





A Note on the Parameterised Complexity of Coverability in Vector Addition Systems

Michał Pilipczuk   

Institute of Informatics, University of Warsaw, Poland

Sylvain Schmitz   

Université Paris Cité, CNRS, IRIF, France

Henry Sinclair-Banks   

Institute of Informatics, University of Warsaw, Poland

Abstract

We investigate the parameterised complexity of the classic coverability problem for vector addition systems (VAS): given a finite set of vectors $V \subseteq \mathbb{Z}^d$, an initial configuration $s \in \mathbb{N}^d$, and a target configuration $t \in \mathbb{N}^d$, decide whether starting from s , one can iteratively add vectors from V to ultimately arrive at a configuration that is larger than or equal to t on every coordinate, while not observing any negative value on any coordinate along the way. We consider two natural parameters for the problem: the dimension d and the size of V , defined as the total bitsize of its encoding. We present several results charting the complexity of those two parameterisations, among which the highlight is that coverability for VAS parameterised by the dimension and with all the numbers in the input encoded in unary is complete for the class XNL under PL-reductions. We also discuss open problems in the topic, most notably the question about fixed-parameter tractability for the parameterisation by the size of V .

2012 ACM Subject Classification Theory of computation \rightarrow Parameterized complexity and exact algorithms; Software and its engineering \rightarrow Model checking; Theory of computation \rightarrow Concurrency

Keywords and phrases vector addition system, Petri net, parameterised complexity, coverability

Digital Object Identifier 10.4230/LIPIcs.IPEC.2025.24

Related Version *Full Version*: <https://arxiv.org/abs/2511.19212>

Funding *Michał Pilipczuk*: This work is a part of project BOBR that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 948057).

Henry Sinclair-Banks: This work is a part of project INFSYS that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 950398).

Acknowledgements This research was initiated during the InfAut 2024 workshop in Warlity Wielkie, Poland. The authors thank the organisers for launching such a fruitful event. They also thank an anonymous reviewer for pointing them to a relevant discussion in Hack's PhD thesis [26].

1 Introduction

Vector Addition Systems. Vector addition systems are a well-established model with a very simple definition: a d -dimensional *vector addition system* (VAS) [31] is a finite set V of vectors in \mathbb{Z}^d , which defines a step relation between configurations in \mathbb{N}^d : $u \rightarrow_V u + v$ for all u in \mathbb{N}^d and v in V , provided that $u + v$ is in \mathbb{N}^d . One usually interprets the d coordinates of a configuration $u \in \mathbb{N}^d$ as d *counters* that may take non-negative integer values; then V represents a set of allowed updates to the counter values, which can be applied only if none of the counters becomes negative.



© Michał Pilipczuk, Sylvain Schmitz, and Henry Sinclair-Banks;
licensed under Creative Commons License CC-BY 4.0

20th International Symposium on Parameterized and Exact Computation (IPEC 2025).

Editors: Akanksha Agrawal and Erik Jan van Leeuwen; Article No. 24; pp. 24:1–24:19

Leibniz International Proceedings in Informatics



Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



In spite of this apparent simplicity, vector addition systems can exhibit highly complex behaviours. Famously, their *reachability problem* is ACKERMANN-complete [35, 34, 13]: given as input a d -dimensional VAS V , an initial configuration $s \in \mathbb{N}^d$, and a target configuration $t \in \mathbb{N}^d$, the reachability problem asks whether t is reachable from $s \in \mathbb{N}^d$ in V , i.e., whether $s \rightarrow_V^* t$. This combination of a simple definition with rich behaviours makes vector addition systems – along with the equivalent model of Petri nets – well-suited whenever one needs to model systems managing multiple discrete resources, e.g., threads in concurrent computations, molecules in chemical reactions [3, 4], organisms in biological processes [5], etc., but also as a theoretical tool involved in establishing decidability and complexity statements for a variety of decision problems in logic, formal languages, verification, etc. [42, Section 5].

The Coverability Problem. In this note, we are interested in a decision problem with a less extreme complexity: given the same input, the *coverability problem* asks whether there exists a coordinate-wise larger or equal configuration $t' \sqsupseteq t$ such that $s \rightarrow_V^* t'$. This relaxation of the reachability problem was first shown decidable by Karp and Miller [31], before being proven EXPSPACE-hard by Lipton [36] and to belong to EXPSPACE by Rackoff [40].

Like reachability, coverability in vector addition systems inter-reduces with numerous decision problems, notably in relation to the automated verification of safety properties in concurrent or distributed systems [30, 29, 22, 16, 18, 23, 2, 6], as well as reasoning on data-aware logics or systems [15, 25, 1]. These applications have motivated a thorough investigation of the problem [31, 36, 40, 9, 33, 32, 44] and the development of several tools specifically targeted at solving it [19, 8, 24].

Towards Parameterised Complexity. Rosier and Yen [41] refined Rackoff’s analysis to identify the contribution of several parameters to the final complexity of the coverability problem. Among those, the main parameter driving the complexity is the *dimension*, i.e., the number of counters of the system. Indeed, Rackoff obtains his result by showing a bound on the length of the shortest covering runs from the source to the target configuration, which is in $n^{2^{\mathcal{O}(d \log d)}}$ when parameterised by the dimension d , where n is the size of the input encoded in unary. (While EXPSPACE-completeness holds with both a unary and a binary encoding, the complexity landscape changes as soon as one isolates the dimension as a parameter).

This upper bound on the length of shortest covering runs in unary-encoded VAS was recently improved to $n^{2^{\mathcal{O}(d)}}$ by Künnemann, Mazowiecki, Schütze, Sinclair-Banks, and Węgrzycki [32, Theorem 3.3], and this matches the $n^{2^{\Omega(d)}}$ lower bound in the family of systems constructed by Lipton [36]. In turn, this upper bound on the length of the shortest covering runs entails that the coverability problem with a unary encoding can be solved either in non-deterministic space $2^{\mathcal{O}(d)} \cdot \log n$ or in deterministic time $n^{2^{\mathcal{O}(d)}}$ [32, Corollary 3.4]; moreover, the latter time bound is also achieved by the classical *backward coverability algorithm* [44, Corollary 4.5]. Finally, as also shown by Künnemann et al. through a parameterised reduction from the parameterised clique problem **p-CLIQUE**, this time bound is optimal if one assumes the Exponential Time Hypothesis: under the ETH, no algorithm running in deterministic time $n^{o(2^d)}$ for coverability may exist [32, Theorem 4.2].

Surprisingly, in spite of this long line of results focusing on coverability in vector addition systems, and in particular when isolating the dimension as a key parameter, the question of the parameterised complexity of the problem has not been considered (see Section 2.4 for literature on other parameterisations on Petri nets, which are not directly comparable).

► **Problem.** $\text{p-dim-COVERABILITY(VAS)}$

input a d -dimensional VAS V , an initial configuration $s \in \mathbb{N}^d$, and a target configuration $t \in \mathbb{N}^d$

parameter d

question does s cover t in V , i.e., does there exist $t' \sqsupseteq t$ such that $s \rightarrow_V^* t'$?

From the viewpoint of parameterised complexity, when using a unary encoding, the $n^{2^{O(d)}}$ deterministic time bounds from [32, 44] immediately yield that $\text{p-dim-COVERABILITY(VAS)}$ is in XP, while the parameterised reduction from p-CLIQUE in [32] shows W[1]-hardness (see Figure 1 for an overview of the parameterised complexity classes discussed in this note).

Fixed Systems and Parameterisation by Size. A long-standing open question on decision problems for VAS, already raised by Hack [26, page 172] and revived in recent years [28, 17, 12], is whether any meaningful statements can be made about the complexity for a *fixed* VAS. In this direction, Draghici, Haase, and Ryzhikov [17] have recently shown that there exists a fixed V such that the coverability problem (and thus also the reachability problem) for V with a binary encoding of the initial/target configurations is already PSPACE-hard. This complexity of coverability in a fixed VAS is also related to a question on the length of shortest covering runs by Czerwiński [12]. A natural way to approach these questions within the framework of parameterised complexity is to ask about the complexity of coverability when parameterised by the *size* of the encoding of the system, which we denote by $\|V\|$.

► **Problem.** $\text{p-size-COVERABILITY(VAS)}$

input a d -dimensional VAS V , an input configuration $s \in \mathbb{N}^d$, and a target configuration $t \in \mathbb{N}^d$

parameter $\|V\|$

question does s cover t in V , i.e., does there exist $t' \sqsupseteq t$ such that $s \rightarrow_V^* t'$?

This is an a priori easier problem than the one parameterised by the dimension, as there is a straightforward parameterised reduction from this problem to the one of coverability parameterised by dimension (see Remark 2.1).

Contributions. Our main contribution in this note is to refine the bounds and pinpoint the exact parameterised complexity of $\text{p-dim-COVERABILITY(VAS)}$: the problem is XNL-complete under PL reductions with a unary encoding (see Section 3), and para-PSPACE-complete under FPT reductions with a binary encoding (see Section 4). These results are mainly applications of the rich literature dedicated to the coverability problem in vector addition systems: the upper bounds stem from the $2^{O(d)} \cdot \log n$ non-deterministic space bounds from Rackoff's and subsequent works [40, 41, 32], while para-PSPACE-hardness with a binary encoding follows from the PSPACE-hardness of coverability in a fixed VAS [17, Corollary 2]. The XNL lower bound is a new proof, which relies on a result of Wehar [46] on the intersection non-emptiness problem for finite automata. Importantly given the dearth of “natural” complete problems for these two parameterised complexity classes and in particular for XNL, these results significantly enrich the library of known complete problems, as most of the inter-reducible problems we mentioned earlier also have natural parameters that correspond to the dimension.

As a consequence of these results, we also conclude that $\text{p-size-COVERABILITY(VAS)}$ is para-PSPACE-complete when using a binary encoding (see Section 4), and in XNL when using a unary encoding (see Section 3). Our motivation here is to restate the questions about the complexity of coverability and reachability in fixed VAS [28, 17, 12] within the framework of parameterised complexity, which we discuss more extensively in Section 5.

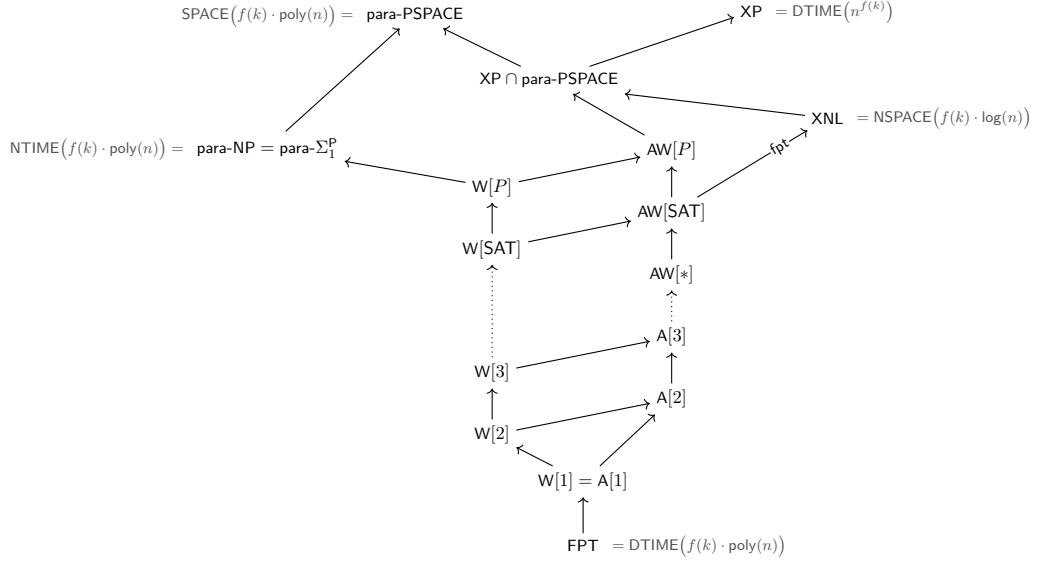


Figure 1 The parameterised complexity classes considered in this note and their known relations to the W- and A-hierarchies. Arrows $C \rightarrow D$ denote inclusions $C \subseteq D$; an arrow $C \xrightarrow{\text{fpt}} D$ denotes an inclusion $C \subseteq [D]^{\text{fpt}}$ into the closure of D under FPT reductions.

2 Preliminaries

2.1 Parameterised Problems, Classes, and Reductions

Let Σ be a finite alphabet. While a decision problem is just a language $L \subseteq \Sigma^*$, a *parameterised problem* is a set $P \subseteq \Sigma^* \times \mathbb{N}$. If the pair $(x, k) \in \Sigma^* \times \mathbb{N}$ is an instance of a parameterised problem, we refer to x as the *input* to the problem and we refer to k as the *parameter*. We use the shorthand $n \stackrel{\text{def}}{=} |x|$ for the length of x , whenever this does not create confusion.

The framework of parameterised complexity has led to rich hierarchies of complexity classes, notably refining the distinction between P and NP in the presence of parameters through the class FPT of fixed-parameter tractable problems, along with classes of intractable parameterised problems: the W- and A-hierarchies, and the complexity classes para-NP and XP; see Figure 1 for a depiction of these classes. A broad introduction to parameterised complexity theory can be found in the book of Flum and Grohe [21], see also the survey of de Haan and Szeider [14] for a comprehensive overview of parameterised classes of high complexity. We provide below a basic recollection of material that will be relevant to us, based on these two sources.

Parameterised Reductions. An *FPT reduction* (respectively a *PL reduction*, which stands for *parameterised logspace*) from the parameterised problem $P_1 \subseteq \Sigma_1^* \times \mathbb{N}$ to the parameterised problem $P_2 \subseteq \Sigma_2^* \times \mathbb{N}$ is a mapping $R: \Sigma_1^* \times \mathbb{N} \rightarrow \Sigma_2^* \times \mathbb{N}$ such that

- (1) for all $(x, k) \in \Sigma_1^* \times \mathbb{N}$, we have $(x, k) \in P_1$ if and only if $R(x, k) \in P_2$;
- (2) there is a computable function $f: \mathbb{N} \rightarrow \mathbb{N}$ and a constant $c \in \mathbb{N}$ such that R is computable in time $f(k) \cdot n^c$ (respectively computable in space $f(k) + c \cdot \log(n)$); and
- (3) there is a computable function $g: \mathbb{N} \rightarrow \mathbb{N}$ such that for all $(x, k) \in \Sigma_1^* \times \mathbb{N}$, say with $R(x, k) = (x', k')$, we have $k' \leq g(k)$.

Parameterised Space Complexity Classes. Although these classes are perhaps less well-known, the parameterised paradigm can be also applied to space complexity.

Regarding nondeterministic logarithmic space complexity, a parameterised problem $P \subseteq \Sigma \times \mathbb{N}$ is in (uniform) XNL [11, Proposition 18] if there is a computable function $f: \mathbb{N} \rightarrow \mathbb{N}$ and a non-deterministic algorithm that, given a pair $(x, k) \in \Sigma^* \times \mathbb{N}$, decides whether $(x, k) \in P$ in space at most $f(k) \cdot \log(n)$. In particular, AW[SAT] is included in the closure of XNL under FPT reductions by [11, Proposition 23]. When working with parameterised complexity classes like FPT, XP, para-PSPACE, etc., FPT reductions suffice, but for XNL, one needs to work with PL reductions. When stating completeness results, we always specify what type of reductions we have in mind.

Regarding polynomial space complexity, a parameterised problem $P \subseteq \Sigma^* \times \mathbb{N}$ is in para-PSPACE if there is a computable function $f: \mathbb{N} \rightarrow \Sigma^*$ and a problem $L \subseteq \Sigma^*$ such that $L \in \text{PSPACE}$ and for all instances $(x, k) \in \Sigma^* \times \mathbb{N}$, we have $(x, k) \in P$ if and only if $(x, f(k)) \in L$. Thus, para-PSPACE consists of all problems that can be decided using at most polynomial space after some precomputation that only involves the parameter. One can also view para-PSPACE as the space complexity-concerned analogue of FPT. In fact, it is easy to prove (see [21, Exercise 8.40]) that a parameterised problem P is in para-PSPACE if and only if it can be solved in FPT space; that is, in space bounded by $f(k) \cdot n^c$ for some computable function $f: \mathbb{N} \rightarrow \mathbb{N}$ and $c \in \mathbb{N}$. In order to establish para-PSPACE-hardness under FPT reductions, it suffices to prove that there exists some fixed value of the parameter k for which the problem is PSPACE-hard under polynomial-time reductions (see [21, Corollary 2.16]).

2.2 Vector Addition Systems and Related Models

As explained in the introduction, vector addition systems are equivalent to a number of models; in particular, *vector addition systems with states* are more convenient in order to prove complexity lower bounds, while *Petri nets* are better suited for modelling concurrent systems. This section succinctly introduces vector addition systems with states and Petri nets and here we argue that the usual reduction between their versions of the coverability problem holds also in the parameterised setting.

Basic Notation. We use bold font for vectors. We index the i -th component of a vector \mathbf{v} with square brackets by writing $\mathbf{v}[i]$. Given two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{Z}^d$, we write $\mathbf{u} \sqsubseteq \mathbf{v}$ if $\mathbf{u}[i] \leq \mathbf{v}[i]$ for all $i \in \{1, \dots, d\}$. Given a vector $\mathbf{v} \in \mathbb{Z}^d$, we define $\|\mathbf{v}\|_1 \stackrel{\text{def}}{=} |\mathbf{v}[1]| + \dots + |\mathbf{v}[d]|$ and $\|\mathbf{v}\|_\infty \stackrel{\text{def}}{=} \max\{|\mathbf{v}[1]|, \dots, |\mathbf{v}[d]|\}$. When working with unary encodings, we define the size of a vector $\mathbf{v} \in \mathbb{Z}^d$ as $\text{size}(\mathbf{v}) \stackrel{\text{def}}{=} \|\mathbf{v}\|_1$, and when using a binary encoding, as $\text{bitsize}(\mathbf{v}) \stackrel{\text{def}}{=} d \cdot (\log_2(\|\mathbf{v}\|_\infty + 1))$. We write $\|\mathbf{v}\|$ when the encoding is implicit. Then the size of a vector addition system \mathbf{V} in either encoding is $\|\mathbf{V}\| \stackrel{\text{def}}{=} \sum_{\mathbf{v} \in \mathbf{V}} \|\mathbf{v}\|$, thus defining $\text{size}(\mathbf{V})$ when encoded in unary and $\text{bitsize}(\mathbf{V})$ when encoded in binary. Note that in both cases, we have $\|\mathbf{V}\| \geq d$ in a non-trivial VAS \mathbf{V} of dimension d .

► **Remark 2.1.** For an input VAS \mathbf{V} of some dimension d , an initial configuration \mathbf{s} , and a target configuration \mathbf{t} , the map from $(\langle \mathbf{V}, \mathbf{s}, \mathbf{t} \rangle, \|\mathbf{V}\|)$ to $(\langle \mathbf{V}, \mathbf{s}, \mathbf{t} \rangle, d)$ is a PL reduction (and thus also an FPT reduction) from the p-size-COVERABILITY(VAS) problem to the p-dim-COVERABILITY(VAS) problem in both a unary and a binary encoding.

Vector Addition Systems with States. A d -dimensional *vector addition system with states* (d -VASS) [27] is a pair $\mathcal{V} = (Q, T)$ consisting of a finite set of states Q and a finite set of transitions $T \subseteq Q \times \mathbb{Z}^d \times Q$. To discuss the complexity of decision problems for VASS, we define the size of a VASS $\mathcal{V} = (Q, T)$ as $\|\mathcal{V}\| \stackrel{\text{def}}{=} |Q| + \sum_{(p, \mathbf{x}, q) \in T} \|\mathbf{x}\|$.

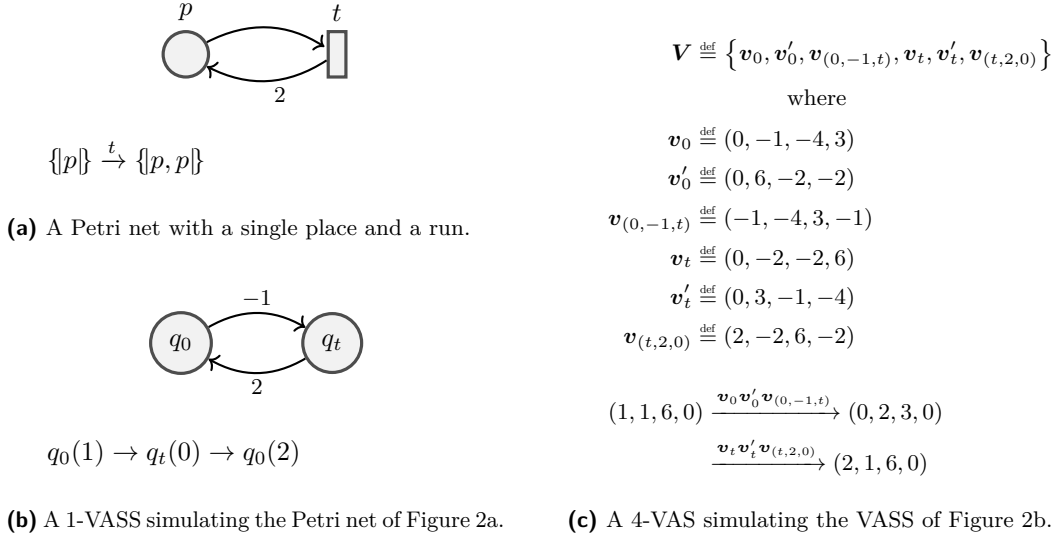


Figure 2 A Petri net, its representation as a VASS, and its representation as a VAS, along with a run in all three systems.

Each index $i \in \{1, \dots, d\}$ can be seen as a counter with a valuation in \mathbb{N} : 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 the current counter values \mathbf{v} ; we use the notation $q(\mathbf{v})$ for configurations. Given two configurations $p(\mathbf{u})$ and $q(\mathbf{v})$, there is a step $p(\mathbf{u}) \rightarrow q(\mathbf{v})$ if there exists a transition $t = (p, \mathbf{x}, q) \in T$ such that $\mathbf{u} + \mathbf{x} = \mathbf{v}$; we may refer to \mathbf{x} as the *update* of the transition $t = (p, \mathbf{x}, q)$, and will also write $p(\mathbf{u}) \xrightarrow{t} q(\mathbf{v})$ to emphasise that transition t was taken from $p(\mathbf{u})$ to $q(\mathbf{v})$.

A *path* in a VASS is a sequence of transitions $((p_1, \mathbf{x}_1, q_1), \dots, (p_m, \mathbf{x}_m, q_m))$ such that $p_{i+1} = q_i$ for every $i \in \{1, \dots, m-1\}$; then its *length* $|\pi| \stackrel{\text{def}}{=} m$ is its number of transitions. A *run* in a VASS is a sequence of configurations $(q_0(\mathbf{v}_0), \dots, q_m(\mathbf{v}_m))$ such that, for every $i \in \{1, \dots, m\}$, $q_{i-1}(\mathbf{v}_{i-1}) \rightarrow q_i(\mathbf{v}_i)$; in other words, $(q_{i-1}, \mathbf{v}_i - \mathbf{v}_{i-1}, q_i) \in T$. Let $\pi = (t_1, \dots, t_m)$ be a path and let $(q_0(\mathbf{v}_0), \dots, q_m(\mathbf{v}_m))$ be a run such that, for every $i \in \{1, \dots, m\}$, $q_{i-1}(\mathbf{v}_{i-1}) \xrightarrow{t_i} q_i(\mathbf{v}_i)$. Then we write $q_0(\mathbf{v}_0) \xrightarrow{\pi} q_m(\mathbf{v}_m)$ to denote that the run is performed along the path π , and $q_0(\mathbf{v}_0) \xrightarrow{*} q_m(\mathbf{v}_m)$ if the precise path is not relevant.

The *coverability problem* for VASS takes as input a VASS \mathcal{V} , an initial configuration $p(\mathbf{u})$, and a target configuration $q(\mathbf{v})$, and asks whether there exists $\mathbf{v}' \sqsupseteq \mathbf{v}$ such that there is a run $p(\mathbf{u}) \xrightarrow{*} q(\mathbf{v}')$. We denote the parameterisations of this decision problem by **p-dim-COVERABILITY(VASS)** and **p-size-COVERABILITY(VASS)**.

A VAS can therefore be seen as a VASS with a single state, which is dropped entirely from the definition. Conversely, by a well-known result from the seventies by Hopcroft and Pansiot, one can simulate the states of a VASS at the cost of three extra dimensions in a VAS [27, Lemma 2.1]. For clarity, the obtained VAS has an equivalent reachability relation between configurations; a configuration $q(\mathbf{x})$ in the original VASS corresponds to configurations of the form $(\mathbf{x}, a_q, b_q, c_q)$ in the VAS, where a_q, b_q , and c_q are values bounded by $(|Q| + 1)^2$ that represent the state q , and such that $(a_p, b_p, c_p) \not\sqsubseteq (a_q, b_q, c_q)$ for $p \neq q$ throughout the computation. Each transition of the VASS is simulated by a sequence of three transitions of the VAS, the first two checking the current state and the last one performing the update. See Figure 2 for an illustration of this construction.

► **Fact 2.2** (c.f. [27, Lemma 2.1]). *The parameterised problems p-dim-COVERABILITY(VAS) and p-dim-COVERABILITY(VASS) are equivalent up to PL reductions, and so are the parameterised problems p-size-COVERABILITY(VAS) and p-size-COVERABILITY(VASS), all this both with a unary and with a binary encoding.*

Petri nets. A *Petri net* [38] is a tuple $\mathcal{N} = (P, T, W)$ where P is a finite set of places, T is a finite set of transitions, and $W: (P \times T) \cup (T \times P) \rightarrow \mathbb{N}$ is a (weighted) flow function. It defines a transition system with configurations in \mathbb{N}^P , i.e. multisets of places (aka *markings*) that can equivalently be seen as vectors in $\mathbb{N}^{|P|}$, and steps $\mathbf{m} \xrightarrow{t} \mathbf{m}'$ whenever $\mathbf{m}(p) \geq W(p, t)$ and $\mathbf{m}'(p) = \mathbf{m}(p) - W(p, t) + W(t, p)$ for all p in P . We write $\mathbf{m} \rightarrow_{\mathcal{N}}^* \mathbf{m}'$ if the marking \mathbf{m}' is reachable from the marking \mathbf{m} by a finite sequence of such steps. The *dimension* of a Petri net is $|P|$, its number of places; its *size* is $\|\mathcal{N}\| \stackrel{\text{def}}{=} |P| + |T| + \sum_{p \in P, t \in T} (\|W(p, t)\| + \|W(t, p)\|)$. The *coverability problem* for Petri nets takes as input a Petri net \mathcal{N} , an initial marking $\mathbf{m} \in \mathbb{N}^P$, and a target marking \mathbf{m}' , and asks whether there exists $\mathbf{m}'' \sqsupseteq \mathbf{m}'$ such that $\mathbf{m} \rightarrow_{\mathcal{N}}^* \mathbf{m}''$. We denote the parameterisations of this decision problem by p-dim-COVERABILITY(PN) and p-size-COVERABILITY(PN).

See Figure 2a on Page 6 for a depiction of a Petri net $(\{p\}, \{t\}, W)$ with $W(p, t) = 1$ and $W(t, p) = 2$ as a directed bipartite graph, with places represented by circles, transitions by rectangles, and arcs from places to transitions representing the input flow $W(p, t)$ and arcs from transitions to places representing the output flows $W(t, p)$; the absence of an arc denotes a flow of 0, and an arc without weight denotes a flow of 1.

As is well-known, a Petri net (P, T, W) can be encoded as an equivalent $|P|$ -dimensional VASS with $|T| + 1$ states (see Figure 2b on Page 6), and conversely a d -dimensional VAS \mathbf{V} can be encoded as an equivalent Petri net with d places and one transition per vector $\mathbf{v} \in \mathbf{V}$, with input flow the absolute values of the negative values in \mathbf{v} and output flow the positive values in \mathbf{v} . Thus we have the following equivalence in terms of parameterised problems.

► **Fact 2.3.** *The problems p-dim-COVERABILITY(VAS) and p-dim-COVERABILITY(PN) are equivalent up to PL reductions, and so are the problems p-size-COVERABILITY(VAS) and p-size-COVERABILITY(PN), all this both with a unary and with a binary encoding.*

2.3 The Complexity of Coverability

Upper Bounds. Regarding the complexity upper bounds, consider an input of the coverability problem for a d -dimensional VAS \mathbf{V} with initial configuration \mathbf{s} and target configuration \mathbf{t} , encoded in unary with size $n = \text{size}(\mathbf{V}) + \|\mathbf{s}\|_1 + \|\mathbf{t}\|_1$. The approach pioneered by Rackoff [40] to establish the EXPSpace upper bound is the following, and will also be pertinent for the discussion in Section 5. A finite sequence of steps $\mathbf{s} \rightarrow_{\mathbf{V}}^* \mathbf{t}'$ for some $\mathbf{t}' \sqsupseteq \mathbf{t}$ is a *covering run* or *coverability witness*; its length is its number of steps. What Rackoff showed is that, if there is a covering run, then there exists one of length bounded by $n^{2^{\mathcal{O}(d \log d)}}$. The best known (and actually optimal) upper bound for this length is the following.

► **Fact 2.4** ([32, Theorem 3.3]). *Shortest covering witnesses for an instance of coverability encoded in unary have length bounded by $n^{2^{\mathcal{O}(d)}}$.*

As a direct corollary, using algorithms that exhaustively explore the space of bounded configurations, one can determine the complexity of coverability in VAS.

► **Corollary 2.5** (cf. [32, Corollary 3.4]). *There exists both a deterministic $n^{2^{\mathcal{O}(d)}}$ -time algorithm and a non-deterministic $2^{\mathcal{O}(d)} \cdot \log(n)$ -space algorithm for coverability, where n is the size of the input encoded in unary.*

In the multi-parameter analysis of the complexity, one usually focuses on the deterministic algorithm, which yields that coverability can be solved in pseudo-polynomial deterministic time when the dimension is fixed. In the parameterised setting, this also yields that $\text{p-dim-COVERABILITY}(\text{VAS})$ with a unary encoding is in XP.

In this paper however, we are more interested in the other, non-deterministic algorithm, which we repeat from [32, Corollary 3.4] for the sake of completeness. The algorithm starts with the initial configuration and iteratively guesses the consecutive transitions in a run while keeping track of the current configuration and the total number of transitions applied so far (all the numbers are encoded in binary). If during this iteration, it encounters a configuration covering the target configuration, then it accepts, and if the bound on the length of $n^{2^{\mathcal{O}(d)}}$ provided by Fact 2.4 is exceeded without encountering a covering configuration, it rejects. The correctness of the algorithm follows directly from Fact 2.4. As for the space complexity, observe that every configuration encountered during the run will have all the counters bounded by $n^{2^{\mathcal{O}(d)}}$, and the same can be also said about the length counter. Hence, the space needed to store all the relevant information is bounded by $\mathcal{O}(d \cdot \log n^{2^{\mathcal{O}(d)}}) \leq 2^{\mathcal{O}(d)} \cdot \log n$.

Lower Bounds. Regarding the lower bounds, Lipton [36] was the first to show the EXPSPACE -hardness of the reachability problem, with a proof that also applies to coverability. The idea of the construction is to define by induction over d a family of VAS \mathbf{V}_d of dimension $\mathcal{O}(d)$ that builds up counter values bounded by 2^{2^d} , allowing to simulate zero tests on those counters. This leads to a simulation of a 3-counter machine with counter values bounded by 2^{2^d} , thereby proving EXPSPACE -hardness. By a slight modification of the construction, one also obtains the following counterpart to Fact 2.4.

► **Fact 2.6** (c.f. [9, Theorem 3]). *Shortest covering witnesses for an instance of coverability encoded in unary may have length as large as $n^{2^{\Omega(d)}}$.*

Lipton's construction was exploited as part of a parameterised reduction from p-CLIQUE to $\text{p-dim-COVERABILITY}(\text{VAS})$ [32, Theorem 4.2], showing the $\text{W}[1]$ -hardness of the problem.

2.4 Further Related Work

Watel, Weisser, and Barth [45] investigate an optimisation variant of the coverability problem in weighted Petri nets: each transition has a non-negative weight in $\mathbb{R}_{\geq 0}$, and the goal is to find a covering witness where the sum of weights is minimal. The parameters they consider are $\max_{p \in P, t \in T} W(p, t)$, $\max_{p \in P, t \in T} W(t, p)$ the maximal input and output flow weights, $\|\mathbf{m}'\|$ the size of the target marking, and the number of steps in the sought covering witness, none of which is comparable to our parameterisations by dimension or size of the Petri net.

Praveen [39] defines a graph associated with a Petri net \mathcal{N} whose vertex set is P – the set of places – and edges $\{p, p'\}$ whenever there exists a transition t with $W(p, t) > 0$ and $W(t, p') > 0$. Praveen then shows that the coverability problem parameterised by both the size of a vertex cover and by the maximal flow weight $\max\{W(p, t), W(t, p) \mid p \in P, t \in T\}$ is in para-PSPACE [39, Theorem 4.5]. While this is in essence a refinement of the parameterisation by the dimension $|P|$ of the Petri net (which bounds the size of a vertex cover), the added flow weight parameter makes this result not directly comparable.

3 Unary Encoding

This section is dedicated to proving the following theorem pinpointing the exact complexity of the coverability problem in vector addition systems, when parameterised by the dimension and using a unary encoding. This is a significant improvement over the XP upper bound and W[1]-hardness mentioned in [32].

► **Theorem 3.1.** *p-dim-COVERABILITY(VAS) is XNL-complete under PL reductions when using a unary encoding.*

Using the reduction from Remark 2.1, this shows that the same upper bound also applies to the parameterisation by size.

► **Corollary 3.2.** *p-size-COVERABILITY(VAS) is in XNL when using a unary encoding.*

The membership to XNL follows from Rackoff's upper bound, that is, Fact 2.4. The XNL lower bound, however, requires some attention. We prove it using a reduction from the intersection non-emptiness problem for deterministic finite automata, which is presented in Lemma 3.3 below. To state it, we first need to establish terminology related to automata.

Finite Automata. A *deterministic finite automaton* (DFA) $\mathcal{D} = (\Sigma, Q, T, q^{\text{init}}, F)$ over a finite alphabet Σ , consists of a finite set Q of states, a set $T \subseteq Q \times \Sigma \times Q$ of transitions, an initial state $q^{\text{init}} \in Q$, and a set $F \subseteq Q$ of *final states*, with the restriction that, for every state $p \in Q$ and letter $\sigma \in \Sigma$, there is at most one state $q \in Q$ such that $(p, \sigma, q) \in T$. For a transition $t = (p, \sigma, q)$; for a word $w = \sigma_1 \sigma_2 \cdots \sigma_n$ is a sequence $(q_0, \sigma_1, q_1), (q_1, \sigma_2, q_2) \cdots (q_{n-1}, \sigma_n, q_n)$; it is *accepting* if $q_n \in F$ is final. The *language* $L(\mathcal{D}) \subseteq \Sigma^*$ recognised by \mathcal{D} is the set of words for which there exists some accepting run. We shall assume that a DFA is specified by its adjacency matrix, and therefore that the *size* of a given DFA $\mathcal{D} = (\Sigma, Q, T, q^{\text{init}}, F)$ is $|\mathcal{D}| \stackrel{\text{def}}{=} |Q| \cdot |\Sigma|$.

Our interest for DFAs stems from their intersection non-emptiness problem, a well-known PSPACE-complete problem, which when parameterised by the number of DFAs is one the very few known XNL-complete problems in the literature [46, Corollary 3.5].

► **Problem.** p-INTERSECTION-NONEMPTINESS(DFA)

input k DFAs $\mathcal{D}_1, \dots, \mathcal{D}_k$ over the same alphabet Σ

parameter k

question is $\bigcap_{1 \leq i \leq k} L(\mathcal{D}_i) \neq \emptyset$, i.e., does there exist a word $w \in \Sigma^*$ with an accepting run in each \mathcal{D}_i ?

Reduction to Coverability. The lower bound in Theorem 3.1 follows from the PL reduction from p-INTERSECTION-NONEMPTINESS(DFA) to p-dim-COVERABILITY(VASS) captured by the following statement.

► **Lemma 3.3.** *Let $\mathcal{D}_1, \dots, \mathcal{D}_k$ be DFAs over a common alphabet Σ . One can compute a $2k$ -VASS \mathcal{V} , an initial configuration $p(\mathbf{u})$, and a target configuration $q(\mathbf{v})$ in space $\mathcal{O}(\log(k) + \log(|\mathcal{D}_1| + \dots + |\mathcal{D}_k|))$ such that $\bigcap_{1 \leq i \leq k} L(\mathcal{D}_i) \neq \emptyset$ if and only if there exists $\mathbf{v}' \geq \mathbf{v}$ and a run from $p(\mathbf{u})$ to $q(\mathbf{v}')$ in \mathcal{V} .*

Proof. Suppose $\mathcal{D}_i = (\Sigma, Q_i, T_i, q_i^{\text{init}}, F_i)$ for every $1 \leq i \leq k$. Without loss of generality, we shall assume that $s = |Q_1| = \dots = |Q_k|$. It will also be convenient for us to number the states; $Q_i = \{q_{i,1}, \dots, q_{i,s}\}$ and we may also assume without loss of generality that $q_i^{\text{init}} = q_{i,1}$.

Overview. We construct a VASS \mathcal{V} with $2k$ counters that come in k pairs: for each $i \in \{1, \dots, k\}$, there are counters x_i and y_i used to store the current state of automaton \mathcal{D}_i . Precisely, that the current state of \mathcal{D}_i is $q_{i,j}$, for some $1 \leq j \leq s$, is reflected by counter values $x_i = j$ and $y_i = s - j$. The construction of \mathcal{V} ensures that, except for some intermediate configurations, the following invariant is maintained: for each $i \in \{1, \dots, k\}$, we have $x_i + y_i = s$ and $x_i \geq 1$.

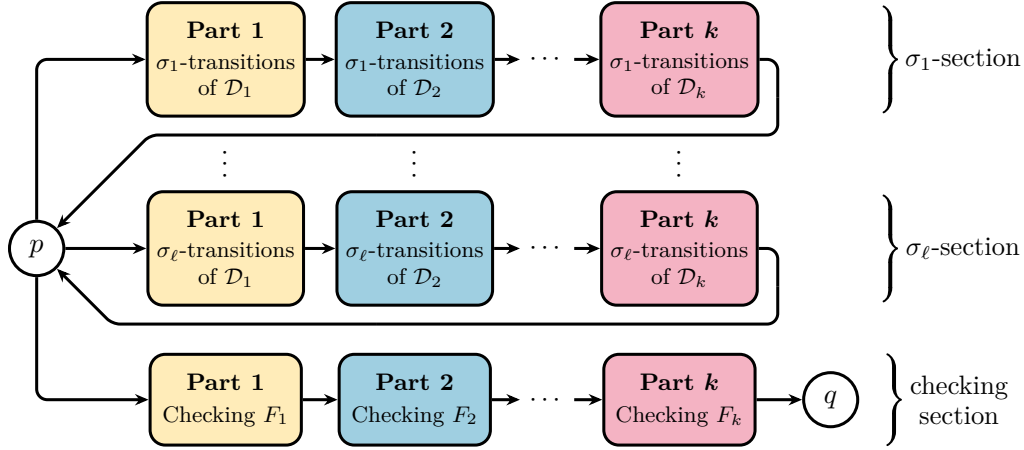
With this way of encoding of a selection of states of $\mathcal{D}_1, \dots, \mathcal{D}_k$, we emulate applying a transition of any \mathcal{D}_i through a pair of transitions in \mathcal{V} as follows. Suppose we would like to apply a transition of \mathcal{D}_i that goes from $q_{i,a}$ to $q_{i,b}$, for some $a, b \in \{1, \dots, s\}$: then, we may apply two transitions in \mathcal{V}

- first, one that subtracts a from x_i and $s - a$ from y_i , and
- then, one that adds b to x_i and $s - b$ to y_i .

Thanks to the invariant, the first transition can be fired only if we indeed have $x_i = a$ and $y_i = s - a$ (and the state encoded by those counters is $q_{i,a}$), and firing it brings both counters to 0. Then the second transition sets $x_i = b$ and $y_i = s - b$. Thus, firing the two transitions in \mathcal{V} both verifies that the current state is $q_{i,a}$, and updates this state to $q_{i,b}$.

With this basic gadget understood, we may explain the overall construction of \mathcal{V} , depicted in Figure 3. We create two special states: the starting state p and the ending state q . The initial configuration is $p(\mathbf{u})$ where $\mathbf{u} \stackrel{\text{def}}{=} (1, s - 1, \dots, 1, s - 1)$, which corresponds to setting every automaton \mathcal{D}_i to its initial state. Next, for each $\sigma \in \Sigma$ we construct a σ -section, which is essentially a loop that goes from p back to p , and a *checking section*, which is essentially a path from p to q . Thus, any run from p to q first loops some number of times through the σ -sections, and finally proceed through the checking section to q . The intention is that the effect of performing a loop through the σ -section, for some $\sigma \in \Sigma$, is an update of the values of the counters of \mathcal{V} that corresponds to performing a σ -transition in each of the automata $\mathcal{D}_1, \dots, \mathcal{D}_k$. This is done within k consecutive parts of the σ -section, each responsible for updating the state of a different \mathcal{D}_i using the gadgets described in the previous paragraph. Thus, choosing consecutive loops through the σ -sections corresponds to choosing consecutive letters of a word w over Σ , and simulating the runs of all the automata $\mathcal{D}_1, \dots, \mathcal{D}_k$ on w . Finally, we construct the checking section using a similar method so that it can be traversed if and only if each automaton \mathcal{D}_i is in a final state; this verifies that w is in the intersection of the languages of $\mathcal{D}_1, \dots, \mathcal{D}_k$. Again, the checking section consist of k parts, each responsible for checking the state of a different \mathcal{D}_i . See Section 3.1 for an illustration of this construction.

Formal Construction. We start by constructing the two special states p and q . Next, for every $\sigma \in \Sigma$ we construct the σ -section. It contains one part for each $i \in \{1, \dots, k\}$ defined as follows. Let $m_{\sigma,i}$ be the number of σ -transitions in \mathcal{D}_i . Then we introduce $m_{\sigma,i} + 2$ many states: one starting state, one ending state, and one intermediate state for each σ -transition in \mathcal{D}_i . Furthermore, for every σ -transition $t = (q_{i,a}, \sigma, q_{i,b})$ in \mathcal{D}_i , we construct two transitions in \mathcal{V} : the first transition from the starting state to the intermediate state q_t that updates the counters by subtracting a from x_i and $s - a$ from y_i (and does not update the other counters); the second transition from the intermediate state q_t to the ending state that updates the counters similarly by adding b to x_i and $s - b$ to y_i (and does not update the other counters). Finally, we connect all the parts of the σ -section into a path as follows by adding for each $i \in \{1, \dots, k - 1\}$, a counter-neutral (not changing the values of the counters) transition from the ending state of the i -th part to the starting state of the $(i + 1)$ -st part. The section is connected to the rest of the VASS by counter-neutral transitions from p to the starting state of the first part, and from the ending state of the last part to q .



■ **Figure 3** Overall structure of \mathcal{V} in the proof of Lemma 3.3.

Next, we present the construction of the checking section, which amounts to constructing the i -th part, for each $i \in \{1, \dots, k\}$, each with two states: one starting state and one ending state. Further, we construct $|F_i|$ many transitions: one for each final state of \mathcal{D}_i . Precisely, for each final state $q_{i,f}$ of \mathcal{D}_i (for some $f \in \{1, \dots, s\}$), we construct a transition in \mathcal{V} from the starting state to the ending state that subtracts f from x_i and $s - f$ from y_i (and does not update the other counters). It is important to note that this is only possible if $x_i = f$ and $y_i = s - f$ (i.e., when \mathcal{D}_i is currently at f). Accordingly, the only possible way to go is through the i -th part of the checking section is if \mathcal{D}_i is currently in a final state. We connect again all the parts of the checking section into a path by adding counter-neutral transitions from p to the starting state of the first part, from the ending state of the last part to q , and from the ending state of the i -th part to the starting state of the $(i + 1)$ -st part, for each $i \in \{1, \dots, k - 1\}$.

Finally, we set the initial configuration of \mathcal{V} to $p(\mathbf{u})$, where \mathbf{u} sets $x_i = 1$ and $y_i = s - 1$ for every $i \in \{1, \dots, k\}$, and the target configuration to $q(\mathbf{0})$ (so $\mathbf{v} = \mathbf{0}$). The following claim verifies the correctness of the reduction.

▷ **Claim 3.4.** The following conditions are equivalent.

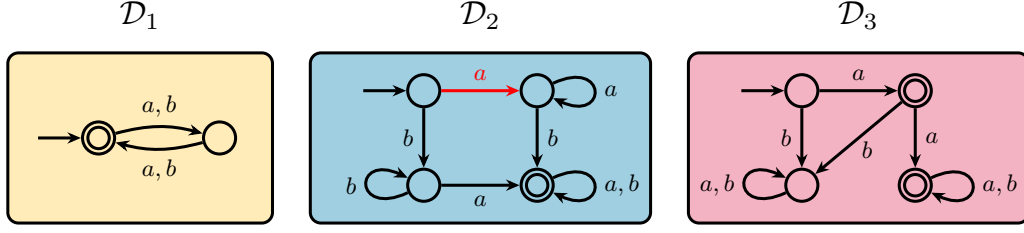
- There exists a word over Σ that is accepted by all the automata $\mathcal{D}_1, \dots, \mathcal{D}_k$.
- There is run of \mathcal{V} from configuration $p(\mathbf{u})$ to some configuration with state q .

The proof of Claim 3.4 follows easily from the construction along the lines sketched at the beginning of the proof; we give a more formal explanation in Section A.1.

Space Complexity. Let us finally observe that the VASS \mathcal{V} can be constructed in space $\mathcal{O}(\log(k) + \log(|\mathcal{D}_1| + \dots + |\mathcal{D}_k|))$, because the states and transitions of \mathcal{V} can be computed using three pointers

- $i \in \{1, \dots, k\}$ for the current automaton \mathcal{D}_i ,
- $\sigma \in \Sigma$ for the current letter, and
- $q_{i,a} \in Q_i$, which together with σ determines the transition $(q_{i,a}, \sigma, q_{i,b})$ in the deterministic automaton \mathcal{D}_i .

This sums up to $\mathcal{O}(\log(k \cdot (|\Sigma| \cdot |Q_1| + \dots + |\Sigma| \cdot |Q_k|)))$ space for the reduction; as furthermore $|\mathcal{D}_i| = |\Sigma| \cdot |Q_i|$ for each i , this is indeed the same as $\mathcal{O}(\log(k) + \log(|\mathcal{D}_1| + \dots + |\mathcal{D}_k|))$ space. ◀



■ **Figure 4** Three DFAs \mathcal{D}_1 , \mathcal{D}_2 , and \mathcal{D}_3 .

We may now conclude this section by proving Theorem 3.1.

Proof of Theorem 3.1. Membership in XNL follows from the algorithm in nondeterministic space $2^{\mathcal{O}(d)} \cdot \log n$ from Corollary 2.5 (c.f. [32, Corollary 3.4]). The XNL-hardness follows from Fact 2.2 and the PL reduction described by Lemma 3.3 along with the XNL-hardness of $\text{p-INTERSECTION-NONEMPTINESS(DFA)}$ proven by Wehar [46, Corollary 3.5]. ◀

3.1 Example

Let us illustrate the construction on the three DFAs over the alphabet $\{a, b\}$ of Figure 4, and construct a 6-VASS for which coverability between two specified configurations holds if and only if the intersection of the languages of the three DFAs is non-empty.

The languages of the three DFAs are the following.

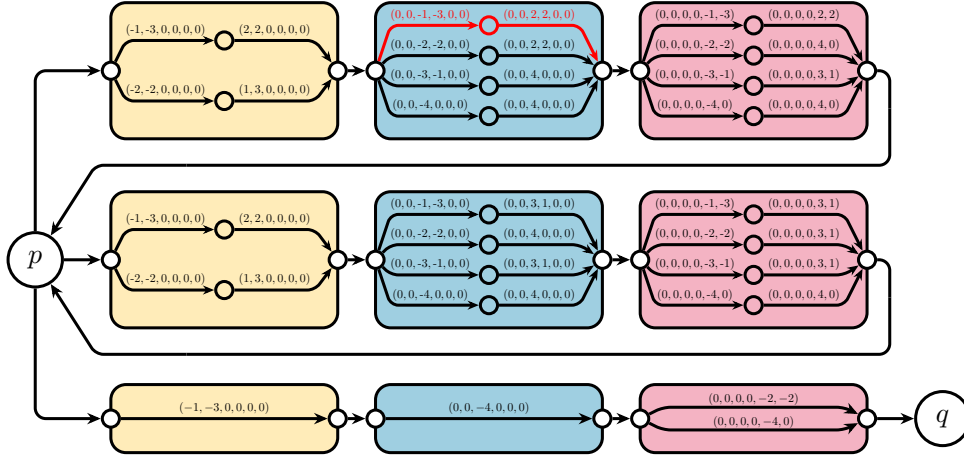
- \mathcal{D}_1 accepts all words of even length.
- \mathcal{D}_2 accepts all words that contain at least one “a” and at least one “b”.
- \mathcal{D}_3 accepts all non-empty words where the first and second letter is not “b”.

The intersection of these languages is not empty; the shortest words in the intersection of their languages are “aaab” and “aaba”.

The resulting 6-VASS \mathcal{V} is presented in Figure 5. We set $s = 4$ to be the greatest number of states of any of the DFAs. The VASS has three sections: the a -section (the top row of the figure), the b -section (the middle row of the figure), and the checking section (the bottom row of the figure). There are three parts in each section, one for each of the three DFAs.

First, let us consider the second part in the a -section; here there is one pair of transitions for each a -transition in \mathcal{D}_2 . For example, consider the transitions *coloured in red* in Figure 4 and Figure 5. The red pair of transitions in \mathcal{V} corresponds to the red a -transition from the first state (top left) to the second state (top right) in \mathcal{D}_2 . The two transitions in \mathcal{V} have the updates $(0, 0, -1, -3, 0, 0)$ and $(0, 0, 2, 2, 0, 0)$, respectively. The first red transition in \mathcal{V} checks if the current state of \mathcal{D}_2 is the first state ($x_2 = 1$ and $y_2 = s - 1 = 3$). The second red transition in \mathcal{V} then updates the counters so that the current state of \mathcal{D}_2 is the second state ($x_2 = 2$ and $y_2 = s - 2 = 2$).

Second, let us consider the third part of the checking section; here there are two transitions, one for each of the two final states of \mathcal{D}_3 . If the current counter values have $x_3 = 2$ and $y_3 = s - 2 = 2$ or the counter values have $x_3 = 4$ and $y_3 = s - 4 = 0$, then it is possible to pass through this part. Indeed, when the counter values of \mathcal{V} contain $x_3 = 2$ and $y_3 = 2$, then the current state of \mathcal{D}_3 is the second state (top right) which is a final state. Similarly, when the counter values of \mathcal{V} contain $x_3 = 4$ and $y_3 = 0$, then the current state of \mathcal{D}_3 is the fourth state (bottom right) which is also a final state.



■ **Figure 5** The 6-VASS \mathcal{V} constructed from the DFAs of Figure 4.

As the intersection of the languages recognised by the DFAs is non-empty, we know by Claim 3.4 that there is a run from $p(1, 3, 1, 3, 1, 3)$ to $q(0, 0, 0, 0, 0, 0)$. The run corresponding to the word $aaab$ follows a path of over 50 transitions; listing every intermediate configuration of the run is impractical, so we list below the main features of the run.

1. From $p(1, 3, 1, 3, 1, 3)$, pass through the a -section to reach $p(2, 2, 2, 2, 2, 2)$.
2. From $p(2, 2, 2, 2, 2, 2)$, pass through the a -section to reach $p(1, 3, 2, 2, 4, 0)$.
3. From $p(1, 3, 2, 2, 4, 0)$, pass through the a -section to reach $p(2, 2, 2, 2, 4, 0)$.
4. From $p(2, 2, 2, 2, 4, 0)$, pass through the b -section to reach $p(1, 3, 4, 0, 4, 0)$.
5. From $p(1, 3, 4, 0, 4, 0)$, pass through the checking section to finally reach $q(0, 0, 0, 0, 0, 0)$.

4 Binary Encoding

In this section, we show the **para-PSPACE-completeness** of the coverability problem parameterised either by dimension or size, when the input is encoded in binary. As in the case of a unary encoding, the upper bound follows from the $2^{O(d)} \cdot \log n$ non-deterministic space bounds from Rackoff's and subsequent works [40, 41, 32]. Regarding the lower bound, we rely on the PSPACE-hardness of coverability in a fixed VAS shown by Draghici, Haase, and Ryzhikov [17].

► **Theorem 4.1.** *The problems **p-dim-COVERABILITY(VAS)** and **p-size-COVERABILITY(VAS)** are para-PSPACE-complete under FPT reductions when using a binary encoding.*

Proof. Regarding the upper bound, recall that the coverability problem can be solved in non-deterministic space $2^{O(d)} \cdot \log n$ where d is the dimension and $n = \text{size}(\mathbf{V}) + \|\mathbf{s}\|_1 + \|\mathbf{t}\|_1$ is the size of the input in unary [32, Corollary 3.4]. This in turn provides an algorithm working in non-deterministic space $2^{O(d)} \cdot N$ where $N = \text{bitsize}(\mathbf{V}) + d(\log_2(\|\mathbf{s}\|_\infty + 1) + \log_2(\|\mathbf{t}\|_\infty + 1))$ is the size of the input in binary, showing that **p-dim-COVERABILITY(VAS)** is in **para-PSPACE**. By the parameterised reduction from Remark 2.1, this also shows that **p-size-COVERABILITY(VAS)** is in **para-PSPACE**.

Regarding the lower bound, [17, Corollary 2] shows that there exists a fixed VASS \mathcal{V}_0 (thus of fixed size, and actually of dimension 6) such that the coverability problem for \mathcal{V}_0 is PSPACE-hard under the assumption that the starting and the target configuration are

encoded in binary. By Fact 2.2, there is a fixed VAS V_0 (of dimension 9) with the same property, thus showing that $\text{p-size-COVERABILITY(VAS)}$ and $\text{p-dim-COVERABILITY(VAS)}$ are para-PSPACE-hard under FPT reductions.¹ ◀

A consequence is that, while coverability with either parameterisation is in XP when encoded in unary, if the problem with a binary encoding were to belong to XP, then $P = \text{PSPACE}$. Indeed, if this were the case, then coverability in the fixed VAS V_0 provided by [17, Corollary 2] would be in P.

5 Discussion and Open Problems

Our results shed some light on the broader landscape of the parameterised complexity of decision problems related to vector addition systems. Let us consider the coverability problem and the reachability problem; both have meaningful parameterisations either by dimension or by size. A first set of questions is simply whether the parameterised problem is in FPT or not. A second set of questions concerns bounding the length of witness runs, and is motivated by the discussion in Section 2.3, where we saw that this approach has been instrumental in the understanding of the complexity of coverability. Given an initial configuration s and a target configuration t in a VAS V , a *shortest witness* is a run $s \rightarrow_V^* t'$ for some $t' \sqsupseteq t$ in the case of coverability or a run $s \rightarrow_V^* t$ in the case of reachability, such that the number of steps in the run is minimal. For a parameter k (the dimension or the size of V), we will say that shortest witnesses have *fixed-parameter length* (FPT length) if there is a computable function f and constant $c \in \mathbb{N}$ such that shortest witnesses have length at most $f(k) \cdot n^c$.

This setup leads to four questions, that can each be asked with a parameterisation by dimension or by size, and using a unary or a binary encoding.

- (CP) Is the parameterised coverability problem in FPT?
- (RP) Is the parameterised reachability problem in FPT?
- (CL) Are shortest coverability witnesses of FPT length?
- (RL) Are shortest reachability witnesses of FPT length?

Negative Cases. Quite a few of these questions have negative answers. In the case of parameterisation by dimension, we have the following:

- A positive answer to (CP) would entail $\text{XNL} \subseteq \text{FPT}$ in the case of a unary encoding by Theorem 3.1, which would bring in particular a collapse of the W- and A-hierarchies down to FPT, because $\text{AW[SAT]} \subseteq [\text{XNL}]^{\text{fpt}}$ by [11, Proposition 23] (recall Figure 1). It would entail $\text{FPT} = \text{para-PSPACE}$ in the case of a binary encoding by Theorem 4.1, which is if and only if $P = \text{PSPACE}$.
- (CL) has an unconditional negative answer regardless of the choice of encoding by Fact 2.6: Lipton's construction [36] yields a family of inputs to the coverability problem in dimension $\mathcal{O}(d)$ where the shortest witnesses have length at least n^{2^d} .
- (RP) and (RL) have a negative answer regardless of the choice of encoding since the reachability problem in dimension $\mathcal{O}(d)$ is hard for the fast-growing complexity class \mathbf{F}_d [34, 13], and the shortest witnesses in their instances have length at least $F_d(n)$ where F_d is the d th fast-growing function (see [43] for a reference on fast-growing complexity).

¹ The $\text{para-PSPACE-hardness}$ of $\text{p-dim-COVERABILITY(VAS)}$ also follows from the fact that coverability in binary-encoded 2-VASS is PSPACE-hard [20, 7], which implies that coverability in binary-encoded 5-VAS is PSPACE-hard .

Turning our attention to parameterisation by size and a binary encoding, we have:

- A positive answer to either **(CP)** or **(RP)** would again entail $P = PSPACE$ by Theorem 4.1.
- Finally, **(CL)** and **(RL)** have a negative answer, since with a binary encoding and a fixed VAS of constant size, one can easily ensure that shortest witnesses have length exponential in the size of the input.

Open Questions. Our questions are therefore only relevant in the case of a unary encoding and a parameterisation by size. In that setting, **(CL)** is essentially Question 3 by Czerwiński [12]² and **(RL)** is essentially a conjecture due Hack [26, page 172] that has recently been restated by Jecker [28]. Both of these questions were originally phrased in terms of an $\mathcal{O}(n)$ upper bound in a fixed VAS; here we cast them in the terminology of parameterised complexity.

Those two questions are connected to **(CP)** and **(RP)**: an FPT length on shortest witnesses implies the existence of a nondeterministic algorithm that guesses and checks this witness. With a unary encoding, this entails that the problem is in **para-NP**, while with a binary encoding, this entails that the problem is in **para-PSPACE**. Recall that all we know at the moment are **XNL** and **para-PSPACE** upper bounds for coverability with unary and binary encoding, and no reasonable upper bounds for reachability. Thus **(CL)** and **(RL)** provide an original angle of attack on **(CP)** and **(RP)**, respectively, and a positive answer would yield

- the **para-PSPACE**-completeness of reachability parameterised by size with a binary encoding, and
- an indication that the complexity of the problems parameterised by size with a unary encoding is likely lower than **XNL** or **para-NP**.

In summary, we believe that **(CP)** and **(RP)** are excellent open questions in parameterised complexity that deserve further investigation, whereas **(CL)** and **(RL)** are auxiliary conjectures whose resolution might be very insightful for **(CP)** and **(RP)**.

As a further indication on the easiness of coverability and reachability parameterised by size with a unary encoding, note that in a *fixed* VAS V_0 , the coverability problem and the reachability problem are *sparse* languages, meaning that there exists a polynomial p such that, for all n , the number of instances x of the language of length $|x| = n$ is bounded by $p(n)$. Indeed, as V_0 and thus its dimension d are fixed, there are at most $\binom{n}{2d} \leq n^{2d}$ possible pairs of initial and target configurations such that $\|s\|_1 + \|t\|_1 = n$. This tells us that if there exists a fixed V_0 such that either coverability or reachability for V_0 with unary encoding is NP-hard, then $P = NP$ by Mahaney’s Theorem [37, Theorem 3.1]. Similarly, NL-hardness entails $L = NL$ [10, Corollary 3].

References

- 1 Parosh Aziz Abdulla, C. Aiswarya, Mohamed Faouzi Atig, Marco Montali, and Othmane Rezine. Complexity of reachability for data-aware dynamic systems. In *Proceedings of ACSD 2018*, pages 11–20. IEEE Computer Society, 2018. doi:10.1109/ACSD.2018.000-3.
- 2 Kaleb Alpernas, Aurojit Panda, Alexander Rabinovich, Mooly Sagiv, Scott Shenker, Sharon Shoham, and Yaron Velner. Some complexity results for stateful network verification. *Formal Methods in System Design*, 54:191–231, 2019. doi:10.1007/s10703-018-00330-9.

² Hack observed [26, page 171] that shortest coverability witnesses were of linear length if, in addition to the VAS, the initial configuration was fixed (thanks to the coverability tree construction [31]). On a similar note, Rackoff’s analysis [40] even yields a constant bound on the length of shortest coverability witnesses if we fix both the VAS and the target configuration.

- 3 Rutherford Aris. Prolegomena to the rational analysis of systems of chemical reactions. *Archive for Rational Mechanics and Analysis*, 19(2):81–99, 1965. doi:10.1007/BF00282276.
- 4 Rutherford Aris. Prolegomena to the rational analysis of systems of chemical reactions II. Some addenda. *Archive for Rational Mechanics and Analysis*, 27(5):356–364, 1968. doi:10.1007/BF00251438.
- 5 Paolo Baldan, Nicoletta Cocco, Andrea Marin, and Marta Simeoni. Petri nets for modelling metabolic pathways: A survey. *Natural Computing*, 9(4):955–989, 2010. doi:10.1007/S11047-010-9180-6.
- 6 Pascal Baumann, Rupak Majumdar, Ramanathan S. Thinniyam, and Georg Zetsche. Context-bounded verification of thread pools. In *Proceedings of POPL 2012*, volume 6 of *Proceedings of the ACM on Programming Languages*. ACM, 2022. doi:10.1145/3498678.
- 7 Michael Blondin, Alain Finkel, Stefan Göller, Christoph Haase, and Pierre McKenzie. Reachability in two-dimensional vector addition systems with states is PSPACE-complete. In *Proceedings of LICS 2015*, pages 32–43. IEEE Computer Society, 2015. doi:10.1109/LICS.2015.14.
- 8 Michael Blondin, Alain Finkel, Christoph Haase, and Serge Haddad. Approaching the coverability problem continuously. In *Proceedings of TACAS 2016*, volume 9636 of *Lecture Notes in Computer Science*, pages 480–496. Springer, 2016. doi:10.1007/978-3-662-49674-9_28.
- 9 Laura Bozzelli and Pierre Ganty. Complexity analysis of the backward coverability algorithm for VASS. In *Proceedings of RP 2011*, volume 6945 of *Lecture Notes in Computer Science*, pages 96–109. Springer, 2011. doi:10.1007/978-3-642-24288-5_10.
- 10 Jin-Yi Cai and D. Sivakumar. Resolution of Hartmanis’ conjecture for NL-hard sparse sets. *Theoretical Computer Science*, 240(2):257–269, 2000. doi:10.1016/S0304-3975(99)00234-0.
- 11 Yijia Chen, Jörg Flum, and Martin Grohe. Bounded nondeterminism and alternation in parameterized complexity theory. In *Proceedings of CCC 2003*, pages 13–29. IEEE Computer Society, 2003. doi:10.1109/CCC.2003.1214407.
- 12 Wojciech Czerwiński. Open problems in infinite-states systems, 2024. Webpage accessed on 2024-12-07. URL: <https://www.mimuw.edu.pl/~wczerwin/research.html>.
- 13 Wojciech Czerwiński and Łukasz Orlikowski. Reachability in vector addition systems is ACKERMANN-complete. In *Proceedings of FOCS 2021*, pages 1229–1240. IEEE Computer Society, 2022. doi:10.1109/FOCS52979.2021.00120.
- 14 Ronald de Haan and Stefan Szeider. A compendium of parameterized problems at higher levels of the polynomial hierarchy. *Algorithms*, 12(9):188, 2019. doi:10.3390/A12090188.
- 15 Normann Decker, Peter Habermehl, Martin Leucker, and Daniel Thoma. Ordered navigation on multi-attributed data words. In *Proceedings of CONCUR 2014*, volume 8704 of *Lecture Notes in Computer Science*, pages 497–511. Springer, 2014. doi:10.1007/978-3-662-44584-6_34.
- 16 Roberto Di Cosmo, Jacopo Mauro, Stefano Zacchiroli, and Gianluigi Zavattaro. Component reconfiguration in the presence of conflicts. In *Proceedings of ICALP 2013*, volume 7966 of *Lecture Notes in Computer Science*, pages 187–198. Springer, 2013. doi:10.1007/978-3-642-39212-2_19.
- 17 Andrei Draghici, Christoph Haase, and Andrew Ryzhikov. Reachability in fixed VASS: expressiveness and lower bounds. In *Proceedings of FoSSaCS 2024*, volume 14575 of *Lecture Notes in Computer Science*, pages 185–205. Springer, 2024. doi:10.1007/978-3-031-57231-9_9.
- 18 Javier Esparza. Keeping a crowd safe: On the complexity of parameterized verification. In *Proceedings of STACS 2014*, volume 25 of *Leibniz International Proceedings in Informatics*, pages 1–10. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2014. doi:10.4230/LIPIcs.STACS.2014.1.
- 19 Javier Esparza, Ruslán Ledesma-Garza, Rupak Majumdar, Philipp Meyer, and Filip Nikić. An SMT-based approach to coverability analysis. In *Proceedings of CAV 2014*, volume 8559 of *Lecture Notes in Computer Science*, pages 603–619. Springer, 2014. doi:10.1007/978-3-319-08867-9_40.

- 20 John Fearnley and Marcin Jurdziński. Reachability in two-clock timed automata is PSPACE-complete. *Information and Computation*, 243:26–36, 2015. doi:10.1016/j.ic.2014.12.004.
- 21 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. Springer, 2006. doi:10.1007/3-540-29953-X.
- 22 Pierre Ganty and Rupak Majumdar. Algorithmic verification of asynchronous programs. *ACM Transactions on Programming Languages and Systems*, 34(1), 2012. doi:10.1145/2160910.2160915.
- 23 Gilles Geeraerts, Alexander Heußner, and Jean-François Raskin. On the verification of concurrent, asynchronous programs with waiting queues. *ACM Transactions on Embedded Computing Systems*, 14(3), 2015. doi:10.1145/2700072.
- 24 Thomas Geffroy, Jérôme Leroux, and Grégoire Sutre. Occam’s Razor applied to the Petri net coverability problem. *Theoretical Computer Science*, 750:38–52, 2018. doi:10.1016/j.tcs.2018.04.014.
- 25 Radu Grigore and Nikos Tzevelekos. History-register automata. *Logical Methods in Computer Science*, 12(1), 2016. doi:10.2168/LMCS-12(1:7)2016.
- 26 Michel Hack. *Decidability questions for Petri Nets*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1976. URL: <https://hdl.handle.net/1721.1/27441>.
- 27 John Hopcroft and Jean-Jacques Pansiot. On the reachability problem for 5-dimensional vector addition systems. *Theoretical Computer Science*, 8(2):135–159, 1979. doi:10.1016/0304-3975(79)90041-0.
- 28 Ismaël Jecker. Open problems in automata theory: 22.1 Complexity of fixed VAS reachability, 2022. Webpage accessed on 2025-05-13. URL: <https://automata.exchange/22.01-complexity-fixed-vas-reachability/>.
- 29 Alexander Kaiser, Daniel Kroening, and Thomas Wahl. Dynamic cutoff detection in parameterized concurrent programs. In *Proceedings of CAV 2010*, volume 6174 of *Lecture Notes in Computer Science*, pages 645–659. Springer, 2010. doi:10.1007/978-3-642-14295-6_55.
- 30 Max Kanovich, Paul Rowe, and Andre Scedrov. Policy compliance in collaborative systems. In *Proceedings of CSF 2009*, pages 218–233. IEEE Computer Society, 2009. doi:10.1109/CSF.2009.19.
- 31 Richard M. Karp and Raymond E. Miller. Parallel program schemata. *Journal of Computer and System Sciences*, 3(2):147–195, 1969. doi:10.1016/S0022-0000(69)80011-5.
- 32 Marvin Künnemann, Filip Mazowiecki, Lia Schütze, Henry Sinclair-Banks, and Karol Węgrzycki. Coverability in VASS revisited: Improving Rackoff’s bound to obtain conditional optimality. In *Proceedings of ICALP 2023*, volume 261 of *Leibniz International Proceedings in Informatics*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ICALP.2023.131.
- 33 Ranko Lazić and Sylvain Schmitz. The ideal view on Rackoff’s coverability technique. *Information and Computation*, 277:104582, 2021. doi:10.1016/j.ic.2020.104582.
- 34 Jérôme Leroux. The reachability problem for Petri nets is not primitive recursive. In *Proceedings of FOCS 2021*, pages 1241–1252. IEEE Computer Society, 2022. doi:10.1109/FOCS52979.2021.00121.
- 35 Jérôme Leroux and Sylvain Schmitz. Reachability in vector addition systems is primitive-recursive in fixed dimension. In *Proceedings of LICS 2019*, pages 1–13. IEEE Computer Society, 2019. doi:10.1109/LICS.2019.8785796.
- 36 Richard J. Lipton. The reachability problem requires exponential space. Technical Report 62, Department of Computer Science, Yale University, January 1976. URL: <http://www.cs.yale.edu/publications/techreports/tr63.pdf>.
- 37 Stephen R. Mahaney. Sparse complete sets for NP: Solution of a conjecture of Berman and Hartmanis. *Journal of Computer and System Sciences*, 25(2):130–143, 1982. doi:10.1016/0022-0000(82)90002-2.
- 38 Carl A. Petri. *Kommunikation mit Automaten*. PhD thesis, Universität Bonn, 1962. URL: <http://edoc.sub.uni-hamburg.de/informatik/volltexte/2011/160/>.

- 39 M. Praveen. Small vertex cover makes Petri net coverability and boundedness easier. *Algorithmica*, 65(4):713–753, 2013. doi:10.1007/S00453-012-9687-6.
- 40 Charles Rackoff. The covering and boundedness problems for vector addition systems. *Theoretical Computer Science*, 6:223–231, 1978. doi:10.1016/0304-3975(78)90036-1.
- 41 Louis E. Rosier and Hsu-Chun Yen. A multiparameter analysis of the boundedness problem for vector addition systems. *Journal of Computer and System Sciences*, 32(1):105–135, 1986. doi:10.1016/0022-0000(86)90006-1.
- 42 Sylvain Schmitz. Automata column: The complexity of reachability in vector addition systems. *ACM SIGLOG News*, 3(1):3–21, 2016. doi:10.1145/2893582.2893585.
- 43 Sylvain Schmitz. Complexity hierarchies beyond ELEMENTARY. *ACM Transactions on Computation Theory*, 8(1), 2016. doi:10.1145/2858784.
- 44 Sylvain Schmitz and Lia Schütze. On the length of strongly monotone descending chains over \mathbb{N}^d . In *Proceedings of ICALP 2024*, volume 297 of *Leibniz International Proceedings in Informatics*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ICALP.2024.153.
- 45 Dimitri Watel, Marc-Antoine Weisser, and Dominique Barth. Parameterized complexity and approximability of coverability problems in weighted Petri nets. In *Proceedings of Petri Nets 2017*, volume 10258 of *Lecture Notes in Computer Science*, pages 330–349. Springer, 2017. doi:10.1007/978-3-319-57861-3_19.
- 46 Michael Wehar. *On the Complexity of Intersection Non-Emptiness Problems*. PhD thesis, University at Buffalo, State University of New York, USA, 2016. URL: http://www.michaelwehar.com/documents/mwehar_dissertation.pdf.

A

 Details for Section 3

A.1 Proof of Claim 3.4

We give here a formal verification of the correctness of the reduction; that is, we prove Claim 3.4, recalled below for convenience.

▷ **Claim 3.4.** The following conditions are equivalent.

- There exists a word over Σ that is accepted by all the automata $\mathcal{D}_1, \dots, \mathcal{D}_k$.
- There is run of \mathcal{V} from configuration $p(\mathbf{u})$ to some configuration with state q .

Proof. We verify the two implications in order.

First direction. We first argue that if there is a word in accepted all the automata $\mathcal{D}_1, \dots, \mathcal{D}_k$, then there exists a run in \mathcal{V} from $p(\mathbf{u})$ to $q(\mathbf{v})$ for some $\mathbf{v} \geq \mathbf{0}$. In fact, we will show that there is a run from $p(\mathbf{u})$ to $q(\mathbf{0})$.

Suppose $w \in \Sigma^*$ is a word that is accepted by \mathcal{D}_i for every $i \in \{1, \dots, k\}$. We shall use vector-indexing notation $(w[1], w[2], \dots)$ to refer to the letters that comprise w . We construct a run of \mathcal{V} starting from $p(\mathbf{u})$ as follows. First, for each consecutive $j \in \{1, \dots, |w|\}$, the run proceed through the $w[j]$ -section as follows. Suppose that after the first $j - 1$ letters have been read, the current state of \mathcal{D}_i is $q_{i,a}$ (for some $1 \leq a \leq s$). Since w is accepted by every automaton \mathcal{D}_i , there is some $w[j]$ -transition in \mathcal{D}_i that is taken; suppose it is the transition $t_i = (q_{i,a}, w[j], q_{i,b})$. Then when the run traverses the $w[j]$ -section from p back to p by using the pair of transitions that go through the intermediate state q_{t_i} , for each $i \in \{1, \dots, k\}$. It is possible to indeed take these transitions because if \mathcal{D}_i was in state $q_{i,a}$, then $x_i = a$ and $y_i = s - a$. Then, after t_i is taken in \mathcal{D}_i , the current state becomes $q_{i,b}$ which is reflected in the run in \mathcal{V} since, after the second transition in the pair is taken, $x_i = b$ and $y_i = s - b$ is achieved.

Lastly, once every letter $w[1], \dots, w[|w|]$ has been read, we know that the current state of \mathcal{D}_i is a final state, for every i . Precisely, suppose that the current state of each automaton \mathcal{D}_i is q_{i,f_i} , for some $1 \leq f_i \leq s$. Thus, in \mathcal{V} , the configuration $p(\mathbf{z})$ that is reached has the vector of counter values \mathbf{z} for which $x_i = f_i$ and $x_i = s - f_i$. This means that in each of the parts of the checking section, a transition can be taken to subtract f_i from x_i and $s - f_i$ from y_i . Thus, the constructed run can traverse the checking section and at the end reach configuration $q(\mathbf{0})$, as required.

Second direction. We now prove that if there is a run in \mathcal{V} from $p(\mathbf{u})$ to $q(\mathbf{v})$ for some $\mathbf{v} \geq \mathbf{0}$, then there exists a word that is accepted by each automaton \mathcal{D}_i , for $i \in \{1, \dots, k\}$.

Let π be a path such that $p(\mathbf{u}) \xrightarrow{\pi} q(\mathbf{v})$. We may split π into subpaths $\pi_1, \dots, \pi_m, \pi_{m+1}$ such that for every $j \in \{1, \dots, m\}$, π_j is a path from p back to p that only visits p at the start and the end, and π_{m+1} is a path from p to q . Due to the overall structure of \mathcal{V} (see Figure 3), we know that for all $j \in \{1, \dots, m\}$, π_j is a path inside a σ_j -section, for some $\sigma_j \in \Sigma$, whereas π_{m+1} traverses the checking section. We will argue that the word

$$w \stackrel{\text{def}}{=} \sigma_1 \sigma_2 \cdots \sigma_m$$

is accepted by every automaton \mathcal{D}_i , for $i \in \{1, \dots, k\}$.

For every $j \in \{1, \dots, m\}$, let $p(\mathbf{u}_j)$ be the intermediate configuration between π_j and π_{j+1} . That is, we have:

$$p(\mathbf{u}) \xrightarrow{\pi_1} p(\mathbf{u}_1) \xrightarrow{\pi_2} p(\mathbf{u}_2) \rightarrow \cdots \rightarrow p(\mathbf{u}_{m-1}) \xrightarrow{\pi_m} p(\mathbf{u}_m) \xrightarrow{\pi_{m+1}} q(\mathbf{v}).$$

Initially, in \mathbf{u} , we know that for all $i \in \{1, \dots, k\}$, the invariant $x_i + y_i = s$ and $x_i \geq 1$ holds, and thus x_i contains the index of a state of \mathcal{D}_i . By the design of \mathcal{V} , we deduce that this invariant holds true also for all intermediate configurations $p(\mathbf{u}_1), p(\mathbf{u}_2), \dots, p(\mathbf{u}_m)$.

Now, consider the sub-run $p(\mathbf{u}_j) \xrightarrow{\pi_j} p(\mathbf{u}_{j+1})$. In order to leave state p , go through the σ_j -section, and return to p , it must be the case that each part is successfully passed. Suppose that at in \mathbf{u}_j , it is true that $x_i = a$ (and, due to the invariant, we know that $y_i = s - a$). In order to pass the i -th part, it must be true that there must be a transition, say t , that subtracts a from x_i and $s - a$ from y_i . This is true because all transitions in the i -th part subtract x from x_i and $s - x$ from y_i for some $1 \leq x \leq s$. Inside the i -th part of the σ_j -section, after t is taken, there is a following transition, say t' , that adds b to x_i and $s - b$ to y_i for some $1 \leq b \leq s$. Now since t and t' are present in i -th part of the $\sigma_{f(j)}$ -section, then we know there is a transition $(q_{i,a}, \sigma_{f(j)}, q_{i,b})$ in \mathcal{D}_i . Indeed, the run taking t, t' is simulating \mathcal{D}_i using $(q_{i,a}, \sigma_j, q_{i,b})$ (and so \mathcal{D}_i has now read the j -th letter of the word). A straightforward induction now shows that for each $j \in \{1, \dots, m\}$, vector \mathbf{u}_j encodes, in its counters, the states in which the automata $\mathcal{D}_1, \dots, \mathcal{D}_k$ are after reading the first j letters of w .

We conclude by analysing the final sub-run $p(\mathbf{u}_m) \xrightarrow{\pi_{m+1}} q(\mathbf{v})$, and arguing that the state in which each automaton \mathcal{D}_i is after reading w must be a final state. This holds for the same reason used before: the transition that subtracts a from x_i and $s - a$ from y_i in the i -th part of the checking section is only present when $q_{i,a}$ is a final state of \mathcal{D}_i . Thus, in order to reach q , one transition from each part of the checking section must be taken, which can only be true when the current state of automaton \mathcal{D}_i is a final state. Thus, the word $w = \sigma_1 \sigma_2 \cdots \sigma_m$ is accepted by all the automata $\mathcal{D}_1, \dots, \mathcal{D}_k$. \triangleleft