The Complexity of Reachability in Affine Vector Addition Systems with States

Michael Blondin*
michael.blondin@usherbrooke.ca
Université de Sherbrooke
Sherbrooke, Québec, Canada

Mikhail Raskin†
raskin@in.tum.de
Technische Universität München
Munich, Bavaria, Germany

Abstract

Vector addition systems with states (VASS) are widely used for the formal verification of concurrent systems. Given their tremendous computational complexity, practical approaches have relied on techniques such as reachability relaxations, *e.g.*, allowing for negative intermediate counter values. It is natural to question their feasibility for VASS enriched with primitives that typically translate into undecidability. Spurred by this concern, we pinpoint the complexity of integer relaxations w.r.t. arbitrary classes of affine operations.

More specifically, we provide a trichotomy on the complexity of integer reachability in VASS extended with affine operations (affine VASS). Namely, we show that it is NP-complete for VASS with resets, PSPACE-complete for VASS with (pseudo-)transfers and VASS with (pseudo-)copies, and undecidable for any other class. We further present a dichotomy for standard reachability in affine VASS: it is decidable for VASS with permutations, and undecidable for any other class. This yields a complete and unified complexity landscape of reachability in affine VASS.

CCS Concepts: • **Theory of computation → Logic and verification**; **Complexity classes**; **Automata over infinite objects**.

Keywords: vector addition system, affine transformation, reachability, computational complexity

*Supported by a Discovery Grant from the Natural Sciences and Engineering Research Council of Canada (NSERC) and by the Fonds de recherche du Québec – Nature et technologies (FRQNT)
†Supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 787367 (PaVeS).

LICS '20, July 8–11, 2020, Saarbrücken, Germany

ACM ISBN 978-1-4503-7104-9/20/07…$15.00
https://doi.org/10.1145/3373718.3394741

ACM Reference Format:
Michael Blondin and Mikhail Raskin. 2020. The Complexity of Reachability in Affine Vector Addition Systems with States. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '20), July 8–11, 2020, Saarbrücken, Germany.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3373718.3394741

1 Introduction

Vector addition systems with states (VASS), which can equivalently be seen as Petri nets, form a widespread general model of infinite-state systems with countless applications ranging from the verification of concurrent programs to the modeling of biological, chemical and business processes (see, *e.g.*, [13, 18, 20, 23, 37]). They comprise a finite-state controller with counters ranging over $\mathbb{N}$ and updated via instructions of the form $x \leftarrow x + c$ which are executable if $x + c \geq 0$. The central decision problem concerning VASS is the *reachability problem*: given configurations x and y, is it possible to reach y starting from x? Such queries allow, *e.g.*, to verify whether unsafe states can be reached in concurrent programs. The notorious difficulty of the reachability problem led to many proofs of its decidability over the last decades [24–28, 32, 34]. While the problem has been known to be EXPSPACE-hard since 1976 [31], its computational complexity has remained unknown until very recently, where it was shown to be TOWER-hard [10] and solvable in Ackermannian time [29, 30].

Given the potential applications on the one hand, and the tremendously high complexity on the other hand, researchers have investigated relaxations of VASS in search of a tradeoff between expressiveness and algorithmic complexity. Two such relaxations consist in permitting either:

(a) transitions to be executed fractionally, and consequently counters to range over $\mathbb{Q}_{\geq 0}$ (*continuous reachability*); or
(b) counters to range over $\mathbb{Z}$ (*integer reachability*).

In both cases, the complexity drops drastically: continuous reachability is P-complete and NP-complete for Petri nets and VASS respectively [4, 17], while integer reachability is NP-complete for both models [9, 19]. Moreover, these two types of reachability have been used successfully to prove safety of real-world instances like multithreaded program skeletons, *e.g.*, see [2, 3, 14].

Although VASS are versatile, they are sometimes too limited to model common primitives. Consequently, their modeling power has been extended with various operations. For example, *(multi-)transfers*, *i.e.* operations of the form

$$x \leftarrow x + \sum_{i=1}^{n} y_i;\ y_1 \leftarrow 0;\ y_2 \leftarrow 0;\ \cdots;\ y_n \leftarrow 0,$$

allow, *e.g.*, for the verification of multi-threaded C and Java program skeletons with communication primitives [11, 23]. Another example is the case of *resets*, *i.e.* operations of the form $x \leftarrow 0$, which allow, *e.g.*, for the validation of some business processes [38], and the generation of program loop invariants [35]. Many such extensions fall under the generic family of *affine VASS*, *i.e.* VASS with instructions of the form $\boldsymbol{x} \leftarrow \mathbf{A} \cdot \boldsymbol{x} + \boldsymbol{b}$. As a general rule of thumb, reachability is undecidable for essentially any class of affine VASS introduced in the literature; in particular, for transfers and resets [1, 12].

Given the success of relaxations for the practical analysis of (standard) VASS, it is tempting to employ the same approach for affine VASS. Unfortunately, continuous reachability becomes undecidable for mild affine extensions such as resets and transfers. However, *integer reachability* was recently shown decidable for affine operations such as resets (NP-complete) and transfers (PSPACE-complete) [5, 19]. While such complexity results do not translate immediately into practical procedures, they arguably guide the design of algorithmic verification strategies.

Contribution. Thus, these recent results raise two natural questions: for *what classes* of affine VASS is integer reachability decidable? And, whenever it is decidable, what is its *exact* computational complexity? We fully answer these questions in this paper by giving a precise trichotomy: integer reachability is NP-complete for VASS with resets, PSPACE-complete for VASS with (pseudo-)transfers and VASS with (pseudo-)copies, and undecidable for *any* other class. In particular, this answers a question left open in [5]: integer reachability is undecidable for *any* class of affine VASS with *infinite* matrix monoid.

This clear complexity landscape is obtained by formalizing classes of affine VASS and by carefully analyzing the structure of arbitrary affine transformations; which could be of independent interest. In particular, it enables us to prove a dichotomy on *(standard) reachability* for affine VASS: it is decidable for VASS with permutations, and undecidable for *any* other class. To the best of our knowledge, this is the first proof of the folkore rule of thumb stating that "reachability is undecidable for essentially any class of affine VASS".

Related work. Our work is related to [5] which shows that integer reachability is decidable for affine VASS whose matrix monoid is finite; and more particularly PSPACE-complete in general for VASS with transfers and VASS with copies. While it is also recalled in [5] that integer reachability is undecidable in general for affine VASS, the authors do not provide any necessary condition for undecidability to hold. Moreover, the complexity landscape for affine VASS with finite monoids is left blurred, *e.g.*, it does not give necessary conditions for PSPACE-hardness results to hold, and the complexity remains unknown for monoids with negative coefficients. This paper completes the work initiated in [5] by providing a unified framework, which includes the notion of *matrix class*, that allows us to precisely characterize the complexity of integer reachability for *any* class of affine VASS.

Our work is also loosely related to a broader line of research on (variants of) affine VASS dealing with, *e.g.*, modeling power [36], accelerability [16], formal languages [8], coverability [7], and the complexity of integer reachability for restricted counters [15] and structures [22].

Structure of the paper. Section 2 introduces general notation and affine VASS. In Section 3, we prove our main result, namely the complexity trichotomy for integer reachability in affine VASS. In Section 4, we show a dichotomy on (standard) reachability in affine VASS. Finally, we conclude in Section 5. To avoid cluttering the presentation with too many technical details, some proofs are deferred to the appendix.

2 Preliminaries

Notation. Let $\mathbb{Z}, \mathbb{N}, [a, b]$ and $[k]$ denote respectively the sets $\{\ldots, -1, 0, 1, \ldots\}$, $\{0, 1, 2, \ldots\}$, $\{a, a+1, \ldots, b\}$ and $[1, k]$. For every pair of vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{Z}^k$, let $\boldsymbol{u} + \boldsymbol{v}$ be the vector $\boldsymbol{w} \in \mathbb{Z}^k$ such that $\boldsymbol{w}(i) \coloneqq \boldsymbol{u}(i) + \boldsymbol{v}(i)$ for every $i \in [k]$. Let $\boldsymbol{e}_i$ be the unit vector such that $\boldsymbol{e}_i(i) = 1$ and $\boldsymbol{e}_i(j) = 0$ for every $j \neq i$. We do not specify the arity of $\boldsymbol{e}_i$ as we will use it without ambiguity in various dimensions. For every square matrix $\mathbf{A} \in \mathbb{Z}^{k \times k}$, let $\dim \mathbf{A} \coloneqq k$ and let

$$\|\mathbf{A}\| \coloneqq \max\{|\mathbf{A}[i, j]| : i, j \in [k]\}.$$

We naturally extend the latter notation to any set X of matrices, *i.e.* $\|X\| \coloneqq \sup\{\|\mathbf{A}\| : \mathbf{A} \in X\}$. Throughout the paper, we will often refer to matrix and vector indices as *counters*. We will also often describe permutations in cycle notation, where elements are separated by semicolons for readability, *e.g.*, $(i; j)$ denotes the permutation that swaps i and j.

Affine VASS. An *affine vector addition system with states (affine VASS)* is a tuple $\mathcal{V} = (d, Q, T)$ where:

- $d \geq 1$ is the *number of counters* of $\mathcal{V}$;
- Q is a finite set of elements called *control-states*;
- $T \subseteq Q \times \mathbb{Z}^{d \times d} \times \mathbb{Z}^d \times Q$ is a finite set of elements called *transitions*.

For every transition $t = (p, \mathbf{A}, \boldsymbol{b}, q)$, let $\mathrm{src}(t) \coloneqq p$, $M(t) \coloneqq \mathbf{A}$, $\Delta(t) \coloneqq \boldsymbol{b}$ and $\mathrm{tgt}(t) \coloneqq q$. A *configuration* is a pair $(q, \boldsymbol{v}) \in Q \times \mathbb{Z}^d$ written $q(\boldsymbol{v})$. For all $t \in T$ and $\mathbb{D} \in \{\mathbb{Z}, \mathbb{N}\}$, we write

$$p(\boldsymbol{u}) \xrightarrow{t}_{\mathbb{D}} q(\boldsymbol{v})$$

if $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{D}^d$, $\mathrm{src}(t) = p$, $\mathrm{tgt}(t) = q$, and $\boldsymbol{v} = M(t) \cdot \boldsymbol{u} + \Delta(t)$. The relation $\rightarrow_{\mathbb{D}}$ is naturally extended to sequences of transitions,

i.e. for every $w \in T^k$ we let

$$\xrightarrow{w}_{\mathbb{D}} \coloneqq \xrightarrow{w_k}_{\mathbb{D}} \circ \cdots \circ \xrightarrow{w_2}_{\mathbb{D}} \circ \xrightarrow{w_1}_{\mathbb{D}} .$$

Moreover, we write

$$p(\boldsymbol{u}) \to_{\mathbb{D}} q(\boldsymbol{v}) \quad \text{if } p(\boldsymbol{u}) \xrightarrow{t}_{\mathbb{D}} q(\boldsymbol{v}) \text{ for some } t \in T, \text{ and}$$

$$p(\boldsymbol{u}) \xrightarrow{*}_{\mathbb{D}} q(\boldsymbol{v}) \quad \text{if } p(\boldsymbol{u}) \xrightarrow{w}_{\mathbb{D}} q(\boldsymbol{v}) \text{ for some } w \in T^*.$$

As an example, let us consider the affine VASS of Figure 1, *i.e.* where $d = 2$, $Q = \{p, q, r\}$ and T is as depicted graphically. We have:

$$p(3,1) \xrightarrow{s}_{\mathbb{Z}} q(1,0) \xrightarrow{t}_{\mathbb{Z}} r(1,0) \xrightarrow{u}_{\mathbb{Z}} q(1,1) \xrightarrow{t}_{\mathbb{Z}} r(2,0) \xrightarrow{u}_{\mathbb{Z}} q(2,2) \xrightarrow{t}_{\mathbb{Z}} r(4,0).$$

More generally, $p(x,1) \xrightarrow{*}_{\mathbb{Z}} r(2^k, 0)$ for all $x \in \mathbb{Z}$ and $k \in \mathbb{N}_{>0}$.

However, $p(3,0) \xrightarrow{s}_{\mathbb{N}} q(1,-1)$ does *not* hold as counters are not allowed to become negative under this semantics.

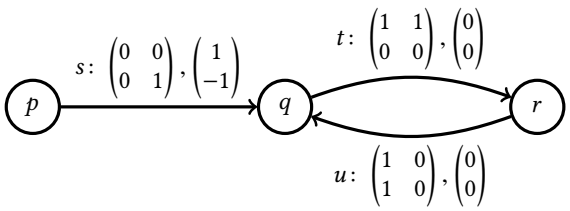

Figure 1. Example of an affine VASS.

Classes of matrices. Let us formalize the informal notion of *classes* of affine VASS, such as "VASS with resets", "VASS with transfers", "VASS with doubling", etc., used throughout the literature.

Such classes depend on the extra operations they provide, *i.e.* by their affine transformations. Since affine VASS extend standard VASS, they always include the identity matrix, which amounts to not applying any extra operation. Moreover, as transformations can be composed along sequences of transitions, their matrices are closed under multiplication. In other words, they form a *monoid*. In addition, general classes do not pose restrictions on the number of counters that can be used, or on the subset of counters on which operations can be applied. In other words, their affine transformations can be extended to *arbitrary dimensions* and can be applied on *any subset of counters*.

We formalize these observations as follows. For every $k \geq 1$, let $\mathbf{I}_k$ be the $k \times k$ identity matrix and let $\mathcal{S}_k$ denote the set of permutations over $[k]$. Let $\mathbf{P}_\sigma \in \{0,1\}^{k \times k}$ be the permutation matrix of $\sigma \in \mathcal{S}_k$. For every matrix $\mathbf{A} \in \mathbb{Z}^{k \times k}$, every permutation $\sigma \in \mathcal{S}_k$ and every $n \geq 1$, let

$$\sigma(\mathbf{A}) \coloneqq \mathbf{P}_\sigma \cdot \mathbf{A} \cdot \mathbf{P}_{\sigma^{-1}},$$

and let $\mathbf{A}\restriction_n \in \mathbb{Z}^{(k+n)\times(k+n)}$ be the matrix such that:

$$\mathbf{A}\restriction_n \coloneqq \begin{pmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_n \end{pmatrix}.$$

A *class (of matrices)* is a set of matrices $C \subseteq \bigcup_{k \geq 1} \mathbb{Z}^{k \times k}$ that satisfies $\{\sigma(\mathbf{A}), \mathbf{A}\restriction_n, \mathbf{I}_n, \mathbf{A} \cdot \mathbf{B}\} \subseteq C$ for every $\mathbf{A}, \mathbf{B} \in C$, every $\sigma \in \mathcal{S}_{\dim \mathbf{A}}$ and every $n \geq 1$. In other words, C is closed under counter renaming; each matrix of C can be extended to larger dimensions; and $C \cap \mathbb{Z}^{k \times k}$ is a monoid under matrix multiplication for every $k \geq 1$.

Note that "counter renaming" amounts to choosing a set of counters on which to apply a given transformation, *i.e.* it renames the counters, applies the transformation, and renames the counters back to their original names. Let us illustrate this. Consider the classical case of transfer VASS, *i.e.* where the contents of a counter can be transferred onto another counter with operations of the form "$x \leftarrow x+y;\ y \leftarrow 0$". In matrix notation, this amounts to:

$$\mathbf{O} \coloneqq \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}.$$

Now, consider a system with three counters c_1, c_2 and c_3. This system should be able to compute "$c_1 \leftarrow c_1 + c_2 + c_3;\ c_2 \leftarrow 0;\ c_3 \leftarrow 0$", but matrix $\mathbf{O}$ cannot achieve this on its own. However, it can be done with the following matrix:

$$\mathbf{O}' \coloneqq \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

We have $\mathbf{O}' = \mathbf{O}\restriction_1 \cdot \sigma(\mathbf{O}\restriction_1)$ where $\sigma \coloneqq (2;3)$. Thus, the operation can be achieved by any class containing $\mathbf{O}$. The symmetric operation "$c_3 \leftarrow c_1 + c_2 + c_3;\ c_1 \leftarrow 0;\ c_2 \leftarrow 0$", *e.g.*, can also be achieved with appropriate permutations. Hence, this corresponds to the usual notion of transfers: we are allowed to choose some counters and apply transfers in either direction.

Note that requiring $\mathbf{P}_\sigma \cdot \mathbf{A} \in C$ for classes would be too strong as it would allow to permute the contents of counters even for classes with no permutation matrix, such as resets.

Classes of interest. We say that a matrix $\mathbf{A} \in \mathbb{Z}^{k \times k}$ is a *pseudo-reset*, *pseudo-transfer* or *pseudo-copy* matrix if $\mathbf{A} \in \{-1, 0, 1\}^{k \times k}$ and if it also satisfies the following:

- pseudo-reset matrix:

  $\mathbf{A}$ is a diagonal matrix;

- pseudo-transfer matrix:

  $\mathbf{A}$ has at most one nonzero entry per column;

- pseudo-copy matrix:

  $\mathbf{A}$ has at most one nonzero entry per row.

We omit the prefix "pseudo-" if $\mathbf{A} \in \{0,1\}^{k \times k}$. Note that the sets of (pseudo-)reset matrices, (pseudo-)transfer matrices, and (pseudo-)copy matrices all form classes. Moreover, (pseudo-)reset matrices are both (pseudo-)transfer and (pseudo-)copy matrices.

Note that the terminology of "reset", "transfer" and "copy" comes from the fact that such matrices implement operations like "$x \leftarrow 0$", "$x \leftarrow x + y;\ y \leftarrow 0$" and "$x \leftarrow x;\ y \leftarrow x$", as

achieved respectively by the matrices of transitions s (*reset*), t (*transfer*) and u (*copy*) illustrated in Figure 1.

Reachability problems. We say that an affine VASS $\mathcal{V} = (k, Q, T)$ *belongs to* a class of matrices C if $\{M(t) : t \in T\} \subseteq C$, *i.e.* if all matrices appearing on its transitions belong to C. The *reachability problem* and *integer reachability problem* for a fixed class C are respectively defined as:

$\underline{\text{Reach}_C}$

Input: an affine VASS $\mathcal{V}$ that belongs to C, and two configurations $p(\boldsymbol{u}), q(\boldsymbol{v})$;
Decide: $p(\boldsymbol{u}) \xrightarrow{*}_{\mathbb{N}} q(\boldsymbol{v})$ in $\mathcal{V}$?

$\underline{\mathbb{Z}\text{-Reach}_C}$

Input: an affine VASS $\mathcal{V}$ that belongs to C, and two configurations $p(\boldsymbol{u}), q(\boldsymbol{v})$;
Decide: $p(\boldsymbol{u}) \xrightarrow{*}_{\mathbb{Z}} q(\boldsymbol{v})$ in $\mathcal{V}$?

3 A complexity trichotomy for integer reachability

This section is devoted to the proof of our main result, namely the trichotomy on $\mathbb{Z}$-Reach$_C$:

Theorem 3.1. *The integer reachability problem $\mathbb{Z}$-Reach$_C$ is:*

(i) NP*-complete if C only contains reset matrices;*
(ii) PSPACE*-complete, otherwise, if either C only contains pseudo-transfer matrices or C only contains pseudo-copy matrices;*
(iii) Undecidable otherwise.

It is known from [19, Cor. 10] that NP-hardness already holds for affine VASS using only the identity matrix, and that NP membership holds for any class of reset matrices. Hence, (i) follows immediately. Thus, the rest of this section is dedicated to proving (ii) and (iii).

3.1 PSPACE-hardness

For the rest of this subsection, let us fix some class C that either only contains pseudo-transfer matrices or only contains pseudo-copy matrices. We prove PSPACE-hardness of $\mathbb{Z}$-Reach$_C$ by first proving that PSPACE-hardness holds if either:

- C contains a matrix with an entry equal to -1; or
- C contains a matrix with entries from $\{0, 1\}$ and a nonzero entry outside of its diagonal.

For these two cases, we first show that C can implement operations $x \leftarrow -x$ or $(x, y) \leftarrow (y, x)$ respectively, *i.e. sign flips* or *swaps*. Essentially, each of these operations is sufficient to simulate linear bounded automata. Before investigating these two cases, let us carefully formalize what it means to implement an operation:

Definition 3.2. Let $f : \mathbb{Z}^k \mapsto \mathbb{Z}^k$ and let $\tau \in \{0, ?\}$. Given a set of counters $X \subseteq [m]$, let

$$V_X := \left\{ \boldsymbol{v} \in \mathbb{Z}^m : \bigwedge_{j \notin X} \boldsymbol{v}(j) = 0 \right\} \text{ if } \tau = 0,$$

and let $V_X := \mathbb{Z}^m$ otherwise. We say that C *τ-implements* f if for every $n \geq k$, there exist counters $X = \{x_1, x_2, \ldots, x_n\}$, matrices $\{\mathbf{F}_\sigma : \sigma \in \mathcal{S}_k\} \subseteq C$ and $m \geq n$ such that the following holds for every $\sigma \in \mathcal{S}_k$ and $\boldsymbol{v} \in V_X$:

(a) $\dim \mathbf{F}_\sigma = m$;
(b) $(\mathbf{F}_\sigma \cdot \boldsymbol{v})(x_{\sigma(i)}) = f(x_{\sigma(1)}, x_{\sigma(2)}, \ldots, x_{\sigma(k)})(i)$ for all $i \in [k]$;
(c) $(\mathbf{F}_\sigma \cdot \boldsymbol{v})(x_{\sigma(i)}) = \boldsymbol{v}(x_{\sigma(i)})$ for every $i \notin [k]$;
(d) $\mathbf{F}_\sigma \cdot \boldsymbol{v} \in V_X$.

We further say that C *implements* f if it either 0-implements or ?-implements f.

Definition 3.2 (b) and (c) state that it is possible to obtain arbitrarily many counters X such that f can be applied on any k-subset of X, provided that the counter values belong to V_X. Moreover, (d) states that vectors resulting from applying operation f also belong to V_X, which ensures that f can be applied arbitrarily many times. Note that (a) allows for extra auxiliary counters whose values are only restricted by V_X.

Informally, ?-implementation means that we use additional counters that can hold arbitrary values, while 0-implementation requires the extra counters to be initialized with zeros but promises to keep them in this state. It turns out that pseudo-transfer matrix classes 0-implement the functions we need, while pseudo-copy matrix classes ?-implement them.

Proposition 3.3. *If C contains a matrix with some entry equal to -1, then it implements sign flips, i.e. the operation $f : \mathbb{Z} \to \mathbb{Z}$ such that $f(x) := -x$.*

Proof. Let $n \geq 1$ and $\mathbf{A} \in C$ be such that $\mathbf{A}[a, b] = -1$ for some counters a and b. Let $d := \dim \mathbf{A}$. We extend $\mathbf{A}$ with $n+2$ counters $X' := X \cup \{y, z\}$, where $X := \{x_i : i \in [n]\}$ are the counters for which we wish to implement sign flips, and $\{y, z\}$ are auxiliary counters. More formally, let $\mathbf{A'} := \mathbf{A}\restriction_{n+2}$ where $X' = [d+1, d']$ and $d' := d + n + 2$.

For every $s, t \in X'$ such that $s \neq t$, let $\mathbf{B}_{s,t} := \pi_{s,t}(\mathbf{A'})$ and let $\mathbf{C}_t := \sigma_t(\mathbf{A'})$ where $\pi_{s,t} := (a; t)(b; s)$ and $\sigma_t := (a; t)$. For every $x \in X$, let

$$\mathbf{F}_x := \begin{cases} \mathbf{B}_{z,x} \cdot \mathbf{B}_{y,z} \cdot \mathbf{B}_{x,y} & \text{if } a \neq b, \\ \mathbf{C}_x & \text{otherwise.} \end{cases}$$

Intuitively, $\mathbf{B}_{s,t}$ (resp. $\mathbf{C}_t$) flips the sign from source counter s (resp. t) to target counter t. If $a \neq b$, then matrix $\mathbf{F}_x$ implements a sign flip in three steps using auxiliary counters y and z, as illustrated in Figure 2. Otherwise, $\mathbf{F}_x$ implements sign flip directly in one step.

Let us consider the case where $\mathbf{A}$ is a pseudo-transfer matrix. From the definition of $\mathbf{B}_{s,t}$ and $\mathbf{C}_t$, it can be shown that for every $s, t, u \in X'$ such that $s \neq t$ and $u \notin \{s, t\}$, the following holds:

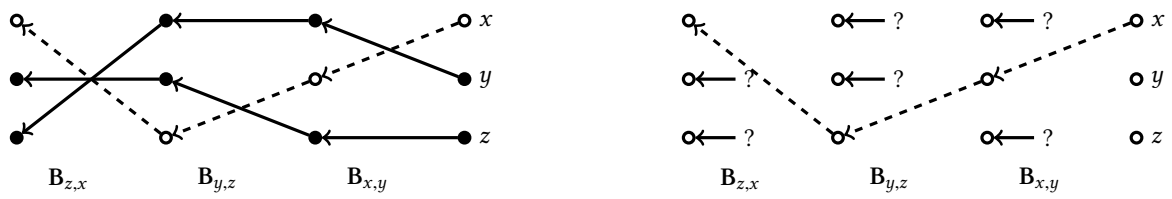

Figure 2. Effect of applying $\mathbf{F}_x$ for the case $a \neq b$, where the left (resp. right) diagram depicts the case where $\mathbf{A}$ is a pseudo-transfer (resp. pseudo-copy) matrix. A solid or dashed edge from counter s to counter t represents operation $s \leftarrow t$ or $s \leftarrow -t$ respectively. Filled nodes indicate counters that necessarily hold 0. Symbol "?" stands for an integer whose value is irrelevant and depends on $\mathbf{A}$ and the counter values.

(i) $\mathbf{B}_{s,t} \cdot \boldsymbol{e}_s = \mathbf{C}_t \cdot \boldsymbol{e}_t = -\boldsymbol{e}_t$, and
(ii) $\mathbf{B}_{s,t} \cdot \boldsymbol{e}_u = \mathbf{C}_t \cdot \boldsymbol{e}_u = \boldsymbol{e}_u$.

Let us show that we 0-implement sign flips, so let

$$V \coloneqq \left\{ \boldsymbol{v} \in \mathbb{Z}^{d'} : \bigwedge_{j \notin X} \boldsymbol{v}(j) = 0 \right\}.$$

Let $\boldsymbol{v} \in V$ and $x \in X$. We have $\boldsymbol{v} = \sum_{y \in X} \boldsymbol{v}(y) \cdot \boldsymbol{e}_y$ by definition of V. Let $\boldsymbol{v}' \coloneqq \sum_{j \in X \setminus \{x\}} \boldsymbol{v}(j) \cdot \boldsymbol{e}_j$. Items (b), (c) and (d) of Definition 3.2 are satisfied since:

$$\begin{aligned}
\mathbf{F}_x \cdot \boldsymbol{v} &= \mathbf{F}_x \cdot \boldsymbol{v}(x) \cdot \boldsymbol{e}_x + \mathbf{F}_x \cdot \boldsymbol{v}' \\
&= \boldsymbol{v}(x) \cdot \mathbf{F}_x \cdot \boldsymbol{e}_x + \boldsymbol{v}' && \text{(by (ii) and def. of } \mathbf{F}_x) \\
&= \boldsymbol{v}(x) \cdot -\boldsymbol{e}_x + \boldsymbol{v}' && \text{(by (i), (ii) and def. of } \mathbf{F}_x) \\
&= -\boldsymbol{v}(x) \cdot \boldsymbol{e}_x + \boldsymbol{v}'.
\end{aligned}$$

The proof of (i) and (ii), and the similar proof for the case where $\mathbf{A}$ is a pseudo-copy matrix, are deferred to the appendix. □

Proposition 3.4. *$\mathbb{Z}$-Reach$_C$ is* PSPACE*-hard if C has a matrix with an entry equal to -1.*

Proof. We give a reduction, partially inspired by [5, Thm. 10], from the membership problem for linear bounded automata, which is PSPACE-complete (*e.g.*, see [21, Sect. 9.3 and 13]).

Let $w \in \{0,1\}^k$ and let $\mathcal{A} = (P, \Sigma, \delta, p_{\text{init}}, p_{\text{acc}})$ be a linear bounded automaton where:

- P is its finite set of control-states;
- $\Sigma = \{0,1\}$ is its input and tape alphabet;
- $\delta \colon P \times \Sigma \to P \times \Sigma \times \{\textsc{Left}, \textsc{Right}\}$ is its transition function; and
- p_{init} and p_{acc} are its initial and accepting control-states, respectively.

We construct an affine VASS $\mathcal{V} = (d, Q, T)$ and configurations $p(\boldsymbol{u})$, $q(\boldsymbol{v})$ such that $\mathcal{V}$ belongs to C, and

$$p(\boldsymbol{u}) \xrightarrow{*}_{\mathbb{Z}} q(\boldsymbol{v}) \iff \mathcal{A} \text{ accepts } w.$$

For every control-state p and head position j of $\mathcal{A}$, there is a matching control-state in $\mathcal{V}$, *i.e.* $Q \coloneqq \{q_{p,j} : p \in P, 1 \leq j \leq k\} \cup \overline{Q}$, where $\overline{Q}$ will be auxiliary control-states. We associate two counters to each tape cell of $\mathcal{A}$, *i.e.* $d \coloneqq 2 \cdot k$. For readability, let us denote these counters $\{x_j, y_j : j \in [k]\}$.

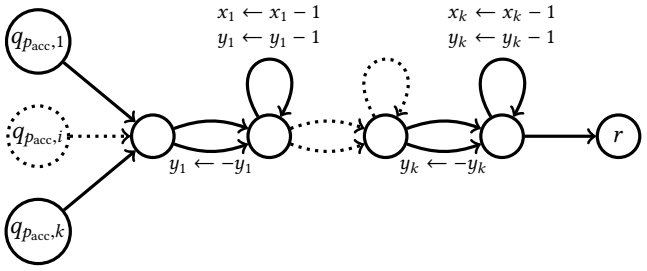

Figure 3. Gadget of $\mathcal{V}$ for tesing whether $\mathcal{A}$ was faithfully simulated and has accepted w.

We represent the contents of tape cell i by the sign of counter y_j, *i.e.* $y_j > 0$ represents 0, and $y_j < 0$ represents 1. We will ensure that y_j is never equal to 0, which would otherwise be an undefined representation. Since $\mathcal{V}$ cannot directly test the sign of a counter, it will be possible for $\mathcal{V}$ to commit errors during the simulation of $\mathcal{A}$. However, we will construct $\mathcal{V}$ in such a way that erroneous simulations are detected.

The gadget depicted in Figure 4 simulates a transition of $\mathcal{A}$ in three steps:

- x_i is incremented;
- y_i is incremented (resp. decremented) if the letter a to be read is 0 (resp. 1);
- the sign of y_i is flipped if the letter b to be written differs from the letter a to be read.

Let $\boldsymbol{u} \in \mathbb{Z}^d$ be the vector such that for every $j \in [k]$:

$$\boldsymbol{u}(x_j) \coloneqq 1 \text{ and } \boldsymbol{u}(y_j) \coloneqq (-1)^{w_j}.$$

Provided that $\mathcal{V}$ starts in vector $\boldsymbol{u}$, we claim that:

- $\bigwedge_{j=1}^{k} (|x_j| \geq |y_j| > 0)$ is an invariant;
- $\mathcal{V}$ has faithfully simulated $\mathcal{A}$ so far if and only if $\bigwedge_{j=1}^{k} (|x_j| = |y_j|)$ holds;
- if $\mathcal{V}$ has faithfully simulated $\mathcal{A}$ so far, then the sign of y_j represents w_j for every $j \in [k]$.

Let us see why this claim holds. Let $i \in [k]$. Initially, we have $|x_i| = |y_i|$ and the sign of y_i set correctly. Assume we execute the gadget of Figure 4, resulting in new values x_i' and y_i'. Let $\lambda \geq 0$ be such that $|x_i| = |y_i| + \lambda$. Let $c \in \{0,1\}$

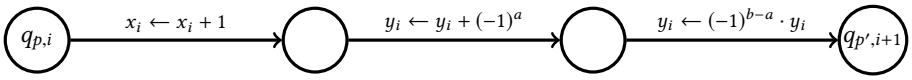

Figure 4. Gadget of $\mathcal{V}$ simulating transition $\delta(p, a) = (p', b, \text{Right})$ of $\mathcal{A}$. The gadget for direction Left is the same except for $q_{p',i+1}$ which is replaced by $q_{p',i-1}$. Note that a and b are fixed, hence expressions such as $(-1)^a$ are constants; they do not require exponentiation.

be the letter represented by y_i. If $c = a$, then $|x'_i| = |y'_i| + \lambda$ and the sign of y'_i represents b as desired. If $c \neq a$, then $|x'_i| = |y'_i| + (\lambda + 1)$. Thus, we have $|x'_i| = |y'_i|$ if and only if no error was made before and during the execution of the gadget.

From the above observations, we conclude that $\mathcal{A}$ accepts w if and only if there exist $i \in [k]$ and $\boldsymbol{v} \in \mathbb{Z}^d$ such that

$$q_{p_{\text{init}},1}(\boldsymbol{u}) \xrightarrow{*}_{\mathbb{Z}} q_{p_{\text{acc}},i}(\boldsymbol{v})$$

and $|\boldsymbol{v}(x_j)| = |\boldsymbol{v}(y_j)|$ for every $j \in [k]$. This can be tested using the gadget depicted in Figure 3, which:

- detects nondeterministically that some control-state of the form $q_{p_{\text{acc}},i}$ has been reached;
- attempts to set y_j to its absolute value for every $j \in [k]$;
- decrements x_j and y_j simultaneously for every $j \in [k]$.

Due to the above observations, it is *only* possible to reach $r(\mathbf{0})$ if $|x_j| = |y_j|$ for every $j \in [k]$ before entering the gadget of Figure 3. Thus, we are done proving the reduction since $\mathcal{A}$ accepts w if and only if

$$q_{p_{\text{init}},1}(\boldsymbol{u}) \xrightarrow{*}_{\mathbb{Z}} r(\mathbf{0}).$$

Sign flips. The above construction considers sign flips as a "native" operation. However, this is not necessarily the case, and instead relies on the fact that class C either 0-implements or ?-implements sign flips, by Proposition 3.3. Thus, the reachability question must be changed to

$$q_{p_{\text{init}},1}(\boldsymbol{u}, \mathbf{0}) \xrightarrow{*}_{\mathbb{Z}} r(\mathbf{0}, \mathbf{0})$$

to take auxiliary counters into account. Moreover, if C ?-implements sign flips, then extra transitions $(r, \mathbf{I}, \boldsymbol{e}_j, r)$ and $(r, \mathbf{I}, -\boldsymbol{e}_j, r)$ must be added to T, for every auxiliary counter j, to allow counter j to be set back to 0. □

In the two forthcoming propositions, we prove PSPACE-hardness of the remaining case.

Proposition 3.5. *If C contains a matrix with entries from $\{0, 1\}$ and a nonzero entry outside of its main diagonal, then it implements swaps, i.e. the operation $f : \mathbb{Z}^2 \to \mathbb{Z}^2$ such that $f(x, y) \coloneqq (y, x)$.*

Proof. Let $n \geq 2$ and let $\mathbf{A} \in C$ be a matrix with entries from $\{0, 1\}$ and a nonzero entry outside of its main diagonal. Let $d \coloneqq \dim \mathbf{A}$. There exist $a, b \in [d]$ such that $\mathbf{A}[a, b] = 1$ and $a \neq b$. Let us extend $\mathbf{A}$ with $n + 1$ counters $X' \coloneqq X \cup \{z\}$ where $X \coloneqq \{x_i : i \in [n]\}$. More formally, let $\mathbf{A}' \coloneqq \mathbf{A}\restriction_{n+1}$, $X' = [d + 1, d']$ and $d' \coloneqq d + n + 1$.

For all $s, t \in X'$ such that $s \neq t$, let $\mathbf{B}_{s,t} \coloneqq \pi_{s,t}(\mathbf{A}')$ where $\pi_{s,t} \coloneqq (b; s)(a; t)$. For every distinct counters $x, y \in X$, let

$$\mathbf{F}_{x,y} \coloneqq \mathbf{B}_{z,x} \cdot \mathbf{B}_{x,y} \cdot \mathbf{B}_{y,z}.$$

Intuitively, $\mathbf{B}_{s,t}$ moves the contents from some source counter s to some target counter t, and $\mathbf{F}_{x,y}$ implements a swap in three steps using an auxiliary counter z as depicted in Figure 5. In the case where $\mathbf{A}$ is a transfer matrix, $\mathbf{B}_{s,t}$ resets s, provided that t held value 0.

Let us consider the case where $\mathbf{A}$ is a transfer matrix. From the definition of $\mathbf{B}_{s,t}$, it can be shown that for every $s, t, u \in X'$ such that $s \neq t$ and $u \notin \{s, t\}$, the following holds:

(i) $\mathbf{B}_{s,t} \cdot \boldsymbol{e}_s = \boldsymbol{e}_t$, and
(ii) $\mathbf{B}_{s,t} \cdot \boldsymbol{e}_u = \boldsymbol{e}_u$.

Let us show that we 0-implement swaps, so let

$$V_X \coloneqq \left\{ \boldsymbol{v} \in \mathbb{Z}^{d'} : \bigwedge_{j \notin X} \boldsymbol{v}(j) = 0 \right\}.$$

Let $\boldsymbol{v} \in V_X$ and let $x, y \in X$ be such that $x \neq y$. We have $\boldsymbol{v} = \sum_{j \in X} \boldsymbol{v}(j) \cdot \boldsymbol{e}_j$ by definition of V_X. Let

$$\boldsymbol{v}' \coloneqq \sum_{j \in X \setminus \{x,y\}} \boldsymbol{v}(j) \cdot \boldsymbol{e}_j.$$

Items (b), (c) and (d) of Definition 3.2 are satisfied since we obtain the following by applications of (i) and (ii):

$$\begin{aligned}
\mathbf{F}_{x,y} \cdot \boldsymbol{v} &= \mathbf{F}_{x,y} \cdot \left(\boldsymbol{v}(x) \cdot \boldsymbol{e}_x + \boldsymbol{v}(y) \cdot \boldsymbol{e}_y\right) + \mathbf{F}_{x,y} \cdot \boldsymbol{v}' \\
&= \mathbf{F}_{x,y} \cdot \left(\boldsymbol{v}(x) \cdot \boldsymbol{e}_x + \boldsymbol{v}(y) \cdot \boldsymbol{e}_y\right) + \boldsymbol{v}' \\
&= \mathbf{B}_{z,x} \cdot \mathbf{B}_{x,y} \cdot \mathbf{B}_{y,z} \cdot (\boldsymbol{v}(x) \cdot \boldsymbol{e}_x + \boldsymbol{v}(y) \cdot \boldsymbol{e}_y) + \boldsymbol{v}' \\
&= \mathbf{B}_{z,x} \cdot \mathbf{B}_{x,y} \cdot (\boldsymbol{v}(x) \cdot \boldsymbol{e}_x + \boldsymbol{v}(y) \cdot \boldsymbol{e}_z) + \boldsymbol{v}' \\
&= \mathbf{B}_{z,x} \cdot (\boldsymbol{v}(x) \cdot \boldsymbol{e}_y + \boldsymbol{v}(y) \cdot \boldsymbol{e}_z) + \boldsymbol{v}' \\
&= \boldsymbol{v}(x) \cdot \boldsymbol{e}_y + \boldsymbol{v}(y) \cdot \boldsymbol{e}_x + \boldsymbol{v}'.
\end{aligned}$$

The proof of (i) and (ii), and the similar proof for the case where $\mathbf{A}$ is a copy matrix, are deferred to the appendix. □

Proposition 3.6. *$\mathbb{Z}$-Reach$_C$ is* PSPACE*-hard if C contains a matrix with entries from $\{0, 1\}$ and a nonzero entry outside of its main diagonal.*

Proof. It is shown in [6] that $\mathbb{Z}$-reachability is PSPACE-hard for affine VASS with swaps, using a reduction from the membership problem for linear bounded automata.

Here, we may not have swaps as a "native" operation. However, by Proposition 3.5, class C implements swaps. Thus, as

Figure 5. Effect of applying $\mathbf{F}_{x,y}$, where the left (resp. right) diagram depicts the case where $\mathbf{A}$ is a transfer (resp. copy) matrix. An edge from counter s to counter t represents operation $s \leftarrow t$. Filled nodes indicate counters that necessarily hold 0. Symbol "?" stands for an integer whose value is irrelevant and depends on $\mathbf{A}$ and the counter values.

in the proof of Proposition 3.4, if the reachability question is of the form

$$p(\boldsymbol{u}) \xrightarrow{*}_{\mathbb{Z}} q(\boldsymbol{v}),$$

then it must be changed to

$$p(\boldsymbol{u}, \mathbf{0}) \xrightarrow{*}_{\mathbb{Z}} q(\boldsymbol{v}, \mathbf{0}).$$

Moreover, if the class C ?-implements swaps, then new transitions must be introduced to allow auxiliary counters to be set back to 0. □

We now proceed to prove the main result of this subsection, namely Theorem 3.1 (ii):

Proof of Theorem 3.1 (ii). Let $\mathcal{M}_k \coloneqq C \cap \mathbb{Z}^{k\times k}$ for every $k \geq 1$. Theorem 7 of [5] shows that $\mathbb{Z}$-Reach$_C$ belongs to PSPACE if each $\mathcal{M}_k$ is a finite monoid of at most exponential norm and size in k. Let us show that this is the case. First, since C is a class, each $\mathcal{M}_k$ is a (finite) monoid. Moreover, by definitions of pseudo-transfer and pseudo-copy matrices, each such matrix can be described by cutting it into k lines and specifying for each line either the position of the unique nonzero entry (which is -1 or 1), or the lack of such entry. Therefore, for every $k \geq 1$, it is the case that $\|\mathcal{M}_k\| \leq 1$ and

$$\begin{aligned} |\mathcal{M}_k| &\leq (2k+1)^k \\ &\leq (4k)^k \\ &= 2^{2k + k\log k} \\ &\leq 2^{\mathrm{poly}(k)}. \end{aligned}$$

It remains to show PSPACE-hardness. By assumption, C contains a nonreset matrix $\mathbf{A}$. Since $\|C\| \leq 1$, we have $\|\mathbf{A}\| = 1$ as no class can be such that $\|C\| = 0$. If $\mathbf{A}$ contains an entry equal to -1, then we are done by Proposition 3.4. Otherwise, $\mathbf{A}$ only has entries from $\{0, 1\}$, and hence we are done by Proposition 3.6. □

3.2 Undecidability

In this subsection, we first show that any class C, that does not satisfy the requirements for $\mathbb{Z}$-Reach$_C \in$ {NP-complete, PSPACE-complete}, must be such that $\|C\| \geq 2$. We then show that this is sufficient to mimic doubling, *i.e.* the operation $x \mapsto 2x$, even if C does not contain a doubling matrix. In more details, we will (a) construct a matrix $\mathbf{C}$ that provides a sufficiently fast growth; which will (b) allow us to derive undecidability by revisiting a reduction from the Post correspondence problem which depends on doubling.

Proposition 3.7. *Let C be a class that contains some matrices $\mathbf{A}$ and $\mathbf{B}$ which are respectively not pseudo-copy and pseudo-transfer matrices. It is the case that $\|C\| \geq 2$.*

Proof. By assumption, $\mathbf{A}$ and $\mathbf{B}$ respectively have a row and a column with at least two nonzero entries. We make use of the following lemma whose proof is deferred to the appendix:

> if C contains a matrix which has a row (resp. column) with at least two nonzero entries, then C also contains a matrix which has a row (resp. column) with at least two nonzero entries with the *same sign*.

Since C is a class, we can assume that $\dim \mathbf{A} = \dim \mathbf{B} = d$ for some $d \geq 2$, as otherwise the smallest matrix can be enlarged. Thus, there exist $i, i' j, k \in [d]$ and $a, b, a', b' \neq 0$ such that:

- $j \neq k$;
- $\mathbf{A}[i, j] = a$ and $\mathbf{A}[i, k] = b$;
- $\mathbf{B}[j, i'] = a'$ and $\mathbf{B}[k, i'] = b'$; and
- $a > 0 \iff b > 0$ and $a' > 0 \iff b' > 0$.

Note that the reason we can assume $\mathbf{A}$ and $\mathbf{B}$ to share counters j and k is due to C being closed under counter renaming.

We wish to obtain a matrix with entry

$$\mathbf{A}[i, j] \cdot \mathbf{B}[j, i'] + \mathbf{A}[i, k] \cdot \mathbf{B}[k, i'] = a \cdot a' + b \cdot b'.$$

We cannot simply pick $\mathbf{A} \cdot \mathbf{B}$ as $(\mathbf{A} \cdot \mathbf{B})[i, i']$ may differ from this value due to other nonzero entries. Hence, we rename all counters of $\mathbf{B}$, except for j and k, with fresh counters. This way, we avoid possible overlaps and we can select precisely the four desired entries. More formally, let $\mathbf{A}' \coloneqq \mathbf{A}\restriction_d$, $\mathbf{B}' \coloneqq \mathbf{B}\restriction_d$ and $\mathbf{C} \coloneqq \mathbf{A}' \cdot \sigma(\mathbf{B}')$, where $\sigma \colon [2d] \to [2d]$ is the permutation:

$$\sigma \coloneqq \prod_{\ell \in [d] \setminus \{j,k\}} (\ell; \ell + d).$$

Let $i'' \coloneqq i' + d$ if $i' \notin \{j, k\}$ and $i'' \coloneqq i'$ otherwise. We have:

$$\begin{aligned}
\mathbf{C}[i, i''] &= \sum_{\ell \in [2d]} \mathbf{A}'[i, \ell] \cdot \mathbf{B}'[\sigma(\ell), i'] && (1)\\
&= \sum_{\ell \in [d] \text{ s.t. } \sigma(\ell) \in [d]} \mathbf{A}'[i, \ell] \cdot \mathbf{B}'[\sigma(\ell), i'] && (2)\\
&= \mathbf{A}'[i, j] \cdot \mathbf{B}'[j, i'] + \mathbf{A}'[i, k] \cdot \mathbf{B}'[k, i']\\
&= a \cdot a' + b \cdot b',
\end{aligned}$$

where (1) follows by definition of $\mathbf{C}$ and by $\sigma(i'') = i'$; and (2) follows by definition of $\mathbf{A}'$ and $\mathbf{B}'$, and by $i, i' \in [d]$.

Since a and b (resp. a' and b') have the same sign, and since $a, b, a', b' \neq 0$, we conclude that $|\mathbf{C}[i, i'']| \geq 2$ and consequently that $\|C\| \geq \|\mathbf{C}\| \geq 2$. □

To avoid cumbersome subscripts, we write $\boldsymbol{e}$ for $\boldsymbol{e}_1$ in the rest of the section. Moreover, let $\lambda_\ell(\mathbf{C}) \coloneqq (\mathbf{C}^\ell \cdot \boldsymbol{e})(1)$ for every matrix $\mathbf{C}$ and $\ell \in \mathbb{N}$.

The following technical lemma will be key to mimic doubling. It shows that, from any class of norm at least 2, we can extract a matrix with sufficiently fast growth.

Lemma 3.8. *For every class of matrices C such that $\|C\| \geq 2$, there exists $\mathbf{C} \in C$ with $\lambda_{n+1}(\mathbf{C}) \geq 2 \cdot \lambda_n(\mathbf{C})$ for every $n \in \mathbb{N}$.*

Proof. Let $\mathbf{A} \in C$ be a matrix with some entry c such that $|c| \geq 2$. We can assume that $c \geq 2$. Indeed, if it is negative, then we can multiply $\mathbf{A}$ by a suitable permutation of itself to obtain an entry equal to $c \cdot c$ (see the proof of Lemma A.1 in the appendix which achieves this). We can further assume that c is the largest positive coefficient occurring within $\mathbf{A}$, and that it lies on the first column of $\mathbf{A}$, *i.e.* $\mathbf{A}[k, 1] = c$ for some $k \in [d]$ where $d \coloneqq \dim \mathbf{A}$. We consider the case where $k = 1$. The case where $k \neq 1$ will be discussed later.

For readability, we rename counters $\{1, 2, \ldots, d\}$ respectively by $X \coloneqq \{x_1, x_2, \ldots, x_d\}$. Note that $(\mathbf{A} \cdot \boldsymbol{e})(x_1) = c \geq 2 \cdot \boldsymbol{e}(x_1)$ as desired. However, vector $\mathbf{A} \cdot \boldsymbol{e}$ may now hold nonzero values in counters $x_2, \ldots, x_d$. Therefore, if we multiply this vector by $\mathbf{A}$, some "noise" will be added to counter x_1. If this noise is too large, then it may cancel the growth of x_1 by $\approx c$. We address this issue by introducing extra auxiliary counters replacing $x_2, \ldots, x_d$ at each "iteration". Of course, we cannot have infinitely many auxiliary counters. Fortunately, after a sufficiently large number m of iterations, the auxiliary counters used at the first iteration will contain sufficiently small noise so that the process can restart from there.

More formally, let $\mathbf{A}' \coloneqq \mathbf{A}\restriction_{|Y|}$ where $Y \coloneqq \{y_{i,j} : 0 \leq i < m, j \in [2, d]\}$ is the set of auxiliary counters, and $m \geq 1$ is a sufficiently large constant whose value will be picked later. Let V be the set of vectors $\boldsymbol{v} \in \mathbb{Z}^{|X|+|Y|}$ satisfying $\boldsymbol{v}(x_1) > 0$ and

$$|\boldsymbol{v}(y_{i,j})| \leq \left(\frac{3c}{4}\right)^i \cdot \frac{\boldsymbol{v}(x_1)}{4d} \text{ for every } y_{i,j} \in Y.$$

Let us fix some vector $\boldsymbol{v}_0 \in V$. For every $0 \leq i < m$, let $\mathbf{B}_i \coloneqq \sigma_i(\mathbf{A}')$ and $\boldsymbol{v}_{i+1} \coloneqq \mathbf{B}_i \cdot \boldsymbol{v}_i$ where σ_i is the permutation

$$\sigma_i \coloneqq \prod_{j \in [2, d]} (x_j; y_{i,j}).$$

We claim that:

$$\boldsymbol{v}_m(x_1) \geq 2 \cdot \boldsymbol{v}_0(x_1) \text{ and } \boldsymbol{v}_m \in V.$$

The validity of this claim proves the lemma. Indeed, $\mathbf{C} \cdot \boldsymbol{v}_0 = \boldsymbol{v}_m$ where $\mathbf{C} \coloneqq \mathbf{B}_{m-1} \cdots \mathbf{B}_1 \cdot \mathbf{B}_0$. Hence, an application of $\mathbf{C}$ yields a vector whose first component has at least doubled. Since $\boldsymbol{e} \in V$ and the resulting vector also belong to V, this can be iterated arbitrarily many times.

Let us first establish the following properties for every $0 \leq i < m$ and $j \in [2, d]$:

(a) $\boldsymbol{v}_i(y_{i,j}) = \boldsymbol{v}_0(y_{i,j})$ and $\boldsymbol{v}_m(y_{i,j}) = \boldsymbol{v}_{i+1}(y_{i,j})$;
(b) $\boldsymbol{v}_{i+1}(x_1) \in \left[\frac{3c}{4} \cdot \boldsymbol{v}_i(x_1), \frac{5c}{4} \cdot \boldsymbol{v}_i(x_1)\right]$; and
(c) $|\boldsymbol{v}_{i+1}(y_{i,j})| \leq 2c \cdot \boldsymbol{v}_i(x_1)$.

Property (a), which follows from the definition of $\mathbf{B}_i$, essentially states that the contents of counter $y_{i,j}$ is only altered from $\boldsymbol{v}_i$ to $\boldsymbol{v}_{i+1}$. Properties (b) and (c) bound the growth of the counters in terms of x_1. Let us prove these two latter properties by induction on i.

By definition of $\boldsymbol{v}_{i+1}$, we have $\boldsymbol{v}_{i+1}(x_1) = c \cdot \boldsymbol{v}_i + \delta$ where

$$\delta \coloneqq \sum_{z \neq x_1} \mathbf{B}_i[x_1, z] \cdot \boldsymbol{v}_i(z).$$

Therefore, $\boldsymbol{v}_{i+1}(x_1) \in [c \cdot \boldsymbol{v}_i(x_1) - |\delta|, c \cdot \boldsymbol{v}_i(x_1) + |\delta|]$, and hence property (b) follows from:

$$\begin{aligned}
|\delta| &\leq \sum_{\substack{z \in X \cup Y \\ z \neq x_1}} |\mathbf{B}_i[x_1, z]| \cdot |\boldsymbol{v}_i(z)|\\
&= \sum_{j \in [2, d]} |\mathbf{A}'[x_1, x_j]| \cdot |\boldsymbol{v}_i(y_{i,j})| && \text{(by def. of } \mathbf{B}_i \text{ and } \sigma_i)\\
&= \sum_{j \in [2, d]} |\mathbf{A}'[x_1, x_j]| \cdot |\boldsymbol{v}_0(y_{i,j})| && \text{(by (a))}\\
&\leq dc \cdot (3c/4)^i \cdot \frac{\boldsymbol{v}_0(x_1)}{4d} && \text{(by } \boldsymbol{v}_0 \in V \text{ and by maximality of } c)\\
&\leq dc \cdot \frac{\boldsymbol{v}_i(x_1)}{4d} && (3)\\
&= \frac{c}{4} \cdot \boldsymbol{v}_i(x_1),
\end{aligned}$$

where (3) holds by $\boldsymbol{v}_i(x_1) \geq (3c/4)^i \cdot \boldsymbol{v}_0(x_1)$ from (b).

Similarly, property (c) holds since, for every $j \in [2,d]$:

$$\begin{aligned}
|\boldsymbol{v}_{i+1}(y_{i,j})| &\leq |\mathbf{A}'[x_j,x_1]| \cdot \boldsymbol{v}_i(x_1) + \sum_{\ell\in[2,d]} |\mathbf{A}'[x_j,x_\ell]| \cdot |\boldsymbol{v}_i(y_{i,\ell})| \\
&\qquad \text{(by def. of } \mathbf{B}_i \text{ and } \sigma_i) \\
&\leq c \cdot \boldsymbol{v}_i(x_1) + dc \cdot (3c/4)^i \cdot \frac{\boldsymbol{v}_0(x_1)}{4d} \\
&\qquad \text{(by (a) and } \boldsymbol{v}_0 \in V) \\
&\leq c \cdot \boldsymbol{v}_i(x_1) + dc \cdot \frac{\boldsymbol{v}_i(x_1)}{4d} \\
&\qquad \text{(by (b))} \\
&\leq 2c \cdot \boldsymbol{v}_i(x_1).
\end{aligned}$$

We may now prove the claim. Let m be sufficiently large so that $(3c/4)^m \geq 8cd$. We have $\boldsymbol{v}_m(x_1) \geq (3c/4)^m \cdot \boldsymbol{v}_0(x_1) \geq 8cd \cdot \boldsymbol{v}_0(x_1)$ by (b) and definition of m. Hence, since $c \geq 2$ and $d \geq 1$, we have $\boldsymbol{v}_m(x_1) \geq 2 \cdot \boldsymbol{v}_0(x_1)$, which satisfies the first part of the claim. Moreover, the second part of the claim, namely $\boldsymbol{v}_m \in V$, holds since for every $y_{i,j} \in Y$, we have:

$$\begin{aligned}
|\boldsymbol{v}_m(y_{i,j})| &= |\boldsymbol{v}_{i+1}(y_{i,j})| && \text{(by (a))} \\
&\leq 2c \cdot \boldsymbol{v}_i(x_1) && \text{(by (c))} \\
&\leq 2c \cdot \frac{\boldsymbol{v}_m(x_1)}{(3c/4)^{m-i}} && (4) \\
&= 2c \cdot (3c/4)^i \cdot \frac{\boldsymbol{v}_m(x_1)}{(3c/4)^m} \\
&\leq 2c \cdot (3c/4)^i \cdot \frac{\boldsymbol{v}_m(x_1)}{8cd} && \text{(by def. of } m) \\
&= (3c/4)^i \cdot \frac{\boldsymbol{v}_m(x_1)}{4d},
\end{aligned}$$

where (4) holds by $\boldsymbol{v}_m(x_1) \geq (3c/4)^{m-i} \cdot \boldsymbol{v}_i(x_1)$ from (b).

We are done proving the lemma for the case $\mathbf{A}[k,1] = c \geq 2$ with $k = 1$. This case is slightly simpler as c lies on the main diagonal of $\mathbf{A}$ which means that $\boldsymbol{v}_{i+1}(x_1) \approx c \cdot \boldsymbol{v}_i(x_1)$. If $k \neq 1$, then we have $\boldsymbol{v}_{i+1}(x_k) \approx c \cdot \boldsymbol{v}_i(x_1)$ instead, which breaks composability for the next iteration. However, this is easily fixed by swapping the names of counters x_k and x_1. □

Let us fix some class C such that $\|C\| \geq 2$ and the matrix $\mathbf{C}$ obtained for C from Lemma 3.8. For simplicity, we will write λ_ℓ instead of $\lambda_\ell(\mathbf{C})$. We prove two intermediary propositions that essentially show that $\mathbf{C}$ can encode binary strings. Let $f_b(\boldsymbol{v}) \coloneqq \mathbf{C} \cdot \boldsymbol{v} + b \cdot \boldsymbol{e}$ for both $b \in \{0,1\}$ and every $\boldsymbol{v} \in \mathbb{Z}^{\dim \mathbf{C}}$. Let f_ε be the identity function, and let

$$f_x \coloneqq f_{x_n} \circ \cdots \circ f_{x_2} \circ f_{x_1} \text{ for every } x \in \{0,1\}^n.$$

Let $\gamma_x \coloneqq f_x(\boldsymbol{e})(1)$ for every $x \in \{0,1\}^*$. Let $[\![\varepsilon]\!] \coloneqq \emptyset$ and let $[\![w]\!] \coloneqq \{i \in [k] : w_i = 1\}$ be the "support" of w for every sequence $w \in \{0,1\}^+$ of length $k > 0$.

Proposition 3.9. *For every $x \in \{0,1\}^*$, the following holds: $\gamma_x = \lambda_{|x|} + \sum_{i\in[\![x]\!]} \lambda_{|x|-i}$.*

Proof. It suffices to show that $f_x(\boldsymbol{e}) = \mathbf{C}^{|x|}\cdot\boldsymbol{e} + \sum_{i\in[\![x]\!]} \mathbf{C}^{|x|-i}\cdot\boldsymbol{e}$ for every $x \in \{0,1\}^*$. Let us prove this by induction on $|x|$. If $|x| = 0$, then $x = \varepsilon$, and hence $f_x(\boldsymbol{e}) = \boldsymbol{e} = \mathbf{C}^0 \cdot \boldsymbol{e}$. Assume that $|x| > 0$ and that the claim holds for sequences of length $|x| - 1$. There exist $b \in \{0,1\}$ and $w \in \{0,1\}^*$ such that $x = wb$. We have:

$$\begin{aligned}
f_x(\boldsymbol{e}) &= \mathbf{C} \cdot f_w(\boldsymbol{e}) + b \cdot \boldsymbol{e} \\
&= \mathbf{C} \cdot \left(\mathbf{C}^{|w|} \cdot \boldsymbol{e} + \sum_{i\in[\![w]\!]} \mathbf{C}^{|w|-i} \cdot \boldsymbol{e} \right) + b \cdot \boldsymbol{e} && (5) \\
&= \left(\mathbf{C}^{|w|+1} \cdot \boldsymbol{e} + \sum_{i\in[\![w]\!]} \mathbf{C}^{|w|+1-i} \cdot \boldsymbol{e} \right) + b \cdot \boldsymbol{e} \\
&= \left(\mathbf{C}^{|x|} \cdot \boldsymbol{e} + \sum_{i\in[\![x]\!]\setminus\{|x|\}} \mathbf{C}^{|x|-i} \cdot \boldsymbol{e} \right) + b \cdot \mathbf{C}^{|x|-|x|} \cdot \boldsymbol{e} \\
&= \mathbf{C}^{|x|} \cdot \boldsymbol{e} + \sum_{i\in[\![x]\!]} \mathbf{C}^{|x|-i} \cdot \boldsymbol{e}, && (6)
\end{aligned}$$

where (5) follows by induction hypothesis, and (6) by definition of $[\![x]\!]$. □

Proposition 3.10. *For every $x, y \in \{0,1\}^*$, it is the case that $x = y$ if and only if $\gamma_x = \gamma_y$.*

Proof. Let $<_{\text{lex}}$ denote the lexicographical order over $\{0,1\}^*$. It is sufficient to show that for every $x, y \in \{0,1\}^*$ the following holds: if $x <_{\text{lex}} y$, then $\gamma_x < \gamma_y$. Indeed, if this claim holds, then for every $x, y \in \{0,1\}^*$ such that $x \neq y$, we either have $x <_{\text{lex}} y$ or $y <_{\text{lex}} x$, which implies $\gamma_x \neq \gamma_y$ in both cases.

Let us prove the claim. Let $x, y \in \{0,1\}^*$ be such that $x <_{\text{lex}} y$. We either have $|x| < |y|$ or $|x| = |y|$. If the former holds, then the claim follows from:

$$\begin{aligned}
\gamma_x &= \lambda_{|x|} + \sum_{i\in[\![x]\!]} \lambda_{|x|-i} && \text{(by Proposition 3.9)} \\
&\leq \lambda_{|x|} + \sum_{i=1}^{|x|} \lambda_{|x|}/2^i && \text{(by Lemma 3.8)} \\
&= \lambda_{|x|} \cdot \left(1 + \sum_{i=1}^{|x|} 1/2^i\right) \\
&= \lambda_{|x|} \cdot (2 - 1/2^{|x|}) \\
&< 2 \cdot \lambda_{|x|} && \text{(since } \lambda_{|x|} > 0) \\
&\leq \lambda_{|y|} && \text{(by Lemma 3.8)} \\
&\leq \gamma_y && \text{(by Proposition 3.9).}
\end{aligned}$$

It remains to prove the case where $|x| = |y| = k$ for some $k > 0$. Since $x <_{\text{lex}} y$, there exist $u, v, w \in \{0,1\}^*$ such that $x = u0v$ and $y = u1w$. Let $\ell \coloneqq k - |u| - 1$. Note that

$\ell = |v| = |w|$. The proof is completed by observing that:

$$\begin{aligned}
\gamma_y - \gamma_x &= \lambda_\ell + \sum_{i \in [\![w]\!]} \lambda_{\ell-i} - \sum_{i \in [\![v]\!]} \lambda_{\ell-i} && \text{(by Proposition 3.9)} \\
&\geq \lambda_\ell - \sum_{i=1}^{|v|} \lambda_{\ell-i} \\
&\geq \lambda_\ell - \sum_{i=1}^{|v|} \lambda_\ell / 2^i && (7) \\
&= \lambda_\ell \cdot \left(1 - \sum_{i=1}^{\ell} 1/2^i\right) && \text{(by } |v| = \ell\text{)} \\
&= \lambda_\ell / 2^\ell \\
&> 0,
\end{aligned}$$

where (7) holds by $\lambda_\ell \geq 2^i \cdot \lambda_{\ell-i}$ from Lemma 3.8. □

We may finally prove the last part of our trichotomy.

Theorem 3.11. *$\mathbb{Z}$-Reach$_C$ is undecidable if $\|C\| \geq 2$.*

Proof. We give a reduction from the Post correspondence problem inspired by [33]. There, counter values can be doubled as a "native" operation. Here, we adapt the construction with our emulation of doubling. Let us consider an instance of the Post correspondence problem over alphabet $\{0, 1\}$:

$$\Gamma := \left\{ \left[\frac{u_1}{v_1}\right], \left[\frac{u_2}{v_2}\right], \ldots, \left[\frac{u_\ell}{v_\ell}\right] \right\}.$$

We say that Γ *has a match* if there exists $w \in \Gamma^+$ such that the underlying top and bottom sequences of w are equal.

Let $\mathbf{C}$ be the matrix obtained for C from Lemma 3.8, let $d := \dim \mathbf{C}$, and let $\boldsymbol{e}$ be of size d. For every $x \in \{0, 1\}^*$, let g_x and h_x be the linear mappings over $\mathbb{Z}^{2d}$ defined as f_x, but operating on counters $1, 2, \ldots, d$ and counters $d+1, d+2, \ldots, 2d$ respectively.

Let $\mathcal{V} := (2d, Q, T)$ be the affine VASS such that Q and T are as depicted in Figure 6. Note that $\mathcal{V}$ belongs to C. Indeed, g_x and h_x can be obtained from matrix $\mathbf{C} \in C$ and the fact that C is a class, and hence closed under counter renaming and larger dimensions. We claim that

$$p(\boldsymbol{e}, \boldsymbol{e}) \xrightarrow{*}_{\mathbb{Z}} r(\boldsymbol{e}, \boldsymbol{e}) \text{ if and only if } \Gamma \text{ has a match.}$$

Note that any sequence $w \in T^+$ from p to p computes $g_{w_x} \circ h_{w_y}$ for some word

$$\left[\frac{w_x}{w_y}\right] \in \Gamma^+.$$

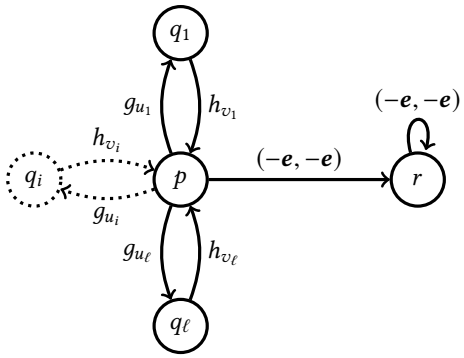

Figure 6. Affine VASS $\mathcal{V}$ for the Post correspondence problem. Arcs labeled by mappings of the form g_x and h_x each stand for finite sequences of $|x|$ transitions implementing g_x and h_x.

Thus:

$$\begin{aligned}
& p(\boldsymbol{e}, \boldsymbol{e}) \xrightarrow{*}_{\mathbb{Z}} r(\boldsymbol{e}, \boldsymbol{e}) \\
\iff\ & \exists w \in T^+, \boldsymbol{v} \in \mathbb{Z}^{2d} : p(\boldsymbol{e}, \boldsymbol{e}) \xrightarrow{w}_{\mathbb{Z}} p(\boldsymbol{v}) \xrightarrow{*}_{\mathbb{Z}} r(\boldsymbol{e}, \boldsymbol{e}) \\
\iff\ & \exists w \in T^+, \boldsymbol{v} \in \mathbb{Z}^{2d} : && (8) \\
& \qquad p(\boldsymbol{e}, \boldsymbol{e}) \xrightarrow{w}_{\mathbb{Z}} p(\boldsymbol{v}) \text{ and } \boldsymbol{v}(1) = \boldsymbol{v}(d+1) \\
\iff\ & \exists w \in T^+ : \gamma_{w_x} = \gamma_{w_y} && (9) \\
\iff\ & \exists w \in T^+ : w_x = w_y && (10) \\
\iff\ & \Gamma \text{ has a match,}
\end{aligned}$$

where (9) follows by definitions of g, h and γ, and where (10) follows from Proposition 3.10. □

We conclude this section by proving Theorem 3.1 (iii) which can be equivalently formulated as follows:

Corollary 3.12. *$\mathbb{Z}$-Reach$_C$ is undecidable if C does not only contain pseudo-transfer matrices and does not only contain pseudo-copy matrices.*

Proof. We have $\|C\| \geq 2$ by Proposition 3.7, hence undecidability follows from Theorem 3.11. □

4 A complexity dichotomy for reachability

This section is devoted to the following complexity dichotomy on Reach$_C$, which is mostly proven by exploiting notions and results from the previous section:

Theorem 4.1. *The reachability problem Reach$_C$ is equivalent to the (standard) VASS reachability problem if C only contains permutation matrices, and is undecidable otherwise.*

4.1 Decidability

Note that the (standard) VASS reachability problem is the problem Reach$_{\mathcal{I}}$ where $\mathcal{I} := \bigcup_{n \geq 1} \mathbf{I}_n$. Clearly Reach$_{\mathcal{I}} \leq$ Reach$_C$ for any class C. Therefore, it suffices to show the following:

Proposition 4.2. *$\textsc{Reach}_C \leq \textsc{Reach}_I$ for every C that only contains permutation matrices.*

Proof. Let $\mathcal{V} = (d, Q, T)$ be an affine VASS that belongs to C. We construct a (standard) VASS $\mathcal{V}' = (d, Q', T')$ that simulates $\mathcal{V}$. Recall that a (standard) VASS is an affine VASS that only uses the identity matrix. For readability, we omit the identity matrix on the transitions of $\mathcal{V}'$. We assume without loss of generality that each transition $t \in T$ satisfies either $\Delta(t) = \mathbf{0}$ or $M(t) = \mathbf{I}$. Indeed, since permutation matrices are nonnegative, every transition of T can be splitted in two parts: first applying its matrix, and then its vector.

The control-states and transitions of $\mathcal{V}'$ are defined as $Q' \coloneqq \{q_\sigma : q \in Q, \sigma \in \mathcal{S}_d\}$ and $T' \coloneqq S_\cup \cup S_{\text{vec}}$, which are to be defined shortly. Intuitively, each control-state of $\mathcal{V}'$ stores the current control-state of $\mathcal{V}$ together with the current renaming of its counters. Whenever a transition $t \in T$ such that $\Delta(t) = \mathbf{0}$ is to be applied, this means that the counters must be renamed by the permutation $M(t)$. This is achieved by:

$$S_\cup \coloneqq \{(p_\sigma, \mathbf{0}, q_{\pi\circ\sigma}) : (p, \mathbf{P}_\pi, \mathbf{0}, q) \in T, \sigma \in \mathcal{S}_d\}.$$

Similarly, whenever a transition $t \in T$ such that $M(t) = \mathbf{I}$ is to be applied, this means that $\Delta(t)$ should be added to the counters, but in accordance to the current renaming of the counters. This is achieved by:

$$S_{\text{vec}} \coloneqq \{(p_\sigma, \mathbf{P}_\sigma \cdot \boldsymbol{b}, q_\sigma) : (p, \mathbf{I}, \boldsymbol{b}, q), \sigma \in \mathcal{S}_d\}.$$

A routine induction shows that

$$p(\boldsymbol{u}) \xrightarrow{*}_{\mathbb{N}} q(\boldsymbol{v}) \text{ in } \mathcal{V} \text{ iff } p_\varepsilon(\boldsymbol{u}) \xrightarrow{*}_{\mathbb{N}} q_\sigma(\mathbf{P}_\sigma \cdot \boldsymbol{v}) \text{ in } \mathcal{V}',$$

where ε denotes the identity permutation. Since this amounts to finitely many reachability queries, *i.e.* $|\mathcal{S}_d| = d!$ queries, this yields a Turing reduction[1]. □

4.2 Undecidability

We show undecidability by considering three types of classes: (1) classes with matrices with negative entries; (2) nontransfer *and* noncopy classes; and (3) transfer *or* copy classes. In each case, we will argue that an "undecidable operation" can be simulated, namely: zero-tests, doubling and resets.

Proposition 4.3. *$\textsc{Reach}_C$ is undecidable for every class C that contains a matrix with some negative entry.*

Proof. Let $\mathbf{A} \in C$ be a matrix such that $\mathbf{A}[i, j] < 0$ for some $i, j \in [d]$ where $d \coloneqq \dim \mathbf{A}$. We show how a two counter Minsky machine $\mathcal{M}$ can be simulated by an affine VASS $\mathcal{V}$ belonging to C. Note that we only have to show how to simulate zero-tests. The affine VASS $\mathcal{V}$ has $2d$ counters: counters j and $j + d$ which represent the two counters x and y of $\mathcal{M}$; and $2d - 2$ auxiliary counters which will be permanently set to value 0.

[1] Although it is not necessary for our needs, the reduction can be made *many-one* by weakly computing a matrix multiplication by $\mathbf{P}_{\sigma^{-1}}$ onto d new counters, from each control-state q_σ to a common state r.

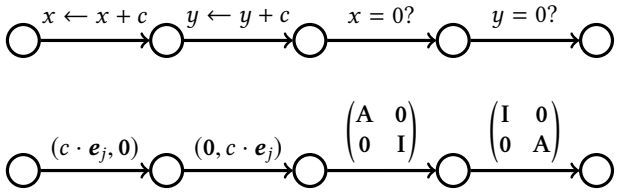

Figure 7. *Top*: example of a two counter machine $\mathcal{M}$. *Bottom*: an affine VASS $\mathcal{V}$ simulating $\mathcal{M}$.

Observe that for every $\lambda \in \mathbb{N}$, the following holds:

$$\mathbf{A} \cdot \lambda \boldsymbol{e}_j = \begin{cases} \mathbf{0} & \text{if } \lambda = 0, \\ \lambda \cdot \mathbf{A}[\star, j] & \text{otherwise.} \end{cases}$$

Thus, $\mathbf{A}$ simulates a zero-test as it leaves all counters set to zero if counter j holds value zero, and it generates a vector with some negative entry otherwise, which is an invalid configuration under $\mathbb{N}$-reachability. Figure 7 shows how each transition of $\mathcal{M}$ is replaced in $\mathcal{V}$. We are done since

$$(m, n) \xrightarrow{*}_{\mathbb{N}} (m', n') \text{ in } \mathcal{M}$$
$$\iff (m \cdot \boldsymbol{e}_j, n \cdot \boldsymbol{e}_j) \xrightarrow{*}_{\mathbb{N}} (m' \cdot \boldsymbol{e}_j, n' \cdot \boldsymbol{e}_j) \text{ in } \mathcal{V}. \qquad \square$$

Proposition 4.4. *$\textsc{Reach}_C$ is undecidable if C does not only contain transfer matrices and does not only contain copy matrices.*

Proof. If C contains a matrix with some negative entry, then we are done by Proposition 4.3. Thus, assume C only contains nonnegative matrices. By Proposition 3.7, we have $\|C\| \geq 2$. Let $\mathbf{C}$ be the matrix obtained for C from Lemma 3.8. Since $\mathbf{C} \geq \mathbf{0}$, we have $\mathbf{C} \cdot \boldsymbol{v} \geq \mathbf{0}$ for every $\boldsymbol{v} \geq \mathbf{0}$. Hence, multiplication by $\mathbf{C}$ is always allowed under $\mathbb{N}$-reachability. Thus, the reduction from the Post correspondence problem given in Theorem 3.11 holds here under $\mathbb{N}$-reachability, as the only possibly (relevant) negative values arose from $\mathbf{C}$. □

We may finally prove the last part of our dichotomy:

Theorem 4.5. *$\textsc{Reach}_C$ is undecidable for every class C with some nonpermutation matrix.*

Proof. Let $\mathbf{A} \in C$ be a matrix which is not a permutation matrix. By Proposition 4.3 and Proposition 4.4, we may assume that $\mathbf{A}$ is either a transfer or a copy matrix. Hence, $\mathbf{A}$ must have a column or a row equal to $\mathbf{0}$, as otherwise it would be a permutation matrix. Thus, we either have $\mathbf{A}[\star, i] = \mathbf{0}$ or $\mathbf{A}[i, \star] = \mathbf{0}$ for some $i \in [d]$ where $d \coloneqq \dim \mathbf{A}$.

We show that C implements resets, *i.e.* the operation $f : \mathbb{Z} \to \mathbb{Z}$ such that $f(x) \coloneqq 0$. This is sufficient to complete the proof since reachability for VASS with resets is undecidable [1].

Let $X \coloneqq \{d + 1, d + 2, \ldots, d + n\}$ be the counters for which we wish to implement resets. Let $\mathbf{A}' \coloneqq \mathbf{A}\restriction_n$ and let $\mathbf{B}_x \coloneqq \sigma_x(\mathbf{A}')$ where $\sigma_x \coloneqq (x; i)$. Let $x, y \in X$ be counters such that $x \neq y$.

Case $\mathbf{A}[\star, i] = \mathbf{0}$. We have: $\mathbf{B}_x \cdot \boldsymbol{e}_x = \mathbf{B}_x[\star, x] = \mathbf{A}'[\star, i] = \mathbf{0}$. Similarly, it can be shown that $\mathbf{B}_x \cdot \boldsymbol{e}_y = \boldsymbol{e}_y$. Hence, class C 0-implements resets.

Case $\mathbf{A}[i, \star] = \mathbf{0}$. The following holds for every $\boldsymbol{v} \in \mathbb{Z}^{d+n}$:

$$\begin{aligned}(\mathbf{B}_x \cdot \boldsymbol{v})(x) &= \sum_{\ell \in [d+n]} \mathbf{B}[x, \ell] \cdot \boldsymbol{v}(\ell) \\ &= \sum_{\ell \in [d+n]} \mathbf{A}'[i, \sigma_x(\ell)] \cdot \boldsymbol{v}(\ell) \\ &= 0.\end{aligned}$$

Similarly, we have $(\mathbf{B}_x \cdot \boldsymbol{v})(y) = \boldsymbol{v}(y)$. Hence, class C ?-implements resets. □

5 Conclusion and further work

Motivated by the use of relaxations to alleviate the tremendous complexity of reachability in VASS, we have studied the complexity of integer reachability in affine VASS.

Namely, we have shown a trichotomy on the complexity of integer reachability for affine VASS: it is NP-complete for any class of reset matrices; PSPACE-complete for any class of pseudo-transfers matrices and any class of pseudo-copies matrices; and undecidable for *any* other class.

Moreover, the notions and techniques introduced along the way allowed us to give a complexity dichotomy for (standard) reachability in affine VASS: it is decidable for any class of permutation matrices, and undecidable for any other class. This provides a complete general landscape of the complexity of reachability in affine VASS.

A further direction of study is the range of possible complexities for integer reachability relations for *specific* affine VASS instances and *specific* matrix monoids. For the former question, we conjecture that the computational complexity can be completely arbitrary across a very wide range going from polynomial complexity to undecidability. We are currently studying a specific construction that is likely to provide a proof of that conjecture. The case of *fixed specific* matrix monoids is entirely open.

References

[1] Toshiro Araki and Tadao Kasami. 1976. Some Decision Problems Related to the Reachability Problem for Petri Nets. *Theoretical Computer Science* 3, 1 (1976), 85–104. https://doi.org/10.1016/0304-3975(76)90067-0

[2] Konstantinos Athanasiou, Peizun Liu, and Thomas Wahl. 2016. Unbounded-Thread Program Verification using Thread-State Equations. In *Proc.* 8^{th} *International Joint Conference on Automated Reasoning (IJCAR)*. 516–531. https://doi.org/10.1007/978-3-319-40229-1_35

[3] Michael Blondin, Alain Finkel, Christoph Haase, and Serge Haddad. 2017. The Logical View on Continuous Petri Nets. *ACM Transactions on Computational Logic (TOCL)* 18, 3 (2017), 24:1–24:28. https://doi.org/10.1145/3105908

[4] Michael Blondin and Christoph Haase. 2017. Logics for continuous reachability in Petri nets and vector addition systems with states. In *Proc.* 32^{nd} *Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. 1–12. https://doi.org/10.1109/LICS.2017.8005068

[5] Michael Blondin, Christoph Haase, and Filip Mazowiecki. 2018. Affine Extensions of Integer Vector Addition Systems with States. In *Proc.* 29^{th} *International Conference on Concurrency Theory (CONCUR)*. 14:1–14:17. https://doi.org/10.4230/LIPIcs.CONCUR.2018.14

[6] Michael Blondin, Christoph Haase, Filip Mazowiecki, and Mikhail A. Raskin. 2019. Affine Extensions of Integer Vector Addition Systems with States. *CoRR* (2019). arXiv:1909.12386

[7] Rémi Bonnet, Alain Finkel, and M. Praveen. 2012. Extending the Rackoff technique to Affine nets. In *Proc.* 32^{nd} *Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*. 301–312. https://doi.org/10.4230/LIPIcs.FSTTCS.2012.301

[8] Michaël Cadilhac, Alain Finkel, and Pierre McKenzie. 2012. Affine Parikh automata. *RAIRO – Theoretical Informatics and Applications* 46, 4 (2012), 511–545. https://doi.org/10.1051/ita/2012013

[9] Dmitry Chistikov, Christoph Haase, and Simon Halfon. 2018. Context-free commutative grammars with integer counters and resets. *Theoretical Computer Science* 735 (2018), 147–161. https://doi.org/10.1016/j.tcs.2016.06.017

[10] Wojciech CzerwiÅĎski, SÅĆawomir Lasota, Ranko Lazić, Jérôme Leroux, and Filip Mazowiecki. 2019. The reachability problem for Petri nets is not elementary. In *Proc.* 51^{st} *Annual ACM SIGACT Symposium on Theory of Computing (STOC)*. 24–33. https://doi.org/10.1145/3313276.3316369

[11] Giorgio Delzanno, Jean-François Raskin, and Laurent Van Begin. 2002. Towards the Automated Verification of Multithreaded Java Programs. In *Proc.* 8^{th} *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*. 173–187. https://doi.org/10.1007/3-540-46002-0_13

[12] Catherine Dufourd, Alain Finkel, and Philippe Schnoebelen. 1998. Reset Nets Between Decidability and Undecidability. In *Proc.* 25^{th} *International Colloquium on Automata, Languages and Programming (ICALP)*. 103–115. https://doi.org/10.1007/BFb0055044

[13] Javier Esparza, Pierre Ganty, Jérôme Leroux, and Rupak Majumdar. 2017. Verification of population protocols. *Acta Informatica* 54, 2 (2017), 191–215. https://doi.org/10.1007/s00236-016-0272-3

[14] Javier Esparza, Ruslán Ledesma-Garza, Rupak Majumdar, Philipp J. Meyer, and Filip NikÅąiÄĞ. 2014. An SMT-Based Approach to Coverability Analysis. In *Proc.* 26^{th} *International Conference on Computer Aided Verification (CAV)*. 603–619. https://doi.org/10.1007/978-3-319-08867-9_40

[15] Alain Finkel, Stefan Göller, and Christoph Haase. 2013. Reachability in Register Machines with Polynomial Updates. In *Proc.* 38^{th} *International Symposium on Mathematical Foundations of Computer Science (MFCS)*. 409–420. https://doi.org/10.1007/978-3-642-40313-2_37

[16] Alain Finkel and Jérôme Leroux. 2002. How to Compose Presburger-Accelerations: Applications to Broadcast Protocols. In *Proc.* 22^{nd} *Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*. 145–156. https://doi.org/10.1007/3-540-36206-1_14

[17] Estíbaliz Fraca and Serge Haddad. 2015. Complexity Analysis of Continuous Petri Nets. *Fundamenta Informaticae* 137, 1 (2015), 1–28. https://doi.org/10.3233/FI-2015-1168

[18] Steven M. German and A. Prasad Sistla. 1992. Reasoning about Systems with Many Processes. *Journal of the ACM* 39, 3 (1992), 675–735. https://doi.org/10.1145/146637.146681

[19] Christoph Haase and Simon Halfon. 2014. Integer Vector Addition Systems with States. In *Proc.* 8^{th} *International Workshop on Reachability Problems (RP)*. 112–124. https://doi.org/10.1007/978-3-319-11439-2_9

[20] Monika Heiner, David R. Gilbert, and Robin Donaldson. 2008. Petri Nets for Systems and Synthetic Biology. In *Formal Methods for Computational Systems Biology*. 215–264. https://doi.org/10.1007/978-3-540-68894-5_7

[21] John E. Hopcroft and Jeffrey D. Ullman. 1979. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley.

[22] Radu Iosif and Arnaud Sangnier. 2016. How Hard is It to Verify Flat Affine Counter Systems with the Finite Monoid Property?. In *Proc.* 14^{th}

International Symposium on Automated Technology for Verification and Analysis (ATVA). 89–105. https://doi.org/10.1007/978-3-319-46520-3_6

[23] Alexander Kaiser, Daniel Kroening, and Thomas Wahl. 2014. A Widening Approach to Multithreaded Program Verification. *ACM Trans. on Prog. Languages and Systems* 36, 4 (2014), 14:1–14:29. https://doi.org/10.1145/2629608

[24] S. Rao Kosaraju. 1982. Decidability of Reachability in Vector Addition Systems (Preliminary Version). In *Proc. 14th Symposium on Theory of Computing (STOC)*. 267–281. https://doi.org/10.1145/800070.802201

[25] Jean-Luc Lambert. 1992. A Structure to Decide Reachability in Petri Nets. *Theoretical Computer Science* 99, 1 (1992), 79–104. https://doi.org/10.1016/0304-3975(92)90173-D

[26] Jérôme Leroux. 2010. The General Vector Addition System Reachability Problem by Presburger Inductive Invariants. *Logical Methods in Computer Science* 6, 3 (2010). https://doi.org/10.2168/LMCS-6(3:22)2010

[27] Jérôme Leroux. 2011. Vector addition system reachability problem: a short self-contained proof. In *Proc. 38th Symposium on Principles of Programming Languages (POPL)*. 307–316. https://doi.org/10.1145/1926385.1926421

[28] Jérôme Leroux. 2012. Vector Addition Systems Reachability Problem (A Simpler Solution). In *Turing-100 – The Alan Turing Centenary*. 214–228.

[29] Jérôme Leroux and Sylvain Schmitz. 2015. Demystifying Reachability in Vector Addition Systems. In *Proc. 30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. 56–67. https://doi.org/10.1109/LICS.2015.16

[30] Jérôme Leroux and Sylvain Schmitz. 2019. Reachability in Vector Addition Systems is Primitive-Recursive in Fixed Dimension. In *Proc. 34th Symposium on Logic in Computer Science (LICS)*.

[31] Richard J. Lipton. 1976. *The Reachability Problem Requires Exponential Space*. Technical Report 63. Department of Computer Science, Yale University.

[32] Ernst W. Mayr. 1981. An Algorithm for the General Petri Net Reachability Problem. In *Proc. 13th Symposium on Theory of Computing (STOC)*. 238–246. https://doi.org/10.1145/800076.802477

[33] Julien Reichert. 2015. *Reachability games with counters: decidability and algorithms*. Ph.D. Dissertation. École normale supérieure de Cachan, France.

[34] George S. Sacerdote and Richard L. Tenney. 1977. The Decidability of the Reachability Problem for Vector Addition Systems (Preliminary Version). In *Proc. 9th Symposium on Theory of Computing (STOC)*. 61–76. https://doi.org/10.1145/800105.803396

[35] Jake Silverman and Zachary Kincaid. 2019. Loop Summarization with Rational Vector Addition Systems. In *Proc. 31st International Conference on Computer Aided Verification (CAV)*. 97–115. https://doi.org/10.1007/978-3-030-25543-5_7

[36] Rüdiger Valk. 1978. Self-Modifying Nets, a Natural Extension of Petri Nets. In *Proc. Fifth Colloquium on Automata, Languages and Programming (ICALP)*. 464–476. https://doi.org/10.1007/3-540-08860-1_35

[37] Wil van der Aalst. 1998. The Application of Petri Nets to Workflow Management. *Journal of Circuits, Systems, and Computers* 8, 1 (1998), 21–66. https://doi.org/10.1142/S0218126698000043

[38] Moe Thandar Wynn, Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, and David Edmond. 2009. Synchronization and Cancelation in Workflows Based on Reset Nets. *International Journal of Cooperative Information Systems* 18, 1 (2009), 63–114. https://doi.org/10.1142/S0218843009002002

A Appendix

A.1 Details for the proof of Proposition 3.3

Pseudo-transfer matrix. Let us first prove properties (i) and (ii) stated within the proof of Proposition 3.5 for the case where $\mathbf{A}$ is a pseudo-transfer matrix. The validity of (i) for $\mathbf{B}_{s,t}$ follows from:

$$\begin{aligned}(\mathbf{B}_{s,t}\cdot\boldsymbol{e}_s)(k) &= \mathbf{B}_{s,t}[k,s]\\ &= \mathbf{A}'[\pi_{s,t}(k),b] && \text{(since } \pi_{s,t}(s)=b\text{)}\\ &= -1 \text{ if } \pi_{s,t}(k)=a \text{ else } 0 && (11)\\ &= -1 \text{ if } k=t \text{ else } 0 && \text{(since } \pi_{s,t}(t)=a\text{)}\end{aligned}$$

where (11) follows from $\mathbf{A}'[a,b]=-1$ and the fact that $\mathbf{A}'$ is a pseudo-transfer matrix. The validity of (ii) $\mathbf{B}_{s,t}$ follows from:

$$\begin{aligned}(\mathbf{B}_{s,t}\cdot\boldsymbol{e}_u)(k) &= \mathbf{B}_{s,t}[k,u]\\ &= \mathbf{A}'[\pi_{s,t}(k),u]=1 && \text{(since } u\in X'\setminus\{s,t\}\text{)}\\ &= 1 \text{ if } \pi_{s,t}(k)=u \text{ else } 0 && \text{(by def. of } \mathbf{A}'\text{)}\\ &= 1 \text{ if } k=u \text{ else } 0 && \text{(since } \pi_{s,t}(u)=u\text{).}\end{aligned}$$

The same proofs apply mutatis mutandis for $\mathbf{C}_t$.

Pseudo-copy matrix. Let us now prove Proposition 3.3 for the case where $\mathbf{A}$ is a pseudo-copy matrix. For this case, we consider ?-implementation and hence $V_X=\mathbb{Z}^{d'}$. Similarly to the case of pseudo-transfer matrices, we claim that for every $\boldsymbol{v}\in V_X$ and every $s,t,u\in X'$ such that $s\neq t$ and $u\notin\{s,t\}$, the following holds:

(1) $(\mathbf{B}_{s,t}\cdot\boldsymbol{v})(t)=(\mathbf{C}_t\cdot\boldsymbol{v})(t)=-\boldsymbol{v}(s)$;
(2) $(\mathbf{B}_{s,t}\cdot\boldsymbol{v})(u)=(\mathbf{C}_t\cdot\boldsymbol{v})(u)=\boldsymbol{v}(u)$.

Indeed, we have:

$$\begin{aligned}(\mathbf{B}_{s,t}\cdot\boldsymbol{v})(t) &= \sum_{\ell\in[d]\cup X'}\mathbf{B}_{s,t}[t,\ell]\cdot\boldsymbol{v}(\ell)\\ &= \sum_{\ell\in[d]\cup X'}\mathbf{A}'[a,\pi_{s,t}(\ell)]\cdot\boldsymbol{v}(\ell) && \text{(since } \pi_{s,t}(t)=a\text{)}\\ &= \mathbf{A}'[a,b]\cdot\boldsymbol{v}(s)+\sum_{\substack{\ell\in[d]\cup X'\\ \ell\neq t}}\mathbf{A}'[a,\pi_{s,t}(\ell)]\cdot\boldsymbol{v}(\ell) && \text{(since } \pi_{s,t}(s)=b\text{)}\\ &= -\boldsymbol{v}(s) && \text{(by } \mathbf{A}'[a,j]\neq 0 \iff j=b\text{),}\end{aligned}$$

where the last equality follows from $\mathbf{A}'[a,b]=-1$ and the fact that $\mathbf{A}'$ is a pseudo-copy matrix. Moreover, we have:

$$\begin{aligned}(\mathbf{B}_{s,t}\cdot\boldsymbol{v})(u) &= \sum_{\ell\in[d]\cup X'}\mathbf{B}_{s,t}[u,\ell]\cdot\boldsymbol{v}(\ell)\\ &= \mathbf{A}'[u,u]\cdot\boldsymbol{v}(u)+\sum_{\substack{\ell\in[d]\cup X'\\ \ell\neq u}}\mathbf{A}'[a,\pi_{s,t}(\ell)]\cdot\boldsymbol{v}(\ell) && \text{(since } \pi_{s,t}(u)=u\text{)}\\ &= \boldsymbol{v}(u) && \text{(by } \mathbf{A}'[u,j]\neq 1 \iff j=u\text{).}\end{aligned}$$

Again, the same proofs apply mutatis mutandis for $\mathbf{C}_t$.

We now prove the proposition. Let $x\in X$ and $\boldsymbol{v}\in V_X$. We obviously have $\mathbf{F}_x\cdot\boldsymbol{v}\in V_X$. If $a\neq b$, we have:

$$\begin{aligned}(\mathbf{F}_x\cdot\boldsymbol{v})(x) &= (\mathbf{B}_{z,x}\cdot(\mathbf{B}_{y,z}\cdot\mathbf{B}_{x,y}\cdot\boldsymbol{v}))(x) && \text{(by def. of } \mathbf{F}_x\text{)}\\ &= -(\mathbf{B}_{y,z}\cdot(\mathbf{B}_{x,y}\cdot\boldsymbol{v}))(z) && \text{(by (1))}\\ &= (\mathbf{B}_{x,y}\cdot\boldsymbol{v})(y) && \text{(by (1))}\\ &= -\boldsymbol{v}(x) && \text{(by (1)),}\end{aligned}$$

and if $a=b$, we have:

$$\begin{aligned}(\mathbf{F}_x\cdot\boldsymbol{v})(x) &= (\mathbf{C}_x\cdot\boldsymbol{v})(x) && \text{(by def. of } \mathbf{F}_x\text{)}\\ &= -\boldsymbol{v}(x) && \text{(by (1)).}\end{aligned}$$

Similarly, by applying (2) repeatedly, we derive $(\mathbf{F}_x\cdot\boldsymbol{v})(y)=\boldsymbol{v}(y)$ for every $y\in X\setminus\{x\}$. □

A.2 Details for the proof of Proposition 3.5

The details of the proof are similar to those of the proof of Proposition 3.3.

Transfer matrix. Let us first prove properties (i) and (ii) stated within the proof of Proposition 3.5 for the case where $\mathbf{A}$ is a transfer matrix. The validity of (i) follows from:

$$\begin{aligned}
(\mathbf{B}_{s,t} \cdot \boldsymbol{e}_s)(k) = 1 &\iff \mathbf{B}_{s,t}[k,s] = 1 \\
&\iff \mathbf{A}'[\pi_{s,t}(k), b] = 1 && \text{(since } \pi_{s,t}(s) = b\text{)} \\
&\iff \pi_{s,t}(k) = a && \text{(since } \mathbf{A} \text{ is a transfer matrix)} \\
&\iff k = t.
\end{aligned}$$

The validity of (ii) follows from:

$$\begin{aligned}
(\mathbf{B}_{s,t} \cdot \boldsymbol{e}_u)(k) = 1 &\iff \mathbf{B}_{s,t}[k,u] = 1 \\
&\iff \mathbf{A}'[\pi_{s,t}(k), u] = 1 && \text{(since } u \in X' \setminus \{s,t\}\text{)} \\
&\iff \pi_{s,t}(k) = u && \text{(by def. of } \mathbf{A}'\text{)} \\
&\iff k = u.
\end{aligned}$$

Copy matrix. Let us now prove Proposition 3.5 for the case where $\mathbf{A}$ is a copy matrix. For this case, we consider ?-implementation and hence $V_X = \mathbb{Z}^{d'}$. Similarly to the case of transfer matrices, we claim that for every $\boldsymbol{v} \in V_X$ and every $s, t, u \in X'$ such that $s \neq t$ and $u \notin \{s,t\}$, the following holds:

(1) $(\mathbf{B}_{s,t} \cdot \boldsymbol{v})(t) = \boldsymbol{v}(s)$;
(2) $(\mathbf{B}_{s,t} \cdot \boldsymbol{v})(u) = \boldsymbol{v}(u)$.

Indeed, we have:

$$\begin{aligned}
(\mathbf{B}_{s,t} \cdot \boldsymbol{v})(t) &= \sum_{\ell \in [d] \cup X'} \mathbf{B}_{s,t}[t,\ell] \cdot \boldsymbol{v}(\ell) \\
&= \sum_{\ell \in [d] \cup X'} \mathbf{A}'[a, \pi_{s,t}(\ell)] \cdot \boldsymbol{v}(\ell) && \text{(since } \pi_{s,t}(t) = a\text{)} \\
&= \mathbf{A}'[a,b] \cdot \boldsymbol{v}(s) + \sum_{\substack{\ell \in [d] \cup X' \\ \ell \neq t}} \mathbf{A}'[a, \pi_{s,t}(\ell)] \cdot \boldsymbol{v}(\ell) && \text{(since } \pi_{s,t}(s) = b\text{)} \\
&= \boldsymbol{v}(s) && \text{(by } \mathbf{A}'[a,j] = 1 \iff j = b\text{)},
\end{aligned}$$

where the last equality follows from $\mathbf{A}'[a,b] = 1$ and the fact that $\mathbf{A}'$ is a copy matrix. Morever, we have:

$$\begin{aligned}
(\mathbf{B}_{s,t} \cdot \boldsymbol{v})(u) &= \sum_{\ell \in [d] \cup X'} \mathbf{B}_{s,t}[u,\ell] \cdot \boldsymbol{v}(\ell) \\
&= \mathbf{A}'[u,u] \cdot \boldsymbol{v}(u) + \sum_{\substack{\ell \in [d] \cup X' \\ \ell \neq u}} \mathbf{A}'[a, \pi_{s,t}(\ell)] \cdot \boldsymbol{v}(\ell) && \text{(since } \pi_{s,t}(u) = u\text{)} \\
&= \boldsymbol{v}(u) && \text{(by } \mathbf{A}'[u,j] = 1 \iff j = u\text{)}.
\end{aligned}$$

We may now prove the proposition. Let $\boldsymbol{v} \in V_X$ and let $x, y \in X$ be such that $x \neq y$. We obviously have $\mathbf{F}_{x,y} \cdot \boldsymbol{v} \in V_X$. Moreover, we have:

$$\begin{aligned}
(\mathbf{F}_{x,y} \cdot \boldsymbol{v})(x) &= (\mathbf{B}_{z,x} \cdot (\mathbf{B}_{x,y} \cdot \mathbf{B}_{y,z} \cdot \boldsymbol{v}))(x) && \text{(by def. of } \mathbf{F}_{x,y}\text{)} \\
&= (\mathbf{B}_{x,y} \cdot (\mathbf{B}_{y,z} \cdot \boldsymbol{v}))(z) && \text{(by (1))} \\
&= (\mathbf{B}_{y,z} \cdot \boldsymbol{v})(z) && \text{(by (2))} \\
&= \boldsymbol{v}(y) && \text{(by (1))},
\end{aligned}$$

and symmetrically $(\mathbf{F}_{x,y} \cdot \boldsymbol{v})(y) = \boldsymbol{v}(x)$. Similarly, by applying (2) three times, we derive $(\mathbf{F}_{x,y} \cdot \boldsymbol{v})(u) = \boldsymbol{v}(u)$ for every $u \in X \setminus \{x,y\}$. □

A.3 Details for the proof of Proposition 3.7

Let us prove the technical lemma invoked within the proof of Proposition 3.7:

Lemma A.1. *For every class C, if C contains a matrix which has a row (resp. column) with at least two nonzero elements, then C also contains a matrix which has a row (resp. column) with at least two nonzero elements with the same sign.*

Proof. We first consider the case of rows. Let $\mathbf{A} \in C$, $i, j, k \in [d]$ and $a, b \neq 0$ be such that $\mathbf{A}[i,j] = a$, $\mathbf{A}[i,k] = b$ and $j \neq k$. If a and b have the same sign, then we are done. Thus, assume without loss of generality that $a < 0$ and $b > 0$.

Let us first give an informal overview of the proof where we see $\mathbf{A}$ as an operation over some counters. We have two counters x and y that we wish to sum up (with some positive integer coefficients), using a supply of counters set to zero. We apply $\mathbf{A}$ to x and some zero counters to produce $a \cdot x$ in some counter (while discarding extra noise into some other ones), and we then apply $\mathbf{A}$ again to $a \cdot x$, y and some zero counters in such a way that we get $a^2 \cdot x + b \cdot y$. The matrix achieving this procedure will have a^2 and b on a common row.

More formally, we wish to obtain a matrix $\mathbf{D}$ with positive entries a^2 and b, and more precisely such that $\mathbf{D}[i,j'] = a^2$ and $\mathbf{D}[i,k] = b$ for some counter j'. Let $\mathbf{B} \coloneqq \mathbf{A}\restriction_d$, $\mathbf{C} \coloneqq \sigma(\mathbf{B})$ and $\mathbf{D} \coloneqq \mathbf{B} \cdot \mathbf{C}$ where $\sigma \colon [2d] \to [2d]$ is the following permutation:

$$\sigma \coloneqq \begin{cases} (j; i; i+d) \cdot \prod_{\ell \in [d] \setminus \{i,j\}} (\ell; \ell + d) & \text{if } j \neq i, \\ \prod_{\ell \in [d] \setminus \{i\}} (\ell; \ell + d) & \text{if } j = i. \end{cases}$$

We claim that $\mathbf{D}$ is as desired. First, observe that:

$$\begin{aligned} \mathbf{D}[i,k] &= \sum_{\ell \in [2d]} \mathbf{B}[i,\ell] \cdot \mathbf{B}[\sigma(\ell), k+d] && \text{(by def. of } \mathbf{D} \text{ and } \sigma(k) = k+d) \\ &= \mathbf{B}[i,k] \cdot \mathbf{B}[k+d, k+d] && \text{(since } \mathbf{B}[\sigma(\ell), k+d] \neq 0 \iff \sigma(\ell) = k+d) \\ &= b && \text{(since } \mathbf{B}[k+d, k+d] = 1). \end{aligned}$$

Thus, $\mathbf{D}$ has a positive entry on row i. It remains to show that $\mathbf{D}$ has another positive entry on row i. We make a case distinction on whether $j = i$.

Case $j \neq i$. Note that $k \neq i + d$. Hence, we are done since:

$$\begin{aligned} \mathbf{D}[i, i+d] &= \sum_{\ell \in [2d]} \mathbf{B}[i,\ell] \cdot \mathbf{B}[\sigma(\ell), j] && \text{(by def. of } \mathbf{D} \text{ and } \sigma(i+d) = j) \\ &= \sum_{\ell \colon \ell, \sigma(\ell) \in [d]} \mathbf{B}[i,\ell] \cdot \mathbf{B}[\sigma(\ell), j] && \text{(since } \mathbf{B} = \mathbf{A}\restriction_d \text{ and } i, j \in [d]) \\ &= \mathbf{B}[i,j] \cdot \mathbf{B}[i,j] && \text{(by def. of } \sigma) \\ &= a^2. \end{aligned}$$

Case $j = i$. Note that $k \neq j = i$. Hence, we are done since:

$$\begin{aligned} \mathbf{D}[i,i] &= \sum_{\ell \in [2d]} \mathbf{B}[i,\ell] \cdot \mathbf{B}[\sigma(\ell), i] && \text{(by def. of } \mathbf{D} \text{ and } \sigma(i) = i) \\ &= \sum_{\ell : \ell, \sigma(\ell) \in [d]} \mathbf{B}[i,\ell] \cdot \mathbf{B}[\sigma(\ell), i] && \text{(since } \mathbf{B} = \mathbf{A}\restriction_d \text{ and } i \in [d]) \\ &= \mathbf{B}[i,i] \cdot \mathbf{B}[i,i] && \text{(by def. of } \sigma) \\ &= a^2 && \text{(since } i = j). \end{aligned}$$

We are done proving the proposition for the case of rows.

For the case of columns, we can instead assume that $\mathbf{A}^\mathsf{T} \in C$, *i.e.* the transpose of $\mathbf{A}$ belongs to C. Since $\mathbf{D}^\mathsf{T}$ is as desired, we simply have to show that $\mathbf{D}^\mathsf{T} \in C$. This is the case since:

$$\begin{aligned}
\mathbf{D}^\mathsf{T} &= (\mathbf{B} \cdot \mathbf{C})^\mathsf{T} \\
&= \mathbf{C}^\mathsf{T} \cdot \mathbf{B}^\mathsf{T} \\
&= (\mathbf{P}_\sigma \cdot \mathbf{B} \cdot \mathbf{P}_{\sigma^{-1}})^\mathsf{T} \cdot \mathbf{B}^\mathsf{T} && \text{(since } \mathbf{C} = \sigma(\mathbf{B})\text{)} \\
&= (\mathbf{P}_\sigma \cdot \mathbf{B}^\mathsf{T} \cdot \mathbf{P}_{\sigma^{-1}}) \cdot \mathbf{B}^\mathsf{T} && \text{(since } \mathbf{P}_{\pi^{-1}} = \mathbf{P}_\pi^\mathsf{T} \text{ for every perm. } \pi\text{)} \\
&= \sigma(\mathbf{B}^\mathsf{T}) \cdot \mathbf{B}^\mathsf{T} \\
&= \sigma((\mathbf{A}\restriction_d)^\mathsf{T}) \cdot (\mathbf{A}\restriction_d)^\mathsf{T} \\
&= \sigma((\mathbf{A}^\mathsf{T})\restriction_d) \cdot (\mathbf{A}^\mathsf{T})\restriction_d && \text{(since } (\mathbf{A}\restriction_d)^\mathsf{T} = (\mathbf{A}^\mathsf{T})\restriction_d\text{).} \\
&\in C && \text{(since } \mathbf{A}^\mathsf{T} \in C\text{).}
\end{aligned}$$

□

A.4 Details for the proof of Theorem 4.5

We prove the missing details for both cases:

Case $\mathbf{A}[\star, i] = \mathbf{0}$. We have $\mathbf{B}_x \cdot \boldsymbol{e}_y = \boldsymbol{e}_y$ since:

$$\begin{aligned}
(\mathbf{B}_x \cdot \boldsymbol{e}_y)(k) &= \mathbf{B}_x[k, y] \\
&= \mathbf{A}'[\sigma_x(k), y] && \text{(since } y \neq x\text{)} \\
&= 1 \iff k = y && \text{(by def. of } \mathbf{A}' \text{ and } \sigma_x\text{).}
\end{aligned}$$

Case $\mathbf{A}[i, \star] = \mathbf{0}$. We have:

$$\begin{aligned}
(\mathbf{B}_x \cdot \boldsymbol{v})(y) &= \sum_{\ell \in [d+n]} \mathbf{B}[y, \ell] \cdot \boldsymbol{v}(\ell) \\
&= \mathbf{A}'[y, y] \cdot \boldsymbol{v}(y) && \text{(by } y \neq x \text{ and def. of } \mathbf{A}' \text{ and } \sigma_x\text{)} \\
&= \boldsymbol{v}(y).
\end{aligned}$$

□