z}^d$. Vectors in $\mathbb{Q}_+^d$ are represented by bold font e.g. $\mathbf{u}$.

The following lemma uses the idea of simulating grammar derivations by applying "the left part forward" and "the right part backward", which is a recurring theme in the literature on context-free grammars (see, e.g. [Baumann et al. 2023; Berstel 1979; Lohrey et al. 2022; Reps et al. 2016; Rosenberg 1967]) and has been applied to PVASS by Leroux et al. [2015, Section 5].

Lemma 3.5. *Given a grammar $\mathcal{G}$ over $\mathbb{Z}^d$ and a non-terminal A in $\mathcal{G}$, one can compute, in polynomial time, a context-free language $K \subseteq (\mathbb{Z}^{2d})^*$ such that (i) K is a semigroup and (ii) for any $\mathbf{u}, \mathbf{v}, \mathbf{u}', \mathbf{v}' \in \mathbb{Q}^d$, we have $(\mathbf{u}, \mathbf{v}') \xrightarrow{K}_{\mathbb{Q}_+} (\mathbf{u}', \mathbf{v})$ if and only if $(\mathbf{u}, \mathbf{v}, \mathbf{u}', \mathbf{v}') \in P_A$.*

Suppose (S, N, Σ, P) is a context-free grammar in Chomsky normal form, with $\Sigma \subseteq \mathbb{Z}^d$ and $A \in N$. The idea is to take a derivation tree for $A \overset{*}{\Rightarrow} wAw'$ and consider the path from the root to the A in the derived word, see Fig. 3 on the left. We transform the tree as follows. Each subtree on the left of this path (ℓ_1 and ℓ_2 in the figure) is left unchanged, except that each produced vector $a \in \mathbb{Z}^d$ is padded so as to obtain $(a, 0, \ldots, 0) \in \mathbb{Z}^{2d}$. In the figure, the resulting subtrees are $\overrightarrow{\ell_1}$, $\overrightarrow{\ell_2}$. Each subtree on the right (r_1 and r_2 in the figure), however, is moved to the left side of the path and it is *reversed*, meaning in particular that the word produced by it is reversed. Moreover, each vector $b \in \mathbb{Z}^d$ occurring at a leaf is turned into $(0, \ldots, 0, -b) \in \mathbb{Z}^{2d}$.

Then, every word generated by the new grammar will be of the form $\overrightarrow{x_1}\overleftarrow{y_n} \cdots \overrightarrow{x_n}\overleftarrow{y_1}$, where $x_1 \cdots x_n y_1 \cdots y_n$ is the word produced by the original grammar. Here, for a word $w \in (\mathbb{Z}^d)^*$, $\overrightarrow{w}$ is obtained from w by replacing each vector $a \in \mathbb{Z}^d$ in w by $(a, 0, \ldots, 0) \in \mathbb{Z}^{2d}$, and $\overleftarrow{w}$ is obtained from w by reversing the word and replacing each $a \in \mathbb{Z}^d$ by $(0, \ldots, 0, -a) \in \mathbb{Z}^{2d}$. Conversely, for every $A \overset{*}{\Rightarrow} wAw'$, we can find a word $\overrightarrow{x_1}\overleftarrow{y_n} \cdots \overrightarrow{x_n}\overleftarrow{y_1}$ in the new grammar such that $x_1 \cdots x_n = w$ and $y_1 \cdots y_n = w'$. Thus, we clearly have $(\mathbf{u}, \mathbf{v}') \xrightarrow{K}_{\mathbb{Q}_+} (\mathbf{u}', \mathbf{v})$ if and only if $(\mathbf{u}, \mathbf{v}, \mathbf{u}', \mathbf{v}') \in P_A$ for every $\mathbf{u}, \mathbf{v} \in \mathbb{Q}_+^d$.

Formally, in the new grammar for K, we have three copies of the non-terminals in N, hence we have $N' = \overleftarrow{N} \cup \overrightarrow{N} \cup \hat{N}$, where $\overleftarrow{N} = \{\overleftarrow{B} \mid B \in N\}$, $\overrightarrow{N} = \{\overrightarrow{B} \mid B \in N\}$ and $\hat{N} = \{\hat{B} \mid B \in N\}$ are disjoint copies of N. The productions in the new grammar are as follows: For every production $B \to CD$ in P, we include productions $\overrightarrow{B} \to \overrightarrow{C}\,\overrightarrow{D}$, $\hat{B} \to \overrightarrow{C}\hat{D}$ and $\hat{B} \to \overleftarrow{D}\hat{C}$, $\overleftarrow{B} \to \overleftarrow{D}\,\overleftarrow{C}$. Moreover, for every $B \to b$ with $b \in \mathbb{Z}^d$, we include the productions $\overrightarrow{B} \to (b, 0, \ldots, 0) \in \mathbb{Z}^{2d}$ and $\overleftarrow{B} \to (0, \ldots, 0, -b) \in \mathbb{Z}^{2d}$. Finally, we add $\hat{A} \to \varepsilon$ and set the start symbol to $\hat{A}$. Observe that the generated language is

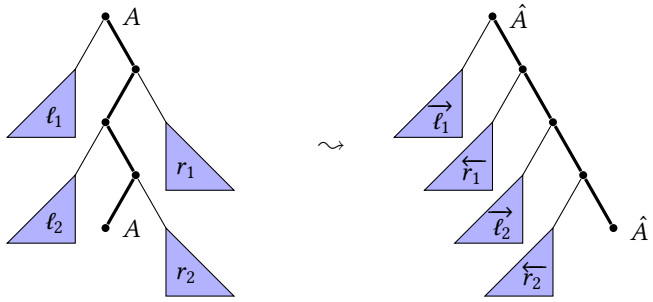


Fig. 3. Illustration of Lemma 3.5. A derivation of the original grammar (shown on the left) is transformed into a derivation of the new grammar (on the right).

closed under concatenation. This is because at any point in any derivation, there is exactly one hat nonterminal symbol which always occurs as the last symbol and only $\hat{A}$ can be replaced by ε. This means whenever $\hat{A} \stackrel{*}{\Rightarrow} u$, it is the case that $\hat{A} \stackrel{*}{\Rightarrow} u\hat{A}$ and hence if $\hat{A} \stackrel{*}{\Rightarrow} v$ as well, then it is the case that $\hat{A} \stackrel{*}{\Rightarrow} uv$ as well as $\hat{A} \stackrel{*}{\Rightarrow} vu$. This grammar achieves the required transformation.

Reduction to letter-uniform semigroup. As a second step, we will further reduce the problem to the case where n all runs, the letters (i.e. added vectors) appear uniformly (in some precise sense). The *support sequence* of a word $w \in \Sigma$ is the tuple $(\Gamma, <)$ where $\Gamma \subseteq \Sigma$ is the subset of letters occuring in w and $<$ is a total order on Γ which corresponds to the order of first occurrence of the letters in w. For example the support sequence of *aacabbc* consists of $\Gamma = \{a, b, c\}$ and the linear ordering $a < c < b$.

A context-free language $K \subseteq (\mathbb{Z}^d)^*$ is *letter-uniform* if any two words in K have the same support sequence. Let $\Sigma \subseteq \mathbb{Z}^d$ be the set of letters occurring in K. Moreover, for every subset $\Gamma \subseteq \Sigma$ and a total order $<$ on $\Gamma = \{\gamma_1, \ldots, \gamma_l\}$ given as $\gamma_1 < \gamma_2 \ldots < \gamma_l$, let $K_{(\Gamma,<)} = \{w \in K \mid \exists u_1, u_2, \ldots, u_l \; w = \gamma_1 u_1 \gamma_2 u_2 \ldots \gamma_l u_l \text{ where } u_i \in \{\gamma_1, \ldots, \gamma_i\}^*\}$ denote the set of all words in K with support sequence $(\Gamma, <)$.

Then we can observe that each $K_{(\Gamma,<)}$ is letter-uniform and also a semigroup: for any two words $u, v \in K_{(\Gamma,<)}$, it is the case that the letters occurring in uv and vu are exactly Γ and furthermore, the order of first occurrence of the letters from Γ in the two words also corresponds to the total order $<$. Furthermore, $uv, vu \in K$ since K is a semigroup. Hence both of these words also belong to $K_{(\Gamma,<)}$. Moreover, we have $\mathbf{u} \xrightarrow{K}_{\mathbb{Q}_+} \mathbf{v}$ if and only if there exists some $\Gamma \subseteq \Sigma$ and total order $<$ on Γ with $\mathbf{u} \xrightarrow{K_{(\Gamma,<)}}_{\mathbb{Q}_+} \mathbf{v}$. We shall prove the following:

Proposition 3.6. *Given a context-free letter-uniform semigroup $K \subseteq (\mathbb{Z}^d)^*$, we can in* NEXPTIME *construct an ELRA formula for the relation R_K.*

Let us see how Theorem 3.4 follows from Proposition 3.6. Given some nonterminal A of a CFG, we want a formula for P_A. We first use Lemma 3.5 to compute a context-free language K such that R_K captures P_A (up to permuting some counters). Suppose $K \subseteq \Sigma^*$ for some $\Sigma \subseteq \mathbb{Z}^d$. For each subset $\Gamma \subseteq \Sigma$ and total order $<$ on Γ, consider the set $K_{(\Gamma,<)}$ as defined earlier. As we already observed, (i) each $K_{(\Gamma,<)}$ is a semigroup, (ii) each $K_{(\Gamma,<)}$ is letter-uniform, and (iii) K is the union of all $K_{(\Gamma,<)}$. Therefore, our construction proceeds as follows. We guess $(\Gamma, <)$ and then apply Proposition 3.6 to compute in NEXPTIME an existential $\mathsf{FO}(\mathbb{Q}, +, <)$ formula for $R_{K_{(\Gamma,<)}}$. Then, the disjunction of all resulting formulas clearly defines R_K. We note that given some $(\Gamma, <)$, a grammar for $K_{(\Gamma,<)}$ can be constructed from the grammar for K in polynomial time. This is because we need

to construct a grammar for the intersection of K with the language of the regular expression given by $R := \gamma_1\gamma_1^*\gamma_2(\gamma_1 + \gamma_2)^*\gamma_3 \dots \gamma_k(\gamma_1 + \gamma_2 \dots \gamma_k)^*$, which only incurs a polynomial blowup. In fact, without the linear order and only the subset Γ, the same construction would lead to an exponential blowup since we would then have to remember all possible subsets of Γ while reading a word.

3.2 Characterizing Reachability by Three Runs

It remains to show Proposition 3.6. The advantage of reducing our problem to the letter-uniform case is that we can employ a characterization of Blondin and Haase [2017] about the existence of runs. In the rest of this section, we will assume that the language K comes with a corresponding support sequence $(\Gamma, <)$.

The following lemma tells us that reachability along a letter-uniform semigroup can be characterized by the existence of three runs: One run that witnesses reachability under $\mathbb{Q}$-semantics, and two runs that witness "admissibility" in both directions. Here, the "only if" direction is trivial, because the run from $\mathbf{u}$ to $\mathbf{v}$ along K is a run of all three types. For the converse, we use the fact that K is a semigroup and letter-uniform to compose the three runs into a run under $\mathbb{Q}_+$-semantics.

Lemma 3.7. *Let $K \subseteq (\mathbb{Z}^d)^*$ be a letter-uniform semigroup. Then we have $\mathbf{u} \xrightarrow{K}_{\mathbb{Q}_+} \mathbf{v}$ if and only if there are $\mathbf{u}', \mathbf{v}' \in \mathbb{Q}^d$ such that:*

$$\mathbf{u} \xrightarrow{K}_{\mathbb{Q}} \mathbf{v}, \qquad \mathbf{u} \xrightarrow{K}_{\mathbb{Q}_+} \mathbf{v}', \qquad \text{and} \qquad \mathbf{u}' \xrightarrow{K}_{\mathbb{Q}_+} \mathbf{v}.$$

Lemma 3.7 is an extension of [Blondin and Haase 2017, Proposition 4.5]. The only difference is that in [Blondin and Haase 2017], K is given by a non-deterministic finite automaton where one state is both the initial and the final state. The "only if" direction is trivial. For the "if" direction, the proof in [Blondin and Haase 2017] takes the three runs and shows that a suitable concatenation of these runs, together with an appropriate choice of multiplicities, yields the desired run $\mathbf{u} \xrightarrow{K}_{\mathbb{Q}_+} \mathbf{v}$. Since K is a semigroup, the same argument yields Lemma 3.7.

Lemma 3.7 allows us to express the reachability relation along K: It tells us that we merely have to express existence of the three simpler types of runs. The first of the three runs is reachability under $\mathbb{Q}$-semantics while the second and third are examples of *admissible* runs under $\mathbb{Q}_+$-semantics. Thus we need to characterize these two types of runs. We will do this in the following two subsections.

Characterizing reachability under $\mathbb{Q}$-semantics. We first show how to construct an ELRA formula for the $\mathbb{Q}$-reachability relation along a letter-uniform context-free K.

Lemma 3.8. *Given a letter-uniform context-free language $K \subseteq (\mathbb{Z}^d)^*$, we can construct in exponential time an ELRA formula for the relation*

$$R_K^{\mathbb{Q}} = \{(\mathbf{u}, \mathbf{v}) \in \mathbb{Q}_+^d \times \mathbb{Q}_+^d \mid \mathbf{u} \xrightarrow{K}_{\mathbb{Q}} \mathbf{v}\}.$$

Our proof relies on the following, which was shown in [Blondin and Haase 2017, Proposition B.4]:

Lemma 3.9 (Blondin and Haase [2017]). *Given an NFA $\mathcal{A}$ over some alphabet $\Sigma \subseteq \mathbb{Z}^d$, one can in polynomial time construct an ELRA formula φ such that for $\mathbf{u}, \mathbf{v} \in \mathbb{Q}_+^d$, we have $\varphi(\mathbf{u}, \mathbf{v})$ if and only if $\mathbf{u} \xrightarrow{L(\mathcal{A})}_{\mathbb{Q}} \mathbf{v}$.*

Proof of Lemma 3.8. The key observation is that in the case of $\mathbb{Q}$-semantics, reachability along a word $w \in (\mathbb{Z}^d)^*$ does not depend on the exact order of the letters in w. Let $\Psi(w) \in \mathbb{N}^{|\Sigma|}$ be the *Parikh image* of w i.e. $\Psi(w)(a)$ for $a \in \Sigma$ denotes the number of times a occurs in w. Formally, if $\Psi(w) = \Psi(w')$, then $\mathbf{u} \xrightarrow{w}_{\mathbb{Q}} \mathbf{v}$ if and only if $\mathbf{u} \xrightarrow{w'}_{\mathbb{Q}} \mathbf{v}$. In particular, for languages $K, K' \subseteq (\mathbb{Z}^d)^*$,

if $\Psi(K) = \Psi(K')$, then $R_K^{\mathbb{Q}} = R_{K'}^{\mathbb{Q}}$. We use this to reduce the case of context-free K to the case of regular languages K.

It is well known that given a context-free grammar, one can construct an NFA of exponential size such that the NFA accepts a language of the same Parikh image as the grammar. For example, a simple construction with a close-to-tight size bound can be found in [Esparza et al. 2011]. Therefore, given K, we can construct an exponential-sized NFA $\mathcal{A}$ such that $\Psi(L(\mathcal{A})) = \Psi(K)$.

Observe that $\Psi(L(\mathcal{A})) = \Psi(K)$ implies that for any $\mathbf{u}, \mathbf{v} \in \mathbb{Q}_+^d$, we have $\mathbf{u} \xrightarrow{K}_{\mathbb{Q}} \mathbf{v}$ if and only if $\mathbf{u} \xrightarrow{L(\mathcal{A})}_{\mathbb{Q}} \mathbf{v}$. Therefore, we apply Lemma 3.9 to compute a formula φ from $\mathcal{A}$. Since $\mathcal{A}$ is exponential in size, this computation takes exponential time and results in an exponential formula φ. Then, for $\mathbf{u}, \mathbf{v} \in \mathbb{Q}_+^d$, we have $\varphi(\mathbf{u}, \mathbf{v})$ if and only if $\mathbf{u} \xrightarrow{L(\mathcal{A})}_{\mathbb{Q}} \mathbf{v}$, which is equivalent to $\mathbf{u} \xrightarrow{K}_{\mathbb{Q}} \mathbf{v}$. □

We note that it is also possible to use a construction of [Verma et al. 2005] to construct a formula for $R_K^{\mathbb{Q}}$ in polynomial time – the reason we chose the current presentation is that applying the result from [Verma et al. 2005] as a black box would result in a formula over *mixed* integer-rational arithmetic (of polynomial size): One would use integer variables to implement the construction from [Verma et al. 2005] and then rational variables to account for continuous semantics. This would yield the same complexity bound in the end (existential mixed linear arithmetic is still in NP), but we preferred not to introduce another logic.

Characterizing admissibility under $\mathbb{Q}_+$-semantics. Finally, we construct an ELRA formula for the set of vectors $\mathbf{u}$ such that there exists a $\mathbf{v}' \in \mathbb{Q}_+^d$ with $\mathbf{u} \xrightarrow{K}_{\mathbb{Q}_+} \mathbf{v}'$. We call such vectors *$K$-admissible* and denote the set of K-admissible vectors as A_K. The key observation is that $\mathbf{u}$ is K-admissible if and only if the total order $<$ satisfies some simple properties. Intuitively, $\mathbf{u}$ is *$<$-admissible* if for each letter that decrements a counter, either (i) that counter is positive in $\mathbf{u}$ or (ii) there is an earlier letter that increments this counter. For $a \in \mathbb{Z}^d$, we denote by $[\![a]\!]$ (resp. $[\![a]\!]^+$ or $[\![a]\!]^-$) the subset of indices i where $a(i) \neq 0$ (resp. $a(i) > 0$ or $a(i) < 0$). Formally, $\mathbf{u}$ is $<$-admissible if for each $a \in \Gamma$ and each $j \in [\![a]\!]^-$, we either have (i) $\mathbf{u}(j) > 0$ or (ii) there is a $b \in \Gamma$ with $b < a$ and $j \in [\![b]\!]^+$. We show the following:

Lemma 3.10. *Let K be letter-uniform and $K \neq \emptyset$. Then $\mathbf{u} \in A_K$ if and only if $\mathbf{u}$ is $<$-admissible.*

The "if" direction is easy. If $\boldsymbol{u}$ is not $< -$*admissible*, this means that there is some index j and some letter γ_i such that γ_i decrements j, $\boldsymbol{u}(j) = 0$ and $\gamma_k(j) \leq 0$ for all $k < i$. This means that starting from $\boldsymbol{u}$, on any word $w \in K$, we would go below 0 in index j when γ_i first occurs in w. Hence $\boldsymbol{u} \notin K$.

For the converse, suppose that $\mathbf{u}$ is $<$-admissible and $w \in K$ be any word. Write $w = w_1 \cdots w_n$ with $w_1, \ldots, w_n \in \Gamma$. For each $i \in \{0, \ldots, n\}$, we define

$$I_i = [\![\boldsymbol{u}]\!]^+ \cup \bigcup_{\ell=1}^{i} [\![w_\ell]\!]^+,$$

i.e. the set of components that are incremented at some point when firing the prefix $w_1 \cdots w_i$.

We will show the following for every i: There exists a $\mathbf{v}_i$ such that $\mathbf{u} \xrightarrow{w_1 \cdots w_i}_{\mathbb{Q}_+} \mathbf{v}_i$ such that $\mathbf{v}_i \in \mathbb{Q}_+$ and $\mathbf{v}_i(j) > 0$ for $j \in I_i$.

We proceed by induction on i. For $i = 0$, the statement clearly holds, because $\mathbf{u}$ is positive on all co-ordinates in $I_0 = [\![\mathbf{u}]\!]^+$ and $\mathbf{u} \in \mathbb{Q}_+$. Now suppose there is a run $\mathbf{u} \xrightarrow{w_1 \cdots w_i}_{\mathbb{Q}_+} \mathbf{v}_i$. We know that $[\![w_{i+1}]\!]^- \subseteq I_i$ and thus $\mathbf{v}_i$ is positive on all indices where w_{i+1} is negative. Let $\alpha_{i+1} = \min\{-\frac{\mathbf{v}_i(j)}{2w_{i+1}(j)} \mid j \in [\![w_{i+1}]\!]^-\}$. Clearly $\mathbf{v}_i + \alpha_{i+1} w_{i+1} = \mathbf{v}_{i+1} \in \mathbb{Q}_+$ and also, $\mathbf{v}_{i+1}(j) > 0$ for all $j \in I_{i+1}$. In particular this means that $\mathbf{u} \xrightarrow{w}_{\mathbb{Q}_+} \mathbf{v}_n$ and thus $\mathbf{u} \in A_K$.

Lemma 3.11. *Given a non-empty letter-uniform context-free language $K \subseteq (\mathbb{Z}^d)^*$, we can construct in polytime an ELRA formula for the relation A_K.*

The formula $\phi_<$, where $\phi_<(\mathbf{u})$ for a vector $\mathbf{u} \in \mathbb{Q}_+^d$ is true iff $\mathbf{u}$ is $<$-admissible can be written as:

$$\phi_<(\mathbf{x}) = \bigwedge_{\gamma \in \Gamma} \bigwedge_{i \in [\![\gamma]\!]^-} \mathbf{x}(i) > 0 \vee \bigvee_{\eta < \gamma} i \in [\![\eta]\!]^+$$

Proof of Proposition 3.6. We are now ready to prove Proposition 3.6. By Lemma 3.7, it suffices to show that there are formulae for the relations $R_K^{\mathbb{Q}}$ and A_K. These formulae have been obtained in Lemma 3.8 and Lemma 3.11 respectively. □

This concludes the proof that reachability in $\mathbb{Q}_+$-PVASS is in NEXPTIME.

State reachability. The material in this section also allows us to derive Theorem 1.3. Since state-reachability is NP-hard already for $\mathbb{Q}_+$-VASS [Blondin and Haase 2017], we only have to show membership in NP. Using the language-theoretic translation from the beginning of this section, state reachability can be phrased as: Given a set of vectors $\Sigma \subseteq \mathbb{Z}^d$, a letter-uniform context-free language $K \subseteq \Sigma^*$ (which comes with an associated $(\Gamma, <)$) and $\mathbf{u} \in \mathbb{Q}_+^d$, decide if $\mathbf{u}$ is K-admissible. This is because, given an arbitrary context-free language K, we can see it as the disjoint union of (exponentially) many $K_{(\Gamma,<)}$ each of which is letter-uniform. Furthermore, we can construct each $K_{(\Gamma,<)}$ in polynomial time and check if they are non-empty. The NP upper bound then follows from Lemma 3.11: We can guess $(\Gamma, <)$, construct $K_{(\Gamma,<)}$ and Lemma 3.11 lets us build an ELRA formula for $A_{K_{(\Gamma,<)}}$. Since the truth problem for ELRA is in NP [Sontag 1985], we can then check whether $\mathbf{u}$ satisfies the constructed formula for A_K.

4 NEXPTIME-**HARDNESS OF** $2\mathsf{CM}_{\mathsf{RL}}^{\cdot 2,+1}$

We now move on to proving the NEXPTIME-hardness of reachability in $\mathbb{Q}_+$-PVASS. As outlined in the introduction, we do this by a chain of reductions. Our reduction chain starts with the machine model $2\mathsf{CM}_{\mathsf{RL}}^{\cdot 2,+1}$. Informally, a $2\mathsf{CM}_{\mathsf{RL}}^{\cdot 2,+1}$ has a finite-state control along with two counters, each of which can hold a non-negative integer. A rule of the machine allows us to move from one state to another whilst either incrementing the value of a counter by 1 or doubling the value of a counter. The set of final configurations of such a machine will be given by a final state and *an equality condition on the two counters*.

Formally, a $2\mathsf{CM}_{\mathsf{RL}}^{\cdot 2,+1}$ is a tuple $\mathcal{M} = (Q, q_{in}, q_f, \Delta)$ where Q is a finite set of states, $q_{in}, q_f \in Q$ are the initial and the final states respectively, and $\Delta \subseteq Q \times \{\mathbf{inc}_0, \mathbf{inc}_1, \mathbf{double}_0, \mathbf{double}_1, \mathbf{nop}\} \times Q$ is a finite set of rules. A configuration of $\mathcal{M}$ is a triple (q, v_0, v_1) where $q \in Q$ is the current state of $\mathcal{M}$ and $v_0, v_1 \in \mathbb{N}$ are the current values of the two counters respectively. Let $r = (q, t, q') \in \Delta$ be a rule. A *step* from a configuration $C = (p, v_0, v_1)$ to another configuration $C' = (p', v_0', v_1')$ by means of the rule r (denoted by $C \xrightarrow{r} C'$) is possible if and only if $p = q, p' = q'$, and

$$\text{If } t = \mathbf{inc}_i \text{ then } v_i' = v_i + 1,\ v_{1-i}' = v_{1-i} \qquad \text{If } t = \mathbf{double}_i \text{ then } v_i' = 2v_i,\ v_{1-i}' = v_{1-i}$$

$$\text{If } t = \mathbf{nop} \text{ then } v_0' = v_0,\ v_1' = v_1$$

We then say that a configuration C can reach another configuration C' if C' can be reached from C by a sequence of steps. The initial configuration of $\mathcal{M}$ is $C_{init} := (q_{in}, 0, 0)$. The set of *final configurations* of $\mathcal{M}$ is taken to be $\{(q_f, n, n) : n \in \mathbb{N}\}$.

The reachability problem for $2\mathsf{CM}_{\mathsf{RL}}^{\cdot 2,+1}$ asks, given a $2\mathsf{CM}_{\mathsf{RL}}^{\cdot 2,+1}$ $\mathcal{M}$ and a number *m in binary*, if the initial configuration can reach *some final configuration* in exactly m steps.

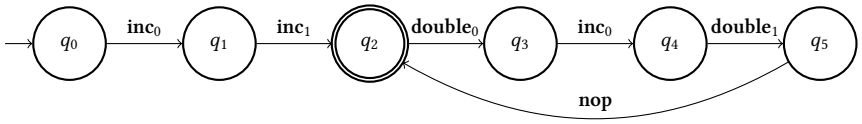


Fig. 4. An example $2\mathsf{CM}^{\cdot 2,+1}_{\mathsf{RL}}$. The initial state is q_0 and the final state is q_2.

Example 4.1. Let us consider the $2\mathsf{CM}^{\cdot 2,+1}_{\mathsf{RL}}$ given in Figure 4, which we shall denote by $\mathcal{M}$. The initial state is q_0 and the final state is q_2. Note that the initial configuration $(q_0, 0, 0)$ can reach $(q_2, 1, 1)$ in exactly 2 steps. Hence if we set the length of the run m to 2, then the instance $\langle \mathcal{M}, m\rangle$ is a positive instance of the reachability problem for $2\mathsf{CM}^{\cdot 2,+1}_{\mathsf{RL}}$. On the other hand, for any other value of m, the instance $\langle \mathcal{M}, m\rangle$ is a negative instance of the reachability problem for $2\mathsf{CM}^{\cdot 2,+1}_{\mathsf{RL}}$. Indeed, first note that in this $2\mathsf{CM}^{\cdot 2,+1}_{\mathsf{RL}}$, each state has exactly one outgoing transition. Hence, there is exactly one run starting from $(q_0, 0, 0)$ and that run is as follows: First it reaches the configuration $(q_2, 1, 1)$ in exactly 2 steps. Then from there, it follows the following (cyclical) pattern.

$$(q_2, x, y) \to (q_3, 2x, y) \to (q_4, 2x+1, y) \to (q_5, 2x+1, 2y) \to (q_2, 2x+1, 2y) \to (q_3, 4x+2, 2y)$$
$$(q_4, 4x+3, 2y) \to (q_5, 4x+3, 4y) \to (q_2, 4x+3, 4y) \ldots$$

This pattern indicates that after the configuration $(q_2, 1, 1)$, whenever the run reaches the state q_2, the first counter has an odd value, whereas the second counter has an even value. Hence, the run will never reach a final configuration and so $\langle \mathcal{M}, m\rangle$ is a negative instance of the reachability problem whenever $m \neq 2$.

We shall prove the following:

Theorem 4.2. *The reachability problem for* $2\mathsf{CM}^{\cdot 2,+1}_{\mathsf{RL}}$ *is* NEXPTIME*-hard.*

Theorem 4.2 is shown using a bounded version of the classical Post Correspondence Problem (PCP). Recall that, in this problem, we are given a set of pairs of words $(u_1, v_1), (u_2, v_2), \ldots, (u_m, v_m)$ over a common alphabet Σ and we are asked to decide if there is a sequence of indices $i_1, i_2, \ldots, i_k$ for some k such that $u_{i_1} \cdot u_{i_2} \cdot \cdots \cdot u_{i_k} = v_{i_1} \cdot v_{i_2} \cdot \cdots \cdot v_{i_k}$. It is well-known that this problem is undecidable [Sipser 2012]. For our purposes, we shall use a bounded version of PCP, called bounded PCP, defined as follows.

Given: A set of pairs of words $(u_1, v_1), (u_2, v_2), \ldots, (u_m, v_m)$ over an alphabet Σ such that none of the given words is the empty string, and a number ℓ encoded in binary.

Question: Is there a sequence of indices $i_1, i_2, \ldots, i_k$ such that $u_{i_1} \cdot u_{i_2} \cdot \cdots \cdot u_{i_k} = v_{i_1} \cdot v_{i_2} \cdot \cdots \cdot v_{i_k}$, and the length of $u_{i_1} \cdots u_{i_k}$ is exactly ℓ.

Note that this problem is decidable – we simply have to guess a sequence of indices of length at most ℓ and check that the resulting words from these indices satisfy the given property. In [Aiswarya et al. 2022, Section 6.1], Bounded-PCP was shown to be NEXPTIME-hard. We now prove Theorem 4.2 by giving a reduction from Bounded-PCP to the reachability problem for $2\mathsf{CM}^{\cdot 2,+1}_{\mathsf{RL}}$.

Let $(u_1, v_1), \ldots, (u_m, v_m)$ be a set of pairs of words over a common alphabet Σ and let $\ell \in \mathbb{N}$. Without loss of generality we assume that $|\Sigma| = 2^k$ for some $k \geq 1$, for instance, by adding at most twice as many dummy letters as the size of the alphabet. With this assumption, there are two essential ideas behind this reduction, which we now briefly outline.

The first idea is as follows: Since the size of Σ is 2^k, we can identify Σ with the set $\{0, 1, \ldots, 2^k - 1\}$, by mapping each letter in Σ to some unique number in $\{0, 1, \ldots, 2^k - 1\}$. This identification means

that, any non-empty word w represents a number n in base $|\Sigma|$ in the most significant bit notation. In this way, to any number n we can bijectively map a non-empty word w.

The second idea is as follows: Assume that we have a word w and its corresponding number n. Suppose we are given another word w' and we are asked to compute the number corresponding to the concatenated word $w \cdot w'$. We can do that as follows: Let $w' = w'_1, \ldots, w'_j$ with each w'_i being a letter. Construct the sequence of numbers $n_0, n_1, \ldots, n_j$ given by $n_0 = n$ and $n_i = |\Sigma| \cdot n_{i-1} + w'_i$. Notice that each n_i is essentially the representation of the string $w \cdot w'_1 \cdot w'_2 \cdots w'_i$ and so n_j is the representation for the word $w \cdot w'$.

These two ideas essentially illustrate the reduction from the Bounded-PCP problem to the reachability problem for $2\mathsf{CM}^{:2,+1}_{\mathsf{RL}}$. Given a Bounded-PCP instance $\langle (u_1, v_1), \ldots (u_m, v_m), \ell \rangle$, we construct a $2\mathsf{CM}^{:2,+1}_{\mathsf{RL}}$ as follows: Initially it starts at an initial state q_{in} with both of its counters set to 0. From here it executes a loop in the following manner: Suppose at some point, the machine is at state q_{in} with counter values n_1 and n_2 corresponding to some strings w_1 and w_2 respectively. Then the machine picks some index between 1 and k and then by the idea given in the previous paragraph, it updates the values of its counters to n'_1 and n'_2 corresponding to the strings $w_1 \cdot u_i$ and $w_2 \cdot v_i$, respectively and then comes back to the state q_{in}.

We can hard-code the rules in this machine so that whenever it has the representation for two strings w, w' in its counters and it wants to compute the representation for $w \cdot u_i$ and $w' \cdot v_i$ for some $1 \leq i \leq k$, it takes *exactly* t steps for some t which is polynomial in the size of the given Bounded-PCP instance. Then clearly, reaching a configuration (q_{in}, z, z) for some number z in the machine in exactly $t\ell$ steps is equivalent to finding a sequence of indices $i_1, \ldots, i_k$ such that $u_{i_1} \cdots u_{i_k} = v_{i_1} \cdots v_{i_k}$ and the length of $u_{i_1} \cdots u_{i_k}$ is exactly ℓ. This completes the reduction.

5 FROM $2\mathsf{CM}^{:2,+1}_{\mathsf{RL}}$ TO $[0,1]$-$\mathsf{VASS}^{0?}_{\mathsf{RL}}$

The next step in our reduction chain moves from $2\mathsf{CM}^{:2,+1}_{\mathsf{RL}}$ to $[0,1]$-$\mathsf{VASS}^{0?}_{\mathsf{RL}}$. Intuitively, a $[0,1]$-$\mathsf{VASS}^{0?}_{\mathsf{RL}}$ has a finite-state control along with some number of *continuous counters*, each of which can only hold *a fractional number* belonging to the interval $[0,1]$. A rule of such a machine allows us to move from one state to another whilst incrementing or decrementing some counters by *some fractional number*. Further a rule can also specify that *the effect of firing that rule* makes some counters 0, thereby allowing us to perform zero-tests. Note that a $[0,1]$-$\mathsf{VASS}^{0?}_{\mathsf{RL}}$ is different from $\mathbb{Q}_+$-VASS in two aspects: First, the counters of a $[0,1]$-$\mathsf{VASS}^{0?}_{\mathsf{RL}}$ can only hold numbers in $[0,1]$, whereas the counters of a $\mathbb{Q}_+$-VASS can hold any rational number. Second, the counters of a $[0,1]$-$\mathsf{VASS}^{0?}_{\mathsf{RL}}$ can be tested for zero, which is not possible in a $\mathbb{Q}_+$-VASS. We now proceed to formally define the model of a $[0,1]$-$\mathsf{VASS}^{0?}_{\mathsf{RL}}$.

More formally, a d-dimensional $[0,1]$-$\mathsf{VASS}^{0?}_{\mathsf{RL}}$ (or d-$[0,1]$-$\mathsf{VASS}^{0?}_{\mathsf{RL}}$ or simply $[0,1]$-$\mathsf{VASS}^{0?}_{\mathsf{RL}}$) is a tuple $C = (Q, T, \Delta)$ where Q is a finite state of states, $T \subseteq \mathbb{Z}^d \times 2^{[d]}$ is a finite set of *transitions* and $\Delta \subseteq Q \times T \times Q$ is a finite set of *rules*. A configuration of C is a tuple $C = (q, \mathbf{v})$ where $q \in Q$ is the current state of C and $\mathbf{v} \in [0,1]^d$ is the vector representing the current values of the counters of C. We use the notations $\mathsf{state}(C), \mathsf{val}(C), C(i)$ to denote $q, \mathbf{v}, \mathbf{v}(i)$, respectively. Let $I = (q, t, q') \in \Delta$ be a rule with $t = (r, s)$ and let $\alpha \in (0,1]$. A *step* from a configuration C to another configuration C' via the pair (α, I) (denoted by $C \xrightarrow{\alpha I} C'$) is possible if and only if $\mathsf{state}(C) = q, \mathsf{state}(C') = q'$ and

$$\mathsf{val}(C') = \mathsf{val}(C) + \alpha r \quad \text{and} \quad \mathsf{val}(C')(i) = 0 \text{ for all } i \in s$$

Note that we implicitly require that $\mathsf{val}(C) + \alpha r \in [0,1]^d$ and also that the value obtained after firing αI is 0 on all the counters in the set s. We define the notions of firing sequences $\alpha_1 I_1, \ldots, \alpha_n I_n$ and reachability between configurations $C \xrightarrow{\alpha_1 I_1, \ldots, \alpha_n I_n} C'$ as for $\mathbb{Q}_+$-VASS, The reachability problem

for $[0,1]$-VASS$^{0?}_{\mathsf{RL}}$ asks, given a $[0,1]$-VASS$^{0?}_{\mathsf{RL}}$ C, two configurations c_{init}, c_{fin} and a number m encoded *in binary*, whether c_{init} can reach c_{fin} in exactly m steps. We show that

Theorem 5.1. *The reachability problem for* $[0,1]$-VASS$^{0?}_{\mathsf{RL}}$ *is* NEXPTIME*-hard.*

We prove this theorem by exhibiting a reduction from the reachability problem for $2\mathsf{CM}^{\cdot 2,+1}_{\mathsf{RL}}$. Fix a $2\mathsf{CM}^{\cdot 2,+1}_{\mathsf{RL}}$ $\mathcal{M}$ and a number m in binary. Since the initial values of both the counters are 0, the largest value we can attain in any counter during a run of length m is at most 2^m (in fact, the bound is 2^{m-1}). Hence, we shall implicitly assume that the set of configurations of $\mathcal{M}$ that are under consideration are those where the counter values are bounded by 2^m.

Overview of the reduction. We want to construct an $[0,1]$-VASS$^{0?}_{\mathsf{RL}}$ C that simulates $\mathcal{M}$. As already mentioned in the introduction, we use *exponential precision* and represent a discrete counter value n in a configuration of $\mathcal{M}$ as the value $\frac{n}{2^m}$ in a continuous counter of C. Furthermore, we want to correctly simulate increment and doubling operations on $\mathcal{M}$ which correspond to addition of $\frac{1}{2^m}$ and doubling in C respectively. Since we do not control the fraction α in a rule, we have to overcome the following challenge:

(C1) How can we create gadgets which simulate addition of $\frac{1}{2^m}$ and doubling?

Towards solving this challenge, we use the following idea: Suppose we are in some configuration C and suppose we want to make a step from C by adding $\frac{1}{2^m}$ to a counter c. Assume that there are two other counters st and te whose values in C are $\frac{1}{2^m}$ and 0, respectively. Suppose I is a rule which decrements st by 1, increments c and te both by 1 and then checks that the value of st (after firing I) is 0. Then, if $C \xrightarrow{\alpha I} C'$ is a step, it must be that α is *exactly* $\frac{1}{2^m}$. This is because, by assumption, before firing this rule the value of st was $\frac{1}{2^m}$ and after firing this rule, the zero-test ensures that the value of st is 0. Hence, the only possible value that α can take is $\frac{1}{2^m}$. Therefore, this rule allows us to add $\frac{1}{2^m}$ to the counter c.

However, note that after firing I, the values of st and te are reversed, i.e., the values of st and te are 0 and $\frac{1}{2^m}$, respectively. This is undesirable, as we might once again want to use st to simulate addition by $\frac{1}{2^m}$. Therefore, we add another rule J, which decrements te by 1, increments st by 1 and then checks that the value of te (after firing J) is 0. Then, a successful firing of the rule J by some fraction β means that $\beta = \frac{1}{2^m}$ (due to the same reasons as above) and so this would mean that the values of st and te after firing J would again become $\frac{1}{2^m}$ and 0, respectively. Hence, the counter te essentially acts as a temporary holder of the value of st and allows us to "refill" the value of st.

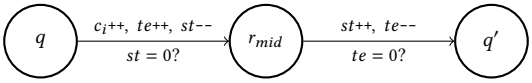
Generalizing this technique allows us to *control the firing fraction* to perform doubling as well. However, this technique has a single obstacle, which we now address.

For this technique to work, we need a counter st initially which stores the value $\frac{1}{2^m}$. It might be tempting to simply declare that the value of st in the initial configuration is $\frac{1}{2^m}$. However, this cannot be done, because the number m is given to us in binary and so the number of bits needed to write down the number $\frac{1}{2^m}$ is exponential in the size of the given input $\langle \mathcal{M}, m \rangle$, which would not give as a polynomial-time reduction. This raises the following challenge as well:

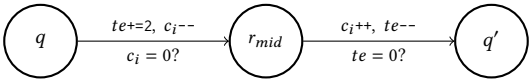
(C2) How can we create a value of $\frac{1}{2^m}$ in a continuous counter?

We show that challenge C2 can also be solved by our idea of controlling the firing fraction.

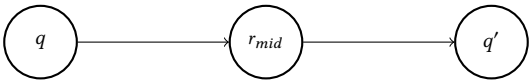
Solving Challenge (C1). From the $2\mathsf{CM}^{\cdot 2,+1}_{\mathsf{RL}}$ $\mathcal{M} = (Q, q_{in}, q_f, \Delta)$, we construct a $[0,1]$-VASS$^{0?}_{\mathsf{RL}}$ C_0 as follows. C_0 will have 4 counters c_0, c_1, st, te, i.e., it will be 4-dimensional. Intuitively, each c_i will store the value of one of the counters of $\mathcal{M}$, st will store the value $1/2^m$ that will be needed for simulating the addition operation, and te will be used to temporarily store the values of c_0, c_1 and st at some points along a run. A rule in C_0 consists of a vector $r \in \mathbb{Z}^4$ and a subset $s \subseteq \{1,2,3,4\}$.



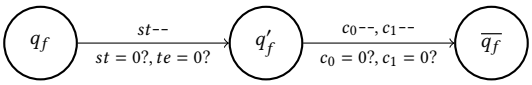
(a) Gadget for the rule $r := (q, \mathbf{inc}_i, q')$.



(b) Gadget for the rule $r := (q, \mathbf{double}_i, q')$.



(c) Gadget for the rule $r := (q, \mathbf{nop}, q')$.



(d) The "finish" gadget.

Fig. 5. Gadgets for simulating rules of $\mathcal{M}$.

For ease of reading, we write the vector r as a sequence of increment or decrement operations c+=n (or c−=n) whose intended meaning is that counter c is incremented (or decremented) by n, followed by a sequence of zero-tests. For example, $t = (r, s)$ where $r = (1, 0, 0, -2)$ and $s = \{1, 3\}$ is represented by c_0+=1, te−=2; $c_0 = 0?, st = 0?$.

C_0 will have all the states of $\mathcal{M}$ and in addition, for every rule r of $\mathcal{M}$, it will have a state r_{mid}. The set of rules of C_0 will be given as follows.

- For the rule $r := (q, \mathbf{inc}_i, q')$ of $\mathcal{M}$, C_0 will have the "increment(i)" gadget given in Figure 5a.
- For the rule $r := (q, \mathbf{double}_i, q')$, C_0 will have the "double(i)" gadget given in Figure 5b.
- For the rule $r := (q, \mathbf{nop}, q')$, C_0 will have the "nop" gadget given in Figure 5c.

Note that for every rule r of $\mathcal{M}$, the corresponding gadget in C_0 has exactly two rules, where the first rule (from q to r_{mid}) will be denoted by r^b and the second rule (from r_{mid} to q') by r^e. We would now like to show that the rules of $\mathcal{M}$ are simulated by their corresponding gadgets. To this end, we first define a mapping g from configurations of $\mathcal{M}$ to configurations of C_0 as follows: If $C = (q, v_0, v_1)$, then $g(C)$ is the configuration of C_0 such that

$$\text{state}(g(C)) = q,\ g(C)(c_0) = v_0/2^m,\ g(C)(c_1) = v_1/2^m,\ g(C)(st) = 1/2^m,\ g(C)(te) = 0$$

We now have the following "gadget simulation" lemma, which solves Challenge (C1).

Lemma 5.2 (Gadget Simulation). *Suppose C is a configuration and r is a rule of $\mathcal{M}$.*

- *Soundness: If $C \xrightarrow{r} C'$, then there exists α, β such that $g(C) \xrightarrow{\alpha r^b, \beta r^e} g(C')$.*
- *Completeness: If $g(C) \xrightarrow{\alpha r^b, \beta r^e} D$ for some α, β and D, then there exists C' such that $D = g(C')$ and $C \xrightarrow{r} C'$.*

Proof sketch. We have already discussed the case of increments in some detail before and so we will concentrate on when r is a doubling rule of the form $(q, \mathbf{double}_i, q')$. The soundness part

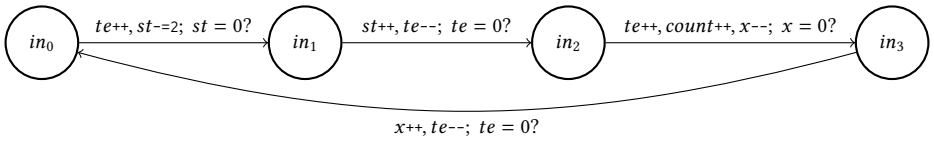


Fig. 6. The "initialization" gadget.

can be easily obtained by setting $\alpha = g(C)(c_i)$ and $\beta = 2 \cdot g(C)(c_i)$. For completeness, note that since r^b has a zero-test on c_i, it must be that $\alpha = g(C)(c_i)$. Hence, after firing αr^b, the value of te must be $2 \cdot g(C)(c_i)$. Now since r^e has a zero-test on te, it must be that $\beta = 2 \cdot g(C)(c_i)$. So the net effect of firing $\alpha r^b, \beta r^e$ is to make the value of c_i to be $2 \cdot g(C)(c_i)$. Hence, if we let C' be such that $C \xrightarrow{r} C'$ in $\mathcal{M}$, it can be verified that $g(C') = D$. □

The "finish" gadget. Before, we solve Challenge (C2), we make a small modification to C_0. Recall that in $\mathcal{M}$, we have a *set of final configurations* given by $F := \{(q_f, n, n) : n \leq 2^m\}$, whereas in a $[0,1]$-VASS$^{0?}_{\mathsf{RL}}$, we are allowed to specify only one final configuration. However, the $[0,1]$-VASS$^{0?}_{\mathsf{RL}}$ C_0 only promises us that the initial configuration c_{init} of $\mathcal{M}$ can reach some configuration in F in m steps iff $g(c_{init})$ can reach some configuration in the set $\{g(D) : D \in F\}$ in $2m$ steps. Hence, we need to make a modification to C_0 which allows us to replace the set of configurations with a single final configuration. To this end, we modify C_0 by adding the "finish gadget" from Figure 5d, where q'_f and $\overline{q_f}$ are two fresh states and the first and the second rule are respectively denoted by f^b and f^e. Let us call the resulting $[0,1]$-VASS$^{0?}_{\mathsf{RL}}$ as C_1.

Note that the effect of firing f^b is to set the values of st and te to 0. Further, if f^e is fired, then c_0 and c_1 are decremented by the same amount and both of them are tested for zero. This means that f^e could be fired successfully only if the counter values of c_0 and c_1 at state q'_f are the same and the effect of firing f^e is to set the values of c_0 and c_1 to 0. This observation along with repeated applications of the Gadget Simulation lemma give us the following Simulation theorem.

Theorem 5.3 (Simulation Theorem). *The initial configuration c_{init} of $\mathcal{M}$ can reach a final configuration in m steps iff $g(c_{init})$ can reach the configuration $(\overline{q_f}, \mathbf{0})$ in $2m + 2$ steps in C_1.*

We now move on to solving Challenge (C2).

Solving Challenge (C2). Thanks to the Simulation theorem, the required reduction is almost over. As we had discussed before, the only remaining part is that since $g(c_{init})(st) = 1/2^m$ and m is already given in binary, we cannot write down $g(c_{init})$ in polynomial time. To handle this challenge (Challenge (C2)), we construct an "initialization" gadget which starts from a "small" initial configuration and then "sets up" the configuration $g(c_{init})$.

The initialization gadget is shown in the Figure 6. The gadget shares the counters st and te with C_1 and has two new counters x and $count$. Initially, the gadget will start in in_0 and will have the values $1, 0, 1/m$ and 0 in st, te, x and $count$ respectively. In each iteration of the gadget, the value of st will be halved. The function of x is to store the value $1/m$ and the function of $count$ is to count the number of executions of this gadget. Initially the value of $count$ is 0 and in every iteration its value will increase by $1/m$. Hence, if we finally require the value of $count$ to be 1, then we would have executed this gadget precisely m times, thereby setting the value of st to $1/2^m$.

The following lemma follows from an analysis of the initalization gadget, similar to the one for the Gadget Simulation lemma.

Lemma 5.4 (The Initialization lemma). *Suppose C is a configuration of the initialization gadget such that* $\mathsf{state}(C) = in_0, C(te) = 0$ *and* $C(x) = 1/m$. *Then we can execute one iteration of the gadget from C to go to a configuration C' if and only if C' is the same as C except that $C'(st) = C(st)/2$ and $C'(count) = C(count) + 1/m$.*

We now construct our final $[0,1]$-VASS$^{0?}_{\mathsf{RL}}$ C as follows: We take the initialization gadget and the $[0,1]$-VASS$^{0?}_{\mathsf{RL}}$ C_1 and we add a rule from in_0 to q_{in} which does not do anything to the counters. Intuitively, we first execute the initialization gadget for some steps and then pass the control flow to C_1. We let d_{init} be the configuration of C whose state is in_0 and whose counter values are all 0, except for $d_{init}(x) = 1/m$ and $d_{init}(st) = 1$. Then, we let d_{fin} be the configuration of C whose state is $\overline{q_f}$ and whose counter values are all 0, except for $d_{fin}(x) = 1/m$ and $d_{fin}(count) = 1$. If we encode d_{init} and d_{fin} in binary, then they can be written down in polynomial time. Since $d_{fin}(count) = 1$, when the control flow passes from the initialization gadget to C_1, the value of st must be $1/2^m$, which is exactly what we want.

Theorem 5.5. *$g(c_{init})$ can reach the configuration $(\overline{q_f}, \mathbf{0})$ in the $[0,1]$-VASS$^{0?}_{\mathsf{RL}}$ C_1 in $2(m+1)$ steps if and only if d_{init} can reach d_{fin} in the $[0,1]$-VASS$^{0?}_{\mathsf{RL}}$ C in $4m + 1 + 2(m+1)$ steps.*

Combining this theorem with Theorem 5.3, proves the correctness of our reduction.

6 FROM $[0,1]$-VASS$^{0?}_{\mathsf{RL}}$ TO $\mathbb{Q}_+$-PVASS

We now move on to the next step in our reduction chain with the following problem called the reachability problem for $\mathbb{Q}_+$-VASS$_{\mathsf{RL}}$, defined as follows: Given a $\mathbb{Q}_+$-VASS $\mathcal{M}$, two configurations c_{init}, c_{fin}, and a number m *in binary*, whether one can reach c_{fin} from c_{init} in exactly m steps.

Theorem 6.1. *The reachability problem for $\mathbb{Q}_+$-VASS$_{\mathsf{RL}}$ is* NEXPTIME*-hard.*

We prove this theorem by giving a reduction from the reachability problem for $[0,1]$-VASS$^{0?}_{\mathsf{RL}}$. Fix a $[0,1]$-VASS$^{0?}_{\mathsf{RL}}$ C, two of its configurations c_{init}, c_{fin}, and a number m. Without loss of generality, we assume that every rule in C performs at least one zero-test.

Overview of the reduction. We want to construct a $\mathbb{Q}_+$-VASS $\mathcal{M}$ that simulates C for m steps. The primary challenge that prevents us from doing this is the following:

(D1) How can we create gadgets to simulate exactly m zero-tests of C in $\mathcal{M}$?

We circumvent this challenge as follows: We know that in a $[0,1]$-VASS$^{0?}_{\mathsf{RL}}$, the value of every counter will always be in the range $[0,1]$. Hence, for every counter x, we introduce another counter $\bar{x}$, called the *complementary counter* of x and maintain the invariant $x + \bar{x} = 1$ throughout a run. Then testing if the value of x is 0, amounts to testing if the value of $\bar{x}$ is at least 1. This allows us to replace a zero-test with a greater than or equal to 1 (geq1) test.

The latter can be implemented as follows: If t and t' are rules which decrement and increment $\bar{x}$ by 1 respectively and $C \xrightarrow{1t} C' \xrightarrow{1t'} C''$ is a run, then we know that the value of $\bar{x}$ in C is at least 1, which lets us implement a geq1 test. Note that for this to succeed, we require that both t and t' are fired completely, i.e., with fraction 1.

To sum this up, this means that if were to simulate a rule $r = (q, (w, s), q')$ of the $[0,1]$-VASS$^{0?}_{\mathsf{RL}}$ C in our new machine with the complementary counters, we need one rule to take care of the updates corresponding to w and two rules to take care of geq1 tests corresponding to the zero tests in s, both of which must be *fired completely*. Hence, simulating m steps of C in our new machine requires $3m$ steps, of which exactly $2m$ steps must be *fired completely*. This leads us to

(D2) How can we force the rules corresponding to geq1 tests to be fired completely, for exactly $2m$ times?

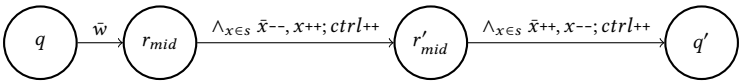


Fig. 7. Gadget for the rule $r := (q, t, q')$ with $t = (w, s)$.

To solve this challenge, we introduce another counter $ctrl$, called the *controlling counter*. We modify every rule corresponding to a geq1 test to also increment the value of the counter $ctrl$ by 1. This means that, if ρ is a run of $3m$ steps such that the value of $ctrl$ after ρ is exactly $2m$, then every rule corresponding to a geq1 test must have been fired completely along the run ρ.

Formal construction. Having given an informal overview of the reduction, we now proceed to the formal construction. We are given an $[0,1]$-VASS$^{0?}_{\mathsf{RL}}$ C and a number m in binary. From the $[0,1]$-VASS$^{0?}_{\mathsf{RL}}$ C, we will construct a $\mathbb{Q}_+$-VASS $\mathcal{M}$ as follows. For every counter x of C, $\mathcal{M}$ will have two counters x and $\bar{x}$. Every transition that increments x will decrement $\bar{x}$ by the same amount and vice-versa, so that the sum of the values of x and $\bar{x}$ will be equal to 1 throughout. Further, $\mathcal{M}$ will have another counter $ctrl$, called the *controlling counter*.

Suppose $r := (q, t, q')$ is a rule of C such that $t = (w, s)$. Denote by $\bar{w}$ the vector such that $w(ctrl) = 0$ and for every counter x of C, $\bar{w}(x) = w(x)$ and $\bar{w}(\bar{x}) = -w(x)$. Then corresponding to the rule r in C, $\mathcal{M}$ will have the gadget in Figure 7, whose first, second and third rules will be denoted by r^b, r^m and r^e respectively.

For any configuration C of C, let $G(C)$ denote the *set of configurations* of $\mathcal{M}$ such that $D \in G(C)$ iff $\mathrm{state}(D) = \mathrm{state}(C)$, $D(x) = C(x)$ and $D(\bar{x}) = 1 - C(x)$ for every counter x of C. Note that any two configurations in $G(C)$ differ only in their value of the counter $ctrl$. For any number α, let $G(C)_\alpha$ denote the unique configuration in $G(C)$ whose $ctrl$ value is α. The following lemma is a consequence of the discussion given in the overview section.

Lemma 6.2 (Control counter simulation).

- *Soundness: If* $C \xrightarrow{\alpha r} C'$ *in* C, *then for any* ζ, $G(C)_\zeta \xrightarrow{\alpha r^b, r^m, r^e} G(C')_{\zeta+2}$.
- *Completeness: If* $G(C)_\zeta \xrightarrow{\alpha r^b, \beta r^m, \gamma r^e} D'$ *for some* $\alpha, \beta, \gamma, \zeta$ *and* D *such that* $D(ctrl) = \zeta + 2$, *then there exists* C' *such that* $D' = G(C')_{\zeta+2}$, $\beta = \gamma = 1$ *and* $C \xrightarrow{\alpha r} C'$.

Repeated applications of the Control Counter Simulation lemma give us the following theorem, which completes our reduction.

Theorem 6.3. c_{init} *can reach* c_{fin} *in* C *in* m *steps iff* $G(c_{init})_0$ *can reach* $G(c_{fin})_{2m}$ *in* $3m$ *steps.*

Example 6.4. Let us see a concrete application of this reduction on some example. To this end, consider the $[0,1]$-VASS$^{0?}_{\mathsf{RL}}$ given in Figure 8. Note that this is essentially a renamed version of the "increment(i)" gadget described in Figure 5a. We consider this version here since it makes it easier to describe the effect of our reduction. The result of the application of the reduction on this $[0,1]$-VASS$^{0?}_{\mathsf{RL}}$ is given in Figure 9.

Suppose for some $u, v \in [0,1]$ with $u + v \leq 1$, we start in state q_0 in the $[0,1]$-VASS$^{0?}_{\mathsf{RL}}$ given in Figure 8 with counter values u, v and 0 for the counters c, st and te, respectively. From the argument given in the previous section, we know that if we fire the $[0,1]$-VASS$^{0?}_{\mathsf{RL}}$ in Figure 8 once, then we will reach the state q_2 with counter values $u + v, v$ and 0 for the counters c, st and te, respectively.

Now, suppose we start in q_0 in the $\mathbb{Q}_+$-VASS$_{\mathsf{RL}}$ given in Figure 9 with counter values $u, 1-u, v, 1-v, 0, 1$ and 0 in $c, \overline{c}, st, \overline{st}, te, \overline{te}$ and $ctrl$, respectively. From the reduction, we know that if we fire the gadget in Figure 9 once, and reach the state q_2 with counter value 4 for the controlling counter $ctrl$, then the counter values for counters $c, \overline{c}, st, \overline{st}, te, \overline{te}$ are $u + v, 1 - u - v, v, 1 - v, 0$ and 1 respectively.

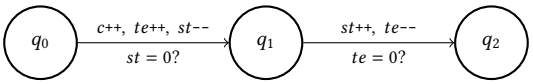


Fig. 8. Renamed version of the "increment(0)" gadget from Figure 5a. The rule from q_0 to q_1 shall be denoted by I and the rule from q_1 to q_2 shall be denoted by J.

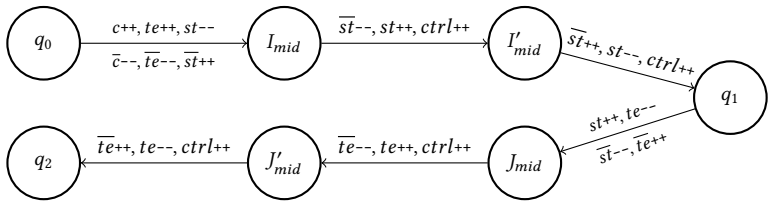


Fig. 9. Application of the reduction described in this section on the example $[0,1]$-VASS$^{0?}_{\mathsf{RL}}$ given in Figure 8.

Wrapping up. We now provide the final steps to prove that reachability for $\mathbb{Q}_+$-PVASS is NEXPTIME-hard. To do this, we recall a well-known folklore fact about pushdown automata. It essentially states that we can implement a binary counter in a PDA.

Lemma 6.5. *For any number m, in polynomial time in $\log(m)$, we can construct a PDA P_m of bounded stack-height and two configurations C and C' such that there is exactly one run from C to C'. Moreover, this run is of length exactly m.*

Proof. The essential idea is to use the stack to do a depth-first search of a binary tree of size $O(m)$. At each point, the PDA will only store at most $O(\log m)$ many entries in its stack, because the depth of the tree is $O(\log m)$. We now give a more precise construction.

Note that when $m = 1$, P_1 can simply be taken to be a PDA with two states and a single transition between the first state and the second state which does nothing to the stack. Now, let us consider the case when $m > 1$ is a power of 2, i.e., $m = 2^k$ for some k. Consider the following PDA P_m with k stack symbols $S^k_1, S^k_2, \ldots, S^k_k$. P_m starts in the state b_m with the empty stack. It then moves to state e_m while pushing S^k_1 onto the stack. The state e_m has k self-loop transitions as follows: For each $1 \leq i < k$, the i^{th} self-loop pops S^k_i and pushes S^k_{i+1} twice. Further, the k^{th} self-loop simply pops S^k_k. It can be easily verified that starting from state b_m with the empty stack, there is exactly one path to the configuration whose state is e_m and whose stack is empty. Moreover this path is of length exactly m. This is because the desired path is essentially the depth-first search traversal of a binary tree of size $m-1$, where the root is labelled by S^k_1 and each node at height i is labelled by S^k_{i+1}. Due to the depth-first search traversal, the number of elements stored in the stack at any point during the run is $O(k)$.

Now for the general case, suppose $m = \sum_{1 \leq i \leq n} 2^{k_i}$ for some $k_1 < k_2 < \cdots < k_n \leq \log(m)$. The desired PDA P_m has $\sum_{1 \leq i \leq n} k_i$ stack symbols given by $S^{k_1}_1, \ldots, S^{k_1}_{k_1}, S^{k_2}_1, \ldots, S^{k_2}_{k_2}, \ldots, S^{k_n}_1, \ldots, S^{k_n}_{k_n}$. Further, P_m has $n+1$ states $b^1_m, \ldots, b^n_m, b^{n+1}_m$. Initially, it starts in the state b^1_m with the empty stack. Then for each $1 \leq i \leq n$, it has a transition from b^i_m to b^{i+1}_m which pushes $S^{k_i}_1$ onto the stack. Then, at state b^{n+1}_m, it has the following set of self-loops: For each $1 \leq i \leq n$ and each $1 \leq j < k_i$, it pops $S^{k_i}_j$ from the stack and pushes $S^{k_i}_{j+1}$ twice. Further for each $1 \leq i \leq n$, it pops $S^{k_i}_{k_i}$. It can now be easily verified that starting from state b^1_m with the empty stack, there is exactly one path to the configuration whose state is b^{n+1}_m and whose stack is empty and also that this path is of length exactly m. □

We now give a reduction from reachability for $\mathbb{Q}_+$-VASS$_{\mathsf{RL}}$ to reachability for $\mathbb{Q}_+$-PVASS. Let $\mathcal{M} = (Q, T, \Delta)$ be a $\mathbb{Q}_+$-VASS such that c_{init} and c_{fin} are two of its configurations and let m be a number, encoded in binary. Construct the pair (P_m, C, C') as given by the Folklore lemma 6.5. We now take the usual cross product, i.e., the Cartesian product between P_m and $\mathcal{M}$, to obtain a $\mathbb{Q}_+$-PVASS C. (This operation is very similar to taking the cross product between a PDA and an NFA). Intuitively, the PDA part of C corresponds to simulating a binary counter, counting till the value m and the $\mathbb{Q}_+$-VASS part of C corresponds to simulating the $\mathbb{Q}_+$-VASS $\mathcal{M}$.

Let $\mathbf{u}$ (resp. $\mathbf{v}$) be the configuration of C such that $\mathrm{state}(\mathbf{u}) = (\mathrm{state}(C), \mathrm{state}(c_{init}))$, $\mathrm{stack}(\mathbf{u}) = \mathrm{stack}(C)$ and $\mathrm{val}(\mathbf{u}) = \mathrm{val}(c_{init})$ (resp. $\mathrm{state}(\mathbf{v}) = (\mathrm{state}(C'), \mathrm{state}(c_{fin}))$, $\mathrm{stack}(\mathbf{v}) = \mathrm{stack}(C')$ and $\mathrm{val}(\mathbf{v}) = \mathrm{val}(c_{fin})$). By construction of C, c_{init} can reach c_{fin} in $\mathcal{M}$ in m steps iff $\mathbf{u}$ can reach $\mathbf{v}$ in C. Hence, we have the following theorem.

Theorem 6.6. *Reachability in* $\mathbb{Q}_+$-PVASS *is* NEXPTIME*-hard.*

7 COVERABILITY, NUMBER OF COUNTERS AND ENCODINGS

The chain of reductions from reachability in $2\mathsf{CM}^{\cdot 2,+1}_{\mathsf{RL}}$ to reachability in $\mathbb{Q}_+$-PVASS prove that the latter is NEXPTIME-hard. The reduction from $2\mathsf{CM}^{\cdot 2,+1}_{\mathsf{RL}}$ to $[0,1]$-VASS$^{0?}_{\mathsf{RL}}$ was accomplished by using 6 counters, and the reduction from $[0,1]$-VASS$^{0?}_{\mathsf{RL}}$ to $\mathbb{Q}_+$-VASS$_{\mathsf{RL}}$ used $2x+1$ counters where x is the number of counters of the $[0,1]$-VASS$^{0?}_{\mathsf{RL}}$ instance. Finally, the reduction from $\mathbb{Q}_+$-VASS$_{\mathsf{RL}}$ to $\mathbb{Q}_+$-PVASS did not add any new counters. It follows that the lower bound already holds for $\mathbb{Q}_+$-PVASS of dimension 13.

We can go one step further. Similar to reachability in $\mathbb{Q}_+$-VASS$_{\mathsf{RL}}$ we can define coverability in $\mathbb{Q}_+$-VASS$_{\mathsf{RL}}$, where want to cover some configuration in a given number of steps. Let us inspect the $\mathbb{Q}_+$-VASS$_{\mathsf{RL}}$ $\mathcal{M}$ that we constructed in Section 6. We claim that

$$G(c_{init})_0 \text{ can reach } G(c_{fin})_{2m} \text{ in } 3m \text{ steps iff } G(c_{init})_0 \text{ can cover } G(c_{fin})_{2m} \text{ in } 3m \text{ steps.}$$

The left-to-right implication is trivial. For the other direction, notice that in any run of $3m$ steps in $\mathcal{M}$ starting from $G(c_{init})_0$, the value of $ctrl$ can be increased by at most $2m$. Further, for every counter $x \neq ctrl$, we maintain the invariant $x + \bar{x} = 1$ throughout. It then follows that the only way to cover $G(c_{fin})_{2m}$ in $3m$ steps is by actually reaching $G(c_{fin})_{2m}$. Hence, coverability in $\mathbb{Q}_+$-VASS$_{\mathsf{RL}}$ is also NEXPTIME-hard. Since the reduction in Section 6 preserves coverability, we obtain:

Theorem 7.1. *The coverability problem for 13-dimensional* $\mathbb{Q}_+$-PVASS *is* NEXPTIME*-hard.*

Let us now consider the encoding of the numbers that we use. It can be easily verified that in the final $\mathbb{Q}_+$-PVASS instance that we construct using our chain of reductions from Sections 4 till 6, all the numbers are fixed constants, except for the numbers appearing in the initial and final configurations, which are encoded in binary. Hence, the above theorem holds for 13-dimensional $\mathbb{Q}_+$-PVASS where the numbers are encoded in binary. We show that it is possible to strengthen this result to unary-encoded numbers at the cost of increasing the number of counters by a constant. More specifically, in the full version of this paper, we present an alternate reduction, which given an instance of reachability in $[0,1]$-VASS$^{0?}_{\mathsf{RL}}$ over x counters produces an instance of coverability in $\mathbb{Q}_+$-VASS$_{\mathsf{RL}}$ over $10x + 25$ counters where all numbers are encoded in unary. (We have already discussed the idea of this reduction in the Introduction). Since the proof in Section 5 shows that reachability in $[0,1]$-VASS$^{0?}_{\mathsf{RL}}$ is NEXPTIME-hard already over 6 counters, this will prove that coverability in $\mathbb{Q}_+$-VASS$_{\mathsf{RL}}$ over 85 counters where all numbers are encoded in unary is also NEXPTIME-hard. Since the reduction given in Section 6 from $\mathbb{Q}_+$-VASS$_{\mathsf{RL}}$ to $\mathbb{Q}_+$-PVASS produces a $\mathbb{Q}_+$-PVASS of bounded stack-height, does not add any new counters and does not change the encodings of the numbers, we can now conclude the following theorem.

Theorem 7.2. *The coverability problem for* $\mathbb{Q}_+$-PVASS *is* NEXPTIME*-hard, already over* $\mathbb{Q}_+$-PVASS*es of dimension 85, bounded stack-height, and when all numbers are encoded in unary.*

This hardness result is very strong, as it simultaneously achieves coverability, bounded stack, constant dimensions, and unary encodings. In contrast, in NEXPTIME, we can decide reachability of $\mathbb{Q}_+$-PVASS over arbitary dimension, even when all the numbers are encoded in binary.

Finally, the reduction from $\mathbb{Q}_+$-VASS$_{\mathsf{RL}}$ to $\mathbb{Q}_+$-PVASS in Section 6 only used the fact that for every m, (1) there is a PDA of size $O(\log(m))$ which can "count" exactly till m and (2) we can take product of a PDA with a $\mathbb{Q}_+$-VASS. For any model of computation that satisfies these two constraints, the corresponding reachability problem over continuous counters should also be NEXPTIME-hard. For instance, if we replace a stack in $\mathbb{Q}_+$-PVASS with *Boolean programs* to define *Boolean programs with continuous counters* then their reachability and coverability problems are also NEXPTIME-hard. A similar result also holds when we replace the stack with a (discrete) one-counter machine which can only increment its counter and whose accepting condition is reaching a particular counter value given in binary. For both models, the reachability and coverability problems must also be in NEXPTIME, because the former can be converted into an exponentially bigger $\mathbb{Q}_+$-VASS, for which these problems are in NP [Blondin and Haase 2017, Theorem 4.14].

8 CONCLUSION

We have shown that the reachability problem for continuous pushdown VASS is NEXPTIME-complete. While our upper bound works for any arbitrary number of counters, our lower bound already holds for the coverability problem for continuous pushdown VASS with constant number of counters, bounded stack-height and when all numbers are encoded in unary.

As part of future work, it might be interesting to study the complexity of coverability and reachability for continuous pushdown VASS over low dimensions. It might also be interesting to study the coverability and reachability problems for extensions of continuous pushdown VASS. For instance, it is already known that reachability in continuous VASS in which the counters are allowed to be tested for zero is undecidable [Blondin and Haase 2017, Theorem 4.17]. It might be interesting to see if this is also the case when the continuous counters are endowed with operations such as resets or transfers. Finally, it would be nice to extend the decidability result here to other machine models, such as continuous VASS with higher-order stacks.

ACKNOWLEDGMENTS

The authors are grateful to the anonymous reviewers for their helpful comments and for pointing out a small (and easily fixable) mistake in an earlier version. This research was sponsored in part by the Deutsche Forschungsgemeinschaft project 389792660 TRR 248–CPEC.

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement number 787367 (PaVeS). Funded by the European Union (ERC, FINABIS, 101077902). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

REFERENCES

C. Aiswarya, Soumodev Mal, and Prakash Saivasan. 2022. On the Satisfiability of Context-free String Constraints with Subword-Ordering. In *LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2 - 5, 2022*, Christel Baier and Dana Fisman (Eds.). ACM, 6:1–6:13. https://doi.org/10.1145/3531130.3533329

Rajeev Alur, Michael Benedikt, Kousha Etessami, Patrice Godefroid, Thomas W. Reps, and Mihalis Yannakakis. 2005. Analysis of recursive state machines. *ACM Trans. Program. Lang. Syst.* 27, 4 (2005), 786–818. https://doi.org/10.1145/1075382.1075387

Rajeev Alur, Ashutosh Trivedi, and Dominik Wojtczak. 2012. Optimal scheduling for constant-rate multi-mode systems. In *Hybrid Systems: Computation and Control (part of CPS Week 2012), HSCC'12, Beijing, China, April 17-19, 2012*, Thao Dang and Ian M. Mitchell (Eds.). ACM, 75–84. https://doi.org/10.1145/2185632.2185647

Mohamed Faouzi Atig, Ahmed Bouajjani, and Shaz Qadeer. 2009. Context-Bounded Analysis for Concurrent Programs with Dynamic Creation of Threads. In *Tools and Algorithms for the Construction and Analysis of Systems, 15th International Conference, TACAS 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings (Lecture Notes in Computer Science, Vol. 5505)*, Stefan Kowalewski and Anna Philippou (Eds.). Springer, 107–123. https://doi.org/10.1007/978-3-642-00768-2_11

Mohamed Faouzi Atig and Pierre Ganty. 2011. Approximating Petri Net Reachability Along Context-free Traces. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2011) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 13)*, Supratik Chakraborty and Amit Kumar (Eds.). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 152–163. https://doi.org/10.4230/LIPIcs.FSTTCS.2011.152

A. R. Balasubramanian, Rupak Majumdar, Ramanathan S. Thinniyam, and Georg Zetzsche. 2023. Reachability in Continuous Pushdown VASS. *arXiv* (2023). https://arxiv.org/abs/2310.16798

Pascal Baumann, Moses Ganardi, Rupak Majumdar, Ramanathan S. Thinniyam, and Georg Zetzsche. 2023. Context-Bounded Verification of Context-Free Specifications. *Proc. ACM Program. Lang.* 7, POPL (2023), 2141–2170. https://doi.org/10.1145/3571266

Jean Berstel. 1979. *Transductions and context-free languages.* Teubner, Stuttgart.

Michael Blondin, Alain Finkel, Christoph Haase, and Serge Haddad. 2016. Approaching the Coverability Problem Continuously. In *Tools and Algorithms for the Construction and Analysis of Systems (Lecture Notes in Computer Science)*, Marsha Chechik and Jean-François Raskin (Eds.). Springer, Berlin, Heidelberg, 480–496. https://doi.org/10.1007/978-3-662-49674-9_28

Michael Blondin and Christoph Haase. 2017. Logics for Continuous Reachability in Petri Nets and Vector Addition Systems with States. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE, Reykjavik, Iceland, 1–12. https://doi.org/10.1109/LICS.2017.8005068

Swarat Chaudhuri. 2008. Subcubic algorithms for recursive state machines. In *Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2008, San Francisco, California, USA, January 7-12, 2008*, George C. Necula and Philip Wadler (Eds.). ACM, 159–169. https://doi.org/10.1145/1328438.1328460

Michael Colón, Sriram Sankaranarayanan, and Henny Sipma. 2003. Linear Invariant Generation Using Non-linear Constraint Solving. In *Computer Aided Verification, 15th International Conference, CAV 2003, Boulder, CO, USA, July 8-12, 2003, Proceedings (Lecture Notes in Computer Science, Vol. 2725)*, Warren A. Hunt Jr. and Fabio Somenzi (Eds.). Springer, 420–432. https://doi.org/10.1007/978-3-540-45069-6_39

Wojciech Czerwiński and Lukasz Orlikowski. 2022. Reachability in Vector Addition Systems Is Ackermann-complete. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*. 1229–1240. https://doi.org/10.1109/FOCS52979.2021.00120

René David. 1987. Continuous Petri nets. In *Proc. 8th European Workshop on Appli. & Theory of Petri nets, 1987.*

Matthias Englert, Piotr Hofman, Sławomir Lasota, Ranko Lazić, Jérôme Leroux, and Juliusz Straszyński. 2021. A Lower Bound for the Coverability Problem in Acyclic Pushdown VAS. *Inform. Process. Lett.* 167 (April 2021), 106079. https://doi.org/10.1016/j.ipl.2020.106079

Javier Esparza, Pierre Ganty, Stefan Kiefer, and Michael Luttenberger. 2011. Parikh's theorem: A simple and direct automaton construction. *Inf. Process. Lett.* 111, 12 (2011), 614–619. https://doi.org/10.1016/j.ipl.2011.03.019

Estíbaliz Fraca and Serge Haddad. 2015. Complexity analysis of continuous Petri nets. *Fundamenta informaticae* 137, 1 (2015), 1–28.

Moses Ganardi, Rupak Majumdar, Andreas Pavlogiannis, Lia Schütze, and Georg Zetzsche. 2022. Reachability in Bidirected Pushdown VASS. In *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 229)*, Mikołaj Bojańczyk, Emanuela Merelli, and David P. Woodruff (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 124:1–124:20. https://doi.org/10.4230/LIPIcs.ICALP.2022.124

Michel Henri Théodore Hack. 1976. *Decidability questions for Petri Nets.* Ph. D. Dissertation. Massachusetts Institute of Technology.

Richard M. Karp and Raymond E. Miller. 1969. Parallel Program Schemata. *J. Comput. System Sci.* 3, 2 (May 1969), 147–195. https://doi.org/10.1016/S0022-0000(69)80011-5

Adam Husted Kjelstrøm and Andreas Pavlogiannis. 2022. The decidability and complexity of interleaved bidirected Dyck reachability. *Proc. ACM Program. Lang.* 6, POPL (2022), 1–26. https://doi.org/10.1145/3498673

Jérôme Leroux. 2022. The reachability problem for Petri nets is not primitive recursive. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 1241–1252.

Jérôme Leroux and Sylvain Schmitz. 2019. 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*. IEEE, 1–13. https://doi.org/10.1109/LICS.2019.8785796

Jérôme Leroux, Grégoire Sutre, and Patrick Totzke. 2015. On the Coverability Problem for Pushdown Vector Addition Systems in One Dimension. In *Automata, Languages, and Programming*, Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann (Eds.). Vol. 9135. Springer Berlin Heidelberg, Berlin, Heidelberg, 324–336. https://doi.org/10.1007/978-3-662-47666-6_26

Yuanbo Li, Qirun Zhang, and Thomas W. Reps. 2021. On the complexity of bidirected interleaved Dyck-reachability. *Proc. ACM Program. Lang.* 5, POPL (2021), 1–28. https://doi.org/10.1145/3434340

Markus Lohrey, Andreas Rosowski, and Georg Zetzsche. 2022. Membership Problems in Finite Groups. In *47th International Symposium on Mathematical Foundations of Computer Science, MFCS 2022, August 22-26, 2022, Vienna, Austria (LIPIcs, Vol. 241)*, Stefan Szeider, Robert Ganian, and Alexandra Silva (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 71:1–71:16. https://doi.org/10.4230/LIPICS.MFCS.2022.71

Christos H. Papadimitriou. 2007. *Computational complexity.* Academic Internet Publ.

Charles Rackoff. 1978. The Covering and Boundedness Problems for Vector Addition Systems. *Theor. Comput. Sci.* 6 (1978), 223–231. https://doi.org/10.1016/0304-3975(78)90036-1

Klaus Reinhardt. 2008. Reachability in Petri nets with inhibitor arcs. *Electronic Notes in Theoretical Computer Science* 223 (2008), 239–264.

Thomas W. Reps, Susan Horwitz, and Shmuel Sagiv. 1995. Precise Interprocedural Dataflow Analysis via Graph Reachability. In *Conference Record of POPL'95: 22nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Francisco, California, USA, January 23-25, 1995*, Ron K. Cytron and Peter Lee (Eds.). ACM Press, 49–61. https://doi.org/10.1145/199448.199462

Thomas W. Reps, Emma Turetsky, and Prathmesh Prabhu. 2016. Newtonian program analysis via tensor product. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 - 22, 2016*, Rastislav Bodík and Rupak Majumdar (Eds.). ACM, 663–677. https://doi.org/10.1145/2837614.2837659

Arnold L Rosenberg. 1967. A machine realization of the linear context-free languages. *Information and Control* 10, 2 (1967), 175–188. https://doi.org/10.1016/S0019-9958(67)80006-8

Louis E. Rosier and Hsu-Chun Yen. 1986. A Multiparameter Analysis of the Boundedness Problem for Vector Addition Systems. *J. Comput. Syst. Sci.* 32, 1 (1986), 105–135. https://doi.org/10.1016/0022-0000(86)90006-1

Sylvain Schmitz and Georg Zetzsche. 2019. Coverability Is Undecidable in One-Dimensional Pushdown Vector Addition Systems with Resets. In *13th International Conference on Reachability Problems (RP 2019) (Lecture Notes in Computer Science, Vol. 11674)*, Emmanuel Filiot, Raphaël Jungers, and Igor Potapov (Eds.). Springer International Publishing, Brussels, Belgium, 193–201. https://doi.org/10.1007/978-3-030-30806-3_15

Michael Sipser. 2012. *Introduction to the Theory of Computation.* Cengage Learning.

Eduardo D. Sontag. 1985. Real Addition and the Polynomial Hierarchy. *Inform. Process. Lett.* 20, 3 (April 1985), 115–120. https://doi.org/10.1016/0020-0190(85)90076-6

Saurabh Srivastava, Sumit Gulwani, and Jeffrey S. Foster. 2010. From program verification to program synthesis. In *Proceedings of the 37th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2010, Madrid, Spain, January 17-23, 2010*, Manuel V. Hermenegildo and Jens Palsberg (Eds.). ACM, 313–326. https://doi.org/10.1145/1706299.1706337

Kumar Neeraj Verma, Helmut Seidl, and Thomas Schwentick. 2005. On the Complexity of Equational Horn Clauses. In *Automated Deduction - CADE-20, 20th International Conference on Automated Deduction, Tallinn, Estonia, July 22-27, 2005, Proceedings (Lecture Notes in Computer Science, Vol. 3632)*, Robert Nieuwenhuis (Ed.). Springer, 337–352. https://doi.org/10.1007/11532231_25

Mihalis Yannakakis. 1990. Graph-Theoretic Methods in Database Theory. In *Proceedings of the Ninth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, April 2-4, 1990, Nashville, Tennessee, USA*. ACM Press, 230–242.

Received 2023-07-11; accepted 2023-11-07