Coverability in VASS Revisited: Improving Rackoff's Bounds to Obtain Conditional Optimality

MARVIN KÜNNEMANN, Karlsruhe Institute of Technology, Karlsruhe, Germany
FILIP MAZOWIECKI, University of Warsaw, Warszawa, Poland
LIA SCHÜTZE, Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany
HENRY SINCLAIR-BANKS*, Department of Computer Science, University of Warwick, Coventry, United Kingdom of Great Britain and Northern Ireland and Mathematics, Informatics and Mechanics, University of Warsaw, Warszawa, Poland
KAROL WĘGRZYCKI, Max Planck Institute for Informatics, Saarbrücken, Germany

Seminal results establish that the coverability problem for Vector Addition Systems with States (VASS) is in EXPSPACE (Rackoff, '78) and is EXPSPACE-hard already under unary encodings (Lipton, '76). More precisely, Rosier and Yen later utilise Rackoff's bounding technique to show that if coverability holds then there is a run of length at most $n^{2^{\mathcal{O}(d\log(d))}}$, where d is the dimension and n is the size of the given unary VASS. Earlier, Lipton showed that there exist instances of coverability in d-dimensional unary VASS that are only witnessed by runs of length at least $n^{2^{\Omega(d)}}$. Our first result closes this gap. We improve the upper bound by removing the twice-exponentiated $\log(d)$ factor, thus matching Lipton's lower bound. This closes the corresponding gap for the exact space required to decide coverability. This also yields a deterministic $n^{2^{\mathcal{O}(d)}}$-time algorithm for coverability. Our second result is a matching lower bound, that there does not exist a deterministic $n^{2^{o(d)}}$-time algorithm, conditioned upon the exponential time hypothesis.

*http://henry.sinclair-banks.com

Research of Marvin Künnemann is partially supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) grant number 462679611.
Filip Mazowiecki is supported by Polish National Science Centre SONATA BIS-12 grant number 2022/46/E/ST6/00230.
Lia Schütze is supported 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.
Henry Sinclair-Banks was supported by EPSRC Standard Research Studentship (DTP), grant number EP/T5179X/1 and is supported by the ERC grant INFSYS, agreement no. 950398.
Research of Karol Węgrzycki is partially supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) grant number 559177164.
Authors' Contact Information: Marvin Künnemann, Karlsruhe Institute of Technology, Karlsruhe, Baden-Württemberg, Germany; e-mail: marvin.kuennemann@kit.edu; Filip Mazowiecki, University of Warsaw, Warszawa, Poland; e-mail: f.mazowiecki@uw.edu.pl; Lia Schütze, Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Rheinland-Pfalz, Germany; e-mail: lschuetze@mpi-sws.org; Henry Sinclair-Banks, Department of Computer Science, University of Warwick, Coventry, United Kingdom of Great Britain and Northern Ireland and Mathematics, Informatics and Mechanics, University of Warsaw, Warszawa, Poland; e-mail: hsb@mimuw.edu.pl; Karol Węgrzycki, Max Planck Institute for Informatics, Saarbrücken, Saarland, Germany; e-mail: kwegrzyc@mpi-inf.mpg.de.

ACM 0004-5411/2025/10-ART33
https://doi.org/10.1145/3762178

When analysing coverability, a standard proof technique is to consider VASS with bounded counters. Bounded VASS make for an interesting and popular model due to strong connections with timed automata. Withal, we study a natural setting where the counter bound is linear in the size of the VASS. Here the trivial exhaustive search algorithm runs in $\mathcal{O}(n^{d+1})$ time. We give evidence to this being near-optimal. We prove that in dimension one this trivial algorithm is conditionally optimal, by showing that $n^{2-o(1)}$ time is required under the k-cycle hypothesis. In general, for any fixed dimension $d \geq 4$, we show that $n^{d-2-o(1)}$ time is required under the 3-uniform hyperclique hypothesis.

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

Additional Key Words and Phrases: Vector Addition System, Coverability, Reachability, Fine-Grained Complexity, Exponential Time Hypothesis, k-Cycle Hypothesis, Hyperclique Hypothesis

ACM Reference Format:
Marvin Künnemann, Filip Mazowiecki, Lia Schütze, Henry Sinclair-Banks, and Karol Węgrzycki. 2025. Coverability in VASS Revisited: Improving Rackoff's Bounds to Obtain Conditional Optimality. *J. ACM* 72, 5, Article 33 (October 2025), 27 pages. https://doi.org/10.1145/3762178

1 Introduction

Vector Addition Systems with States (VASS) are a popular model of concurrency with a number of applications in database theory [10], business processes [59], and more (see the survey [55]). A d-dimensional VASS (d-VASS) consists of a finite automaton equipped with d non-negative integer counters that can be updated by transitions. A configuration in a d-VASS consists of a state and a d-dimensional vector over the naturals. One of the central decision problems for VASS is the *coverability problem*, that asks whether there is a run from a given initial configuration to some configuration with at least the counter values of a given target configuration. Coverability finds applications in the verification of safety conditions, which often equate to whether or not a particular state can be reached without any precise counter values [17, 30]. Roughly speaking, one can use VASS as a modest model for concurrent systems where the dimension corresponds with the number of locations a process can be in and each counter value corresponds with the number of processes in a particular location [27, 31].

In 1978, Rackoff [53] showed that coverability is in EXPSPACE, by proving that if coverability holds then there exists a run of double-exponential length. Following, Rosier and Yen [54] analysed and discussed Rackoff's ideas in more detail and argued that if coverability holds then it is witnessed by a run of length at most $n^{2^{\mathcal{O}(d \log(d))}}$, where n is the size of the given unary encoded d-VASS. Furthermore, this yields a $2^{\mathcal{O}(d \log(d))} \cdot \log(n)$-space algorithm for coverability. Prior to this in 1976, Lipton [44] proved that coverability is EXPSPACE-hard even when VASS is encoded in unary, by constructing an instance of coverability witnessed only by a run of double-exponential length $n^{2^{\Omega(d)}}$. Rosier and Yen [54] also presented a proof that generalised Lipton's construction to show that $2^{\Omega(d)} \cdot \log(n)$ space is required for coverability. Although this problem is EXPSPACE-complete in terms of classical complexity, a gap was left open for the exact space needed for coverability [54, Section 1]. By using an approach akin to Rackoff's argument, we close this fourty-year-old gap by improving the upper bound to match Lipton's lower bound.

Result 1: If coverability holds then there exists a run witnessing coverability of length at most $n^{\mathcal{O}(d \cdot 2^d)}$ (Theorem 3.3). Accordingly, we obtain an optimal non-deterministic $2^{\mathcal{O}(d)} \cdot \log(n)$-space algorithm that decides coverability (Corollary 3.4).

Apart from closing the gap for the exact space needed to decide coverability, we would like to highlight the further relevance of this result. The upper bound of Result 1 relies on careful analysis of the minimal length of runs. In doing so, we introduce the notion of thin configurations (Definition 3.6). This enriches the ever-growing collection of notions and techniques used when tackling problems about infinite-state systems. We note that thin configurations have already been used in a recent paper [56] to improve the running time of other algorithms for coverability [39] and to improve the upper bound (and close the complexity gap) for coverability in invertible affine VASS [6]. We state this, in part, to emphasise that our main contribution is the conceptual notion of thin configurations, a notion we only discovered by considering the intersection of fine-grained complexity with traditional problems from formal methods. We believe that further investigation into this infrequently studied intersection will be fruitful for finding new concepts, techniques, and points-of-view.

Our bound also implies the existence of a deterministic $n^{2^{\mathcal{O}(d)}}$-time algorithm for coverability. We complement this with a matching lower bound on the deterministic running time, that is, conditioned upon the **Exponential Time Hypothesis** (**ETH**).

Result 2: Under ETH, there is no deterministic $n^{2^{o(d)}}$-time algorithm deciding coverability in unary d-VASS (Theorem 4.2).

One can see that the algorithm, that is, obtained from the upper bound (Result 1) is just a simple exhaustive search in a directed graph with a doubly exponential number of nodes. The lower bound (Result 2) shows that, under ETH, this simple algorithm is essentially optimal.

While our results establish a fast-increasing, conditionally optimal exponent of $2^{\Theta(d)}$ in the time complexity of the coverability problem, they rely on careful constructions that enforce the observation of large counter values. In certain settings, however, it is natural to instead consider a restricted version of coverability, where all counter values remain *bounded*. This yields one of the simplest models, fixed dimension bounded unary VASS, for which we obtain even tighter results. Decision problems for B-bounded VASS, where B forms part of the input, have been studied due to their strong connections to timed automata [28, 33, 49]. We consider linearly-bounded unary VASS, that is, when the maximum counter value is bounded above by a constant multiple of the size of the VASS. Interestingly, coverability and reachability are equivalent in linearly-bounded unary VASS. The trivial algorithm that employs depth-first search on the space of configurations runs in $\mathcal{O}(n^{d+1})$ time for both coverability and reachability. We provide evidence that the trivial algorithm is optimal. In the following two results, the $o(1)$ terms are sub-constants that may only depend on the size of the VASS n (and not the dimension d, which is fixed).

Result 3: Reachability in linearly-bounded unary 1-VASS requires $n^{2-o(1)}$ time, subject to the k-cycle hypothesis (Theorem 5.4).

This effectively demonstrates that the trivial algorithm is optimal in the one-dimensional case. For the case of large dimensions, we show that the trivial algorithm only differs from an optimal deterministic-time algorithm by at most an $n^{3+o(1)}$ factor.

Result 4: Reachability in linearly-bounded unary d-VASS requires $n^{d-2-o(1)}$ time, subject to the 3-uniform k-hyperclique hypothesis (Theorem 5.8).

Broadly speaking, these results add a time complexity perspective to the already known result about the space complexity of coverability: for any fixed dimension $d \geq 0$, coverability in unary d-VASS is NL-complete [53].

Organisation and Overview

Section 3 contains our first main result, the improved upper bound on the space required for coverability. Most notably, in Theorem 3.3, we show that if coverability holds then there exists a run of length at most $n^{\mathcal{O}(d \cdot 2^d)}$. Then, in Corollary 3.4 we are able to obtain a non-deterministic $\mathcal{O}(d^2 \cdot 2^d \cdot \log(n))$-space algorithm and a deterministic $n^{\mathcal{O}(d^2 \cdot 2^d)}$-time algorithm for coverability. In much of the same way as Rackoff, we proceed by induction on the dimension. The difference is in the inductive step; Rackoff's inductive hypothesis dealt with a case where all counters are bounded by the same well-chosen value. Intuitively speaking, the configurations are bounded within a d-hypercube. This turns out to be suboptimal. This is due to the fact that the volume of a d-hypercube with sides of length ℓ is ℓ^d; unrolling the induction steps gives a bound of roughly $n^{d \cdot (d-1) \cdot \ldots \cdot 1} = n^{d!} = n^{2^{\mathcal{O}(d \log(d))}}$, hence the twice-exponentiated $\log(d)$ factor. The key ingredient in our proof is to replace the d-hypercubes with a collection of hyperrectangles with greatly reduced volume, thus reducing the number of configurations in a run witnessing coverability.

Section 4 contains our second main result, the matching lower bound on the time required for coverability, that is, conditioned upon ETH. In Lemma 4.3, we first reduce from finding a k-clique in a graph to an instance of coverability in bounded unary 2-VASS with zero tests. Then, via Lemma 4.4, we implement the aforementioned technique of Rosier and Yen to, when there is a counter bound, remove the zero tests at the cost of increasing to a d-dimensional unary VASS. Then, in Theorem 4.2, by carefully selecting a value of $k = 2^{\Theta(d)}$, we are able to conclude that if ETH holds, then there does not exist a deterministic $n^{2^{o(d)}}$-time algorithm for coverability in unary d-VASS. This is because ETH implies that there is no $f(k) \cdot r^{o(k)}$-time algorithm for finding a k-clique in a graph with r vertices (Theorem 4.1).

Section 5 contains our other results where we study bounded fixed dimension unary VASS. Firstly, Theorem 5.4 states that under the k-cycle hypothesis (Hypothesis 5.2), there does not exist a deterministic $n^{2-o(1)}$-time algorithm deciding reachability in linearly-bounded unary 1-VASS. Further, we conclude in Corollary 5.5, if the k-cycle hypothesis is assumed then there does not exist a deterministic $n^{2-o(1)}$-time algorithm for coverability in (not bounded) unary 2-VASS. Following, we prove Theorem 5.8, that shows there does not exist a deterministic $n^{d-o(1)}$-time algorithm for reachability in linearly-bounded unary $(d+2)$-VASS under the 3-uniform k-hyperclique hypothesis (Hypothesis 5.7). We achieve this with two components. First, in Lemma 5.9, we reduce from finding a $4d$-hyperclique to an instance of reachability in a bounded unary $(d+1)$-VASS with a fixed number of zero tests. Second, via Lemma 5.10, we use the recently developed "controlling counter technique" [21] to remove the fixed number of zero tests at the cost of increasing the dimension by one.

Related Work

Despite being studied since the seventies, structural properties of the coverability problem for VASS still receive active attention. The set of configurations from which the target can be covered is upwards-closed, meaning that coverability still holds if the initial counter values are increased. An alternative approach, the *backwards algorithm* for coverability, relies on this phenomenon. Starting from the target configuration, one computes the set of configurations from which it can be covered [1]. Thanks to the upwards-closed property, it suffices to maintain the collection of minimal configurations. The backwards algorithm terminates due to Dickson's lemma, however, using Rackoff's bound one can show it runs in double-exponential time [11]. This technique has been analysed both for coverability in VASS and some extensions [29, 39]. Subsequently to our work, it was shown that thinness arises inherently in the configurations explored by the backwards coverability algorithm. Accordingly, an $n^{2^{\mathcal{O}(d)}}$ upper bound on the running time is attained [56]. Despite the high worst-case complexity, there are many implementations of coverability algorithms relying on the backwards algorithm that work well in practice. Intuitively, the idea is to prune the set of configurations, using relaxations that can be efficiently implemented in SMT solvers [8, 27].

A recently tested family of algorithms that explore the configuration graph (akin to the simple algorithm arising from Result 1) rely on heuristics such as A* and greedy best-first search [9].

Another central decision problem for VASS is the *reachability problem*, asking whether a run from a given initial configuration to a given target configuration exists. Reachability is a provably harder problem. In essence, reachability differs from coverability by allowing one zero test to each counter. Counter machines, well-known to be equivalent to Turing machines [51], can be seen as VASS with the ability to arbitrarily zero-test counters; coverability and reachability are equivalent here and are undecidable. In 1981, Mayr proved that reachability in VASS is decidable [47], making VASS one of the richest decidable variants of counter machines. Only recently, after decades of work, has the complexity of reachability in VASS been determined to be Ackermann-complete [20, 21, 41, 42]. A widespread technique for obtaining lower bounds for coverability and reachability problems in VASS is to simulate counter machines with some restrictions. Our overall approach to obtaining lower bounds follows suit; we first reduce finding cliques in graphs, finding cycles in graphs, and finding hypercliques in hypergraphs to various intermediate instances of coverability in VASS with additional restrictions or capabilities, such as bounded counters or a fixed number of zero tests. These VASS, which in some sense are restricted counter machines, are then simulated by standard higher-dimensional VASS. Such simulations are brought about by the two previously developed techniques. Rosier and Yen leverage Lipton's construction to obtain VASS that can simulate counter machines with bounded counters [54]. Czerwiński and Orlikowski have shown that the presence of an additional counter in a VASS, with carefully chosen transition effects and reachability condition, can be used to implicitly perform a limited number of zero tests [21].

Another studied variant, *bidirected* VASS, has the property that for every transition (p, x, q) the reverse transition $(q, -\mathrm{x}, p)$ is also present. The reachability problem in bidirected VASS is equivalent to the uniform word problem in commutative semigroups, both of which are EXPSPACE-complete [48]; not to be confused with the reversible (or mutual) reachability problem in general VASS which is also EXPSPACE-complete [40]. In 1982, Meyer and Mayr listed an open problem that stated, in terms of commutative semigroups, the best-known upper bound for coverability in general VASS [53], the best-known lower bound for coverability in bidirected VASS [44], and asked for improvements to these bounds [48, Section 8, Problem 3]. Subsequently, Rosier and Yen refined the upper bound for coverability in general VASS to $2^{\mathcal{O}(d \log(d))} \cdot \log(n)$ space [54]. Finally, Koppenhagen and Mayr showed that the coverability problem in bidirected VASS can be decided in $2^{\mathcal{O}(n)}$ space [37], matching the lower bound.

Recently, some work has been dedicated to the coverability problem for low-dimensional VASS [3, 50]. Furthermore, reachability in low-dimensional VASS has been given plenty of attention, in particular for 1-VASS [32, 58] and for 2-VASS [7, 35]. In the restricted class of flat VASS, other fixed dimensions have also been studied [16, 19, 22].

The Dyck reachability problem has been studied from the fine-grained complexity perspective [12, 13, 38, 46]). Coverability in unary 1-VASS is a special case of Dyck-1 reachability; in fact reachability in unary 1-VASS is equivalent to Dyck-1 reachability. Our lower bounds (in Section 5) for reachability in linearly-bounded unary 1-VASS also applies to the Dyck-1 reachability problem. The complexity of coverability in 1-VASS is also closely related to a recently studied problem about reachability for electric cars [25].

2 Preliminaries

We use $\mathbb{Z}$ to denote the set of integers, $\mathbb{N}$ to denote the set of non-negative integers, and $\mathbb{R}_+$ to denote the set of positive real numbers. Throughout, we assume that log has base 2. We say that function $f(x) = \mathcal{O}(g(x))$ for some functions $f, g : \mathbb{R}_+ \to \mathbb{R}_+$ if there exist constants c and N such that $f(x) \leq c \cdot g(x)$ for all $x > N$. Similarly, $f(x) = o(g(x))$ if for every $c > 0$ there exists an N

such that, for all $x > N$, $f(x) < c \cdot g(x)$ [57]. We use $\mathrm{poly}(n)$ to denote $n^{\mathcal{O}(1)}$. We use bold font for vectors. We index the ith component of a vector $\mathbf{v}$ by writing $\mathbf{v}[i]$. Given two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{Z}^d$ we write $\mathbf{u} \leq \mathbf{v}$ if $\mathbf{u}[i] \leq \mathbf{v}[i]$ for each $1 \leq i \leq d$. For every $1 \leq i \leq d$, we write $\mathbf{e}_i \in \mathbb{Z}^d$ to represent the ith standard basis vector that has $\mathbf{e}_i[i] = 1$ and $\mathbf{e}_i[j] = 0$ for all $j \neq i$. Given a vector $\mathbf{v} \in \mathbb{Z}^d$ we define $\|\mathbf{v}\| = \max\{1, |\mathbf{v}[1]|, \ldots, |\mathbf{v}[d']|\}$. For convenience, given two vectors $\mathbf{u} \in \mathbb{Z}^d$ and $\mathbf{v} \in \mathbb{Z}^{d'}$, we use $(\mathbf{u}, \mathbf{v})$ to denote the $(d + d')$-dimensional vector $(u[1], \ldots, u[d], v[1], \ldots, v[d'])$. Through this article, we analyse the running time of algorithms in the standard "*word RAM model*" with $\mathcal{O}(\log(n))$-bit words, where n is the size of the input (see [34] for a survey).

A *d-dimensional Vector Addition System with States* (d-VASS) $\mathcal{V} = (Q, T)$ consists of a non-empty finite set of states Q and a non-empty set of transitions $T \subseteq Q \times \mathbb{Z}^d \times Q$. A *configuration* of a d-VASS is a pair $(q, \mathbf{v}) \in Q \times \mathbb{N}^d$ consisting of the current state q and current counter values $\mathbf{v}$, denoted $q(\mathbf{v})$. Given two configurations $p(\mathbf{u})$, $q(\mathbf{v})$, we write $p(\mathbf{u}) \rightarrow q(\mathbf{v})$ if there exists $t = (p, \mathbf{x}, q) \in T$ where $\mathbf{u} + \mathbf{x} = \mathbf{v}$. We may refer to $\mathbf{x}$ as the *effect* or *update* of a transition and may also write $p(\mathbf{u}) \xrightarrow{t} q(\mathbf{v})$ to emphasise the transition t taken.

A *path* in a VASS is a (possibly empty) sequence of transitions $((p_1, \mathbf{x}_1, q_1), \ldots, (p_\ell, \mathbf{x}_\ell, q_\ell))$, where $(p_i, \mathbf{x}_i, q_i) \in T$ for all $1 \leq i \leq \ell$ and such that the start and end states of consecutive transitions match $q_i = p_{i+1}$ for all $1 \leq i \leq \ell - 1$. The *effect* of a path is the sum of the effects of the transitions in the path. A *run* π in a VASS is a sequence of configurations $\pi = (q_0(\mathbf{v}_0), \ldots, q_\ell(\mathbf{v}_\ell))$ such that $q_i(\mathbf{v}_i) \rightarrow q_{i+1}(\mathbf{v}_{i+1})$ for all $1 \leq i \leq \ell - 1$. We denote the length of the run by $\mathrm{len}(\pi) := \ell + 1$; we denote the effect of the run by $\mathrm{eff}(\pi) := \mathbf{v}_\ell - \mathbf{v}_0$. If there is such a run π, we can write $q_0(\mathbf{v}_0) \xrightarrow{\pi} q_\ell(\mathbf{v}_\ell)$. We may also write $p(\mathbf{s}) \xrightarrow{*} q(\mathbf{t})$ if there exists a run from $p(\mathbf{s})$ to $q(\mathbf{t})$. The *underlying path* of a run $\pi = (q_0(\mathbf{v}_0), \ldots, q_\ell(\mathbf{v}_\ell))$ is the sequence of transitions $(t_1, \ldots, t_\ell)$ such that, for every $0 \leq i \leq \ell - 1$, $q_i(\mathbf{v}_i) \xrightarrow{t_i} q_{i+1}(\mathbf{v}_{i+1})$.

We do allow for zero-dimensional VASS, that is, VASS with no counters, which can be seen as just directed graphs. A *hypergraph* is a generalisation of the graph. Formally, a hypergraph is a tuple $H = (V, E)$ where V is a set of vertices and E is a collection of non-empty subsets of V called *hyperedges*. For an integer μ, a hypergraph is *μ-uniform* if each hyperedge has cardinality μ. Note that a 2-uniform hypergraph is a standard graph.

We study the complexity of the *coverability problem*. An instance $(\mathcal{V}, p(\mathbf{s}), q(\mathbf{t}))$ of coverability asks whether there is a run in the given VASS $\mathcal{V}$ from the given initial configuration $p(\mathbf{s})$ to a configuration $q(\mathbf{t}')$ with counter values $\mathbf{t}' \geq \mathbf{t}$. At times, we also consider the *reachability problem* that additionally requires $\mathbf{t}' = \mathbf{t}$ so that the target configuration is reached exactly.

To measure the complexity of these problems we need to discuss the encoding used. In *unary encoding*, a d-VASS $\mathcal{V} = (Q, T)$ has *size* $\|\mathcal{V}\| = |Q| + \sum_{(p,\mathbf{x},q)\in T} \|\mathbf{x}\|$. An instance of coverability in a unary-encoded d-VASS $(\mathcal{V}, p(\mathbf{s}), q(\mathbf{t}))$ has size $\|\mathcal{V}\| + \|\mathbf{s}\| + \|\mathbf{t}\|$. We define a *unary* d-VASS $\mathcal{U} = (Q', T')$ to have restricted transitions $T' \subseteq Q' \times \{-1, 0, 1\}^d \times Q'$, the size is therefore $\|\mathcal{U}\| = |Q'| + |T'|$. For any unary encoded d-VASS $\mathcal{V}$ there exists an equivalent unary d-VASS $\mathcal{U}$ such that $\|\mathcal{U}\| = \|\mathcal{V}\|$.

It is well known that coverability in d-VASS can be reduced, in logarithmic space, to the reachability problem. Indeed, for an instance $(\mathcal{V}, p(\mathbf{s}), q(\mathbf{t}))$ of coverability, define $\mathcal{V}' = (Q, T')$ that has additional decrementing transitions at the target state q, precisely $T' = T \cup \{(q, -\mathbf{e}_i, q) : 1 \leq i \leq d\}$. It is clear that $p(\mathbf{s}) \xrightarrow{*} q(\mathbf{t}')$, for some $\mathbf{t}' \geq \mathbf{t}$, in $\mathcal{V}$ if and only if $p(\mathbf{s}) \xrightarrow{*} q(\mathbf{t})$ in $\mathcal{V}'$.

A *B-bounded d-VASS*, in short (B, d)-VASS, is given as an integer upper bound on the counter values $B \in \mathbb{N}$ and d-VASS $\mathcal{V}$. A configuration in a (B, d)-VASS is a pair $q(\mathbf{v}) \in Q \times \{0, \ldots, B\}^d$. The notions of paths and runs in bounded VASS remain the same as for VASS, but are accordingly adapted for the appropriate bounded configurations. We note that one should think that B forms part of the problem statement, not the input, as it will be given implicitly by a function depending on the size of the VASS. For example, we later consider *linearly-bounded* d-VASS, in which $B = \mathcal{O}(\|\mathcal{V}\|)$.

A *d-dimensional Vector Addition System* (d-VAS) $\mathcal{U}$ is a system without states, consisting only of a non-empty collection of transitions $\mathcal{U} \subseteq \mathbb{Z}^d$. All definitions, notations, and problems carry over for VAS except that, for simplicity, we drop the states across the board. For example, a configuration in a VAS is just a vector $\mathrm{v} \in \mathbb{N}^d$. A well-known result from the seventies by Hopcroft and Pansiot is that one can simulate the states of a VASS at the cost of three extra dimensions in a VAS [35].

Lemma 2.1 (See [35, Lemma 2.1]). *Let $\mathcal{V} = (Q, T)$ be a d-VASS. There exists a $(d+3)$-VAS $\mathcal{U}$ that* simulates *$\mathcal{V}$ and such that $\|\mathcal{U}\| = poly(\|\mathcal{V}\|)$. Precisely, there exist an injective function $f : Q \to \mathbb{N}^3$ such that there is a run $p(\mathrm{s}) \xrightarrow{\pi} q(\mathrm{t})$ in $\mathcal{V}$ if and only if there is a run $(f(p), \mathrm{s}) \xrightarrow{\rho} (f(q), \mathrm{t})$ in $\mathcal{U}$. Moreover, f can be computed in $poly(\|\mathcal{V}\|)$ time and* $\mathrm{len}(\rho) = 3 \cdot \mathrm{len}(\pi)$.

Roughly speaking, the VAS obtained has an equivalent reachability relation between configurations; a configuration $q(\mathrm{x})$ in the original VASS corresponds with a configuration $((a, b, c), \mathrm{x})$ in the VAS, where $a, b, c \leq 2|Q|^2$ are carefully chosen values that together represent the state q. Each transition of the d-VASS $\mathcal{V}$ is split into a triplet of vectors that get added to the $(d+3)$-VAS $\mathcal{U}$ where the combined effect of such a triplet is indeed the same as the effect of the original transition in $\mathcal{V}$. Importantly, during a run in $\mathcal{U}$, each triplet of vectors must be taken in sequence so a run of length ℓ in $\mathcal{V}$ corresponds to a run of length 3ℓ in $\mathcal{U}$ and vice versa.

3 Improved Bounds on the Maximum Counter Value

This section is devoted to our improvement of the seminal result of Rackoff. Throughout, we fix our attention to the arbitrary instance $(\mathcal{V}, p(\mathrm{s}), q(\mathrm{t}))$ of the coverability problem in a d-VASS $\mathcal{V} = (Q, T)$ from the initial configuration $p(\mathrm{s})$ to a configuration $q(\mathrm{t}')$ with at least the counter values of the target configuration $q(\mathrm{t})$. We denote $n = \|\mathcal{V}\| + \|\mathrm{s}\| + \|\mathrm{t}\|$. The following two theorems follow from Rackoff's technique and subsequent work by Rosier and Yen, in particular see [53, Lemma 3.4 and Theorem 3.5] and [54, Theorem 2.1 and Lemma 2.2].

Theorem 3.1 (Corollary of [53, Lemma 3.4] and [54, Theorem 2.1]). *Suppose $p(\mathrm{s}) \xrightarrow{*} q(\mathrm{t}')$ for some $\mathrm{t}' \geq \mathrm{t}$. Then there exists a run π such that $p(\mathrm{s}) \xrightarrow{\pi} q(\mathrm{t}'')$ for some $\mathrm{t}'' \geq \mathrm{t}$ and* $\mathrm{len}(\pi) \leq n^{2^{\mathcal{O}(d \log(d))}}$.

Theorem 3.2 (cf. [53, Theorem 3.5]). *For a given d-VASS $\mathcal{V}$, integer ℓ, and two configurations $p(\mathrm{s})$ and $q(\mathrm{t})$, there is an algorithm that determines the existence of a run π of length* $\mathrm{len}(\pi) \leq \ell$ *that witnesses coverability, so $p(\mathrm{s}) \xrightarrow{\pi} q(\mathrm{t}')$ for some $\mathrm{t}' \geq \mathrm{t}$. The algorithm can be implemented to run in non-deterministic $\mathcal{O}(d \log(n \cdot \ell))$ space or deterministic $2^{\mathcal{O}(d \log(n \cdot \ell))}$ time.*

Proof. In runs whose length is bounded by ℓ, the observed counter values are trivially bounded by $n \cdot \ell$. Notice that every configuration can be written in $\mathcal{O}(d \log(n \cdot \ell))$ space. A non-deterministic algorithm can therefore decide coverability by guessing a run on-the-fly by maintaining the current configuration and the length of the run (using a step counter). The algorithm accepts if and only if t is covered by the final configuration.

The second part follows from the standard construction that if a problem can be solved in $S(n)$ non-deterministic space then it can be solved in $2^{\mathcal{O}(S(n))}$ deterministic time. Indeed, one can construct the graph of all configurations and check whether there is a path from the initial configuration to the final configuration. Since there are at most $2^{\mathcal{O}(d \log(n \cdot \ell))}$ many configurations, this can be completed in $2^{\mathcal{O}(d \log(n \cdot \ell))}$ time. □

Note that Theorem 3.1 combined with Theorem 3.2 yield non-deterministic $2^{\mathcal{O}(d\log(d))}\cdot\log(n)$-space and deterministic $n^{2^{\mathcal{O}(d\log(d))}}$-time algorithms for coverability. Our result improves this by an $\mathcal{O}(\log(d))$ factor in the second exponent.

Theorem 3.3. *Suppose* $p(\mathrm{s})\xrightarrow{*}q(\mathrm{t}')$ *for some* $\mathrm{t}'\geq\mathrm{t}$. *Then there exists a run* π *such that* $p(\mathrm{s})\xrightarrow{\pi}q(\mathrm{t}'')$ *for some* $\mathrm{t}''\geq\mathrm{t}$ *and* $\mathrm{len}(\pi)\leq n^{\mathcal{O}(d\cdot 2^d)}$.

Combining Theorem 3.2 with Theorem 3.3 yields the following corollary.

Corollary 3.4. *Coverability in d-VASS can be decided by both a non-deterministic* $\mathcal{O}(d^2\cdot 2^d\cdot\log(n))$-*space algorithm and a deterministic* $n^{\mathcal{O}(d^2\cdot 2^d)}$-*time algorithm.*

Proof. Let $\ell=n^{c\cdot d\cdot 2^d}$ for some constant c such that Theorem 3.3 tells us that if $p(\mathrm{s})\xrightarrow{*}q(\mathrm{t}')$ for some $\mathrm{t}'\geq\mathrm{t}$, then there exists a run π such that $p(\mathrm{s})\xrightarrow{\pi}q(\mathrm{t}'')$ for some $\mathrm{t}''\geq\mathrm{t}$ and $\mathrm{len}(\pi)\leq\ell$. By Theorem 3.2, we know that there exists a non-deterministic algorithm that decides coverability between $p(\mathrm{s})$ and $q(\mathrm{t})$ in the following space.

$$\begin{aligned}\mathcal{O}(d\log(n\cdot\ell))=\mathcal{O}(d\log(n)+d\log(\ell))&=\mathcal{O}\left(d\log(n)+d\log\left(n^{c\cdot d\cdot 2^d}\right)\right)\\&=\mathcal{O}\left(d\log(n)+d\log(n)\cdot c\cdot d\cdot 2^d\right)\\&=\mathcal{O}\left(d^2\cdot 2^d\cdot\log(n)\right).\end{aligned}$$

Also by Theorem 3.2, we know that there exists a deterministic algorithm that decides coverability between $p(\mathrm{s})$ and $q(\mathrm{t})$ in $2^{\mathcal{O}(d\log(n\cdot\ell))}=2^{\mathcal{O}(d^2\cdot 2^d\cdot\log(n))}=n^{\mathcal{O}(d^2\cdot 2^d)}$ time. □

The rest of this section is dedicated to the proof of Theorem 3.3. Theorem 3.3 is a corollary of Lemma 2.1 and the following lemma (Lemma 3.5); a formal proof can be found at the end of this section. By Lemma 2.1, instead of handling a given VASS, we may instead handle an equivalent VAS with three additional counters whose size is polynomial in n. Recall that, as there are no states, a d-VAS consists only of a set of vectors in $\mathbb{Z}^d$ which we still refer to as transitions. A configuration is just a vector in $\mathbb{N}^d$. Accordingly, we may fix our attention on the instance $(\mathcal{V},\mathrm{s},\mathrm{t})$ of the coverability problem in a d-VAS $\mathcal{V}=\{\mathrm{v}_1,\ldots,\mathrm{v}_m\}$ from the initial configuration s to a configuration t', that is, at least as great as the target configuration t. Although Theorem 3.3 is a stronger statement than Theorem 3.1, we imitate the structure of Rackoff's proof; we proceed by induction on the dimension d.

Lemma 3.5. *Define* $L_i:=n^{i\cdot 2^i}$, *and let* $\mathrm{t}\in\mathbb{N}^d$ *such that* $\|\mathrm{t}\|\leq n$. *For any* $\mathrm{s}\in\mathbb{N}^d$, *if* $\mathrm{s}\xrightarrow{*}\mathrm{t}'$ *for some* $\mathrm{t}'\geq\mathrm{t}$ *then there exists a run* π *such that* $\mathrm{s}\xrightarrow{\pi}\mathrm{t}''$ *for some* $\mathrm{t}''\geq\mathrm{t}$ *and* $\mathrm{len}(\pi)\leq L_d$.

The base case is $d=0$. In a 0-dimensional VAS, the only possible configuration is the empty vector $\boldsymbol{\varepsilon}$ and therefore there is only the trivial run $\boldsymbol{\varepsilon}\xrightarrow{*}\boldsymbol{\varepsilon}$. This trivially satisfies the lemma.

For the inductive step, when $d\geq 1$, we assume that Lemma 3.5 holds for all lower dimensions $0,\ldots,d-1$. Let $\pi=(\mathrm{c}_0,\mathrm{c}_1,\ldots,\mathrm{c}_\ell)$ be a run with minimal length such that $\mathrm{s}\xrightarrow{\pi}\mathrm{t}'$ for some $\mathrm{t}'\geq\mathrm{t}$, so in particular, $\mathrm{c}_0=\mathrm{s}$ and $\mathrm{c}_\ell=\mathrm{t}'$. Our objective is to prove that $\mathrm{len}(\pi)=\ell+1\leq L_d$. Observe that configurations c_i need to be distinct, else π could be shortened trivially. We introduce the notion of a *thin configuration.*

Definition 3.6 (Thin Configuration). In a d-VAS, we say that a configuration $\mathrm{c}\in\mathbb{N}^d$ is *thin* if there exists a permutation σ of $\{1,\ldots,d\}$ such that $\mathrm{c}[\sigma(i)]<M_i$ for every $i\in\{1,\ldots,d\}$, where $M_0:=n$ and for $i\geq 1$, $M_i:=L_{i-1}\cdot n$.

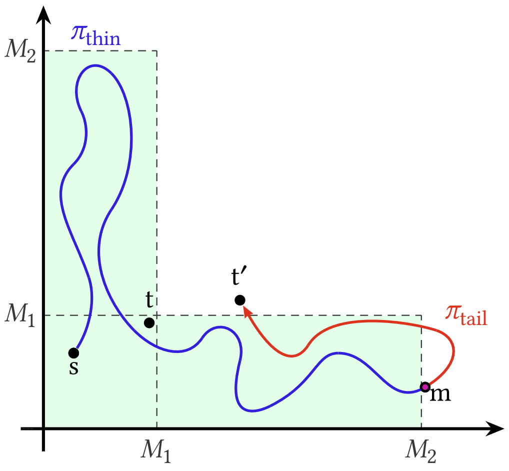


Fig. 1. The schematic view of Claims 3.7 and 3.8, restricted to the two-dimensional case. Here, s is the initial configuration and t is the target configuration. The shortest run witnessing coverability (to $t' \geq t$) is drawn. Every configuration inside the green shaded polygon is thin, where each rectangular component of the green shaded polygon corresponds to a permutation of the indices. Observe that m is the first configuration, just outside the green shaded polygon, that is, not thin. The prefix of the run from s to m is π_{thin} (drawn in blue); Claim 3.7 bounds the maximum possible length of π_{thin} by the volume of the green polygon. The suffix of the run from m to t′ is π_{tail} (drawn in red); Claim 3.8 bounds the maximum possible length of π_{tail} by L_{d-1}.

Recall, from above, the run $\pi = (c_0, c_1, \dots, c_\ell)$. Let $t \in \{0, \dots, \ell\}$ be the first index where c_t is not thin, otherwise let $t = \ell + 1$ if every configuration in π is thin. We decompose the run about the tth configuration $\pi = \pi_{\text{thin}} \cdot \pi_{\text{tail}}$, where $\pi_{\text{thin}} := (c_0, \dots, c_{t-1})$ and $\pi_{\text{tail}} := (c_t, \dots, c_\ell)$. Note that π_{thin} or π_{tail} can be empty. Subsequently, we individually analyse the lengths of π_{thin} and π_{tail} (see Figure 1). We also denote $m = c_t$ to be the first configuration, that is, not thin.

Claim 3.7. $\operatorname{len}(\pi_{\text{thin}}) \leq d! \cdot n^d \cdot L_{d-1} \cdot \ldots \cdot L_0$.

Proof. By definition, every configuration in π_{thin} is thin. Moreover, since π has a minimal length, no configurations in π repeat; the same is therefore true for the prefix π_{thin}. We now count the number of possible thin configurations. There are $d!$ many permutations of $\{1, \dots, d\}$. For a given permutation σ and an index $i \in \{1, \dots, d\}$, we know that for a thin configuration c, $0 \leq c[\sigma(i)] < M_i$, so there are at most $M_i = L_{i-1} \cdot n$ many possible values on the $\sigma(i)$th counter. Hence the total number of thin configurations is at most $d! \cdot \prod_{i=1}^{d}(L_{i-1} \cdot n) = d! \cdot n^d \cdot L_{d-1} \cdot \ldots \cdot L_0$. □

Claim 3.8. $\operatorname{len}(\pi_{\text{tail}}) \leq L_{d-1}$.

Proof. Consider $m \in \mathbb{N}^d$, the first configuration of π_{tail}. Let σ be a permutation such that $m[\sigma(1)] \leq m[\sigma(2)] \leq \ldots \leq m[\sigma(d)]$. Given that m is not thin, for every permutation σ' there exists an $i \in \{1, \dots, d\}$ such that $m[\sigma'(i)] \geq M_i$; in particular, this holds for σ. Note that this also implies $M_i \leq m[\sigma(i+1)] \leq \ldots \leq m[\sigma(d)]$.

We construct an $(i-1)$-VAS $\mathcal{U}$ from $\mathcal{V}$ by ignoring the counters $\sigma(i), \dots, \sigma(d)$. Formally, $u \in \mathcal{U}$ if there is $v \in \mathcal{V}$ such that $u[j] = v[\sigma(j)]$ for each $1 \leq j \leq i-1$. In such a case we say u is the *projection* of v via σ. We will use the inductive hypothesis to show that there is a short path ρ' in $\mathcal{U}$ from (the projection of) m covering (the projection of) t. We will then show that the remaining components of m are large enough that the embedding of ρ' into $\mathcal{V}$ maintains its covering status.

Recall that t' is the final configuration of the run π. Note that the run π_{tail} induces a run π'_{tail} in $\mathcal{U}$ by permuting and projecting every configuration. More precisely, $(\mathrm{m}[\sigma(1)], \ldots, \mathrm{m}[\sigma(i-1)]) \xrightarrow{\pi'_{\mathrm{tail}}} (\mathrm{t}'[\sigma(1)], \ldots, \mathrm{t}'[\sigma(i-1)])$. By the inductive hypothesis there exists a run ρ' in $\mathcal{U}$ such that $(\mathrm{m}[\sigma(1)], \ldots, \mathrm{m}[\sigma(i-1)]) \xrightarrow{\rho'} (\mathrm{t}''[\sigma(1)], \ldots, \mathrm{t}''[\sigma(i-1)])$, such that $(\mathrm{t}''[\sigma(1)], \ldots, \mathrm{t}''[\sigma(i-1)]) \geq (\mathrm{t}[\sigma(1)], \ldots, \mathrm{t}[\sigma(i-1)])$ and $\mathrm{len}(\rho') \leq L_{i-1}$.

Let $(\mathrm{u}_1, \ldots, \mathrm{u}_{\mathrm{len}(\rho')})$ be the underlying path of the run ρ', that is, the sequence of transitions in $\mathcal{U}$ that are sequentially added to form the run ρ'. By construction, each transition vector $\mathrm{u}_i \in \mathcal{U}$ has a corresponding transition vector $\mathrm{v}_i \in \mathcal{V}$ where u_i is the projection of v_i via σ. We will now show that the following run witnesses coverability of t.

$$\rho = \left(\mathrm{m},\ \mathrm{m} + \mathrm{v}_1,\ \mathrm{m} + \mathrm{v}_1 + \mathrm{v}_2,\ \ldots,\ \mathrm{m} + \sum_{j=1}^{\mathrm{len}(\rho')} \mathrm{v}_j\right).$$

To this end, we verify that (i) ρ is a run, that is, all configurations lie in $\mathbb{N}^d$, and (ii) the final configuration indeed covers t. For components $\sigma(1)$, ..., $\sigma(i-1)$, this follows directly from the inductive hypothesis. For all other components we will show that *all* configurations of ρ are covering t in these components. This satisfies both (i) and (ii).

Let j be any of the remaining components. Recall that by the choice of m, $\mathrm{m}[j] \geq M_i = n \cdot L_{i-1}$. Since $n > \|\mathcal{V}\| \geq \|\mathrm{v}_j\|$ for every $1 \leq j \leq \mathrm{len}(\rho')$, this means that in a single step, the value of a counter can change by at most $(n-1)$. Given that $\mathrm{len}(\rho) = \mathrm{len}(\rho') \leq L_{i-1}$, the value on each of the remaining components must be at least n for every configuration in ρ. In particular, observing that $\|\mathrm{t}\| \leq n$, the final configuration of ρ satisfies

$$\mathrm{m} + \sum_{j=1}^{\mathrm{len}(\rho')} \mathrm{v}_j \geq \mathrm{t}.$$

Finally, observe that $\mathrm{len}(\rho) = \mathrm{len}(\rho') \leq L_{i-1} \leq L_{d-1}$. □

Now, we can prove that Lemma 3.5 follows from Claims 3.7 and 3.8.

Proof of Lemma 3.5. From Claims 3.7 and 3.8,

$$\begin{aligned}\mathrm{len}(\pi) \leq \mathrm{len}(\pi_{\mathrm{thin}}) + \mathrm{len}(\pi_{\mathrm{tail}}) &\leq d! \cdot n^d \cdot L_{d-1} \cdot \ldots \cdot L_0 + L_{d-1} \\ &\leq 2 \cdot d! \cdot n^d \cdot L_{d-1} \cdot \ldots \cdot L_0.\end{aligned}$$

Recall that $n \geq 2$ and $d! \leq n^d$; observe that $2 \cdot d! \cdot n^d \leq n^{d^2+1}$. Hence,

$$\mathrm{len}(\pi) \leq n^{d^2+1} \cdot L_{d-1} \cdot \ldots \cdot L_0.$$

Next, we use the definition of $L_i := n^{i \cdot 2^i}$ to show

$$\mathrm{len}(\pi) \leq n^{d^2+1} \cdot \prod_{i=0}^{d-1} n^{i \cdot 2^i} \leq n^{\left(d^2+1+\sum_{i=0}^{d-1} i \cdot 2^i\right)}.$$

Next, we use the fact that $\sum_{i=0}^{d-1} i \cdot 2^i = d \cdot 2^d - 2^{d+1} + 2$, so when $d \geq 1$, $d^2 + 1 + \sum_{i=0}^{d-1} i \cdot 2^i \leq d \cdot 2^d$. Therefore,

$$\mathrm{len}(\pi) \leq n^{d \cdot 2^d} = L_d.$$ □

To conclude this section, we now transfer the upper bound to VASS to prove Theorem 3.3.

Proof of Theorem 3.3. Let $(\mathcal{V}, p(\mathrm{s}), q(\mathrm{t}))$ be an instance of d-VASS coverability of size n. By Lemma 2.1, we can construct a $(d+3)$-dimensional VAS $\mathcal{U}$ of size $\mathrm{poly}(\|\mathcal{V}\|)$ and vectors $\mathrm{p}, \mathrm{q} \in \mathbb{N}^3$ such that $\|\mathrm{p}\|, \|\mathrm{q}\| \leq \mathrm{poly}(\|\mathcal{V}\|)$ and, for some $\mathrm{t}' \geq \mathrm{t}$, there is a run from $p(\mathrm{s})$ to $q(\mathrm{t}')$ in $\mathcal{V}$ if and only if there is a run from (p, s) to $(\mathrm{q}, \mathrm{t}')$ in $\mathcal{U}$.

Suppose $(\mathrm{p}, \mathrm{s}) \xrightarrow{\rho} (\mathrm{q}, \mathrm{t}')$ is the minimal length run in $\mathcal{U}$ such that $\mathrm{t}' \geq \mathrm{t}$. By Lemma 3.5, we know that $\mathrm{len}(\rho) \leq \mathrm{poly}(n)^{(d+3)\cdot 2^{d+3}} = n^{\mathcal{O}(1)\cdot(d+3)\cdot 8\cdot 2^d} = n^{\mathcal{O}(d\cdot 2^d)}$. As stated at the end of Lemma 2.1, there exists a corresponding run $p(\mathrm{s}) \xrightarrow{\pi} q(\mathrm{t}')$ in $\mathcal{V}$ where $\mathrm{t}' \geq \mathrm{t}$ and $\mathrm{len}(\pi) = \frac{\mathrm{len}(\rho)}{3}$. We therefore obtain a run from $p(\mathrm{s})$ that covers $q(\mathrm{t})$ in $\mathcal{V}$ of length $\ell = \frac{\mathrm{len}(\rho)}{3} \leq n^{\mathcal{O}(d\cdot 2^d)}$. □

4 Conditional Time Lower Bound for Coverability

In this section, we present a conditional lower bound based on the ETH [36]. Roughly speaking, ETH is a conjecture that a λ-variable instance of 3-SAT cannot be solved by a deterministic $2^{o(\lambda)}$-time algorithm (for a modern survey, see [45]). In our reductions, it will be convenient for us to reduce via the k-clique problem instead of reducing directly from 3-SAT. In the k-clique problem, the input is a graph $G = (V, E)$, k is a parameter, and the task is to decide whether there is a set of k pairwise adjacent vertices in V. For a graph with r nodes, the naive algorithm for k-clique runs in $\mathcal{O}(r^k)$ time. Even though the exact constant in the dependence on k can be improved [52], ETH implies that the exponent must have a linear dependence on k.

Theorem 4.1 ([14, Theorem 4.2], [15, Theorem 4.5], and [18, Theorem 14.21]). *Assuming the ETH, there is no algorithm running in $f(k) \cdot r^{o(k)}$ time for the k-clique problem for any computable function f. Moreover, one can assume that G is k-partite, i.e. $G = (V_1 \cup \ldots \cup V_k, E)$ and $E \subseteq \{\{u, v\} : u \in V_i, v \in V_j, i \neq j\}$.*

We will use Theorem 4.1 to show the following conditional lower bound for coverability in unary d-VASS, which is proved at the end of this section.

Theorem 4.2. *Assuming the ETH, there does not exist an $n^{2^{o(d)}}$-time algorithm deciding coverability in a d-VASS of size n.*

We remark, as corollary of Theorem 4.2, that there does not exist an $n^{2^{o(d)}}$-time algorithm for coverability in d-VAS under ETH. [1] Here, to be precise, $n = \sum_{\mathrm{x}\in\mathcal{V}} \|\mathrm{x}\|$ refers to the size of a d-VAS $\mathcal{V}$ encoded in unary. This follows from the fact that VAS can simulate VASS [35, Lemma 2.1]; see the text surrounding Lemma 2.1 for a brief discussion about this simulation.

To prove Theorem 4.2, we first reduce the k-clique problem to coverability in bounded 2-VASS with the ability to perform a fixed number of zero tests. We will then leverage a result by Rosier and Yen to construct an equivalent, with respect to coverability, $(\mathcal{O}(\log(k)))$-VASS without zero tests.

Lemma 4.3. *Given a k-partite graph $G = (V_1 \cup \cdots \cup V_k, E)$ with r vertices, there exists a unary $(\mathcal{O}(r^{2k}), 2)$-VASS $\mathcal{T}$ with $\mathcal{O}(k^2)$ zero tests and two designated states q_I, q_F such that there is a k-clique in G if and only if there exists a run from $q_I(0)$ to $q_F(\mathrm{t})$ in $\mathcal{T}$, for some $\mathrm{t} \geq 0$. Moreover, $\|\mathcal{T}\| \leq poly(r+k)$ and $\mathcal{T}$ can be constructed in poly$(r+k)$ time.*

Proof. Without loss of generality, we may assume that each of the k vertex subsets in the graph has the same size $|V_1| = \ldots = |V_k| = \ell$. Thus $r = k \cdot \ell$. For convenience, we denote $V = \{v_{i,j} : i \in \{1, \ldots, k\}, j \in \{1, \ldots, \ell\}\}$.

[1]Recall, from Section 2, that a d-VAS is a d-dimensional vector addition system (without states).

ALGORITHM 1: A counter program for a VASS with zero tests with two counters x and y.

```
input: $x = 0, y = 0$
$x \mathrel{+}= 1$
for $i \leftarrow 1$ to $k$ do
    guess $j \in \{1, \dots, \ell\}$
    Multiply$[x, p_{i,j}]$
end
for $(i, j) \in \{1, \dots, k\}^2, i < j$ do
    guess $e \in \{\{u, v\} \in E : u \in V_i, v \in V_j\}$
    Edge$[e]$
end
```

We begin by sketching the main ideas behind the reduction before they are implemented. We start by finding the first $r = k \cdot \ell$ primes and associating a distinct prime $p_{i,j}$ to each vertex $v_{i,j} \in V$. Note that a product of k different primes uniquely corresponds to a selection of k vertices. Thus the idea is to guess such a product and then test whether the corresponding vertices form a k-clique.

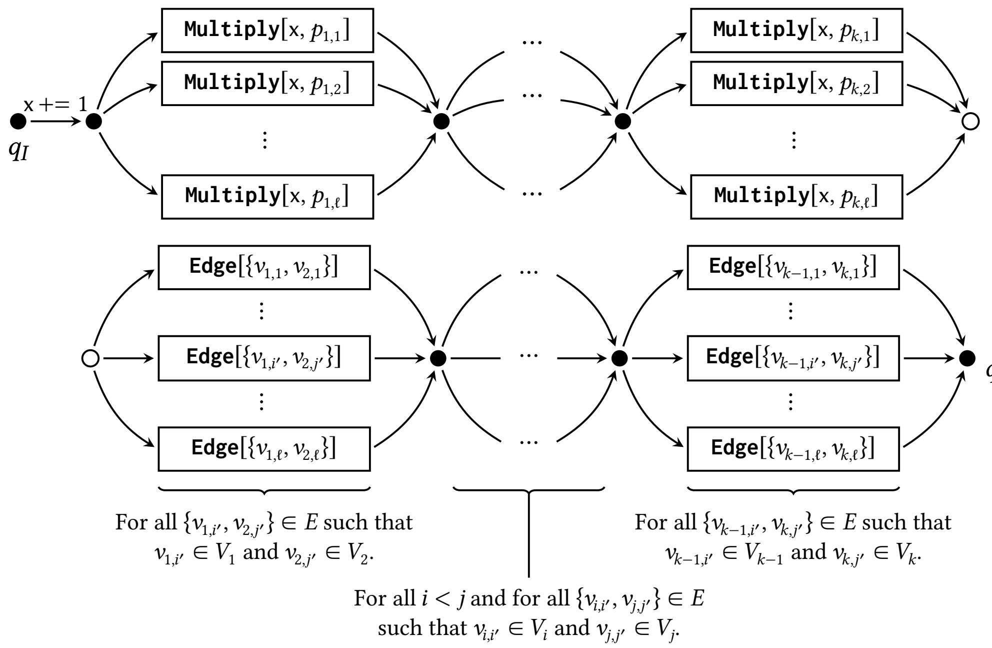
We present an overview of our construction in Algorithm 1. To simplify the presentation, we present VASS also as counter programs, inspired by Esparza's presentation of Lipton's lower bound [26, Section 7]. The program is non-deterministic and contains zero tests. The zero tests are not immediately obvious in Algorithm 1 because they are used in the subprocedures (including **Multiply**). Note that counter y is used only by subprocedures. Also note that the variable i in the first loop and variables i and j in the second loop are just syntactic sugar for copying similar code multiple times. The variable j in the first loop and the variable e in the second loop allow us to neatly represent non-determinism in the VASS corresponding to the program.

At the start of the program presented in Algorithm 1, both counters x and y are initialised to 0 (as specified in the statement of this lemma) and we are interested in the existence of a coverability run that simply reaches the final control state (with any counter values). One should think that coverability holds if and only if there is a run through the program that does not execute an instruction that would take a counter below zero (colloquially, without "getting stuck"). It turns out that a run could only get stuck in the **Edge**$[e]$ subprocedure, which will be explained later.

Algorithm 1 uses the **Multiply**$[x, p]$ and **Edge**$[e]$ subprocedures. These two subprocedures will be implemented later. The subprocedure **Multiply**$[x, p]$ takes a counter x as input as we later reuse this subprocedure when there is more than one counter subject to multiplication. The intended behaviour of **Multiply**$[x, p]$ is that it can be performed if and only if as a result we get $\mathrm{x} = x \cdot p$ from an initial value of $\mathrm{x} = x$ (despite the fact that VASS can only additively increase and decrease counters). The subprocedure **Edge**$[e]$ can be performed if and only if both vertices of the edge e are encoded in the value of counter x. Overall, Algorithm 1 is designed so that in the first part x is multiplied by $p_{i,j}$, where for every i, one j is guessed. This equates to selecting one vertex from each V_i. Then the second part the algorithm checks whether between every pair of selected vertices from V_i and V_j there is an edge. Clearly, there is a run through the program that does not get stuck if and only if there is k-clique in G.

In Figure 2 we present a VASS with zero tests implementing Algorithm 1. The construction will guarantee that $q_F(0)$ can be covered from $q_I(0)$ if and only if there is a k-clique in G.

It remains to define the subprocedures. One should think that every call of a subprocedure corresponds to a unique part of the VASS, like a gadget of sorts. To enter and leave the subprocedure one needs to add trivial transitions that do not change the counter values. All subprocedures rely on the invariant $\mathrm{y} = 0$ at the beginning and admit the invariant at the end.



x += 1
q_I
Multiply[x, $p_{1,1}$]
Multiply[x, $p_{1,2}$]
Multiply[x, $p_{1,\ell}$]
Multiply[x, $p_{k,1}$]
Multiply[x, $p_{k,2}$]
Multiply[x, $p_{k,\ell}$]
Edge[$\{v_{1,1}, v_{2,1}\}$]
Edge[$\{v_{1,i'}, v_{2,j'}\}$]
Edge[$\{v_{1,\ell}, v_{2,\ell}\}$]
Edge[$\{v_{k-1,1}, v_{k,1}\}$]
Edge[$\{v_{k-1,i'}, v_{k,j'}\}$]
Edge[$\{v_{k-1,\ell}, v_{k,\ell}\}$]
q_F
For all $\{v_{1,i'}, v_{2,j'}\} \in E$ such that $v_{1,i'} \in V_1$ and $v_{2,j'} \in V_2$.
For all $\{v_{k-1,i'}, v_{k,j'}\} \in E$ such that $v_{k-1,i'} \in V_{k-1}$ and $v_{k,j'} \in V_k$.
For all $i < j$ and for all $\{v_{i,i'}, v_{j,j'}\} \in E$ such that $v_{i,i'} \in V_i$ and $v_{j,j'} \in V_j$.

Fig. 2. The top part of the VASS implements the first line and the first loop in Algorithm 1; counter x is multiplied by k non-deterministically chosen primes $p_{i,j}$, each corresponding to a vertex in V_i. The bottom part of the VASS implements the second loop in Algorithm 1; for every pair $i \neq j$ the VASS non-deterministically chooses $\{v_{i,i'}, v_{j,j'}\} \in E$ such that $v_{i,i'} \in V_i$ and $v_{j,j'} \in V_j$ and invokes the subprocedure **Edge**[e].

We start with **Multiply**[x, p] and **Divide**[x, p], which indeed multiply and divide x by p, respectively. See Algorithm 2 for the counter program and VASS implementations. Note that **loop** statements correspond to self-loop transitions in VASS. These loops can be taken any non-negative number of times so long as the counters remain non-negative; however, the subsequent zero tests implicity enforce these loops to be iterated exhaustively. In the **Multiply**[x, p] gadget, it is easy to see that a run passes through the procedure if and only if x is multiplied by p. Similarly, in the **Divide**[x, p] gadget, it is easy to see that a run passes through the procedure if and only if x is divided by p wholly. Indeed, the division procedure would get stuck if $p \nmid$ x because it will be impossible to exit the first loop.

The procedure **Edge**[$\{v_{i,i'}, v_{j,j'}\}$] is simply a sequence of four subprocedures, as shown in Algorithm 3. Indeed, to check if the vertices $v_{i,i'}$ and $v_{j,j'}$ are encoded in x we simply check whether x is divisible by the corresponding primes $p_{i,i'}$ and $p_{j,j'}$.

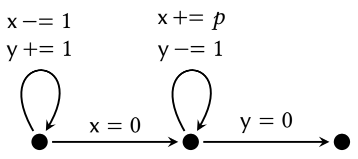
Afterwards we multiply x with the same primes so that the value does not change and it is ready for future edge checks.

It remains to analyse the size of the VASS and its construction time in this reduction. In every run from $q_I(0)$ to $q_F(\mathrm{t})$, for some t ≥ 0, the greatest counter value observable can be bounded above by p^k where p is the rth prime. By the Prime Number Theorem (for example, see [61]), we know that $p^k = \mathcal{O}((r\log(r))^k) = \mathcal{O}(r^{2k})$ is an upper bound on the counter values observed. Hence $\mathcal{T}$ is an $\mathcal{O}(r^{2k})$-bounded unary 2-VASS.

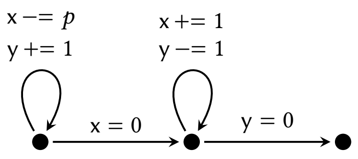
Now, we count the number of zero tests performed in each run from $q_I(0)$ to $q_F(\mathrm{t})$, for some t ≥ 0. The only zero tests occur in the instances of the **Multiply** and **Divide** subprocedures, each performing two zero tests. In the first part of $\mathcal{T}$, a run will encounter k many **Multiply**

ALGORITHM 2: The counter program of **Multiply**[x, p] above its VASS implementation (left) and the counter program of **Divide**[x, p] above its VASS implementation (right).

```
input  : $x = x, y = 0$
output : $x = x \cdot p, y = 0$
loop $x -= 1,\ y += 1$
zero-test[x]
loop $x += p,\ y -= 1$
zero-test[y]
```

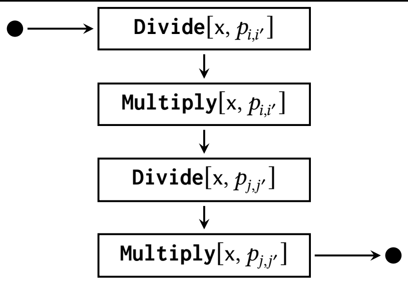


```
input  : $x = x \cdot p, y = 0$
output : $x = x, y = 0$
loop $x -= p,\ y += 1$
zero-test[x]
loop $x += 1,\ y -= 1$
zero-test[y]
```



ALGORITHM 3: The counter program for **Edge**[$\{v_{i,i'} v_{j,j'}\}$] and its VASS implementation.

```
input  : $x = x, y = 0$
output : $x = x, y = 0$
Divide[x, $p_{i,i'}$]
Multiply[x, $p_{i,i'}$]
Divide[x, $p_{j,j'}$]
Multiply[x, $p_{j,j'}$]
```



subprocedures, contributing $2k$ many zero tests. In the second part of $\mathcal{T}$, a run will encounter $\binom{k}{2}$ many **Edge** subprocedures, each containing two **Multiply** subprocedures and two **Divide** subprocedures, in total contributing $8\binom{k}{2}$ many zero tests. Together, every run encounters exactly $2k+8\binom{k}{2} = 2k(2k-1)$ many zero tests. Hence $\mathcal{T}$ is an $\mathcal{O}(r^{2k})$-bounded unary 2-VASS with $2k(2k-1)$ zero tests.

Finally, the **Multiply** and **Divide** subprocedures contain three states and five transitions. Since the rth prime is bounded above by $\mathcal{O}(r\log(r))$, we can unfold the updates in these procedures into an $\mathcal{O}(r\log(r))$-length sequence of unary (-1, 0, or $+1$) updates. Analysing Algorithm 1, it is easy to see that overall the number of states is polynomial in r. Finally, the first r primes can be found in $r^{1+o(1)}$ time [2]. Therefore, in total $\mathcal{T}$ has size $\|\mathcal{T}\| = \text{poly}(r+k)$ and can be constructed in poly$(r+k)$ time. ☐

To attain conditional lower bounds for coverability we must replace the zero tests. We make use of a technique of Rosier and Yen [54] that relies on the construction of Lipton [44]. They show that a 2^{2^d}-bounded counter machine with finite state control can be simulated by a unary $\mathcal{O}(d)$-VASS. As Rosier and Yen detail after their proof, it is possible to apply this technique to multiple counters with zero tests at once [54, Page 127]. This accordingly results in the number of VASS counters increasing, but we instantiate this with just two counters. We remark that the VASS constructed in

Lemma 4.3 is structurally bounded, so for any initial configuration there is a limit on the largest observable counter value, as is the case in the VASS that Lipton constructed [44].

Lemma 4.4 (Corollary of [54, Lemma 4.3]). *For parameters a and d, let $\mathcal{T}$ be an m-state unary $(a^{2^d}, 2)$-VASS with zero tests and two designated states q_I, q_F. Then there exists a poly$(a \cdot m)$-state $(4d)$-VASS $\mathcal{V}$ with two designated states q_I', q_F' such that there is a run from $q_I(\mathbf{0})$ to $q_F(\mathrm{v})$, for some $\mathrm{v} \geq \mathbf{0}$, in $\mathcal{T}$ if and only if there is a run from $q_I'(\mathbf{0})$ to $q_F'(\mathrm{w})$, for some $\mathrm{w} \geq \mathbf{0}$, in $\mathcal{V}$. Moreover, $\mathcal{V}$ has size poly$(a \cdot \|\mathcal{T}\|)$ and can be constructed in the same time.*

Proof. This essentially follows from Rosier and Yen's original construction [54], hence we briefly describe how to ensure that the dimension of the resulting VASS is not too large. Here, we will follow Esparza's detailed analysis [26]. In particular on [26, Page 411], all counters are listed with their properties (this also implies the dimension of the output VASS). Altogether, their construction consists of the following counters with the following properties:

- $\mathrm{x}, \mathrm{y}, \mathrm{s}$ and $\bar{\mathrm{x}}, \bar{\mathrm{y}}, \bar{\mathrm{s}}$; after initialisation, the invariants $\mathrm{x} + \bar{\mathrm{x}} = 2^{2^d}$, $\mathrm{y} + \bar{\mathrm{y}} = 2^{2^d}$, and $\mathrm{s} + \bar{\mathrm{s}} = 2^{2^d}$ are satisfied and the values of $\mathrm{x}, \mathrm{y}, \mathrm{s}$ are bounded by 2^{2^d}, and
- $\mathrm{y}_i, \mathrm{z}_i$ and $\bar{\mathrm{y}}_i, \bar{\mathrm{z}}_i$ for each $0 \leq i \leq d-1$; after initialisation, for all $0 \leq i \leq d-1$, the invariants $\mathrm{y}_i + \bar{\mathrm{y}}_i = 2^{2^i}$ and $\mathrm{z} + \bar{\mathrm{z}}_i = 2^{2^i}$ are satisfied and the values of $\mathrm{y}_i, \mathrm{z}_i$ are bounded by 2^{2^i}.

There are two nuances with the construction. First, the counters x, y are bounded by 2^{2^d} and the counters $\mathrm{y}_i, \mathrm{z}_i$ are bounded by 2^{2^i}; we require the bound a^{2^d}. This is straightforward to fix. Initially, y_0 and z_0 are set to 0 and $\bar{y}_0, \bar{z}_0$ are set to 2. Hence in our construction, it suffices to change the aforementioned 2 to a, which using unary updates requires a additional states and transitions. The sequence of initialisations guarantees that $\mathrm{y}_i + \bar{\mathrm{y}}_i$ and $\mathrm{z}_i + \bar{\mathrm{z}}_i$ are bounded by $(\mathrm{y}_{i-1} + \bar{\mathrm{y}}_{i-1})^2 = (\mathrm{z}_{i-1} + \bar{\mathrm{z}}_{i-1})^2 = (a^{2^{i-1}})^2 = a^{2^i}$. Then, by construction, we will also get $\mathrm{x} + \bar{\mathrm{x}} = \mathrm{y} + \bar{\mathrm{y}} = \mathrm{s} + \bar{\mathrm{s}} = a^{2^d}$, this also provides an a^{2^d} bound on x and y.

The second issue, is that the number of counters is $4d+6$; we require $4d$. However, by construction the counters $\mathrm{y}_i, \mathrm{z}_i$ and $\bar{\mathrm{y}}_i, \bar{\mathrm{z}}_i$ are always bounded by a^{2^i}. Thus, we can do away with the six counters $\mathrm{y}_0, \mathrm{z}_0, \bar{\mathrm{y}}_0, \bar{\mathrm{z}}_0, \mathrm{y}_1$, and z_1 by encoding their value in the states. This only multiplies the number of states, and therefore the size of the VASS, by a polynomial in a. □

With this, we can conclude this section with the proof of its main theorem.

Proof of Theorem 4.2. Let $k = \frac{1}{2} \cdot 2^{d/4}$ and let G be an arbitrary k-partite graph with r nodes. By Lemma 4.3, there exists a unary $(\mathcal{O}(r^{2k}, 2))$-VASS with zero tests $\mathcal{T}$ such that G contains a k-clique if and only if there is a run from $q_I(\mathbf{0})$ to $q_F(\mathrm{t})$, for some $\mathrm{t} \geq \mathbf{0}$, in $\mathcal{T}$.

Given the bound on the counters in $\mathcal{T}$ is $\mathcal{O}(r^{2k}) = \mathcal{O}(r^{2^{d/4}}) = (\mathcal{O}(r))^{2^{d/4}}$, and the size of $\mathcal{T}$ is $\|\mathcal{T}\| = \text{poly}(r + k)$, we can apply Lemma 4.4 to $\mathcal{T}$. There exists a unary d-VASS $\mathcal{V}$ such that G contains a k-clique if and only if there is a run from $q_I'(\mathbf{0})$ to $q_F'(\mathrm{w})$, for some $\mathrm{w} \geq \mathbf{0}$, in $\mathcal{V}$. The size of $\mathcal{V}$ is $\|\mathcal{V}\| = \text{poly}(r + k)$, and since $k \leq r$, we have $\|\mathcal{V}\| = n \leq r^c$ for some constant c.

Assume, for the sake of contradiction, that an $n^{2^{o(d)}}$-time algorithm for coverability in unary d-VASS exists. By the above reduction, it would decide any given k-clique instance in time $n^{2^{o(d)}} = n^{o(2^{d/4})} = n^{o(k)} = r^{o(k)}$, where we used $k = O(2^{d/4})$ in the second equation and $n = \text{poly}(r)$ in the last equation. By Theorem 4.1, such an $r^{o(k)}$-time algorithm for k-clique would refute ETH. □

Remark 4.5. The reduction used to prove Theorem 4.2 actually excludes $f(d) \cdot n^{o(2^{d/4})}$-time algorithms for coverability for any computable function f, assuming ETH. Recall, from Corollary 3.4,

Table 1. Conditional Lower Bounds and Upper Bounds of the Time Complexity of Coverability and Reachability in Unary $(\mathcal{O}(n), d)$-VASS

d	Lower Bound	Upper Bound
0	$\Omega(n)$ (trivial)	$\mathcal{O}(n)$
1	$n^{2-o(1)}$ (Theorem 5.4)	$\mathcal{O}(n^2)$
2	$n^{2-o(1)}$ (from above)	$\mathcal{O}(n^3)$
3	$n^{2-o(1)}$ (from above)	$\mathcal{O}(n^4)$
$d \geq 4$	$n^{d-2-o(1)}$ (Theorem 5.8)	$\mathcal{O}(n^{d+1})$

For clarity, we remark that Theorem 5.4 is subject to Hypothesis 5.2 and that Theorem 5.8 is subject to Hypothesis 5.7. The $o(1)$ terms in the exponents of the lower bounds are sub-constant terms that may only depend on n; the lower bounds are more precisely formulated in Theorem 5.4 and Theorem 5.8. Note that the lower bounds for dimensions $d = 2$ and $d = 3$ follow from Theorem 5.4 by just adding components consisting of only zeros. All upper bounds follow from Observation 5.1.

that coverability can be decided in $n^{\mathcal{O}(d^2 \cdot 2^d)}$ time. Note that our conditional lower bound only differs from the (unconditional) upper bound by a constant multiplicative factor of 4 in the second exponent; indeed, $\mathcal{O}(d^2 \cdot 2^d) = \mathcal{O}(2^{(1+\varepsilon)\cdot d})$ for any positive constant $\varepsilon > 0$.

5 Coverability and Reachability in Bounded Unary VASS

In this section, we give even tighter bounds for coverability in *bounded* fixed dimension unary VASS. Specifically, for a non-decreasing function $B : \mathbb{N} \to \mathbb{N}$, the coverability problem in $(B(n), d)$-VASS asks, for a given $(B(n), d)$-VASS $\mathcal{V} = (Q, T)$ of size n as well as configurations $p(\mathrm{s})$, $q(\mathrm{t})$, whether there is a run in $\mathcal{V}$ from $p(\mathrm{s})$ to $q(\mathrm{t}')$ for some $\mathrm{t}' \geq \mathrm{t}$ such that each counter value remains in $\{0, \dots, B(n)\}$ throughout. We would like to clarify the fact that the bound is not part of the input to the problem. We focus on the natural setting of linearly-bounded fixed dimension VASS, that is, $(\mathcal{O}(n), d)$-VASS. There is a simple algorithm, presented in the proof of Observation 5.1, that yields an immediate $\mathcal{O}(n^{d+1})$ upper bound for the time needed to decide the coverability problem. We accompany this observation with closely matching lower bounds, see Table 1 for an overview.

Observation 5.1. *Coverability in a unary $(B(n), d)$-VASS of size n can be decided in $\mathcal{O}(n(B(n)+1)^d)$ time.*

Proof. Since all configurations in a $(B(n), d)$-VASS belong to the finite set $Q \times \{0, \dots, B(n)\}^d$, we can exhaustively explore all configurations reachable from $p(\mathrm{s})$ using a straightforward depth-first search. Each state $q \in Q$ and each transition $t \in T$ will be considered at most once for each admissible vector in $\{0, \dots, B(n)\}^d$. In total, the algorithm takes at most $\mathcal{O}(n(B(n) + 1)^d)$ time since $|Q|, |T| \leq n$. We accept the instance if and only if we have ever witnessed a configuration $q(\mathrm{t}')$ for some $\mathrm{t}' \geq \mathrm{t}$. □

Lower Bounds for Coverability in Linearly-Bounded VASS

Now, we consider lower bounds for the coverability problem in linearly-bounded fixed dimension unary VASS. Firstly, in dimension one, we show that quadratic running time is conditionally optimal under the k-cycle hypothesis. Secondly, in any fixed dimension $d \geq 4$, under the 3-uniform hyperclique hypothesis, we show that a running time of $\Omega(n^{d-2-\varepsilon})$ is required for every $\varepsilon > 0$. Together, this provides evidence that the simple $\mathcal{O}(n^{d+1})$-time algorithm for coverability in $(\mathcal{O}(n), d)$-VASS is close to optimal, as summarised in Table 1.

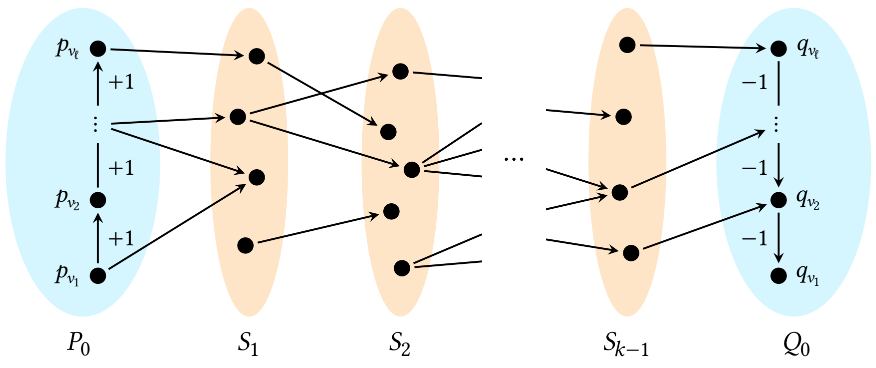


Fig. 3. The $(\mathcal{O}(n), 1)$-VASS $\mathcal{V}$ of size $n = \mathcal{O}(m)$ for detecting a k-cycle in a k-circle-layered graph with m edges. The transitions that are unlabelled have zero effect. Observe that the structure of the graph is mostly copied into the states and transitions of the linearly-bounded 1-VASS. Importantly, two copies of V_0 are created (P_0 and Q_0). Consider a run with initial configuration $p_{v_1}(0)$. First, in P_0, a vertex from V_0 belonging to a k-cycle can be selected by adding a value corresponding to that vertex to the counter. Then the configuration $q_{v_1}(0)$ can be reached if the state first observed in Q_0 corresponds to the vertex originally selected in P_0. Accordingly, there is a run from $p_{v_1}(0)$ to $q_{v_1}(0)$ if and only if there exists a k-cycle, since the states visited in the underlying path of the run correspond to the vertices of the k-cycle.

HYPOTHESIS 5.2 (k-CYCLE HYPOTHESIS). *For every $\varepsilon > 0$ and $m \in \mathbb{N}$, there exists an integer k such that there does not exist an $\mathcal{O}(m^{2-\varepsilon})$-time algorithm for detecting whether there is a k-cycle in a directed graph with m edges.*

The k-cycle hypothesis arises from the challenge of improving upon the state-of-the-art $\mathcal{O}(m^{2-\frac{c}{k}})$-time algorithms for the k-cycle problem [4, 24, 60]; here c is some constant. It has been previously used as an assumption for hardness results, for example, see [5, 23, 43]. A standard observation, due to colour-coding arguments, is that we may without loss of generality assume that the given directed graph is k-circle-layered [43, Lemma 2.2]. Specifically, we can assume that the input graph $G = (V, E)$ has vertex partition $V = V_0 \cup \cdots \cup V_{k-1}$ such that each edge $(u, v) \in E$ is in $V_i \times V_{i+1 \pmod k}$ for some $0 \le i < k$. We may also assume that $|V| \le |E|$.

LEMMA 5.3. *Given a k-circle-layered graph $G = (V_0 \cup \cdots \cup V_{k-1}, E)$ with m edges, there exists a unary $(\mathcal{O}(n), 1)$-VASS $\mathcal{V}$ such that there is a k-cycle in G if and only if there exists a run from $p(0)$ to $q(0)$ in $\mathcal{V}$. Moreover, $\mathcal{V}$ has size $n = \mathcal{O}(m)$ and can be constructed in $\mathcal{O}(m)$ time.*

PROOF. Consider the unary $(\mathcal{O}(m), 1)$-VASS $\mathcal{V} = (Q, T)$, that is, defined as follows (see also Figure 3). For ease of construction let us number the vertices in V_0, so suppose that $V_0 = \{v_1, \dots, v_\ell\}$. Let us first define the set of states Q. There are two copies of the vertex subset V_0, namely $P_0 = \{p_{v_1}, \dots, p_{v_\ell}\}$ and $Q_0 = \{q_{v_1}, \dots, q_{v_\ell}\}$. There are also copies of each of the vertex subsets $V_1, V_2, \dots, V_{k-1}$, namely $S_i = \{s_v : v \in V_i\}$ for each $1 \le i \le k-1$.

$$Q = P_0 \cup S_1 \cup S_2 \cup \cdots \cup S_{k-1} \cup Q_0.$$

Now, we define the set of transitions T. There are three kinds of transitions, the initial *vertex selection* transitions T_I, the intermediate transitions T_E, and the final *vertex checking* transitions T_F.

$$T = T_I \cup T_E \cup T_F.$$

The initial transitions connect states in P_0 sequentially. Each transition increments the counter. Intuitively speaking, the counter takes a value corresponding to the vertex in V_0 that will belong to the k-cycle in G.

$$T_I = \{(p_{v_i}, 1, p_{v_{i+1}}) \,:\, 1 \le i < \ell\}.$$

The intermediate transitions are directed copies of the edges in the original graph. The only difference is that edges between V_0 and V_1 are now transitions from P_0 to S_1 and edges between V_{k-1} and V_0 become transitions from S_{k-1} to Q_0.

$$\begin{aligned}T_E = &\{(p_u, 0, s_v) \,:\, (u, v) \in (V_0 \times V_1) \cap E\} \cup \{(s_u, 0, q_v) \,:\, (u, v) \in (V_{k-1} \times V_0) \cap E\} \cup \\ &\{(s_u, 0, s_v) \,:\, (u, v) \in (V_i \times V_{i+1}) \cap E \text{ for some } 1 \le i < k-1\}.\end{aligned}$$

The final transitions connect the states in Q_0 sequentially. Each such transition decrements the counter. Intuitively speaking, if the state reached in Q_0 matches the counter that has a value corresponding to the vertex in V_0 then the final state q_{v_1} can be reached with counter value zero.

$$T_F = \{(q_{v_{i+1}}, -1, q_{v_i}) \,:\, 1 \le i < \ell\}.$$

Importantly, there is a run from the initial configuration $p_{v_1}(0)$ to the target configuration $q_{v_1}(0)$ in $\mathcal{V}$ if and only if there is a k-cycle in the k-circle-layered graph G. In closing, observe that $|Q| \le 2|V|$ and $|T| \le 2|V| + |E|$. Therefore, $\mathcal{V}$ has size $\mathcal{O}(m)$. We remark that the greatest possible counter value is trivially bounded above by $|Q|$, hence $\mathcal{V}$ is a unary $(\mathcal{O}(m), 1)$-VASS of size $\mathcal{O}(m)$ that can be constructed in $\mathcal{O}(m)$ time. □

Theorem 5.4. *Assuming the k-cycle hypothesis, there does not exist an $\varepsilon > 0$ and an $\mathcal{O}(n^{2-\varepsilon})$-time algorithm that decides reachability (or coverability) in unary $(\mathcal{O}(n), 1)$-VASS of size n.*

Proof. Assume, for the sake of contradiction, that reachability in a unary $(\mathcal{O}(n), 1)$-VASS of size n can be solved in $\mathcal{O}(n^{2-\varepsilon})$ time for some $\varepsilon > 0$. By the k-cycle hypothesis (Hypothesis 5.2), there exists a k such that the problem of detecting a k-cycle in a k-circle layered graph with m vertices cannot be solved in $\mathcal{O}(m^{2-\varepsilon})$ time. Via the reduction presented above in Lemma 5.3, we create a $(\mathcal{O}(n), 1)$-VASS $\mathcal{V}$ of size $n = \mathcal{O}(m)$ together with an initial configuration $p(0)$ and a target configuration $q(0)$, such that deciding reachability from $p(0)$ to $q(0)$ in $\mathcal{V}$ determines the existence of a k-cycle in G. Thus the $\mathcal{O}(n^{2-\varepsilon})$-time algorithm for reachability would yield an $\mathcal{O}(m^{2-\varepsilon})$-time algorithm for detecting k-cycles, contradicting the k-cycle hypothesis.

By the equivalence of coverability and reachability in unary $(\mathcal{O}(n), 1)$-VASS in Lemma 5.6, the same lower bound holds for coverability. □

We can now use Theorem 5.4 to obtain a conditional lower bound on the time required to decide coverability in unary 2-VASS.

Corollary 5.5. *Assuming the k-cycle hypothesis, there does not exist an $\varepsilon > 0$ and an $\mathcal{O}(n^{2-\varepsilon})$-time algorithm that decides coverability in unary 2-VASS of size n.*

Proof. Consider a standard modification of the reduction presented for Lemma 5.3, that is, to increase the dimension of $\mathcal{V} = (Q, T)$ by one by adding an opposite counter of sorts, yielding a 2-VASS $\mathcal{W} = (Q, T')$. For every transition $(p, x, q) \in T$, create a transition also modifying the opposite counter, $(p, (x, -x), q) \in T'$. Now, the instance $(\mathcal{W}, p(0, n), q(0, n))$ of coverability holds if and only if the instance $(\mathcal{V}, p(0), q(0))$ of reachability holds. The rest follows by Theorem 5.4. □

Equivalence of Coverability and Reachability in Linearly-Bounded VASS

Reachability in $(\mathcal{O}(n), d)$-VASS can be decided in $\mathcal{O}(n(B(n)+1)^d)$ time using the simple algorithm for Observation 5.1 with a trivially modified acceptance condition. It turns out that coverability and reachability are equivalent in unary $(\mathcal{O}(n), d)$-VASS. This is true in the sense that it may hold that, for example, coverability in a $(100n, d)$-VASS may be reduced in linear time to reachability in a $(3n, d)$-VASS. Conversely, reachability in some linearly-bounded d-VASS can be reduced in linear time to a corresponding instance of coverability in a linearly-bounded d-VASS. Note that perversely, it appears plausible that instances of coverability in a $(100n, d)$-VASS could in fact be simpler to solve than in a $(3n, d)$-VASS.

Lemma 5.6. *For a $(B(n), d)$-VASS, let $C^{B(n)}(n)$ and $R^{B(n)}(n)$ denote the optimal running times for coverability and reachability, respectively. For any $\gamma > 0$, there exists some $\delta > 0$ such that $C^{\gamma\cdot n}(n) \leq \mathcal{O}(R^{\delta\cdot n}(n))$. Conversely, for any $\gamma > 0$, there exists some $\delta > 0$ such that $R^{\gamma\cdot n}(n) \leq \mathcal{O}(C^{\delta\cdot n}(n))$.*

Proof. Given an instance $(\mathcal{V}, p(\mathbf{s}), q(\mathbf{t}))$, of size n, of coverability in $(B(n), d)$-VASS $\mathcal{V}$. We construct a $(B(n), d)$-VASS $\mathcal{V}'$ from $\mathcal{V}$ by adding transitions $(q, -\mathbf{e}_i, q)$ for every $1 \leq i \leq d$. It is easy to see that there exists a run from $p(\mathbf{s})$ to $q(\mathbf{t})$ in $\mathcal{V}'$ if and only if there exists and a run from $p(\mathbf{s})$ to $q(\mathbf{t}')$ in $\mathcal{V}$ for some $\mathbf{t}' \geq \mathbf{t}$. Since $\|\mathcal{V}'\| = \mathcal{O}(n)$, for $B(n) = \gamma \cdot n$ we can ensure (by possibly adding any number of unreachable states) that $B(n) = \delta \cdot \|\mathcal{V}'\|$ for an appropriately selected δ. Thus, $C^{\gamma n}(n) = \mathcal{O}(R^{\delta n}(\mathcal{O}(n))) = \mathcal{O}(R^{\delta n}(n))$.

Conversely, consider an instance $(\mathcal{V}, p(\mathbf{s}), q(\mathbf{t}))$, of size n, of reachability in a $(B(n), d)$-VASS $\mathcal{V}$, again denote $n = \|\mathcal{V}\|$. We construct the $(B(n), d)$-VASS $\mathcal{V}'$ from $\mathcal{V}$ by adding a path from q to a new state r whose transitions update the counters by $-\mathbf{t}$. This is easily implementable by a path of length at most $B(n)$, for if $\|\mathbf{t}\| > B(n)$ this instance is trivially false. We then append a path from r to a new state r' whose transitions add $B(n)$ to every counter. It is easy to see that there is a run from $p(\mathbf{s})$ to $r'((B(n), \dots, B(n)))$ in $\mathcal{V}'$ if and only if there exists a run from $p(\mathbf{s})$ to $q(\mathbf{t})$ in $\mathcal{V}$. Since $\|\mathcal{V}'\| = \mathcal{O}(n + B(n))$, for $B(n) = \gamma \cdot n$ we can ensure that $B(\|\mathcal{V}'\|) = \delta \cdot \|\mathcal{V}'\|$ for some δ. Thus, $R^{\gamma n}(n) = \mathcal{O}(C^{\delta n}(\mathcal{O}(n))) = \mathcal{O}(C^{\delta n}(n))$. □

Lower Bounds for Reachability in Linearly-Bounded VASS

To obtain further lower bounds for the coverability problem in $(\mathcal{O}(n), d)$-VASS, by Lemma 5.6, we can, equivalently, find lower bounds for the reachability problem in $(\mathcal{O}(n), d)$-VASS. In Theorem 5.8, we will assume a well-established hypothesis concerning the time required to detect a hyperclique in a 3-uniform hypergraph. In fact, Lincoln, Vassilevska Williams, and Williams state and justify an even stronger hypothesis about μ-uniform hypergraphs for every $\mu \geq 3$ [43, Hypothesis 1.4]. We will use this computational complexity hypothesis to expose precise lower bounds on the time complexity of reachability in linearly-bounded fixed dimension unary VASS.

Hypothesis 5.7 (k-Hyperclique Hypothesis [43, Hypothesis 1.4]). *Let $k \geq 3$ be a fixed integer. There does not exist an $\varepsilon > 0$ and an $\mathcal{O}(r^{k-\varepsilon})$-time algorithm for detecting whether there is a k-hyperclique in a 3-uniform hypergraph with r vertices.*

For the remainder of this section, we focus on the proof of the following theorem.

Theorem 5.8. *Let $d \geq 1$ be a fixed integer. Assuming Hypothesis 5.7, there does not exist an $\varepsilon > 0$ and an $\mathcal{O}(n^{d-\varepsilon})$-time algorithm that decides reachability in a unary $(\mathcal{O}(n), d+2)$-VASS of size n.*

The lower bound is obtained via reduction from detecting hyperclique in 3-uniform hypergraphs, hence it is subject to the k-hyperclique hypothesis. We present our reduction in two steps. The first step is an intermediate step, in Lemma 5.9 we offer a reduction to an instance of reachability in unary VASS with a limited number of zero tests. The second step extends the first, in Lemma 5.10 we modify the reduction by adding a counter so zero tests are absented. This extension leverages the recently developed *controlling counter technique* of Czerwiński and Orlikowski [21]. This technique allows for implicit zero tests to be performed in the presence of a dedicated counter whose transition effects and reachability condition ensure the implicit zero tests were indeed performed correctly.

It has been shown that we may assume that the hypergraph is k-partite for k-hyperclique Hypothesis 5.7 [43, Theorem 3.1]. Thus, we may assume that the vertices can be partitioned into k disjoint subsets $V = V_1 \cup \dots \cup V_k$ and every hyperedge $\{u, v, w\}$ comprises of three vertices from distinct subsets $u \in V_{i_1}$, $v \in V_{i_2}$, and $w \in V_{i_3}$ for some $1 \leq i_1 < i_2 < i_3 \leq k$.

Lemma 5.9. *Let $d \geq 1$ be a fixed integer. Given a $4d$-partite 3-uniform hypergraph $H = (V_1 \cup \ldots \cup V_{4d}, E)$ with r vertices, there exists a unary $(\mathcal{O}((r\log(r))^4), d+1)$-VASS with $\mathcal{O}(d^3)$ zero tests $\mathcal{T}$ such that there is a $4d$-hyperclique in H if and only if there is a run from $q_I(0)$ to $q_F(0)$ in $\mathcal{T}$. Moreover, $\mathcal{T}$ can be constructed in poly$(d) \cdot (r\log(r))^4$ time.*

Proof. We will re-employ some of the ideas already used in the constructions of the proof of Lemma 4.3. In particular, we will use **Multiply** and **Divide** subprocedures, see Algorithm 2. Let us denote the $d+1$ counters $\mathsf{x}_1, \dots, \mathsf{x}_d, \mathsf{y}$. The collective role of $\mathsf{x}_1, \dots, \mathsf{x}_d$ is to maintain a representation of the $4d$ vertices forming the $4d$-hyperclique. The role of y is to ensure multiplications and divisions are completed correctly. Just as previously seen, before the execution of **Multiply** or **Divide**, we require $\mathsf{y} = 0$. We will combine these subprocedures to construct new subprocedures for unary $(d+1)$-VASS with zero tests to verify properties related to 3-uniform hypergraphs.

We start by finding the first r primes. We associate a distinct prime p_v to each vertex $v \in V$. Now we encode the chosen $4d$ vertices that will form the $4d$-clique by storing, on d many counters, products of four primes corresponding to four of the selected vertices. Therefore, after the initial guessing part, the value of counter x_i will be $p_t \cdot p_u \cdot p_v \cdot p_w$ for some vertices $t, u, v, w \in V$. Roughly speaking, we store the product of four primes on one counter so that the maximum observable counter value matches the size of the resulting VASS with zero tests.

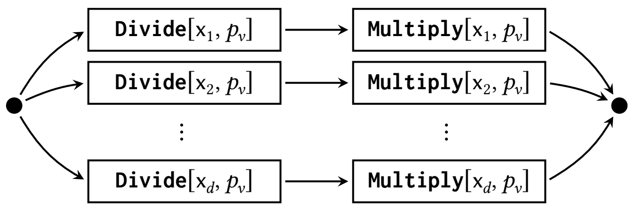
Guessing part. The VASS presented in Algorithm 4 implements the following algorithm: Guess $4d$ vertices, not necessarily distinct, and check whether they form a $4d$-hyperclique. Note that this algorithm is correct because it does not help us to repeatedly guess the same vertex or repeatedly guess vertices that belong to the same vertex subset. In contrast to Algorithm 1, the main difference is that the guessed vertices are encoded as quadruple products of primes across counters x_1, ..., x_d. We do not store the entire product of $4d$ primes explicitly, only the values of each counter.

Checking part. In the second part, we verify that we have selected a $4d$-hyperclique by testing for each of the $\binom{4d}{3}$ hyperedges. This is achieved in essentially the same way as **Edge**$[e]$ was implemented in Algorithm 3. We check that between every triplet of vertex subsets there is a hyperedge that has all of the vertices selected in the first part. At the end of the checking part, there are $d+1$ self-loops that each decrement one of the $d+1$ counters. These loops can be used to reach the target configuration $q_F(0)$ once all hyperedge checks have been succeeded.

For ease of presentation and as previously mentioned, we introduce the **VertexSelected** subprocedure that checks whether a given vertex has been selected; see Figure 4.

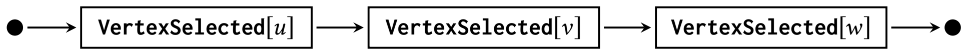
We also implement the **HyperEdge** subprocedure for checking whether the vertices, in a given hyperedge, have been selected; see Figure 5. This subprocedure checks that the three primes corresponding to the three vertices in the hyperedge can divide one of the values stored in $\mathsf{x}_1, \dots, \mathsf{x}_d$.

Now, we use the **HyperEdge** subprocedure to construct the checking part of $\mathcal{T}$. This part consists of a sequence of $\binom{4d}{3}$ non-deterministic branching sections, one for each triplet of vertex subsets V_{i_1}, V_{i_2}, and V_{i_3} where $1 \leq i_1 < i_2 < i_3 \leq 4d$. In each branching section, there is an instance of the



VertexSelected[v]

Fig. 4. The **VertexSelected** subprocedure implemented in a unary $(d+1)$-VASS with zero tests. To instantiate this subprocedure, a vertex v is specified so that the d counters can be checked for divisibility by the prime p_v, with the effect of checking whether the vertex v has been selected. The counter y is used by the **Divide** and **Multiply** subprocedures to ensure the division and multiplications are completed correctly.



HyperEdge[$\{u, v, w\}$]

Fig. 5. The **HyperEdge** subprocedure implemented in a unary $(d + 1)$-VASS with zero tests. Note that the three vertices of the given hyperedge may be stored across any of the d counters $x_1, \dots, x_d$. Therefore, we make use of the **VertexSelected** subprocedure three times to check if indeed u, v, and w have been selected.

ALGORITHM 4: A counter program representing the VASS with zero tests. As seen earlier, the variables in **for** loops are just syntactic sugar for repeating similar lines of code which correspond to states in the VASS. Similarly, the variables in **guess** statements represent non-deterministic branching transitions in a VASS.

```
input: $x_1, \dots, x_d, y = 0$
for $i \leftarrow 1$ to $d$ do
    $x_i$ += 1
    for $g \leftarrow 1$ to 4 do
        guess $j \in \{1, \dots, r\}$
        Multiply[$x_i, p_j$]
    end
end
for $(i_1, i_2, i_3) \in \{1, \dots, 4d\}^3, i_1 < i_2 < i_3$ do
    guess $e \in E \cap \{\{u, v, w\} : u \in V_{i_1}, v \in V_{i_2}, w \in V_{i_3}\}$
    HyperEdge[$e$]
end
for $i \leftarrow 1$ to $d$ do
    loop $x_i$ −= 1
end
loop y −= 1
```

HyperEdge subprocedure for each of the hyperedges $\{u, v, w\} \in E$ such that $u \in V_{i_1}$, $v \in V_{i_2}$, and $w \in V_{i_3}$.

Figure 6 implements the check that the selected vertices indeed form a hyperclique: In order to reach the final state q_F, there must be a hyperedge between each of the $4d$ vertices selected in the

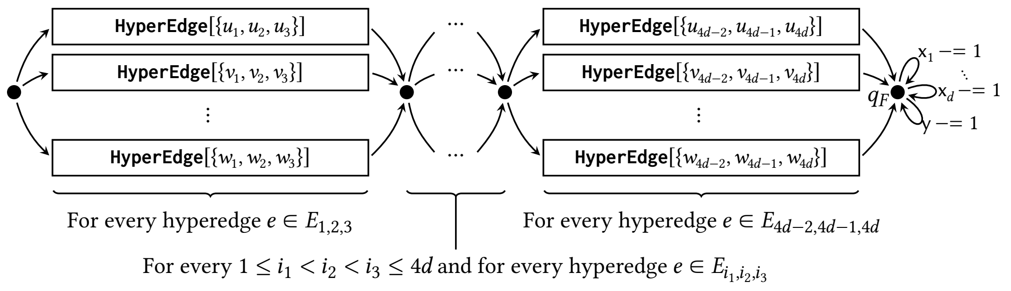

HyperEdge[$\{u_1, u_2, u_3\}$]
HyperEdge[$\{v_1, v_2, v_3\}$]
HyperEdge[$\{w_1, w_2, w_3\}$]
HyperEdge[$\{u_{4d-2}, u_{4d-1}, u_{4d}\}$]
HyperEdge[$\{v_{4d-2}, v_{4d-1}, v_{4d}\}$]
HyperEdge[$\{w_{4d-2}, w_{4d-1}, w_{4d}\}$]
q_F
$x_1 -= 1$
$x_d -= 1$
$y -= 1$
For every hyperedge $e \in E_{1,2,3}$
For every hyperedge $e \in E_{4d-2,4d-1,4d}$
For every $1 \le i_1 < i_2 < i_3 \le 4d$ and for every hyperedge $e \in E_{i_1,i_2,i_3}$

Fig. 6. The check part of the unary $(d+1)$-VASS with zero tests $\mathcal{T}$ for detecting a $4d$-hyperclique in $4d$-partite hypergraph. The set E_{i_1,i_2,i_3} is shorthand for the subset of hyperedges containing vertices belonging to V_{i_1}, V_{i_2}, and V_{i_3}; precisely $E_{i_1,i_2,i_3} = \{\{u, v, w\} \in E : u \in V_{i_1}, v \in V_{i_2}, w \in V_{i_3}\}$.

first part of $\mathcal{T}$. Thus, there is a $4d$-hyperclique in H if and only if there is a run from $q_I(0)$ to $q_F(\mathrm{t})$, for some $\mathrm{t} \ge 0$ in $\mathcal{T}$. Given that there are self-loops decrementing each of the counters, there is always a run from $q_F(\mathrm{t})$ to $q_F(0)$. Therefore, H contains a $4d$-hyperclique if and only if there is a run from $q_I(0)$ to $q_F(0)$ in $\mathcal{T}$.

We are now able to finalize the proof. We will carefully analyse the maximum counter value observed on any run, count the number of zero tests performed on any run, and lastly evaluate the size of $\mathcal{T}$.

The highest counter value observed by each counter $\mathrm{x}_1, \dots, \mathrm{x}_d$ is the product of four primes. The highest counter value observed by y is equal to the highest counter value observed by any of the other counters $\mathrm{x}_1, \dots, \mathrm{x}_d$. Therefore the bound on the highest value observed, altogether can be bounded about by p^4 where p is the rth prime. By the Prime Number Theorem (for example, see [61]) we know that $p \in \mathcal{O}(r\log(r))$. Therefore, every run from $q_I(0)$ to $q_F(0)$ in $\mathcal{T}$ is $\mathcal{O}((r\log(r))^4)$-bounded.

Zero tests are only performed by the **Multiply** and **Divide** subprocedures, each instance of these subprocedures containing two zero tests. We therefore count the number of calls to these subprocedures on any given run. In the guessing part, there is one call to **Multiply** for each of the $4d$ vertices selected to form a $4d$-hyperclique. In the checking part, there is a sequence of $\binom{4d}{3}$ many calls to **HyperEdge**. Each **HyperEdge** call contains three invocations of **VertexSelected**, which, in turn, executes **Divide** and **Multiply** once each. In total, there are $2(4d + 6\binom{4d}{3}) \in \mathcal{O}(d^3)$ many zero tests are performed in any run from $q_I(0)$ to $q_F(0)$ in $\mathcal{T}$.

Finally, each instance of the **Multiply** and **Divide** subprocedures has size $\mathcal{O}(r\log(r))$. Note that the first r primes can be found in $r^{1+o(1)}$ time [2]. In the guessing part, there are $4d \cdot r$ instances of the **Multiply** subprocedure. In the checking part, there is an instance of the **HyperEdge** subprocedure for each edge in the hypergraph. The **HyperEdge** subprocedures themselves appear in $\binom{4d}{3}$ many collections, one for each triplet of vertex subsets. Each **HyperEdge** subprocedure contains three instances of the **VertexSelected** subprocedure, which contains d instances of the **Multiply** subprocedure and d instances of the **Divide** subprocedure. Therefore, in total $\mathcal{T}$ has polynomial size and can be constructed in $\mathcal{O}(d \cdot r^2 \log(r) + m \cdot \binom{4d}{3} \cdot d \cdot r\log(r))$ time, where $m \in \mathcal{O}(r^3)$ is the total number of hyperedges. □

Consider our earlier described two-step approach towards proving Theorem 5.8 by first obtaining a unary $(d+1)$-VASS $\mathcal{T}$ with zero tests and then obtaining a unary $(d+2)$-VASS $\mathcal{V}$ by increasing the dimension by one and removing the zero tests. In actuality, both steps occur together to prove Theorem 5.8. Ultimately, the $d+2$ counters of $\mathcal{V}$ have the following roles. The counters $\mathrm{x}_1, \dots, \mathrm{x}_d$ are used to store the products of primes corresponding to vertices of hyperclique. The counter y is used to complete multiplications and divisions. In the remainder of this section, we add the $(d+2)$-nd counter, that is, used to ensure the (implicit) zero tests are performed faithfully. We achieve this by leveraging the *controlling counter technique* that was developed by Czerwiński and Orlikowski and first formalised in [21]. The following is the restatement of their technique, their lemma has been restricted to our scenario and rewritten using the notation of this article.

Lemma 5.10 ([21, Lemma 10]). *Let ρ be a run in a $(d+2)$-VASS such that $q_I(0) \xrightarrow{\rho} q_F(0)$. Further, let $q_0(\mathrm{v}_0)$, $q_1(\mathrm{v}_1), \ldots, q_r(\mathrm{v}_r)$ be some distinguished configurations observed along the run ρ with $q_0(\mathrm{v}_0) = q_I(0)$ and $q_r(\mathrm{v}_r) = q_F(0)$ and let ρ_j be the segment of ρ, that is, between $q_{j-1}(\mathrm{v}_{j-1})$ and $q_j(\mathrm{v}_j)$, so ρ can be described as:*

$$q_I(0) = q_0(\mathrm{v}_0) \xrightarrow{\rho_1} q_1(\mathrm{v}_1) \to \cdots \to q_{r-1}(\mathrm{v}_{r-1}) \xrightarrow{\rho_r} q_r(\mathrm{v}_r) = q_F(0).$$

Let $S_1, \ldots, S_d, S_{d+1} \subseteq \{0, 1, \ldots, r\}$ be the sets of indices of the distinguished configurations where zero tests could be performed on counters $\mathrm{x}_1, \ldots, \mathrm{x}_d, \mathrm{x}_{d+1}$, respectively. Let $t_{j,i} = |\{s \geq j \,:\, s \in S_i\}|$ be the number of zero test for the counter x_i in the remainder of the run $\rho_{j+1} \cdots \rho_r$. Given that $\mathrm{v}_0 = 0$ and $\mathrm{v}_r = 0$, if

$$\mathrm{eff}(\rho_j)[d+2] = \sum_{i=1}^{d+1} t_{j,i} \cdot \mathrm{eff}(\rho_j)[i], \tag{1}$$

then for every $i \in \{1, \ldots, d, d+1\}$ and $j \in S_i$, we know that $\mathrm{v}_j[i] = 0$.

With Lemma 5.10 in hand, we can ensure that the $\mathcal{O}(d^3)$ zero tests performed by $\mathcal{T}$ from Lemma 5.9 are executed correctly. We conclude this section with a proof of Theorem 5.8.

Proof of Theorem 5.8. Consider the reduction, presented in Lemma 5.9, from detecting a $4d$-hyperclique in a $4d$-partite 3-uniform hypergraph H to reachability in unary $(\mathcal{O}((r\log(r))^4), d+1)$-VASS with $\mathcal{O}(d^3)$ zero tests. Now, given Lemma 5.10, we will add a controlling counter to $\mathcal{T}$ so that the zero tests on the $d+1$ counters $\mathrm{x}_1, \ldots, \mathrm{x}_d, \mathrm{y}$ are instead performed implicitly. To this end, we introduce another counter z that receives updates on transitions, consistent with Equation (1), whenever any of the other counters are updated. Note that counters y and z, for the sake of a succinct and consistent description, are, respectively, referred to as counters x_{d+1} and x_{d+2} in the statement of Lemma 5.10. Moreover, notice that the maximum value of z is bounded by $\mathsf{poly}(d) \cdot (\sum_{i=1}^{d+1} \mathrm{x}_i) \in \mathsf{poly}(d) \cdot (r\log(r))^4$.

Therefore, we have constructed a unary $\big(\mathsf{poly}(d) \cdot (r\log(r))^4, d+2\big)$-VASS $\mathcal{V}$ with the property that there H contains a $4d$-hyperclique if and only if there is a run from $q_I(0)$ to $q_F(0)$ in $\mathcal{V}$. Such a $(\mathsf{poly}(d) \cdot (r\log(r))^4, d+2)$-VASS $\mathcal{V}$ has size $\mathcal{O}(t \cdot |\mathcal{T}|)$ where $t \in \mathsf{poly}(d)$ is the number of zero tests performed on the run from $q_I(0)$ to $q_F(0)$ in $\mathcal{T}$. Moreover, $\mathcal{V}$ can be constructed in $\mathsf{poly}(d) \cdot (r\log(r))^4$ time. Hence, if reachability in unary $(\mathcal{O}(n), d+2)$-VASS of size n can be solved in $n^{d-\varepsilon}$ time for some $\varepsilon > 0$, then one can decide whether there is a $4d$-hyperclique in a 3-uniform hypergraph with r vertices in $r^{4d-\varepsilon'}$ time for some $\varepsilon' > 0$, contradicting Hypothesis 5.7. □

6 Conclusion

Summary

In this article, we have revisited a classical problem of coverability in d-VASS. We have closed the gap left by Rosier and Yen [54] on the length of runs witnessing instances of coverability in d-VASS. We have lowered the upper bound of $n^{2^{\mathcal{O}(d\log(d))}}$, from Rackoff's technique [53], to $n^{\mathcal{O}(d \cdot 2^d)}$ (Theorem 3.3), matching the $n^{2^{\Omega(d)}}$ lower bound from Lipton's construction [44]. This accordingly closes the gap on the exact space required for the coverability problem and yields a deterministic

$n^{\mathcal{O}(d^2 \cdot 2^d)}$-time algorithm for coverability in d-VASS (Corollary 3.4). We complement this with a matching lower bound conditional on ETH; there does not exist a deterministic $n^{2^{o(d)}}$-time algorithm for coverability (Theorem 4.2). As mentioned in Remark 4.5, the difference between the constant multiplicative factors in the second exponent between our upper bound and our conditional lower bound only differ by a factor 4. By and large, this settles the exact space and time complexity of coverability in VASS.

In addition, we study linearly-bounded unary d-VASS. Here, coverability and reachability are equivalent and the trivial exhaustive search $\mathcal{O}(n^{d+1})$ algorithm is near-optimal. We prove that reachability in linearly-bounded 1-VASS requires $n^{2-o(1)}$ time under the k-cycle hypothesis (Theorem 5.4), matching the trivial upper bound. We further prove that reachability in linearly-bounded $(d+2)$-VASS requires $n^{d-o(1)}$ time under the 3-uniform hyperclique hypothesis (Theorem 5.8).

Open Problems

The *boundedness problem*, a problem closely related to coverability, asks whether, from a given initial configuration, the set of all reachable configurations is finite. This problem was also studied by Lipton and Rackoff and is EXPSPACE-complete [44, 53]. Boundedness was further analysed by Rosier and Yen [54, Theorem 2.1] and the same gap also exists for the exact space required. We leave the same improvement, to eliminate the same twice-exponentiated $\log(d)$ factor, as an open problem.

Our lower bounds for the time complexity of coverability and reachability in linearly-bounded unary d-VASS, for $d \geq 2$, leave a gap of up to $n^{3+o(1)}$, see Table 1. We leave it as an open problem to either improve upon the upper bound $\mathcal{O}(n^{d+1})$ given by the trivial algorithm, or to raise our conditional lower bounds.

Acknowledgments

We thank Łukasz Orlikowski for the time he spent checking and discussing a final draft of this article. We would also like to the thank the reviewers for their comments and for their time.

References

[1] Parosh Aziz Abdulla, Karlis Cerans, Bengt Jonsson, and Yih-Kuen Tsay. 2000. Algorithmic analysis of programs with well quasi-ordered domains. *Inf. Comput.* 160, 1-2 (2000), 109–127. DOI: https://doi.org/10.1006/inco.1999.2843

[2] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. 2004. PRIMES is in P. *Ann. Math.* 160, 2 (2004), 781–793. DOI: https://doi.org/10.4007/annals.2004.160.781

[3] Shaull Almagor, Nathann Cohen, Guillermo A. Pérez, Mahsa Shirmohammadi, and James Worrell. 2020. Coverability in 1-VASS with disequality tests. In *Proceedings of the 31st International Conference on Concurrency Theory, CONCUR 2020, September 1-4, 2020, Vienna, Austria (Virtual Conference) (LIPIcs)*, Igor Konnov and Laura Kovács (Eds.), Vol. 171. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 38:1–38:20. DOI: https://doi.org/10.4230/LIPIcs.CONCUR.2020.38

[4] Noga Alon, Raphael Yuster, and Uri Zwick. 1997. Finding and counting given length cycles. *Algorithmica* 17, 3 (1997), 209–223. DOI: https://doi.org/10.1007/BF02523189

[5] Bertie Ancona, Monika Henzinger, Liam Roditty, Virginia Vassilevska Williams, and Nicole Wein. 2019. Algorithms and hardness for diameter in dynamic graphs. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece (LIPIcs)*, Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi (Eds.), Vol. 132. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 13:1–13:14. DOI: https://doi.org/10.4230/LIPIcs.ICALP.2019.13

[6] Michael Benedikt, Timothy Duff, Aditya Sharad, and James Worrell. 2017. Polynomial automata: Zeroness and applications. In *Proceedings of the 32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017.* IEEE Computer Society, 1–12. DOI: https://doi.org/10.1109/LICS.2017.8005101

[7] Michael Blondin, Matthias Englert, Alain Finkel, Stefan Göller, Christoph Haase, Ranko Lazić, Pierre McKenzie, and Patrick Totzke. 2021. The reachability problem for two-dimensional vector addition systems with states. *J. ACM* 68, 5 (2021), 34:1–34:43. DOI: https://doi.org/10.1145/3464794

[8] Michael Blondin, Alain Finkel, Christoph Haase, and Serge Haddad. 2016. Approaching the coverability problem continuously. In *Proceedings of the Tools and Algorithms for the Construction and Analysis of Systems - 22nd International*

Conference, TACAS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings (Lecture Notes in Computer Science), Marsha Chechik and Jean-François Raskin (Eds.), Vol. 9636. Springer, 480–496. DOI: https://doi.org/10.1007/978-3-662-49674-9_28

[9] Michael Blondin, Christoph Haase, and Philip Offtermatt. 2021. Directed reachability for infinite-state systems. In *Proceedings of the Tools and Algorithms for the Construction and Analysis of Systems - 27th International Conference, TACAS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 - April 1, 2021, Proceedings, Part II (Lecture Notes in Computer Science)*, Jan Friso Groote and Kim Guldstrand Larsen (Eds.), Vol. 12652. Springer, 3–23. DOI: https://doi.org/10.1007/978-3-030-72013-1_1

[10] Mikołaj Bojańczyk, Claire David, Anca Muscholl, Thomas Schwentick, and Luc Segoufin. 2011. Two-variable logic on data words. *ACM Trans. Comput. Log.* 12, 4 (2011), 27:1–27:26. DOI: https://doi.org/10.1145/1970398.1970403

[11] Laura Bozzelli and Pierre Ganty. 2011. Complexity analysis of the backward coverability algorithm for VASS. In *Proceedings of the Reachability Problems - 5th International Workshop, RP 2011, Genoa, Italy, September 28-30, 2011. Proceedings (Lecture Notes in Computer Science)*, Giorgio Delzanno and Igor Potapov (Eds.), Vol. 6945. Springer, 96–109. DOI: https://doi.org/10.1007/978-3-642-24288-5_10

[12] Karl Bringmann, Allan Grønlund, Marvin Künnemann, and Kasper Green Larsen. 2024. The NFA acceptance hypothesis: Non-combinatorial and dynamic lower bounds. In *Proceedings of the 15th Innovations in Theoretical Computer Science Conference, ITCS 2024, January 30 to February 2, 2024, Berkeley, CA, USA (LIPIcs)*, Venkatesan Guruswami (Ed.), Vol. 287. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 22:1–22:25. DOI: https://doi.org/10.4230/LIPICS.ITCS.2024.22

[13] Krishnendu Chatterjee, Bhavya Choudhary, and Andreas Pavlogiannis. 2017. Optimal dyck reachability for data-dependence and alias analysis. *Proc. ACM Program. Lang.* 2, POPL (2017), 1–30.

[14] Jianer Chen, Benny Chor, Mike Fellows, Xiuzhen Huang, David W. Juedes, Iyad A. Kanj, and Ge Xia. 2005. Tight lower bounds for certain parameterized NP-hard problems. *Inf. Comput.* 201, 2 (2005), 216–231. DOI: https://doi.org/10.1016/j.ic.2005.05.001

[15] Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. 2006. Strong computational lower bounds via parameterized complexity. *J. Comput. Syst. Sci.* 72, 8 (2006), 1346–1367. DOI: https://doi.org/10.1016/j.jcss.2006.04.007

[16] Dmitry Chistikov, Wojciech Czerwinski, Filip Mazowiecki, Lukasz Orlikowski, Henry Sinclair-Banks, and Karol Wegrzycki. 2024. The tractability border of reachability in simple vector addition systems with states. In *Proceedings of the 65th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2024, Chicago, IL, USA, October 27-30, 2024.* IEEE, 1332–1354. DOI: https://doi.org/10.1109/FOCS61266.2024.00086

[17] Hubert Comon and Yan Jurski. 1998. Multiple counters automata, safety analysis and presburger arithmetic. In *Proceedings of the Computer Aided Verification, 10th International Conference, CAV '98, Vancouver, BC, Canada, June 28 - July 2, 1998, Proceedings (Lecture Notes in Computer Science)*, Alan J. Hu and Moshe Y. Vardi (Eds.), Vol. 1427. Springer, 268–279. DOI: https://doi.org/10.1007/BFb0028751

[18] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. 2015. *Parameterized Algorithms.* Springer. DOI: https://doi.org/10.1007/978-3-319-21275-3

[19] Wojciech Czerwiński, Slawomir Lasota, Ranko Lazic, Jérôme Leroux, and Filip Mazowiecki. 2020. Reachability in fixed dimension vector addition systems with states. In *Proceedings of the 31st International Conference on Concurrency Theory, CONCUR 2020, September 1-4, 2020, Vienna, Austria (Virtual Conference) (LIPIcs)*, Igor Konnov and Laura Kovács (Eds.), Vol. 171. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 48:1–48:21. DOI: https://doi.org/10.4230/LIPIcs.CONCUR.2020.48

[20] Wojciech Czerwiński, Sławomir Lasota, Ranko Lazic, Jérôme Leroux, and Filip Mazowiecki. 2021. The reachability problem for petri nets is not elementary. *J. ACM* 68, 1 (2021), 7:1–7:28. DOI: https://doi.org/10.1145/3422822

[21] Wojciech Czerwiński and Łukasz Orlikowski. 2021. Reachability in vector addition systems is ackermann-complete. In *Proceedings of the 62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022.* IEEE, 1229–1240. DOI: https://doi.org/10.1109/FOCS52979.2021.00120

[22] Wojciech Czerwiński and Łukasz Orlikowski. 2022. Lower bounds for the reachability problem in fixed dimensional VASSes. In *Proceedings of the 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, 40:1–40:12. DOI: https://doi.org/10.1145/3531130.3533357

[23] Mina Dalirrooyfard, Ce Jin, Virginia Vassilevska Williams, and Nicole Wein. 2022. Approximation algorithms and hardness for n-pairs shortest paths and all-nodes shortest cycles. In *Proceedings of the 63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022.* IEEE, 290–300. DOI: https://doi.org/10.1109/FOCS54457.2022.00034

[24] Mina Dalirrooyfard, Thuy Duong Vuong, and Virginia Vassilevska Williams. 2021. Graph pattern detection: Hardness for all induced patterns and faster noninduced cycles. *SIAM J. Comput.* 50, 5 (2021), 1627–1662. DOI: https://doi.org/10.1137/20M1335054

[25] Dani Dorfman, Haim Kaplan, Robert E. Tarjan, and Uri Zwick. 2023. Optimal energetic paths for electric cars. In *Proceedings of the 31st Annual European Symposium on Algorithms, ESA 2023 (LIPIcs)*, Vol. 274. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 42:1–42:17. DOI: https://doi.org/10.4230/LIPICS.ESA.2023.42

[26] Javier Esparza. 1996. Decidability and complexity of petri net problems - an introduction. In *Proceedings of the Lectures on Petri Nets I: Basic Models, Advances in Petri Nets, the volumes are based on the Advanced Course on Petri Nets, held in Dagstuhl, September 1996 (Lecture Notes in Computer Science)*, Wolfgang Reisig and Grzegorz Rozenberg (Eds.), Vol. 1491. Springer, 374–428. DOI: https://doi.org/10.1007/3-540-65306-6_20

[27] Javier Esparza, Ruslán Ledesma-Garza, Rupak Majumdar, Philipp J. Meyer, and Filip Niksic. 2014. An SMT-based approach to coverability analysis. In *Proceedings of the Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings (Lecture Notes in Computer Science)*, Armin Biere and Roderick Bloem (Eds.), Vol. 8559. Springer, 603–619. DOI: https://doi.org/10.1007/978-3-319-08867-9_40

[28] John Fearnley and Marcin Jurdziński. 2013. Reachability in two-clock timed automata is PSPACE-complete. In *Proceedings of the Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II (Lecture Notes in Computer Science)*, Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska, and David Peleg (Eds.), Vol. 7966. Springer, 212–223. DOI: https://doi.org/10.1007/978-3-642-39212-2_21

[29] Diego Figueira, Santiago Figueira, Sylvain Schmitz, and Philippe Schnoebelen. 2011. Ackermannian and primitive-recursive bounds with dickson's lemma. In *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science, LICS 2011, June 21-24, 2011, Toronto, Ontario, Canada.* IEEE Computer Society, 269–278. DOI: https://doi.org/10.1109/LICS.2011.39

[30] Pierre Ganty and Rupak Majumdar. 2012. Algorithmic verification of asynchronous programs. *ACM Trans. Program. Lang. Syst.* 34, 1 (2012), 6:1–6:48. DOI: https://doi.org/10.1145/2160910.2160915

[31] Steven M. German and A. Prasad Sistla. 1992. Reasoning about systems with many processes. *J. ACM* 39, 3 (1992), 675–735. DOI: https://doi.org/10.1145/146637.146681

[32] Christoph Haase, Stephan Kreutzer, Joël Ouaknine, and James Worrell. 2009. Reachability in succinct and parametric one-counter automata. In *Proceedings of the CONCUR 2009 - Concurrency Theory, 20th International Conference, CONCUR 2009, Bologna, Italy, September 1-4, 2009. Proceedings (Lecture Notes in Computer Science)*, Mario Bravetti and Gianluigi Zavattaro (Eds.), Vol. 5710. Springer, 369–383. DOI: https://doi.org/10.1007/978-3-642-04081-8_25

[33] Christoph Haase, Joël Ouaknine, and James Worrell. 2012. On the relationship between reachability problems in timed and counter automata. In *Proceedings of the Reachability Problems - 6th International Workshop, RP 2012, Bordeaux, France, September 17-19, 2012. Proceedings (Lecture Notes in Computer Science)*, Alain Finkel, Jérôme Leroux, and Igor Potapov (Eds.), Vol. 7550. Springer, 54–65. DOI: https://doi.org/10.1007/978-3-642-33512-9_6

[34] Torben Hagerup. 1998. Sorting and searching on the word RAM. In *Proceedings of the STACS 98: 15th Annual Symposium on Theoretical Aspects of Computer Science Paris.* Springer, 366–398.

[35] John E. Hopcroft and Jean-Jacques Pansiot. 1979. On the Reachability Problem for 5-Dimensional Vector Addition Systems. *Theor. Comput. Sci.* 8, 2 (1979), 135–159. DOI: https://doi.org/10.1016/0304-3975(79)90041-0

[36] Russell Impagliazzo and Ramamohan Paturi. 2001. On the Complexity of k-SAT. *J. Comput. Syst. Sci.* 62, 2 (2001), 367–375. DOI: https://doi.org/10.1006/jcss.2000.1727

[37] Ulla Koppenhagen and Ernst W. Mayr. 2000. Optimal algorithms for the coverability, the subword, the containment, and the equivalence problems for commutative semigroups. *Inf. Comput.* 158, 2 (2000), 98–124. DOI: https://doi.org/10.1006/inco.1999.2812

[38] Paraschos Koutris and Shaleen Deep. 2023. The fine-grained complexity of CFL reachability. *Proc. ACM Program. Lang.* 7, POPL, Article 59 (jan 2023), 27 pages. DOI: https://doi.org/10.1145/3571252

[39] Ranko Lazic and Sylvain Schmitz. 2021. The ideal view on Rackoff's coverability technique. *Inf. Comput.* 277 (2021), 104582. DOI: https://doi.org/10.1016/j.ic.2020.104582

[40] Jérôme Leroux. 2013. Vector addition system reversible reachability problem. *Log. Methods Comput. Sci.* 9, 1 (2013). DOI: https://doi.org/10.2168/LMCS-9(1:5)2013

[41] Jérôme Leroux. 2021. The reachability problem for petri nets is not primitive recursive. In *Proceedings of the 62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022.* IEEE, 1241–1252. DOI: https://doi.org/10.1109/FOCS52979.2021.00121

[42] Jérôme Leroux and Sylvain Schmitz. 2019. Reachability in vector addition systems is primitive-recursive in fixed dimension. In *Proceedings of the 34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019.* IEEE, 1–13. DOI: https://doi.org/10.1109/LICS.2019.8785796

[43] Andrea Lincoln, Virginia Vassilevska Williams, and R. Ryan Williams. 2018. Tight hardness for shortest cycles and paths in sparse graphs. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, Artur Czumaj (Ed.). SIAM, 1236–1252. DOI: https://doi.org/10.1137/1.9781611975031.80

[44] Richard Lipton. 1976. The reachability problem requires exponential space. *Dep. Comput. Sci., Yale Univ.* 62 (1976), 1–16.

[45] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. 2013. Lower bounds based on the exponential time hypothesis. *Bulletin of EATCS* 3, 105 (2013), 47–72.

[46] Anders Alnor Mathiasen and Andreas Pavlogiannis. 2021. The fine-grained and parallel complexity of andersen's pointer analysis. *Proc. ACM Program. Lang.* 5, POPL (2021), 1–29. DOI: https://doi.org/10.1145/3434315

[47] Ernst W. Mayr. 1984. An algorithm for the general petri net reachability problem. *SIAM J. Comput.* 13, 3 (1984), 441–460. DOI: https://doi.org/10.1137/0213029

[48] Ernst W. Mayr and Albert R. Meyer. 1982. The complexity of the word problems for commutative semigroups and polynomial ideals. *Adv. Math.* 46, 3 (1982), 305–329. DOI: https://doi.org/10.1016/0001-8708(82)90048-2

[49] Filip Mazowiecki and Michał Pilipczuk. 2019. Reachability for bounded branching VASS. In *Proceedings of the 30th International Conference on Concurrency Theory, CONCUR 2019, August 27-30, 2019, Amsterdam, the Netherlands (LIPIcs)*, Wan J. Fokkink and Rob van Glabbeek (Eds.), Vol. 140. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 28:1–28:13. DOI: https://doi.org/10.4230/LIPIcs.CONCUR.2019.28

[50] Filip Mazowiecki, Henry Sinclair-Banks, and Karol Węgrzycki. 2023. Coverability in 2-VASS with one unary counter is in NP. In *Proceedings of the Foundations of Software Science and Computation Structures*, Orna Kupferman and Paweł Sobociński (Eds.). Springer Nature Switzerland, 196–217. DOI: https://doi.org/10.1007/978-3-031-30829-1_10

[51] Marvin L. Minsky. 1967. *Computation: Finite and Infinite Machines.* Prentice-Hall, Inc.

[52] Jaroslav Nešetřil and Svatopluk Poljak. 1985. On the complexity of the subgraph problem. *Comment. Math. Univ. Carol.* 26, 2 (1985), 415–419.

[53] Charles Rackoff. 1978. The covering and boundedness problems for vector addition systems. *Theor. Comput. Sci.* 6, 2 (1978), 223–231. DOI: https://doi.org/10.1016/0304-3975(78)90036-1

[54] 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. DOI: https://doi.org/10.1016/0022-0000(86)90006-1

[55] Sylvain Schmitz. 2016. The complexity of reachability in vector addition systems. *ACM SIGLOG News* 3, 1 (2016), 4–21. Retrieved from https://dl.acm.org/citation.cfm?id=2893585

[56] Sylvain Schmitz and Lia Schütze. 2024. On the length of strongly monotone descending chains over ℕ^d. In *Proceedings of the 51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024, Tallinn, Estonia (LIPIcs)*, Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson (Eds.), Vol. 297. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 153:1–153:19. DOI: https://doi.org/10.4230/LIPICS.ICALP.2024.153

[57] Michael Sipser. 1996. Introduction to the theory of computation. *ACM Sigact News* 27, 1 (1996), 27–29.

[58] Leslie G. Valiant and Mike Paterson. 1975. Deterministic one-counter automata. *J. Comput. Syst. Sci.* 10, 3 (1975), 340–350. DOI: https://doi.org/10.1016/S0022-0000(75)80005-5

[59] Wil M. P. van der Aalst. 1997. Verification of workflow nets. In *Proceedings of the Application and Theory of Petri Nets 1997, 18th International Conference, ICATPN '97, Toulouse, France, June 23-27, 1997, Proceedings (Lecture Notes in Computer Science)*, Pierre Azéma and Gianfranco Balbo (Eds.), Vol. 1248. Springer, 407–426. DOI: https://doi.org/10.1007/3-540-63139-9_48

[60] Raphael Yuster and Uri Zwick. 2004. Detecting short directed cycles using rectangular matrix multiplication and dynamic programming. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, New Orleans, Louisiana, USA, January 11-14, 2004*, J. Ian Munro (Ed.). SIAM, 254–260. Retrieved from http://dl.acm.org/citation.cfm?id=982792.982828

[61] Don Zagier. 1997. Newman's short proof of the prime number theorem. *Am. Math. Mon.* 104, 8 (1997), 705–708.

Received 19 March 2024; revised 8 April 2025; accepted 29 June 2025