Logical Methods in Computer Science
Volume 21, Issue 1, 2025, pp. 27:1–27:27
https://lmcs.episciences.org/

Submitted Apr. 07, 2023
Published Mar. 20, 2025

LANGUAGE INCLUSION FOR BOUNDEDLY-AMBIGUOUS VECTOR ADDITION SYSTEMS IS DECIDABLE

WOJCIECH CZERWIŃSKI AND PIOTR HOFMAN

University of Warsaw, Poland
e-mail address: wczerwin@mimuw.edu.pl, piotr.hofman@uw.edu.pl

Abstract. We consider the problems of language inclusion and language equivalence for Vector Addition Systems with States (VASS) with the acceptance condition defined by the set of accepting states (and more generally by some upward-closed conditions). In general, the problem of language equivalence is undecidable even for one-dimensional VASS, thus to get decidability we investigate restricted subclasses. On the one hand, we show that the problem of language inclusion of a VASS in k-ambiguous VASS (for any natural k) is decidable and even in Ackermann. On the other hand, we prove that the language equivalence problem is already Ackermann-hard for deterministic VASS. These two results imply Ackermann-completeness for language inclusion and equivalence in several possible restrictions. Some of our techniques can be also applied in much broader generality in infinite-state systems, namely for some subclass of well-structured transition systems.

1. Introduction

Vector Addition Systems (VAS) together with almost equivalent Petri Nets and Vector Addition Systems with States (VASS) are one of the most fundamental computational models with many applications in practice for modelling concurrent behaviour. There is also an active field of theoretical research on VAS, with a prominent example being the reachability problem whose complexity was recently established to be Ackermann-complete [Ler21a, CO21] and [LS19]. An important type of questions that can be asked for any pair of systems is whether they are equivalent in a certain sense. The problem of language equivalence (acceptance by configuration) was already proven to be undecidable in 1975 by Araki and Kasami [AK76] (Theorem 3). They also have shown that the language equivalence (acceptance by configuration) for deterministic VAS is reducible to the reachability problem, thus decidable, as the reachability problem was shown to be decidable by Mayr a few years later in 1981 [May81]. The equality of the reachability sets of two given VAS was also shown undecidable in the 70-ties by Hack [Hac76]. Jančar has proven in 1995 that the most natural behavioural equivalence, namely the bisimilarity equivalence is undecidable for VASS [Jan95].

Key words and phrases: vector addition systems, language inclusion, language equivalence, determinism, unambiguity, bounded ambiguity, Petri nets, well-structured transition systems.

This paper is a full version of [CH22] which appeared at the CONCUR 2022 conference. Both authors are supported by the INFSYS ERC grant, agreement no. 950398.

DOI:10.46298/LMCS-21(1:27)2025
© W. Czerwiński and P. Hofman
Creative Commons

His proof works for only two dimensions (improving the previous results [AK76]) and is applicable also to language equivalence (this time as well for acceptance by states). A few years later in 2001 Jančar has shown in [Jan01] that any reasonable equivalence in-between language equivalence (with acceptance by states) and bisimilarity is undecidable (Theorem 3) and Ackermann-hard even for systems with finite reachability set (Theorem 4). For the language equivalence problem, the state-of-the-art was improved a few years ago. In [HMT13] (Theorem 20) it was shown that already for one-dimensional VASS the language equivalence (and even the trace equivalence, namely language equivalence with all the states accepting) is undecidable.

As the problem of language equivalence (and similar ones) is undecidable for general VASS (even in very small dimensions) it is natural to search for subclasses in which the problem is decidable. The decidability of the problem for deterministic VASS [AK76, May81] suggests that restricting nondeterminism might be a good idea. Recently a lot of attention was drawn to unambiguous systems [Col15], namely systems in which each word is accepted by at most one accepting run, but can potentially have many non-accepting runs. Such systems are often more expressive than the deterministic ones however they share some of their good properties, for example [CFM13]. In particular many problems are more tractable in the unambiguous case than in the general nondeterministic case. This difference is already visible for finite automata. The language universality and the language equivalence problems for unambiguous finite automata are in NC^2 [Tze96] (so also in PTime) while they are in general PSpace-complete for nondeterministic finite automata. Recently it was shown that for some infinite-state systems the language universality, equivalence and inclusion problems are much more tractable in the unambiguous case than in the general one. There was a line of research investigating the problem for register automata [MQ19, BC21, CMQ21] culminating in the work of Bojańczyk, Klin and Moerman [BKM21]. They have shown that for unambiguous register automata with guessing the language equivalence problem is in ExpTime (and in PTime for a fixed number of registers). This result is in a sheer contrast with the undecidability of the problem in the general case even for two register automata without guessing [NSV04] or one register automata with guessing (the proof can be obtained following the lines of [DL09] as explained in [CMQ21]). Recently, it was also shown in [CFH20] that the language universality problem for VASS accepting with states is ExpSpace-complete in the unambiguous case in contrast to Ackermann-hardness in the nondeterministic case (even for one-dimensional VASS) [HT14].

Our contribution. In this article we follow the line of [CFH20] and consider problems of language equivalence and inclusion for unambiguous VASS and also for their generalisations k-ambiguous VASS (for $k \in \mathbb{N}$) in which each word can have at most k accepting runs. The acceptance condition is defined by some upward-closed set of configurations which generalises a bit the acceptance by states considered in [CFH20]. Notice that the equivalence problem can be easily reduced to the inclusion problem, so we prove lower complexity bounds for the equivalence problem and upper complexity bounds for the inclusion problem.

Our only lower bound result is the following one; it is proven by a rather straightforward reduction from the reachability problem for VASS.

Theorem 1.1. *The language equivalence problem for deterministic VASS is* Ackermann-*hard.*

Our main technical result concerns VASS with bounded ambiguity.

Theorem 1.2. *There is an algorithm that for each $k \in \mathbb{N}$ and a k-ambiguous VASS V constructs, in elementary time, a VASS with a downward-closed set of accepting configurations which recognises the complement of the language of V.*

Its rather simple consequence is the following result.

Theorem 1.3. *For each $k \in \mathbb{N}$, the language inclusion problem of a VASS in a k-ambiguous VASS is in* Ackermann.

Briefly speaking to prove Theorem 1.3 we use Theorem 1.2 and check non-emptiness of the intersection of the language of the first VASS with the complement of the language of the second VASS. The problem reduces to the reachability problem in VASS, which is in Ackermann [LS19]. In Section 5 we present the details of the proof. The special case of Theorem 1.3 is the following statement.

Theorem 1.4. *The inclusion problem of a nondeterministic VASS language in an unambiguous VASS language is in* Ackermann.

Despite the fact that Theorem 1.4 is a special case of Theorem 1.3 we decided to present separately the proof of Theorem 1.4 because it introduces a different technique of independent interest. Additionally, it is a good introduction to a more technically challenging proof of Theorem 1.3.

The proof of Theorem 1.4 uses a novel technique but is quite simple. We add a regular lookahead to a VASS and use results on regular separability of VASS from [CLM$^+$18] to reduce the problem, roughly speaking, to the deterministic case. This technique can be applied to more general systems namely well-structured transition-systems [FS01]. We believe that it might be interesting on its own and reveal some connection between separability problems and the notion of unambiguity.

The proof of Theorem 1.2 proceeds in three steps. First, we show that the problem for k-ambiguous VASS can be reduced to the case when the control automaton of the VASS is k-ambiguous. Next, we show that the control automaton can even be made k-deterministic (roughly speaking, for each word there are at most k runs). Finally, we show that for a VASS with k-deterministic control automaton one can compute in elementary time a VASS with a downward-closed set of accepting configurations recognising its complement. We emphasise that to prove our main technical result, namely Theorem 1.2, we do not need most of our contribution from Section 4. In fact, the only used part of Section 4 is the construction of the word and automata decoration.

On a way to show Theorem 1.2 we also prove several other lemmas and theorems, which we believe may be interesting on their own. Theorems 1.1 and 1.3 together easily imply the following corollary.

Corollary 1.5. *The language equivalence problem is* Ackermann-*complete for:*

- *deterministic VASS*
- *unambiguous VASS*
- *k-ambiguous VASS for any $k \in \mathbb{N}$*

Organisation of the paper. In Section 2 we introduce the necessary notions. Then in Section 3 we present results concerning deterministic VASS. First, we show Theorem 1.1. Next, we prove that the inclusion problem of a VASS language in a language of a deterministic VASS, a k-control-deterministic VASS or a VASS with holes (to be defined) is in Ackermann.

This is achieved by reducing to the VASS reachability problem. In Section 4 we define adding a regular lookahead to VASS. Then we show that with a carefully chosen lookahead we can reduce the inclusion problem of a VASS language in an unambiguous VASS language into the inclusion problem of a VASS language in the language of deterministic VASS with holes. The latter is in Ackermann due to Section 3 so the former is also in Ackermann. In Section 5 we present the proof of Theorem 1.2 which is our most technically involved contribution. We also use the idea of a regular lookahead and the result proved in Section 3 about k-control-deterministic VASS. Finally in Section 6 we discuss the implications of our results and sketch possible future research directions.

2. Preliminaries

Basic notions. By $\mathbb{N}$ and $\mathbb{Z}$ we denote natural and integer numbers, respectively. For $a, b \in \mathbb{N}$ we write $[a, b]$ to denote the set $\{a, a+1, \ldots, b-1, b\}$. For a vector $v \in \mathbb{N}^d$ and $i \in [1, d]$ we write $v[i]$ to denote the i-th coordinate of vector v. By 0^d we denote the vector $v \in \mathbb{N}^d$ with all the coordinates equal to zero. For a word $w = a_1 \cdot \ldots \cdot a_n$ and $1 \leq i \leq j \leq n$ we write $w[i..j] = a_i \cdot \ldots \cdot a_j$ for the infix of w starting at position i and ending at position j. We also write $w[i] = a_i$. For any $1 \leq i \leq d$ by $e_i \in \mathbb{N}^d$ we denote the vector with all coordinates equal to zero except the i-th coordinate, which is equal to one. For a finite alphabet Σ we denote $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ the extension of Σ by the empty word ε.

Upward and downward-closed sets. For two vectors $u, v \in \mathbb{N}^d$ we say that $u \preceq v$ if for all $i \in [1, d]$ we have $u[i] \leq v[i]$. A set $S \subseteq \mathbb{N}^d$ is *upward-closed* if for each $u, v \in \mathbb{N}^d$ it holds that $u \in S$ and $u \preceq v$ implies $v \in S$. Similarly, a set $S \subseteq \mathbb{N}^d$ is *downward-closed* if for each $u, v \in \mathbb{N}^d$ it holds that $u \in S$ and $v \preceq u$ implies $v \in S$. For $u \in \mathbb{N}^d$ we write $u{\uparrow} = \{v \mid u \preceq v\}$ for the set of all vectors bigger than u w.r.t. $\preceq$ and $u{\downarrow} = \{v \mid v \preceq u\}$ for the set of all vectors smaller than u w.r.t. $\preceq$. If an upward-closed set is of the form $u{\uparrow}$, we call it an *up-atom*. Notice that if a one-dimensional set $S \subseteq \mathbb{N}$ is downward-closed then either $S = \mathbb{N}$ or $S = [0, n]$ for some $n \in \mathbb{N}$. In the first case, we write $S = \omega{\downarrow}$ and in the second case $S = n{\downarrow}$. If a downward-closed set $D \subseteq \mathbb{N}^d$ is of a form $D = D_1 \times \ldots \times D_d$, where all D_i for $i \in [1, d]$ are downward-closed one dimensional sets then we call D a *down-atom*. In the literature, sometimes *up-atoms* are called principal filters, and *down-atoms* are called ideals. If $D_i = (n_i){\downarrow}$, then we also write $D = (n_1, n_2, \ldots, n_d){\downarrow}$. In that sense, each down-atom is of the form $u{\downarrow}$ for $u \in (\mathbb{N} \cup \{\omega\})^d$. Notice that a down-atom does not have to be of a form $u{\downarrow}$ for $u \in \mathbb{N}^d$, for example $D = \mathbb{N}^d$ is not of this form, but $D = (\omega, \ldots, \omega){\downarrow}$.

The following two propositions will be helpful in our considerations.

Proposition 2.1 ([CLM+18] Lemma 17, [KP92], [Dic13])**.** *Each downward-closed set in $\mathbb{N}^d$ is a finite union of down-atoms. Similarly, each upward-closed set in $\mathbb{N}^d$ is a finite union of up-atoms.*

We represent upward-closed sets as finite unions of up-atoms and downward-closed sets as finite unions of down-atoms, the numbers are encoded in binary. The size of the representation of an upward- or a downward-closed set S is denoted $||S||$. The following proposition helps to control the blowup of the representations of upward- and downward-closed sets.

Proposition 2.2. *Let $U \subseteq \mathbb{N}^d$ be an upward-closed set and $D \subseteq \mathbb{N}^d$ be a downward-closed set. Then the size of the representation of their complements $\overline{U} = \mathbb{N}^d \setminus U$ and $\overline{D} = \mathbb{N}^d \setminus D$ is at most exponential w.r.t. the sizes $||U||$ and $||D||$, respectively and can be computed in exponential time.*

Proof. Here we present only the proof for the complement of the upward-closed set U as the case for downward-closed sets follows the same lines. Let $U = u_1{\uparrow} \cup u_2{\uparrow} \cup \ldots \cup u_n{\uparrow}$. The complement of U is a downward closed set, that can be represented as a union of down-atoms. As each down-atom is represented as a downward closure of a single vector over $\mathbb{N} \cup \{\omega\}$ it is sufficient to calculate these vectors. Each vector x representing a down-atom has two properties:

(1) it is in the complement of U i.e. for every $i \in \{1, \ldots, n\}$ there is a coordinate $f(i) \in \{1, \ldots, d\}$ such that $x[f(i)] < u_i[f(i)]$.
(2) it is maximal, i.e. for every $y \neq x$ that satisfies the first property it holds $x \not\preceq y$.

Intuitively, function f witnesses why x is not in U.

Thus, for every function $f : \{1, \ldots, n\} \to \{1, \ldots, d\}$ we can define one down-atom represented by some vector x_f as the maximal vector, for which function f witnesses that $x_f \notin U$. For a coordinate $i \in [1, d]$ we need that $x_f[i] < u_j[i]$ for each $j \in [1, n]$ such that $j = f(i)$. Therefore

$$x_f[i] = \min\{u_j[i] - 1 \mid j \in [1, n], f(j) = i\},$$

where we put $\min(\emptyset) = \omega$. Notice that the union of $x_f{\downarrow}$ is exactly the complement of U.

Observe that there is at most exponentially many different functions $f \in \{1 \ldots n\} \to \{1, \ldots, d\}$ and numbers that appear in descriptions of vectors x_f are ω or are smaller than the maximal number that appear in the description of U. Therefore, we conclude that the proposition is true. □

For a more general study (for arbitrary well-quasi orders) see [GLHK+20].

Vector Addition Systems with States. A d-dimensional Vector Addition System with States (d-VASS or simply VASS) V consists of a finite *alphabet* Σ, a finite set of *states* Q, a finite set of *transitions* $T \subseteq Q \times \Sigma \times \mathbb{Z}^d \times Q$, a distinguished *initial configuration* $c_I \in Q \times \mathbb{N}^d$, and a set of distinguished *final configurations* $F \subseteq Q \times \mathbb{N}^d$. We write $V = (\Sigma, Q, T, c_I, F)$. Sometimes we ignore some of the components in a VASS if they are not relevant, for example we write $V = (Q, T)$ if Σ, c_I, and F do not matter. A *configuration* of a d-VASS is a pair $(q, v) \in Q \times \mathbb{N}^d$, we often write it $q(v)$ instead of (q, v). We write $\text{state}(q(v)) = q$. The set of all configurations is denoted $\text{Conf} = Q \times \mathbb{N}^d$. For a state $q \in Q$ and a set $U \subseteq \mathbb{N}^d$ we write $q(U) = \{q(u) \mid u \in U\}$. A transition $t = (p, a, u, q) \in T$ can be *fired* in the configuration $r(v)$ if $p = r$ and $u + v \in \mathbb{N}^d$. Then we write $p(v) \xrightarrow{t} q(u + v)$. We say that the transition $t \in T$ is *over* the letter $a \in \Sigma$ or the letter a *labels* the transition t. We write $p(v) \xrightarrow{a} q(u + v)$ slightly overloading the notation, when we want to emphasise that the transition is over the letter a. The *effect* of a transition $t = (p, a, u, q)$ is the vector u, we write $\text{eff}(t) = u$. The size of VASS V is the total number of bits needed to represent the tuple (Σ, Q, T, c_I, F), we do not specify here how we represent F as it may depend a lot on the form of F. A sequence $\rho = (c_1, t_1, c_1'), (c_2, t_2, c_2'), \ldots, (c_n, t_n, c_n') \in (\text{Conf} \times T \times \text{Conf})^*$ is a *run* of VASS $V = (Q, T)$ if for all $i \in [1, n]$ we have $c_i \xrightarrow{t_i} c_i'$ and for all $i \in [1, n-1]$ we have $c_i' = c_{i+1}$. We write $\text{trans}(\rho) = t_1 \cdot \ldots \cdot t_n$. We extend the notion of *labelling* to runs; Labelling of a run ρ is a word w which is the concatenation of labels of transitions of ρ, we also say that the run ρ is

over the word w. This run ρ, if not empty, is *from* the configuration c_1 *to* the configuration c'_n and the configuration c'_n is *reachable* from the configuration c_1 by the run ρ. We write then $c_1 \xrightarrow{\rho} c'_n$, $c_1 \xrightarrow{w} c'_n$ if w labels ρ slightly overloading the notation or simply $c_1 \longrightarrow c'_n$ if the run ρ is not relevant. The empty run can go from any configuration to itself, and it is labeled with ε.

VASS languages. A run ρ is *accepting* if it is from the initial configuration to some final configuration. For a VASS $V = (\Sigma, Q, T, c_I, F)$ we define the language of V as the set of all labellings of accepting runs, namely

$$L(V) = \{w \in \Sigma^* \mid c_I \xrightarrow{w} c_F \text{ for some } c_F \in F\}.$$

For any configuration c of V we define the *language of configuration* c, denoted $L_c(V)$ to be the language of VASS (Σ, Q, T, c, F), namely the language of VASS V with the initial configuration c_I substituted by c. Sometimes we simply write $L(c)$ instead of $L_c(V)$ if V is clear from the context. Further, we say that the *configuration c has the empty language* if $L(c) = \emptyset$. For a VASS $V = (\Sigma, Q, T, c_I, F)$ its *control automaton* is intuitively VASS V after ignoring its counters. Precisely speaking, the control automaton is (Σ, Q, T', q_I, F') where $q_I = \text{state}(c_I)$, $F' = \{q \in Q \mid \exists_{v \in \mathbb{N}^d}\ q(v) \in F\}$ and for each $(q, a, v, q') \in T$ we have $(q, a, q') \in T'$.

Notice that a 0-VASS, namely a VASS with no counters is just a finite automaton, so all the VASS terminology works also for finite automata. In particular, a configuration of a 0-VASS is simply an automaton state. In that special case for each state $q \in Q$ we call the $L(q)$ the language of state q.

A VASS is *deterministic* if for each configuration c reachable from the initial configuration c_I and for each letter $a \in \Sigma$ there is at most one configuration c' such that $c \xrightarrow{a} c'$. A VASS is *k-ambiguous* for $k \in \mathbb{N}$ if for each word $w \in \Sigma^*$ there are at most k accepting runs over w. If a VASS is 1-ambiguous we also call it *unambiguous*.

Note that, the set of languages accepted by unambiguous VASS is a strict superset of the languages accepted by deterministic VASS. To see that unambiguous VASS can accept more, consider a language $(a^*b)^*a^nb^m$ where $n \geq m$. On the one hand, an unambiguous VASS that accepts the language, guesses where the last block of letter a starts, then it counts the number of a's in this last block, and finally counts down reading b's. As there is exactly one correct guess, this VASS is unambiguous. On the other hand, deterministic system can not accept the language, as intuitively speaking it does not know whether the last block of a's has already started or not. To formulate the argument precisely one should use rather easy pumping techniques.

An example language which can be recognised by k-ambiguous VASS, but probably cannot be recognised by a $(k-1)$-ambiguous VASS is the following: language of words over $\{a, b\}$ with exactly k letters b, which divides the word into $k+1$ blocks consisting of letters a, such that the first block is not the shortest one. A k-ambiguous 1-VASS increments the counter on each letter a in the first block, later guesses a block which is the shortest one, and in that guessed block it decrements the counter. As there are at most k other blocks, which may be guessed, there are at most k accepting runs for each word. However, it is not clear how to accept this language by a $(k-1)$-ambiguous VASS. The problematic scenario is if the first block is the longest one.

The following two problems are the main focus of this paper, for different subclasses of VASS:

Inclusion problem for VASS:
Input: Two VASS V_1 and V_2.
Question: Does $L(V_1) \subseteq L(V_2)$ hold?

Equivalence problem for VASS:
Input: Two VASS V_1 and V_2.
Question: Does $L(V_1) = L(V_2)$ hold?

In the sequel, we are mostly interested in VASS with the set of final configurations F of some special form. We extend the order $\preceq$ on the vectors of $\mathbb{N}^d$ to configurations from $Q \times \mathbb{N}^d$ in a natural way: we say that $q_1(v_1) \preceq q_2(v_2)$ if $q_1 = q_2$ and $v_1 \preceq v_2$. We define the notions of upward-closed, downward-closed, up-atom and down-atom the same as for vectors. As Proposition 2.1 holds for any well quasi-order, it also applies to $Q \times \mathbb{N}^d$. Proposition 2.2 applies here as well, as the upper bound on the size can be shown separately for each state. Let the set of final configurations of VASS V be F. If F is upward-closed then we call V an *upward-VASS*. If F is closed downward, then we call V a *downward-VASS*. For two sets $A \subseteq \mathbb{N}^a$, $B \subseteq \mathbb{N}^b$ and a subset of coordinates $J \subseteq [1, a+b]$ by $A \times_J B$ we denote the set of vectors in $\mathbb{N}^{a+b}$ which projected into coordinates in J belong to A and projected into coordinates outside J belong to B. If $F = \bigcup_{i \in [1,n]} q_i(U_i \times_{J_i} D_i)$ where for all $i \in [1, n]$ we have $J_i \subseteq [1, d]$, $U_i \subseteq \mathbb{N}^{|J_i|}$ are up-atoms and $D_i \subseteq \mathbb{N}^{d-|J_i|}$ are down-atoms then we call V an *updown-VASS*. In the sequel we write simply $\times$ instead of $\times_J$, as the set of coordinates J is never relevant. If $F = \{c_F\}$ is a singleton, then we call V a *singleton-VASS*.
Notation. As in this paper we mostly work with upward-VASS we often say simply a VASS instead of an upward-VASS. In other words, if not indicated otherwise, we assume that the set of final configurations F is upward-closed.

For the complexity analysis we assume that whenever F is upward- or downward-closed then it is given as a union of atoms. If $F = \bigcup_{i \in [1,n]} q_i(U_i \times D_i)$ then in the input we get a sequence of q_i and representations of atoms U_i, D_i defining individual sets $q_i(U_i \times D_i)$.

Language emptiness problem for VASS. The following emptiness problem is the central problem for VASS.

Emptiness problem for VASS
Input: A VASS $V = (\Sigma, Q, T, c_I, F)$
Question: Is c_F reachable from c_I in V for some $c_F \in F$?

Observe that the emptiness problem is not influenced in any way by labels of the transitions, so sometimes we will not even specify transition labels when we work with the emptiness problem. If we want to emphasise that transition labels do not matter for some problem, then we write $V = (Q, T, c_I, F)$ ignoring the Σ component. In such cases we also assume that transitions do not contain the Σ component, namely $T \subseteq Q \times \mathbb{Z}^d \times Q$.

Note also that the celebrated reachability problem and the coverability problem for VASS are special cases of the emptiness problem. The reachability problem is the case when F is a singleton set $\{c_F\}$, classically it is formulated as the question whether there is a run from c_I to c_F. The coverability problem is the case when F is an up-atom c_F, classically it is formulated as the question whether there is a run from c_I to any c such that $c_F \preceq c$. Recall that the reachability problem, so the emptiness problem for singleton-VASS is in Ackermann [LS19] and actually Ackermann-complete [Ler21a, CO21].

A special case of the emptiness problem is helpful for us in Section 3.

Lemma 2.3. *The emptiness problem for VASS with the acceptance condition $F = q_F(U \times D)$ where D is a down-atom and U is an up-atom is in* Ackermann.

Proof. We provide a polynomial reduction of the problem to the emptiness problem in singleton-VASS which is in Ackermann. Let $V = (Q, T, c_I, q_F(U \times D))$ be a d-VASS with up-atom $U \subseteq \mathbb{N}^{d_1}$ and down-atom $D \subseteq \mathbb{N}^{d_2}$ such that $d_1 + d_2 = d$. Let $U = u\uparrow$ for some $u \in \mathbb{N}^{d_1}$ and let $D = v\downarrow$ for some $v \in (\mathbb{N} \cup \{\omega\})^{d_2}$. Let us assume wlog of generality that $d_2 = d_U + d_B$ such that for $i \in [1, d_U]$ we have $v[i] = \omega$ and for $i \in [d_U + 1, d_2]$ we have $v[i] \in \mathbb{N}$. Let a d-VASS V' be the VASS V slightly modified in the following way. First we add a new state q'_F and a transition $(q_F, 0^d, q'_F)$. Next, for each coordinate $i \in [1, d_1]$ we add a loop in state q'_F (transition from q'_F to q'_F) with the effect $-e_i$, namely the one decreasing the coordinate i, these are the coordinates corresponding to the up-atom U. Similarly for each coordinate $i \in [d_1 + 1, d_1 + d_U]$ we add in q'_F a loop with the effect $-e_i$, these are the unbounded coordinates corresponding to the down-atom D. Finally, for each coordinate $i \in [d_1 + d_U + 1, d]$ we add in q'_F a loop with the effect e_i (notice that this time we increase the counter values); these are the bounded coordinates corresponding to down-atom D. Let the initial configuration of V' be c_I (the same as in V) and the set of final configurations F' of V' be the singleton set containing $q'_F(u, (0^{d_U}, v[d_U + 1], \ldots, v[d_U + d_B]))$. Clearly V' is a singleton-VASS, so the emptiness problem for V' is in Ackermann. It is easy to see that the emptiness problems in V and in V' are equivalent, which finishes the proof. □

The following is a simple and useful corollary of Lemma 2.3.

Corollary 2.4. *The emptiness problem for updown-VASS is in* Ackermann.

Proof. Recall that for updown-VASS the acceptance condition is a finite union of $q(U \times D)$ for some up-atom $U \subseteq \mathbb{N}^{d_1}$ and down-atom $D \subseteq \mathbb{N}^{d_2}$ where d_1 and d_2 sums to the dimension of the VASS V. Thus, emptiness of the updown-VASS can be reduced to finitely many emptiness queries of the form $q(U \times D)$, which can be decided in Ackermann due to Lemma 2.3. Notice that the number of queries is not bigger than the size of the representation of F thus the emptiness problem for updown-VASS is also in Ackermann. □

By Proposition 2.1 each downward-VASS is also an updown-VASS, thus Corollary 2.4 implies the following one.

Corollary 2.5. *The emptiness problem for downward-VASS is in* Ackermann.

Recall that the coverability problem in VASS is in ExpSpace [Rac78], and the coverability problem is equivalent to the emptiness problem for the set of final configurations being an up-atom. By Proposition 2.1 we have the following simple corollary which creates an elegant duality for the emptiness problems in VASS.

Corollary 2.6. *The emptiness problem for upward-VASS is in* ExpSpace.

Actually, even the following stronger fact is true and helpful for us in the remaining part of the paper. The following proposition is an easy consequence of Corollary 4.6 from [LS21].

Proposition 2.7. *For each upward-VASS the representation of the downward-closed set of configurations with the empty language can be computed in doubly-exponential time.*

Deciding unambiguity and k-ambiguity for VASS. It is a natural question to ask whether a given VASS is unambiguous, or more generally, k-ambiguous for a given $k \in \mathbb{N}$.

Deciding it can be rather easily reduced to the language emptiness problem (and thus to the coverability or the reachability problem for VASS). Intuitively, to check whether a VASS V is unambiguous, one can construct a VASS V' that accepts words having at least two different runs in V. Then V is unambiguous if and only if the language of V' is empty. VASS V' essentially speaking simulates two copies of V and additionally keeps in its state information whether these runs have already differed or not. Thus, for VASS accepting by a set of states, the unambiguity problem can be decided in ExpSpace, for details, see Proposition 20 in [CFH20]. The same complexity can be achieved for upward-VASS in exactly the same way. Also, in the same way, for downward-VASS the unambiguity problem is reduced to the reachability problem for VASS, which is in Ackermann [LS19].

A very similar construction can be used to decide the k-ambiguity of the upward- or downward-VASS for any fixed k. For a d-VASS V one can construct a $d(k+1)$-VASS V', which simulates $k+1$ copies of V, makes sure that all of them accept and runs in all the copies are different. Then V is not k-ambiguous if and only if the language of V' is nonempty. In turn, the problem of checking k-ambiguity is in ExpSpace for upward-VASS and in Ackermann for downward-VASS.

3. Deterministic VASS

3.1. Lower bound.

First we prove a lemma, which easily implies Theorem 1.1.

Lemma 3.1. *For each d-dimensional singleton-VASS V with final configuration being $c_F = q_F(0^d)$ one can construct in polynomial time two deterministic $(d+1)$-dimensional upward-VASSes V_1 and V_2 such that*

$$L(V_1) = L(V_2) \iff L(V) = \emptyset.$$

Notice that Lemma 3.1 shows that the emptiness problem for a singleton-VASS with the final configuration having zero counter values can be reduced in polynomial time to the language equivalence for deterministic VASS. This proves Theorem 1.1 as the emptiness problem, even with zero counter values of the final configuration is Ackermann-hard [Ler21a, CO21]. Also, Lemma 3.1 implies hardness in fixed dimensional VASS. For example the language equivalence problem for deterministic $(2d+5)$-VASS is $\mathcal{F}_d$-hard as the emptiness problem is $\mathcal{F}_d$-hard for $(2d+4)$-dimensional singleton-VASS [Ler21b].

Proof of Lemma 3.1. We first sketch the proof. To show the lemma, we take V and add to it one transition labelled with a new letter. In V_1 the added transition can be performed if we have reached a configuration bigger than or equal to c_F. In V_2 the added transition can be performed only if we have reached a configuration strictly bigger than c_F. Then it is easy to see that $L(V_1) \neq L(V_2)$ if and only if c_F can be reached. A detailed proof follows.

For a given $V = (Q, T, c_I, c_F)$ we construct $V_1 = (\Sigma = T \cup \{a\}, Q \cup \{q'_F\}, T' \cup \{t_1\}, c'_I, q'_F(0^{d+1}\uparrow))$, and $V_2 = (\Sigma = T \cup \{a\}, Q \cup \{q'_F\}, T' \cup \{t_2\}, c'_I, q'_F(0^{d+1}\uparrow))$. Notice that V_1 and V_2 are pretty similar to each other and also to V. Both V_1 and V_2 have the same states as V plus one additional state q'_F. Notice that the alphabet of labels of V_1 and V_2 is the set of transitions T of V plus one additional letter a. For each transition $t = (p, v, q) \in T$ of V we create a transition $(p, t, v', q) \in T'$ where

- for each $i \in [1, d]$ we have $v'[i] = v[i]$; and
- $v'[d+1] = v[1] + \ldots + v[d]$,

so v' is identical as v on the first d coordinates and on the last $(d+1)$-th coordinate it keeps the sum of all the others. Notice that the transitions in T' are used both in V_1 and in V_2.

We also add one additional transition t_1 to V_1 and one t_2 to V_2. To V_1 we add a new a-labelled transition from q_F to q'_F with the effect 0^{d+1} for the additional letter a. To V_2 we also add an a-labelled transition between q_F and q'_F, but with an effect equal $(0^d, -1)$. This -1 on the last coordinate is the only difference between V_1 and V_2. The starting configuration in both V_1 and V_2 is $c'_I = q_I(x_1, x_2, \dots x_d, \sum_{i=1}^{d} x_i)$ where $c_I = q_I(x_1, x_2, \dots x_d)$. The set of accepting configurations is the same in both V_1 and V_2, namely it is $q'_F(0^{d+1}\uparrow)$. Notice that both V_1 and V_2 are deterministic upward-VASS, as required in the lemma statement.

Now we aim to show that $L(V_1) = L(V_2)$ if and only if $L(V) = \emptyset$. First, observe that $L(V_1) \supseteq L(V_2)$. Clearly if $w \in L(V_2)$ then $w = ua$ for some $u \in T^*$, where T is the set of transitions of V. For any word $ua \in L(V_2)$, we have

$$c'_I \xrightarrow{u} q_F(v) \xrightarrow{a} q'_F(v - e_{d+1})$$

in V_2. But, then we have also

$$c'_I \xrightarrow{u} q_F(v) \xrightarrow{a} q'_F(v)$$

in V_1. Thus $ua \in L(V_1)$.

Now we show that, if $L(V) \neq \emptyset$, so $c_I \longrightarrow q_F(0^d)$ in V then $L(V_1) \neq L(V_2)$. Let the run ρ of V be such that $c_I \xrightarrow{\rho} q_F(0^d)$ and let $u = \mathrm{trans}(\rho) \in T^*$. Then clearly $c'_I \xrightarrow{u} q_F(0^{d+1}) \xrightarrow{a} q'_F(0^{d+1})$ and $ua \in L(V_1)$. However $ua \notin L(V_2)$ as the last coordinate on the run of V_2 over ua corresponding to ρ would go below zero and this is the only possible run of V_2 over ua due to determinism of V_2.

It remains to show that if $L(V) = \emptyset$, so $c_I \not\longrightarrow q_F(0^d)$ in V, then $L(V_1) \subseteq L(V_2)$. Let $w \in L(V_1)$. Then $w = ua$ for some $u \in T^*$. Let $c'_I \xrightarrow{\rho} c$ in V_1 such that $\mathrm{trans}(\rho) = u$. As $ua \in L(V_1)$ we know that $c = q_F(v)$. However as $c_I \not\longrightarrow q_F(0^d)$ in V we know that $v \neq 0^{d+1}$. In particular $v[d+1] > 0$. Therefore $w = ua \in L(V_2)$ as the last transition over a may decrease the $(d+1)$-th coordinate and reach an accepting configuration. This finishes the proof. □

3.2. **Upper bounds.** In this Section we prove three results of the form: if V_1 is a VASS and V_2 is a VASS of some special type then deciding whether $L(V_1) \subseteq L(V_2)$ is in Ackermann. Our approach to these problems is the same, namely we first prove that complement of $L(V_2)$ for V_2 of the special type is also a language of some VASS V'_2. Then to decide the inclusion problem it is enough to construct VASS V such that $L(V) = L(V_1) \cap L(V'_2) = L(V_1) \setminus L(V_2)$ and check it for emptiness. In the description above, using the term VASS we do not specify the form of its set of accepting configurations. Starting from now on, we call upward-VASS simply VASS and for VASS with other acceptance conditions we use their full name (like downward-VASS or updown-VASS) to distinguish them from upward-VASS. The following lemma is very useful in our strategy of deciding the inclusion problem for VASS languages.

Lemma 3.2. *There is an algorithm that for a VASS V_1 and a downward-VASS V_2 constructs in polynomial time an updown-VASS V such that $L(V) = L(V_1) \cap L(V_2)$.*

Proof. We construct V as the standard synchronous product of V_1 and V_2. The set of accepting configurations in V is also the product of accepting configurations in V_1 and

accepting configurations in V_2, thus due to Proposition 2.1 a finite union of $q(U \times D)$ for a state q of V, an up-atom U and a down-atom D. □

Deterministic VASS. We first show the following theorem, which will be generalised by the other results in this section. We aim to prove it independently in order to mildly introduce our techniques.

Theorem 3.3. *There is algorithm, that for a deterministic VASS V constructs, in exponential time, a downward VASS that recognises the complement of the language of V.*

Proof of Theorem 3.3. Let $V = (\Sigma, Q, T, c_I, F)$ be a deterministic d-VASS. We aim at constructing a d-dimensional downward-VASS V' such that $L(V') = \overline{L(V)}$. Before constructing V' let us observe that there are three possible scenarios for a word w to be not in $L(V)$. The first scenario (1) is that the only run over w in V finishes in a non-accepting configuration. Another possibility is that there is even no run over w. Namely, for some prefix va of w where $v \in \Sigma^*$ and $a \in \Sigma$ we have $c_I \xrightarrow{v} c$ for some configuration c but there is no transition from c over the letter a as either (2) all possible transitions over a would decrease some of the counters below zero, (3) there is no such transition possible in V in the state of c. For each case we separately design a part of a downward-VASS accepting it. Cases (1) and (3) are simple. For case (2) we nondeterministically guess the moment when the run would go below zero and freeze the configuration at that moment. Then, at the end of the word, we check if our guess was correct. Notice that the set of configurations from which a step labelled with a letter a would take a counter below zero is downward-closed, so we can check the correctness of our guess using a downward-closed accepting condition. A detailed proof follows.

We are ready to describe VASS $V' = (\Sigma, Q', T', c'_I, F')$. Basically speaking, it consists of $2 \cdot |\Sigma| + 1$ copies of V. Concretely, the set of states Q' is the set of pairs $Q \times (\Sigma \times \{2, 3\} \cup \{-\})$. Let $c_I = q_I(v_I)$. Then let $q'_I \in Q'$ be defined as $q'_I = (q_I, -)$ and we define the initial configuration of V as $c'_I = q'_I(v_I)$. The set of accepting configurations $F' = F_1 \cup F_2 \cup F_3$ is a union of three sets F_i, each set F_i for $i \in \{1, 2, 3\}$ is responsible for accepting words rejected by VASS V due to the scenario (i) described above. We successively describe which transitions are added to T' and which configurations are added to F' in order to appropriately handle various scenarios.

We first focus on words fulfilling the scenario (1). For states of a form $(q, -)$ the VASS V' is just as V. Namely for each transition $(p, a, v, q) \in T$ we add (p', a, v, q') to T' where $p' = (p, -)$ and $q' = (q, -)$. We also add to F' the following set $F_1 = \{(q, -)(v) \mid q(v) \notin F\}$. It is easy to see that words that fulfil scenario (1) above are accepted in V' by the use of the set F_1. The size of the description of F_1 is at most exponential compared to the size of the description of F by Proposition 2.2.

Now we describe the second part of V' which is responsible for words rejected by V because of the scenario (2). The idea is to guess when the run over w is finished. For each label $a \in \Sigma$ we add $(p', a, 0^d, p'')$ to T' where $p' = (p, -)$ and $p'' = (p, (a, 2))$. The idea is that the run reaches the configuration in which the transition labeled with a cannot be fired. Now we have to check that our guess is correct. In the state $(p, (a, 2))$ for $t \in T$ no transition changes the configuration. Namely for each $p'' = (p, (a, 2)) \in Q \times \Sigma$ and each $b \in \Sigma$ we add to T' a transition $(p'', b, 0^d, p'')$. We now add to F' the set $F_2 = \{(p, (a, 2))(v) \mid v + \mathrm{eff}(t) \notin \mathbb{N}^d$ for all $t = (p, a, \mathrm{eff}(t), q) \in T$ where $q \in Q\}$. Notice that F_2 can be easily represented as a polynomial union of down-atoms. It is easy to see that

indeed V' accepts by F_2 exactly words w such that there is a run of V over some prefix v of w but reading the next letter would decrease one of the counters below zero. The last part of V' is responsible for the words w rejected by V because of the scenario (3), that is, w has a prefix va such that there is a run over $v \in \Sigma^*$ in V but then in the state of the reached configuration there is no transition over the letter $a \in \Sigma$. To accept such words for each state $p \in Q$ and letter $a \in \Sigma$ such that there is no transition of a form $(p, a, v, q) \in T$ for any $v \in \mathbb{Z}^d$ and $q \in Q$ we add to T' transition $((p, -), a, 0^d, (p, (a, 3)))$. In each state $p' = (p, (a, 3)) \in Q \times \Sigma$, we have a transition $(p', b, 0^d, p')$ for each $b \in \Sigma$. We also add to F' the set $F_3 = \{(p, (a, 3))(v) \mid v \in \mathbb{N}^d$ and there is no $(p, a, u, q) \in T$ for $u \in \mathbb{Z}^d$ and $q \in Q\}$. The size of F_3 is polynomial w.r.t. T.

Summarising V' with the accepting downward-closed set $F = F_1 \cup F_2 \cup F_3$ indeed satisfies $L(V') = \overline{L(V)}$, which finishes the construction and the proof. □

The following theorem is a simple corollary of Theorem 3.3, Lemma 3.2 and Corollary 2.4.

Theorem 3.4. *The inclusion problem of a VASS language in a deterministic VASS language is in* Ackermann.

Deterministic VASS with holes. We define here VASS with holes, which are a useful tool to obtain our results about unambiguous VASS in Section 4. A *d-VASS with holes* (or shortly d-HVASS) V is defined exactly as a standard VASS, but with an additional downward-closed set $H \subseteq Q \times \mathbb{N}^d$ which affects the semantics of V. Namely the set of configurations of V is $Q \times \mathbb{N}^d \setminus H$. Thus each configuration on a run of V needs not only to have nonnegative counters, but in addition to that it can not be in the set of *holes* H. Additionally in HVASS we allow for transitions labelled by the empty word ε, in contrast to the rest of our paper. Due to that fact in this paragraph we often work also with VASS having ε-labelled transitions, we call such VASS the ε-VASS. As an illustration of the HVASS notion, let us consider the zero-dimensional case. In that case, the set of holes is just a subset of states. Therefore, HVASS in dimension zero are exactly VASS in dimension zero, so finite automata. However, for higher dimensions, the notions of HVASS and VASS differ.

We present here a few results on languages for HVASS. First notice that for nondeterministic HVASS it is easy to construct a language equivalent ε-VASS.

Lemma 3.5. *There is an algorithm, that for a given HVASS computes, in exponential time, a language equivalent ε-VASS.*

Proof of Lemma 3.5. We first sketch our solution. At first, we observe that the complement of the set of holes is an upward-closed set U. The idea behind the construction is that after each step we test if the current configuration is in U. We nondeterministically guess a minimal element x_i of U above which the current configuration is, then we subtract x_i and add it back. If our guess was not correct, then the run is blocked. A detailed proof follows.

Let $V = (\Sigma, Q, T, q_I(v_I), F, H)$ be a d-HVASS with the set of holes H. We aim at constructing a d-VASS $V' = (\Sigma, Q', T', c'_I, F')$ such that $L(V) = L(V')$. By Proposition 2.2 we can compute in exponential time an upward-closed set of configurations $U = (Q \times \mathbb{N}^d) \setminus H$. In order to translate V into a d-VASS V' intuitively we need to check that each configuration on the run is not in the set H. To do this, we use the representation of U as a finite union $U = \bigcup_{i \in [1,k]} q_i(u_i\uparrow)$ for $q_i \in Q$ and $u_i \in \mathbb{N}^d$. Now for each configuration c on the run of V the simulating VASS V' needs to check that c belongs to $q_i(u_i\uparrow)$ for some $i \in [1, k]$. That is why in V' after every step simulating a transition of V we go into a testing gadget and

after performing the test we are ready to simulate the next step. For that purpose we define $Q' = (Q \times \{0,1\}) \cup \{r_1, \ldots, r_k\}$. The states in $Q \times \{0\}$ are those before the test and the states in $Q \times \{1\}$ are the ones after the test. The states $r_1, \ldots, r_k$ are used to perform the test. The initial configuration c'_I is defined as $(q_I, 0)(v_I)$ and the set of final configurations is defined as $F' = \{(q,1)(v) \mid q(v) \in F\}$. For each transition (p, a, v, q) in T we add a corresponding transition $((p,1), a, v, (q,0))$ to T'. In each reachable configuration $(q,0)(v)$ the VASS V' nondeterministically guesses for which $i \in [1,k]$ holds $q_i(u_i) \preceq q(v)$ (which guarantees that indeed $q(v) \in U$). In order to implement it for each $q \in Q$ and each $i \in [1,k]$ such that $q = \text{state}(r_i)$ we add two transitions to T': the one from $(q,0)$ to r_i subtracting u_i, namely $((q,0), \varepsilon, -u_i, r_i)$ and the one coming back and restoring the counter values, namely $(r_i, \varepsilon, u_i, (q,1))$. It is easy to see that $(q,0)(v) \xrightarrow{\varepsilon} (q,1)(v)$ if and only if $q(v) \in U$, which finishes the proof. □

It is important to emphasise that the above construction applied to a deterministic HVASS does not give us a deterministic VASS, so we cannot simply reuse Theorem 3.3. Thus in order to prove the decidability of the inclusion problem for HVASS we need to generalise Theorem 3.3 to HVASS.

Theorem 3.6. *There is an exponential time algorithm, that for a deterministic HVASS V computes a downward-ε-VASS, which recognises the complement of the language of V.*

Proof of Theorem 3.6. We first sketch our solution. The proof is very similar to the proof of Theorem 3.3. In case (1) we have to check if the accepting run stays above the holes, to perform it we use the trick from Lemma 3.5. In case (2) we freeze the counter when the run would have to drop below zero or enter the hole. The case (3) is the same as in Theorem 3.3. A more detailed proof follows.

As the proof is very similar to the proof of Theorem 3.3 we only sketch the key differences. Let V be a deterministic HVASS and let $H \subseteq Q \times \mathbb{N}^d$ be the set of its holes. Let $U = (Q \times \mathbb{N}^d) \setminus H$, by Proposition 2.2 we know that $U = \bigcup_{i \in [1,k]} q_i(u_i{\uparrow})$ for some states $q_i \in Q$ and vectors $u_i \in \mathbb{N}^d$, and additionally $||U||$ is at most exponential w.r.t. the size $||H||$.

The construction of V' recognising the complement of $L(V)$ is almost the same as in the proof of Theorem 3.3, we need to introduce only small changes. The biggest changes are in the part of V' that recognises words rejected by V because of scenario (1). We need to check that after each transition, the current configuration is in U (so it is not in any hole from H). We perform it here in the same way as in the proof of Lemma 3.5. Namely, we guess which $q_i(u_i{\uparrow})$ the current configuration belongs to and check it by simple VASS modifications (for details, look at the proof of Lemma 3.5). The size of this part of V' can have a blow-up of at most size of U times, namely the size can be multiplied by some number, which is at most exponential w.r.t. the size $||H||$.

In the part recognising words rejected by V due to scenario (2), we only need to adjust the accepting set F_2. Indeed, we need to accept now if we are in a configuration $(p,t)(v) \in Q \times T$ such that $v + t \notin \mathbb{N}^d$ or $v + t \in H$ (in contrast to only $v + t \notin \mathbb{N}^d$ in the proof of Theorem 3.3). This change does not introduce any new superlinear blow-up.

Finally the part recognising words rejected by V because of scenario (3) does not need adjusting at all. It is not hard to see that the presented construction indeed accepts the complement of $L(V)$ as before. The constructed downward-VASS V' is of at most exponential size w.r.t. the size V as explained above, which finishes the proof. □

Now, the following theorem is an easy consequence of the shown facts. We need only to observe that proofs of Lemma 3.2 and Corollary 2.4 work as well for ε-VASS.

Theorem 3.7. *The inclusion problem of an HVASS language in a deterministic HVASS language is in* Ackermann.

Proof of Theorem 3.7. Let $V_1 = (\Sigma, Q_1, T_1, c_I^1, F_1, H_1)$ be a d_1-HVASS with holes $H_1 \subseteq Q_1 \times \mathbb{N}^{d_1}$ and let $V_2 = (\Sigma, Q_2, T_2, c_I^2, F_2, H_2)$ be a deterministic d_2-HVASS with holes $H_2 \subseteq Q_2 \times \mathbb{N}^{d_2}$. By Lemma 3.5 an ε-VASS V_1' equivalent to V_1 can be computed in exponential time. By Theorem 3.6 a downward-ε-VASS V_2' can be computed in exponential time such that $L(V_2') = \Sigma^* \setminus L(V_2)$. It is enough to check now whether $L(V_1') \cap L(V_2') = \emptyset$. By Lemma 3.2 (extended to ε-VASS) one can compute an updown-ε-VASS V such that $L(V) = L(V_1') \cap L(V_2')$. Finally, by Corollary 2.4 (also extended to ε-VASS) the emptiness problem for updown-ε-VASS is in Ackermann, which finishes the proof. □

Boundedly-control-deterministic VASS. We define here a generalisation of a deterministic VASS, namely a k-control-deterministic VASS for $k \in \mathbb{N}$. Such VASS are later used as a tool for deriving results about k-ambiguous VASS in Section 5.

We say that a finite automaton $\mathcal{A} = (\Sigma, Q, T, q_I, F)$ is *k-deterministic* if for each word $w \in \Sigma^*$ there are at most k *maximal runs over w*. We call a run ρ a *maximal run over w* if either (1) it is a run over w or (2) $w = uav$ for $u, v \in \Sigma^*$, $a \in \Sigma$ such that the run ρ is over the prefix u of w but there is no possible way of extending ρ by any transition labelled with the letter $a \in \Sigma$. Let us emphasise that here we count runs in a subtle way. We do not count only the maximal number of active runs throughout the word, but the total number of different runs that have ever been started during the word. To illustrate the difference better let us consider an example finite automaton $\mathcal{A}$ over $\Sigma = \{a, b\}$ with two states p, q and with three transitions: (p, a, p), (p, a, q) and (q, b, q). Then $\mathcal{A}$ has $n + 1$ maximal runs over the word a^n although only two of these runs actually survive till the end of the input word. So $\mathcal{A}$ is not 2-deterministic even though for each input word it has at most two runs. We say that a VASS $V = (\Sigma, Q, T, c_I, F)$ is *k-control-deterministic* if its control automaton is k-deterministic.

Theorem 3.8. *There is an exponential time algorithm, that for a k-control-deterministic d-VASS V constructs a $(k \cdot d)$-dimensional downward-VASS, which recognises the complement of the language of V.*

Proof of Theorem 3.8. We first sketch our solution. In the algorithm the $(k \cdot d)$-dimensional downward-VASS V' simulates k copies of V which take care of at most k different maximal runs of V. The accepting condition F' of V' verifies whether in all copies there is a reason that the simulated maximal runs do not accept. The reasons why each individual copy does not accept are the same as in Theorem 3.3.

Before starting the proof, let us remark that it would seem natural to first build a deterministic $(k \cdot d)$-VASS equivalent to the input k-control-deterministic d-VASS and then apply construction from the proof of Theorem 3.3 to recognise its complement. However, it is not clear how to construct a deterministic $(k \cdot d)$-VASS equivalent to k-control-deterministic d-VASS, thus we compute directly a VASS recognising the complement of the input VASS language.

Let $V = (\Sigma, Q, T, c_I, F)$ be a k-control-deterministic d-VASS. We aim to construct a $(k \cdot d)$-dimensional downward-VASS $V' = (\Sigma, Q', T', c_I', F')$ such that $L(V') = \Sigma^* \setminus L(V)$.

In this proof, also, we strongly rely on the ideas introduced in the proof of Theorem 3.3. Recall that the idea of the construction is that V' simulates k copies of V which take care of different maximal runs of V. Then the accepting condition F' of V' verifies whether in all copies there is a reason the simulated maximal runs do not accept.

Recall that for a run there are three scenarios in which it is not accepted: (1) it reaches the end of the word, but the reached configuration is not accepted, (2) at some moment it tries to decrease some counter below zero, and (3) at some moment there is no transition available over the input letter. In the proof of Theorem 3.3 it was shown how a VASS can handle all three reasons. In short words: in case (1) it simulates the run till the end of the word and then checks that the reached configuration is not accepting and in cases (2) and (3) it guesses the moment in which there is no valid transition available and keeps this configuration untouched (in other words freezes it) till the end of the run when it checks by the accepting condition that the guess was correct. We only sketch how the downward-VASS V' works without stating explicitly its states and transitions. It starts in the configuration c'_I which consists of k copies of c_I. Then it simulates the run in all the copies in the same way until the first time when there is a choice of transition. Then we enforce that at least one copy follows each choice, but we allow for more than one copy to follow the same choice. In the state of V' we keep the information which copies follow the same maximal run and which have already split. Each copy is exactly as in the proof of Theorem 3.3, it realises one of the scenarios (1), (2) or (3). As we know that V is k-control-deterministic, we are sure that all the possible runs of V can be simulated by V' under the condition that V' correctly guesses which copies should simulate which runs. If the guesses of V' are wrong and at some point it cannot send to each branch a copy, then the run of V' is rejected.

A bit more concretely, the state of V' keeps the following information: (I) for each of the k copies in which state in Q it is, (II) the set of copies which are frozen and for each such copy a transition which caused the freezing, (III) which copies have already split and which have not (formally speaking we keep a partition of the set of copies).

We guarantee that all the branches are explored by some copy in the following way. If in V there are m different transitions over some letter a from a state q then in V' in every state that encodes ℓ copies in state q not all the possible m^ℓ options in the product are allowed. We allow only for these options for which each of the possible m transitions of V are taken in one of the ℓ copies. In particular if $m = \ell$ then instead of m^m options in V' we have exactly $m!$ options. In each of the options, there are actually 2^m possibilities for freezing the copies. This is needed because if the transition would decrease the counters below zero (so scenario (2) is realised), the copy may not actually fire the transition, and we should have a possibility of freezing it. If a copy gets frozen, the control state is updated accordingly.

At the end of the run over the input word w VASS V' checks using the acceptance condition F' that indeed all copies have simulated all the possible maximal runs and that all reject. It is easy to see that F' is a downward-closed set, since, roughly speaking, it is a product of k downward-closed accepting conditions, which finishes the proof. □

Theorem 3.8 together with Lemma 3.2 and Corollary 2.4 easily implies (analogously as in the proof of Theorem 3.7) the following theorem.

Theorem 3.9. *The inclusion problem of a VASS language in a k-control-deterministic VASS language is in* Ackermann.

4. Unambiguous VASS

In this section we aim to prove Theorem 1.4. However, possibly a more valuable contribution of this section is a novel technique which we introduce in order to show Theorem 1.4. The essence of this technique is to introduce a regular lookahead to words, namely to decorate each letter of a word with a piece of information regarding some regular properties of the suffix of this word. For technical reasons, we realise it by using finite monoids.

The high-level intuition behind the proof of Theorem 1.4 is the following. We first introduce the notion of (M, h)-decoration of words, languages and VASS, where M is a monoid and $h : \Sigma^* \to M$ is a homomorphism. Proposition 4.3 states that language inclusion of two VASS can be reduced to language inclusion of its decorations. On the other hand Theorem 4.6 shows that for appropriately chosen pair (M, h) the decorations of unambiguous VASS are deterministic HVASS. Theorem 4.5 states that such an appropriate pair can be computed quickly enough. Thus, the language inclusion of unambiguous VASS reduces to language inclusion of deterministic HVASS, which is in Ackermann due to Theorem 3.7.

Recall that a monoid M together with a homomorphism $h : \Sigma^* \to M$ and an accepting subset $F \subseteq M$ recognises a language L if $L = h^{-1}(F)$. In other words L is exactly the set of words w such that $h(w) \in F$. The following proposition is folklore, for details see [Pin97] (Proposition 3.12).

Proposition 4.1. *A language of finite words is regular if and only if it is recognised by some finite monoid.*

For that reason, monoids are a good tool for working with regular languages. In particular Proposition 4.1 implies that for each finite family of regular languages there is a monoid, which recognises all of them, this fact is useful in Theorem 4.6. Fix a finite monoid M and a homomorphism $h : \Sigma^* \to M$. For a word $w = a_1 \cdot \ldots \cdot a_n \in \Sigma^*$ we define its (M, h)-*decoration* to be the following word over an alphabet $\Sigma_\varepsilon \times M$:

$$(\varepsilon, h(a_1 \cdot \ldots \cdot a_n)) \cdot (a_1, h(a_2 \cdot \ldots \cdot a_n)) \cdot \ldots \cdot (a_{n-1}, h(a_n)) \cdot (a_n, h(\varepsilon)).$$

In other words, the (M, h)-decoration of a word w of length n has length $n + 1$, where the i-th letter has the form $(a_{i-1}, h(a_i \cdot \ldots \cdot a_n))$. We denote the (M, h)-decoration of a word w as $w_{(M,h)}$. If $h(w) = m$ then we say that the word w has *type* $m \in M$. The intuition behind the (M, h)-decoration of w is that for each language L which is recognised by the pair (M, h) the i-th letter of w is extended with an information whether the suffix of w after this letter belongs to L or does not belong. This information can be extracted from the monoid element $h(a_{i+1} \cdot \ldots \cdot a_n)$ by which the letter a_i is extended. As an illustration consider words over alphabet $\Sigma = \{a, b\}$, monoid $M = \mathbb{Z}_2$ counting modulo two and homomorphism $h : \Sigma \to M$ defined as $h(a) = 1$, $h(b) = 0$. In that case for each $w \in \Sigma^*$ the element $h(w)$ indicates whether the number of letters a in the word w is odd or even. The decoration of $w = aabab$ is then $w_{(M,h)} = (\varepsilon, 1)(a, 0)(a, 1)(b, 1)(a, 0)(b, 0)$.

We say that a word $u \in (\Sigma_\varepsilon \times M)^*$ is *well-formed* if $u = (\varepsilon, m_0) \cdot (a_1, m_1) \cdot \ldots \cdot (a_n, m_n)$ such that all $a_i \in \Sigma$, and for each $i \in [0, n]$ the type of $a_{i+1} \cdot \ldots \cdot a_n$ is m_i (in particular type of ε is m_n). We say that such a word u *projects* into word $a_1 \cdot \ldots \cdot a_n$. It is easy to observe that $w_{(M,h)}$ is the only well-formed word that projects into w. The following proposition is useful in Section 5, an appropriate finite automaton can be easily constructed.

Proposition 4.2. *The set of all well-formed words is regular.*

A word is *almost well-formed* if it satisfies all the conditions of well-formedness, but the first letter is not necessarily of the form (ε, m) for $m \in M$, it can as well belong to $\Sigma \times M$.

The (M, h)-decoration of a language L, denoted $L_{(M,h)}$, is the set of all (M, h)-decorations of all words in L, namely

$$L_{(M,h)} = \{w_{(M,h)} \mid w \in L\}.$$

Since the (M, h)-decoration is a function from the set of words over Σ to words over $\Sigma_\varepsilon \times M$ we observe that $u = v$ iff $u_{(M,h)} = v_{(M,h)}$ and clearly the following proposition holds.

Proposition 4.3. *For each finite alphabet Σ, two languages $K, L \subseteq \Sigma^*$, a monoid M and homomorphism $h : \Sigma^* \to M$ we have*

$$K \subseteq L \iff K_{(M,h)} \subseteq L_{(M,h)}.$$

HVASS construction. As defined earlier, an HVASS (VASS with holes) is a VASS with some downward-closed set H of prohibited configurations (see Section 3, paragraph Deterministic VASS with holes). For each d-VASS $V = (\Sigma, Q, T, c_I, F)$, a monoid M and a homomorphism $h : \Sigma^* \to M$ we can define in a natural way a d-HVASS $V_{(M,h)} = (\Sigma_\varepsilon \times M, Q', T', c'_I, F')$ accepting the (M, h)-decoration of $L(V)$. The set of states Q' equals $Q \times (M \cup \{\bot\})$. The intuition is that $V_{(M,h)}$ is designed in such a way that for any state $(q, m) \in Q \times M$ and vector $v \in \mathbb{N}^d$ if $(q, m)(v) \xrightarrow{w'} F'$ then w' is almost well-formed and w' projects into some $w \in \Sigma^*$ such that $h(w) = m$. If $c_I = q_I(v_I)$ then configuration $c'_I = (q_I, \bot)(v_I)$ is the initial configuration of $V_{(M,h)}$. The set of final configurations F' is defined as $F' = \{(q, h(\varepsilon))(v) \mid q(v) \in F\}$. Finally we define the set of transitions T' of V' as follows. First, for each $m \in M$ we add the following transition $((q_I, \bot), (\varepsilon, m), 0^d, (q_I, m))$ to T'. Then for each transition $(p, a, v, q) \in T$ and for each $m \in M$ we add to T' the transition (p', a', v, q') where $a' = (a, m)$, $q' = (q, m)$ and $p' = (p, h(a) \cdot m)$. It is now easy to see that for any word $w = a_1 \cdot \ldots \cdot a_n \in \Sigma^*$ we have

$$q_I(v_I) \xrightarrow{a_1} q_1(v_1) \xrightarrow{a_2} \ldots \xrightarrow{a_{n-1}} q_{n-1}(v_{n-1}) \xrightarrow{a_n} q_n(v_n)$$

if and only if

$$(q_I, \bot)(v_I) \xrightarrow{(\varepsilon, m_1)} (q_I, m_1)(v_I) \xrightarrow{(a_1, m_2)} (q_1, m_2)(v_1) \xrightarrow{(a_2, m_3)} \ldots$$
$$\xrightarrow{(a_{n-1}, m_n)} (q_{n-1}, m_n)(v_{n-1}) \xrightarrow{(a_n, m_{n+1})} (q_n, m_{n+1})(v_n),$$

where $m_i = h(w[i..n])$ for all $i \in [1, n+1]$, in particular $m_{n+1} = h(\varepsilon)$. Therefore, indeed $L(V_{(M,h)}) = L(V)_{(M,h)}$. Until now the defined HVASS is actually a VASS, we have not defined any holes. Our aim is now to remove configurations with the empty language, that is, $(q, m)(v)$ for which there is no word $w \in (\Sigma_\varepsilon \times M)^*$ such that $(q, m)(v) \xrightarrow{w} c'_F$ for some $c'_F \in F'$. Notice that as F' is upward-closed we know that the set of configurations with the empty language is downward-closed. This is how we define the set of holes H, it is exactly the set of configurations with the empty language. We can compute the set of holes in doubly-exponential time by Proposition 2.7.

By Proposition 4.3 we know that for two VASS U, V we have $L(U) \subseteq L(V)$ if and only if $L(U_{(M,h)}) \subseteq L(V_{(M,h)})$. This equivalence is useful, as we show in a moment that for an unambiguous VASS V and suitably chosen (M, h) the HVASS $V_{(M,h)}$ is deterministic.

Regular separability. We use here the notion of regular separability. We say that two languages $K, L \subseteq \Sigma^*$ are *regular-separable* if there exists a regular language $S \subseteq \Sigma^*$ such

that $K \subseteq S$ and $S \cap L = \emptyset$. We then say that S *separates* K and L and S is a *separator* of K and L. We recall here a theorem about regular-separability of VASS languages (importantly upward-VASS languages, not downward-VASS languages) from [CLM+18].

Theorem 4.4 [CLM+18, Theorem 24]**.** *For any two VASS languages $L_1, L_2 \subseteq \Sigma^*$ if $L_1 \cap L_2 = \emptyset$ then L_1 and L_2 are regular-separable and one can compute a regular separator in elementary time.*

Proof. Theorem 24 in [CLM+18] says that there exists a regular separator of L_1 and L_2 of size at most triply-exponential. In order to compute it, we can simply enumerate all the possible separators of at most triply-exponential size and check them one by one. For a given regular language and a given VASS language by Proposition 2.6 one can check in doubly-exponential time whether they intersection is nonempty. □

For our purposes we need a bit stronger version of this theorem. We say that a family of regular languages $\mathcal{F}$ *separates languages of a VASS V* if for any two configurations c_1, c_2 such that languages $L(c_1)$ and $L(c_2)$ are disjoint there exists a language $S \in \mathcal{F}$ that separates $L(c_1)$ and $L(c_2)$.

Theorem 4.5. *There is an algorithm that, for any VASS computes, in an elementary time, a finite family of regular languages that separates its languages.*

Proof. Let us fix a d-VASS $V = (\Sigma, Q, T, c_I, F)$. Let us define the set of pairs of configurations of V with disjoint languages $D = \{(c_1, c_2) \mid L(c_1) \cap L(c_2) = \emptyset\} \subseteq Q \times \mathbb{N}^d \times Q \times \mathbb{N}^d$. One can easily see that the set D is exactly the set of configurations with empty language in the synchronised product of VASS V with itself. Thus, by Proposition 2.7 we can compute in doubly-exponential time its representation as a finite union of down-atoms $D = A_1 \cup \ldots \cup A_n$. We show now that for each $i \in [1, n]$ one can compute in elementary time a regular language S_i such that for all $(c_1, c_2) \in A_i$ the language S_i separates $L(c_1)$ and $L(c_2)$. This will complete the proof showing that one of $S_1, \ldots, S_n$ separates $L(c_1)$ and $L(c_2)$ whenever they are disjoint.

Let $A \subseteq Q \times \mathbb{N}^d \times Q \times \mathbb{N}^d$ be a down-atom. Therefore $A = D_1 \times D_2$ where $D_1 = p_1(u_1{\downarrow})$ and $D_2 = p_2(u_2{\downarrow})$ for some $u_1, u_2 \in (\mathbb{N} \cup \{\omega\})^d$. Let $L_1 = \bigcup_{c \in D_1} L(c)$ and $L_2 = \bigcup_{c \in D_2} L(c)$. Languages L_1 and L_2 are disjoint as $w \in L_1 \cap L_2$ would imply $w \in L(c_1) \cap L(c_2)$ for some $c_1 \in D_1$ and $c_2 \in D_2$. Now, observe that L_1 is not only an infinite union of VASS languages but also a VASS language itself. Indeed, let $V_1 = (\Sigma, Q, T_1, c_{(I,1)}, F_1)$ be the VASS V where all coordinates $i \in [1, d]$ such that $u_1[i] = \omega$ are ignored. Concretely,

- $(p, a, v_1, q) \in T_1$ if there exists $(p, a, v, q) \in T$ such that for every i holds either $v_1[i] = v[i]$ or $v_1[i] = 0$ and $u_1[i] = \omega$,
- $\mathrm{state}(c_{(I,1)}) = p_1$ and for every i holds either $c_{(I,1)}[i] = u_1[i]$ or $c_{(I,1)}[i] = 0$ and $u_1[i] = \omega$,
- $(q, v_1) \in F_1$ if there exists $(q, v) \in F$ such that for every i holds either $v_1[i] = v[i]$ or $u_1[i] = \omega$.

Then it is easy to observe that V_1 accepts exactly the language L_1. Similarly, one can define VASS V_2 accepting the language L_2. By Theorem 4.4 we can compute in elementary time some regular separator S of $L(V_1)$ and $L(V_2)$. It is now easy to see that for any configurations $c_1 \in D_1$ and $c_2 \in D_2$ languages $L(c_1)$ and $L(c_2)$ are separated by S. □

Now we are ready to use the notion of (M, h)-decoration of a VASS language. Let us recall that a regular language L is recognised by a monoid M and homomorphism $h : \Sigma^* \to M$ if there is $F \subseteq M$ such that $L = h^{-1}(F)$.

Theorem 4.6. *Let V be an unambiguous VASS over Σ and $\mathcal{F}$ be a finite family of regular languages separating languages of V. Suppose M is a monoid with homomorphism $h : \Sigma^* \to M$ recognising every language in $\mathcal{F}$. Then the HVASS $V_{(M,h)}$ is deterministic.*

Proof. Let $V = (\Sigma, Q, T, c_I, F)$ and let $c_I = q_I(v_I)$. We aim to show that HVASS $V_{(M,h)} = (\Sigma', Q', T', c'_I, F')$ is deterministic, where $\Sigma' = \Sigma_\varepsilon \times M$ and $Q' = Q \times (M \cup \{\bot\})$. It is easy to see from the definition of $V_{(M,h)}$ that for each $(a, m) \in \Sigma'$ and each $q \in Q$ the state $(q, \bot)$ has at most one outgoing transition over (a, m). Indeed, there is exactly one transition over (ε, m) outgoing from $(q_I, \bot)$ and no outgoing transitions in the other cases. Assume now towards a contradiction that $V_{(M,h)}$ is not deterministic. Then there is some configuration $c = (q, m)(v)$ with $(q, m) \in Q \times M$ such that $c_I \xrightarrow{u} c$ for some word u over Σ' and a letter $(a, m') \in \Sigma'$ such that a transition from c over (a, m') leads to two different configurations $c_1 = (q_1, m')(v_1)$ and $c_2 = (q_2, m')(v_2)$. Recall that a transition over (a, m') has to lead to some state where the second component is equal to m'. As configurations with empty language are not present in $V_{(M,h)}$ we know that there exist words $w_1 \in L(c_1)$ and $w_2 \in L(c_2)$. Recall that as $c_1 = (q_1, m')(v_1)$ and $c_2 = (q_2, m')(v_2)$ we have $h(w_1) = m' = h(w_2)$. We show now that $L(c_1)$ and $L(c_2)$ are disjoint. Assume otherwise that there exists $w \in L(c_1) \cap L(c_2)$. Then there are at least two accepting runs over the word $u \cdot (a, m') \cdot w$ in $V_{(M,h)}$. However, this means that there are at least two accepting runs over the projection of $u \cdot (a, m') \cdot w$ in V, which contradicts the unambiguity of V. Thus, $L(c_1)$ and $L(c_2)$ are disjoint and therefore separable by some language from $\mathcal{F}$. Recall that all languages in $\mathcal{F}$ are recognisable by (M, h), therefore words from $L(c_1)$ should be mapped by homomorphism h to different elements from M than words from $L(c_2)$. However, $h(w_1) = m'$ for $w_1 \in L(c_1)$ and $h(w_2) = m'$ for $w_2 \in L(c_2)$ which leads to the contradiction. □

Now we are ready to prove Theorem 1.4. Let V_1 be a VASS and V_2 be an unambiguous VASS, both with labels from Σ. We first compute a finite family $\mathcal{F}$ separating languages of V_2 which can be performed in elementary time by Theorem 4.5 and then we compute a finite monoid M together with a homomorphism $h : \Sigma^* \to M$ recognising all the languages from $\mathcal{F}$. By Proposition 4.3 we get that $L(V_1) \subseteq L(V_2)$ if and only if $L(V_1)_{(M,h)} \subseteq L(V_2)_{(M,h)}$. We now compute HVASS $V'_1 = V_{1_{(M,h)}}$ and $V'_2 = V_{2_{(M,h)}}$ as described above in the paragraph HVASS construction. By Theorem 4.6 the HVASS V'_2 is deterministic. Thus it remains to check whether the language of a HVASS V'_1 is included in the language of a deterministic HVASS V'_2, which is in Ackermann due to Theorem 3.7.

Remark 4.7. We remark that our technique can be applied not only to VASS but also in a more general setting of well-structured transition systems. In [CLM+18] it was shown that for any well-structured transition systems satisfying some mild conditions (finite branching is enough), the disjointness of two languages implies regular separability of these languages. We claim that an analogue of our Theorem 4.5 can be obtained in that case as well. Assume now that $\mathcal{W}_1, \mathcal{W}_2$ are two classes of finitely branching well-structured transition systems, such that for any two systems $V_1 \in \mathcal{W}_1$, $V_2 \in \mathcal{W}_2$ where V_2 is deterministic the language inclusion problem is decidable. Then this problem is also likely to be decidable if we weaken the condition of determinism to unambiguity. More concretely speaking this seems to be the case if it is possible to perform the construction analogous to Theorem 3.3 in $\mathcal{W}_2$, namely if one can compute the system recognising the complement of deterministic language without leaving the class $\mathcal{W}_2$. We claim that an example of such a class $\mathcal{W}_2$ is the class of VASS with one reset. The emptiness problem for VASS with one zero-test (and thus also for VASS with

one reset) is decidable due to [Rei08, Bon11]. Then following our techniques it seems that one can show that inclusion of a VASS language in a language of an unambiguous VASS with one reset is decidable.

5. Boundedly-ambiguous VASS

In this section we aim to prove Theorem 1.2 and show that Theorem 1.3 is its easy consequence. Recall the statements of both Theorem 1.2 and Theorem 1.3.

Theorem 1.2. There is an algorithm that, for each $k \in \mathbb{N}$ and a k-ambiguous VASS V constructs, in elementary time, a VASS with a downward-closed set of accepting configurations which recognises the complement of the language of V.

Theorem 1.3. For each $k \in \mathbb{N}$ the language inclusion problem of a VASS in a k-ambiguous VASS is in Ackermann.

Let us first show how Theorem 1.2 implies Theorem 1.3. Let V_1 be a VASS and V_2 be a k-ambiguous VASS. By Theorem 1.2 can be computed in elementary time a downward-VASS V_2' such that $L(V_2') = \Sigma^* \setminus L(V_2)$. By Lemma 3.2 one can construct in time a polynomial w.r.t. the size of V_1 and V_2' an updown-VASS V such that $L(V) = L(V_1) \cap L(V_2') = L(V_1) \setminus L(V_2)$. By Corollary 2.4 emptiness of V is decidable in Ackermann which, consequently, proves Theorem 1.3.

The rest of this section focusses on the proof of Theorem 1.2.

Proof of Theorem 1.2. We now prove Theorem 1.2 using Lemmas 5.1 and 5.2. Then in Sections 5.1 and 5.2 we show the formulated lemmas. Let V be a k-ambiguous VASS over an alphabet Σ. First due to Lemma 5.1 proved in Section 5.1 we construct a VASS V^1 which is language equivalent to V and additionally has the control automaton being k-ambiguous.

Lemma 5.1. *There is an algorithm, that for each k-ambiguous VASS V constructs, in doubly-exponential time, a language equivalent VASS V' with the property that its control automaton is k-ambiguous.*

Now our aim is to obtain a k-control-deterministic VASS V^2, which is language equivalent to V^1. We are not able to achieve it literally, but using the notion of (M, h)-decoration from Section 4 we can compute a somehow connected k-control-deterministic VASS V^2. We use the following lemma which is proved in Section 5.2.

Lemma 5.2. *Let $\mathcal{A} = (\Sigma, Q, T, q, F)$ be a k-ambiguous finite automaton for some $k \in \mathbb{N}$. Let M be a finite monoid and $h : \Sigma^* \to M$ be a homomorphism recognising all the state languages of the automaton $\mathcal{A}$. Then the decoration $\mathcal{A}_{(M,h)}$ is a k-deterministic finite automaton.*

Now we consider the control automaton $\mathcal{A}$ of VASS V^1. We compute a monoid M together with a homomorphism $h : \Sigma^* \to M$ that recognises all the state languages of $\mathcal{A}$. Then we construct the automaton $\mathcal{A}_{(M,h)}$. Note that the decoration of a VASS produces an HVASS, but as we decorate an automaton i.e. 0-VASS we get a 0-HVASS which is also a finite automaton. Based on $\mathcal{A}_{(M,h)}$ we construct a VASS V^2. We add a vector to every transition in $\mathcal{A}_{(M,h)}$, to produce a VASS, that recognises the (M, h)-decoration of the language of VASS V^1. Precisely, if we have a transition $((p, m), (a, m'), (q, m'))$ in $\mathcal{A}_{(M,h)}$ then it is created from the transition (p, a, q) in $\mathcal{A}$, which originates from the transition

(p, a, v, q) in V^1. So in V^2 we label $((p, m), (a, m'), (q, m'))$ with v i.e. we have the transition $((p, m), (a, m'), v, (q, m'))$. Similarly, based on V^1, we define initial and final configurations in V^2. It is easy to see that there is a bijection between accepting runs in V^1 and accepting runs in V^2. By Lemma 5.2 $\mathcal{A}_{(M,h)}$ is k-deterministic which immediately implies that V^2 is k-control-deterministic as well.

Now by Theorem 3.8 we compute a downward-VASS V^3 which recognises the complement of $L(V^2)$. Notice that for each $w \in \Sigma^*$ there is exactly one well-formed word in $\Sigma_\varepsilon \times M$ which projects into w, namely the (M, h)-decoration of w. Therefore, V^3 accepts all the non-well-formed words and all the well-formed words that project into the complement of $L(V)$. By Proposition 4.2 the set of all well-formed words is recognised by some finite automaton $\mathcal{B}$. Computing a synchronised product of $\mathcal{B}$ and V^3 one can obtain a downward-VASS V^4 which recognises the intersection of languages $L(\mathcal{B})$ and $L(V^3)$, namely all the well-formed words which project into the complement of $L(V)$. Now, it is easy to compute a downward-ε-VASS V^5 recognising the projection of $L(V^4)$ into the first component of the alphabet $\Sigma_\varepsilon \times M$. We obtain V^5 simply by ignoring the second component of the alphabet. Thus, V^5 recognises exactly the complement of $L(V)$. However V^5 is not a downward-VASS as it contains a few ε-labelled transitions leaving the initial state. We aim to eliminate these ε-labelled transitions. Recall that in the construction of the (M, h)-decoration the (ε, m)-labelled transitions leaving the initial configuration have the effect 0^d. Thus, it is easy to eliminate them and obtain a downward-VASS V^6 which recognises exactly the complement of $L(V)$, which finishes the proof of Theorem 1.2. Let us remark here that even ignoring the last step of elimination and obtaining a downward-ε-VASS recognising the complement of $L(V)$ would be enough to prove Theorem 1.3 along the same lines as it is proved now. □

5.1. **Proof of Lemma 5.1.** We recall the statement of Lemma 5.1.

Lemma 5.1. There is an algorithm, that for each k-ambiguous VASS V constructs, in doubly-exponential time, a language equivalent VASS V' with the property that its control automaton is k-ambiguous.

Proof of Lemma 5.1. Let $V = (\Sigma, Q, T, c_I, F)$ be a k-ambiguous d-VASS for some $k \in \mathbb{N}$. We aim at constructing a language equivalent VASS V' such that its control automaton is k-ambiguous. Notice that if the control automaton of V' is k-ambiguous then clearly V' is k-ambiguous as well. The idea behind the construction of V' is that it behaves as V, but additionally the states of V' keep some finite information about the values of the counters. More precisely counter values are kept exactly until they pass some threshold M. After passing this threshold, its value is not kept in the state, only the information about passing the threshold is remembered. We define the described notion below more precisely as the M-abstraction of a VASS. The control state of V' is the M-abstraction of V for appropriately chosen M; we show how to choose M later.

Definition 5.3. We consider here vectors over $\mathbb{N} \cup \{\omega\}$ where ω is interpreted as a number greater than all natural numbers and fulfilling $\omega + a = \omega$ for any $a \in \mathbb{Z}$. For $v \in (\mathbb{N} \cup \{\omega\})^d$ and $M \in \mathbb{N}$ we define $v_M \in ([0, M-1] \cup \{\omega\})^d$ such that for all $i \in [1, d]$ we have $v_M[i] = v[i]$ if $v[i] < M$ and $v_M[i] = \omega$ otherwise. In other words, all the numbers in v that are at least equal to M are changed into ω in v_M. Let $V = (\Sigma, Q, T, c_I, F)$ be a d-VASS. For $M \in \mathbb{N}$ the *M-abstraction of V* is a finite automaton $V_M = (\Sigma, Q', T', q'_I, Q'_F)$ which roughly speaking behaves like V up to the threshold M. More concretely, we define V_M as follows:

- the set Q' of states equals $Q \times ([0, M-1] \cup \{\omega\})^d$
- if $c_I = q_I(v)$ then the initial state equals $q'_I = q_I(v_M)$
- the set Q'_F of accepting states equals $\{q(v_M) \mid q(v) \in F\}$
- for each transition $t = (q, a, v, q') \in T$ and for each $u \in ([0, M-1] \cup \{\omega\})^d$ such that $u + v \in (\mathbb{N} \cup \{\omega\})^d$ we define transition $(q(u), a, q'(u')) \in T'$ such that $u' = (u+v)_M$.

Now we aim at finding $M \in \mathbb{N}$ such that the M-abstraction of V is k-ambiguous. Notice that it proves Lemma 5.1. Having such an M we substitute the control automaton of V by the M-abstraction of V and obtain a VASS V' that meets the conditions of Lemma 5.1. In other words, each state q of the control automaton of V multiplies itself and now in the control automaton V_M of V' there are $(M+1)^d$ copies of q. The effects of transitions in V' are inherited from the effects of transitions in V. The languages of V and V' are the same as the new control automaton V_M of V' never eliminates any run allowed by the old control automaton in V.

We define now a $d(k+1)$-VASS $\overline{V}$ which roughly speaking simulates $k+1$ copies of V and accepts if all the copies have taken different runs and additionally all of them accept. Notice that it is easy to construct $\overline{V}$: it is just a synchronised product of $k+1$ copies of V which additionally keeps in the state the information which copies follow the same runs and which have already split. The language of $\overline{V}$ contains those words that have at least $k+1$ runs in V. As V is k-ambiguous, we know that $L(\overline{V})$ is empty. Now, we formulate the following lemma.

Lemma 5.4. *For each d-VASS V there is a computable doubly-exponential threshold $M \in \mathbb{N}$ such that if the M-abstraction of V has an accepting run then V has an accepting run.*

Lemma 5.4 is a consequence of the result by Rackoff [Rac78] showing that if there is a covering path in VASS then there is also a covering path of at most doubly-exponential length.

By Lemma 5.4 we can compute a threshold $\overline{M}$ such that the $\overline{M}$-abstraction of $\overline{V}$ has no accepting run. We claim now that the $\overline{M}$-abstraction of V, namely $V_{\overline{M}}$ is k-ambiguous. Assume otherwise; let $V_{\overline{M}}$ have at least $k+1$ runs over some word w. Then it is easy to see that the $\overline{M}$-abstraction of $\overline{V}$ has an accepting run over w, which is a contradiction. Thus indeed $V_{\overline{M}}$ is k-ambiguous and extending the control automaton of V by M-bounded counters in a way as in $V_{\overline{M}}$ (such that the control automaton becomes exactly $V_{\overline{M}}$) finishes the proof of Lemma 5.1. □

Proof of Lemma 5.4. Let $V = (\Sigma, Q, T, c_I, F)$ and let $\mathcal{C}$ be the set of configurations of V from which it is possible to reach F. As the set F is upward closed the same holds for $\mathcal{C}$. Further, let $\bar{\mathcal{C}}$ be $(\mathbb{N} \cup \omega)^d \setminus \mathcal{C}$, so the complement of $\mathcal{C}$ in the space of $(\mathbb{N} \cup \omega)^d$. It is downward closed i.e. it is a union of down-atoms, $d_1 \downarrow, d_2 \downarrow, \ldots, d_n \downarrow$. Let $||V||$ be the norm of VASS V, namely the maximal absolute value occurring on transition of V. Let $M = \max\{d_i[j] : i \leq n, j \leq d, d_i[j] \neq \omega\} + ||V||$ be the maximal constant that appears in the description of the set $\bar{\mathcal{C}}$ plus the norm of VASS V. By [Rac78] the minimal length of a path covering a given configuration is at most doubly-exponential wrt. the size of the VASS and the norm of the target configuration (i.e. the maximal absolute value of coordinates in the target). Therefore all the numbers in vectors d_i are at most doubly-exponential as well. Thus also the constant M is at most doubly-exponential.

By an M-abstraction of a configuration c, denoted by $A_M(c)$, we mean an ω-configuration in $Q \times (\mathbb{N} \cup \{\omega\})^d$ where we substitute in c any value greater or equal M by ω. Concretely:

$$A_M(c)[j] = \begin{cases} c[j] & \text{if } c[j] < M, \\ \omega & \text{if } c[j] \geq M. \end{cases}$$

Observe that function A_M has a following property, which is the consequence of the definition of the value M,

$$c \in \bar{\mathcal{C}} \iff A_M(c) \in \bar{\mathcal{C}}. \tag{5.1}$$

Finally, the image of the function A_M is in one to one correspondence with the states of the M-abstraction of V, so we will use M-abstractions of configurations in both settings: in the VASS V and in the M-abstraction of V.

Now we can start the actual proof of the lemma. We prove it by contradiction. Namely we assume that from some configuration c there is no accepting run in V, but there is an accepting run from the state $A_M(c)$ in the M-abstraction of V. Therefore $c \in \bar{\mathcal{C}}$ and from $A_M(c)$ in the M-abstraction it is possible to reach the set of final states F' of the M-abstraction. We will show that such c cannot exist. First observe that $A_M(c)$ cannot be among final states of the M-abstraction as then $c \in F$ which contradicts the fact that $c \in \bar{\mathcal{C}}$.

So for every $A_M(c)$ there is a shortest path in the M-abstraction to the set of final states F'. Without loss of generality we can assume that c is chosen in such a way that the length of the path from $A_M(c)$ to F' is minimal. We denote this path by σ. As $A_M(c) \notin F'$ we can decompose $\sigma = t\sigma'$ into the head and a tail, where $t = (q, a_t, \delta_t, p)$. Let us consider ω-configuration $c' = A_M(c) + \text{eff}(t)$ denoting an ω-configuration (p, v) where $v[i] = A_M(c)[i] + \delta_t[i]$ for every $i \in [1, d]$. On the one hand we know that $c' \in \bar{\mathcal{C}}$, as $c \in \bar{\mathcal{C}}$. Because of (5.1) we get that $A_M(c') \in \bar{\mathcal{C}}$. On the other hand in the M-abstraction of V the path σ from $A_M(c)$ traverses through $A_M(c')$. Therefore the path σ' from $A_M(c')$ to F' is shorter than the path σ. So we get a contradiction with the choice of c as $c' \in \bar{\mathcal{C}}$, and the path from $A_M(c')$ to F' is shorter than from c. □

5.2. **Proof of Lemma 5.2.** We recall the statement of Lemma 5.2.

Lemma 5.2. Let $\mathcal{A} = (\Sigma, Q, T, q, F)$ be a k-ambiguous finite automaton for some $k \in \mathbb{N}$. Let M be a finite monoid and $h : \Sigma^* \to M$ be a homomorphism recognising all the state languages of the automaton $\mathcal{A}$. Then the decoration $\mathcal{A}_{(M,h)}$ is a k-deterministic finite automaton.

Proof of Lemma 5.2. Let $w \in \Sigma^*$ be any word and $w_{(M,h)} \in (\Sigma_\varepsilon \times M)^*$ be its decoration. We have to prove that there are at most k maximal runs over the word $w_{(M,h)}$. To do so we prove two statements

- All maximal runs over the word $w_{(M,h)}$ have the same length.
- The multiset $X^{w_{(M,h)}}$ of states reachable in $\mathcal{A}_{(M,h)}$ along $w_{(M,h)}$ is of size at most k.

Indeed, if the above holds, then all the maximal runs have the same length, therefore their number is bounded by the size of $X^{w_{(M,h)}}$ i.e. by k as required.

In the first claim actually we show that either there is only one empty run of length zero or all the maximal runs are over the whole word $w_{(M,h)}$. We consider two cases: either $w \notin L(\mathcal{A})$ or $w \in L(\mathcal{A})$.

We claim that in the first case, the only maximal run has length zero. Notice that, then the first step should be $(q, \perp) \xrightarrow{(\epsilon, h(w))} (q, h(w))$. According to the definition of M for every $m \in M$ and every state $p \in Q$ we have either $h^{-1}(m) \subseteq L(p)$ or $h^{-1}(m) \cap L(p) = \emptyset$. Thus, in our case, it is that $h^{-1}(h(w)) \cap L(q) = \emptyset$, because $w \notin L(\mathcal{A})$. This means that the state $(q, h(w))$ is not present in the automaton $\mathcal{A}_{(M,h)}$, since $\mathcal{A}_{(M,h)}$ does not have states with empty language. So, indeed, the only maximal run is the empty one.

We prove the second case in a slightly more general setting: for all the prefixes of a decorated word. The reason is that we show the statement via an induction on the length of this prefix. Suppose u is a prefix of w and $\bar{u}$ is the corresponding prefix of $w_{(M,h)}$. For the empty prefix $u = \varepsilon$ the statement trivially holds. For an induction step, suppose that $\bar{u} = \bar{u}' \cdot (a, m)$. The induction hypothesis says that all the maximal runs along $\bar{u}'$ are over the whole word $\bar{u}'$. Let $X^{\bar{u}'}_{(M,h)} = \{(q_1, m'), (q_2, m'), \ldots, (q_\ell, m')\}_{\text{MUL}}$ be the multiset of states reached by the runs labelled $\bar{u}'$, for $m' = h(a) \cdot m$. Notice that $a \cdot h^{-1}(m) \subseteq h^{-1}(m')$. Recall that by the construction of $\mathcal{A}_{(M,h)}$ the languages of states (q_i, m') are not empty, therefore for each $i \in \{1 \ldots \ell\}$ we have $h^{-1}(m') \subseteq L(q_i)$. The two above inclusions imply that $a \cdot h^{-1}(m) \subseteq L(q_i)$. So, from each state $(q_i, m') \in X^{\bar{u}'}_{(M,h)}$ there is an outgoing transition labelled by (a, m). Thus, all the maximal runs over $\bar{u}$ are over the whole word $\bar{u}$.

To prove the second claim, we assume that w is accepted by the automaton $\mathcal{A}$ as otherwise there is just one maximal run of length zero. Let

$$X^{w_{(M,h)}}_{(M,h)} = \{(q_1, m_1), (q_2, m_1), \ldots, (q_\ell, m_1)\}_{\text{MUL}}$$

be the multiset of states reached by the runs labelled $w_{(M,h)}$. Furthermore, because of the proof that all the maximal runs have the same length, we conclude that there are exactly ℓ runs along $w_{(M,h)}$. Moreover, all of them are accepting, as they end in states with the monoid element $h(\epsilon)$. Observe that the states along the runs of $w_{(M,h)}$ never differ on the second component, as the second component is determined by the image h on the suffix of the word. Thus, if two runs over $w_{(M,h)}$ differ, then they differ at the first component and in consequence they induce two different runs along w in $\mathcal{A}$. For example, such two different runs over $w_{(M,h)}$ with the first difference after the letter (a_{i+1}, m_{i+1}):

$$(q, \perp) \xrightarrow{(\varepsilon, m)} (q, m) \xrightarrow{(a_1, m_1)} (p_1, m_1) \xrightarrow{(a_2, m_2)} \ldots (p_i, m_i) \xrightarrow{(a_{i+1}, m_{i+1})} (p_{i+1}, m_{i+1}) \ldots$$
$$\xrightarrow{(a_{n-1}, m_{n-1})} (p_{n-1}, m_{n-1}) \xrightarrow{(a_n, m_n)} (p_n, h(\epsilon)),$$

$$(q, \perp) \xrightarrow{(\varepsilon, m)} (q, m) \xrightarrow{(a_1, m_1)} (p_1, m_1) \xrightarrow{(a_2, m_2)} \ldots (p_i, m_i) \xrightarrow{(a_{i+1}, m_{i+1})} (r_{i+1}, m_{i+1}) \ldots$$
$$\xrightarrow{(a_{n-1}, m_{n-1})} (r_{n-1}, m_{n-1}) \xrightarrow{(a_n, m_n)} (r_n, h(\epsilon)).$$

induce the following two different runs over w:

$$q \xrightarrow{a_1} p_1 \xrightarrow{a_2} \ldots p_i \xrightarrow{a_{i+1}} p_{i+1} \ldots \xrightarrow{a_{n-1}} p_{n-1} \xrightarrow{a_n} p_n,$$

$$q \xrightarrow{a_1} p_1 \xrightarrow{a_2} \ldots p_i \xrightarrow{a_{i+1}} r_{i+1} \ldots \xrightarrow{a_{n-1}} r_{n-1} \xrightarrow{a_n} r_n.$$

Therefore, as $\mathcal{A}$ is k-ambiguous and $\mathcal{A}$ has at least ℓ accepting runs then we necessarily have $\ell \leq k$, which finishes the proof that size of $X^{\bar{u}}$ is at most k. $\square$

6. Future research

VASS accepting by configuration. In our work we prove Theorem 1.2 stating that for a k-ambiguous upward-VASS one can compute a downward-VASS recognising the complement of its language. This theorem implies all our upper bound results, namely decidability of language inclusion of an upward-VASS in a k-ambiguous upward-VASS and language equivalence of k-ambiguous upward-VASS. The most natural question which can be asked in this context is whether Theorem 1.2 or some of its consequences generalises to singleton-VASS (so VASS accepting by a single configuration) or more generally to downward-VASS. Our results about complementing deterministic VASS apply also to downward-VASS. However, generalising our results for nondeterministic (but k-ambiguous or unambiguous) VASS encounters essential barriers. Techniques from Section 4 do not work as the regular-separability result from [CLM$^+$18] applies only to upward-VASS. Techniques from Section 5 break as the proof of Lemma 5.1 essentially uses the fact that the acceptance condition is upward-closed. Thus it seems that one would need to develop novel techniques to handle the language equivalence problem for unambiguous VASS accepting by configuration.

Weighted models. Efficient decidability procedures for language equivalence were obtained for finite automata and for register automata using weighted models [Sch61, BKM21]. For many kinds of systems one can naturally define weighted models by adding weights and computing value of a word in the field $(\mathbb{Q}, +, \cdot)$. The decidability of equivalence for weighted models easily implies language equivalence for unambiguous models, as accepted words always have the output equal to one, while rejected words always have the output equal zero. Thus one can pose a natural conjecture that decidability of language equivalence for unambiguous models always comes as a byproduct of equivalence of the weighted model. Our results show that this is however not always the case as VASS are a counterexample to this conjecture. In the case of upward-VASS language equivalence for unambiguous models is decidable. However, equivalence for weighted VASS is undecidable, as it would imply decidability of path equivalence (for each word, both systems need to accept by the same number of accepting runs), which is undecidable for VASS [Jan01].

Unambiguity and separability. Our result from Section 4 uses the notion of regular-separability in order to obtain a result for unambiguous VASS. This technique seems to generalise to some other well-structured transition systems. It is natural to ask whether there is a deeper connection between the notions of separability and unambiguity that can be explored in future research.

Acknowledgment

We thank Filip Mazowiecki for asking the question for boundedly-ambiguous VASS and formulating the conjecture that control automata of boundedly-ambiguous VASS can be made boundedly-ambiguous. We also thank him and David Purser for inspiring discussions on the problem. We thank Thomas Colcombet for suggesting the way of proving Theorem 4.5, Mahsa Shirmohammadi for pointing us to the undecidability result [Jan01] and Lorenzo Clemente for inspiring discussions on weighted models. We thank Petr Jančar for many helpful remarks and simplifying the proofs of Proposition 2.2 and Lemma 5.4. We also thank Marin Ricros for his detailed comments useful in improving our paper.

References

[AK76] Toshiro Araki and Tadao Kasami. Some Decision Problems Related to the Reachability Problem for Petri Nets. *Theor. Comput. Sci.*, 3(1):85–104, 1976.

[BC21] Corentin Barloy and Lorenzo Clemente. Bidimensional Linear Recursive Sequences and Universality of Unambiguous Register Automata. In *Proceedings of STACS 2021)*, volume 187 of *LIPIcs*, pages 8:1–8:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.STACS.2021.8.

[BKM21] Mikolaj Bojanczyk, Bartek Klin, and Joshua Moerman. Orbit-Finite-Dimensional Vector Spaces and Weighted Register Automata. In *Proceedings of LICS 2021*, pages 1–13. IEEE, 2021. doi:10.1109/LICS52264.2021.9470634.

[Bon11] Rémi Bonnet. The Reachability Problem for Vector Addition System with One Zero-Test. In *Proceedings of MFCS 2011*, volume 6907 of *Lecture Notes in Computer Science*, pages 145–157. Springer, 2011. doi:10.1007/978-3-642-22993-0_16.

[CFH20] Wojciech Czerwinski, Diego Figueira, and Piotr Hofman. Universality Problem for Unambiguous VASS. In *Proceedings of CONCUR 2020*, pages 36:1–36:15, 2020. doi:10.4230/LIPICS.CONCUR.2020.36.

[CFM13] Michaël Cadilhac, Alain Finkel, and Pierre McKenzie. Unambiguous constrained Automata. *Int. J. Found. Comput. Sci.*, 24(7):1099–1116, 2013. doi:10.1142/S0129054113400339.

[CH22] Wojciech Czerwinski and Piotr Hofman. Language Inclusion for Boundedly-Ambiguous Vector Addition Systems Is Decidable. In *Proceedings of CONCUR 2022*, volume 243 of *LIPIcs*, pages 16:1–16:22, 2022. doi:10.4230/LIPICS.CONCUR.2022.16.

[CLM+18] Wojciech Czerwinski, Slawomir Lasota, Roland Meyer, Sebastian Muskalla, K. Narayan Kumar, and Prakash Saivasan. Regular Separability of Well-Structured Transition Systems. In *Proceedings of CONCUR 2018*, volume 118 of *LIPIcs*, pages 35:1–35:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.CONCUR.2018.35.

[CMQ21] Wojciech Czerwinski, Antoine Mottet, and Karin Quaas. New Techniques for Universality in Unambiguous Register Automata. In *Proceedings of ICALP 2021*, volume 198 of *LIPIcs*, pages 129:1–129:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.ICALP.2021.129.

[CO21] Wojciech Czerwinski and Lukasz Orlikowski. Reachability in Vector Addition Systems is Ackermann-complete. In *Proceedings of FOCS 2021*, pages 1229–1240, 2021. doi:10.1109/FOCS52979.2021.00120.

[Col15] Thomas Colcombet. Unambiguity in Automata Theory. In *Proceedings of DCFS 2015*, pages 3–18, 2015. doi:10.1007/978-3-319-19225-3_1.

[Dic13] L.E. Dickson. Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. *American Journal of Mathematics*, 35((4)):413–422, 1913. doi:10.2307/2370405.

[DL09] Stéphane Demri and Ranko Lazic. LTL with the freeze quantifier and register automata. *ACM Trans. Comput. Log.*, 10(3):16:1–16:30, 2009. doi:10.1145/1507244.1507246.

[FS01] Alain Finkel and Philippe Schnoebelen. Well-structured transition systems everywhere! *Theor. Comput. Sci.*, 256(1-2):63–92, 2001. doi:10.1016/S0304-3975(00)00102-X.

[GLHK+20] Jean Goubault-Larrecq, Simon Halfon, Prateek Karandikar, K. Narayan Kumar, and Philippe Schnoebelen. *The Ideal Approach to Computing Closed Subsets in Well-Quasi-orderings*, pages 55–105. Springer International Publishing, Cham, 2020.

[Hac76] Michel Hack. The Equality Problem for Vector Addition Systems is Undecidable. *Theor. Comput. Sci.*, 2(1):77–95, 1976. doi:10.1016/0304-3975(76)90008-6.

[HMT13] Piotr Hofman, Richard Mayr, and Patrick Totzke. Decidability of Weak Simulation on One-Counter Nets. In *Proceedings of LICS 2013*, pages 203–212. IEEE Computer Society, 2013. doi:10.1109/LICS.2013.26.

[HT14] Piotr Hofman and Patrick Totzke. Trace Inclusion for One-Counter Nets Revisited. In *Proceedings of RP 2014*, volume 8762 of *Lecture Notes in Computer Science*, pages 151–162. Springer, 2014. doi:10.1007/978-3-319-11439-2_12.

[Jan95] Petr Jancar. Undecidability of Bisimilarity for Petri Nets and Some Related Problems. *Theor. Comput. Sci.*, 148(2):281–301, 1995. doi:10.1016/0304-3975(95)00037-W.

[Jan01] Petr Jancar. Nonprimitive recursive complexity and undecidability for Petri net equivalences. *Theor. Comput. Sci.*, 256(1-2):23–30, 2001. doi:10.1016/S0304-3975(00)00100-6.

[KP92] M. Kabil and M. Pouzet. Une extension d'un théorème de P. Jullien sur les âges de mots. *RAIRO - Theoretical Informatics and Applications - Informatique Théorique et Applications*, 26(5):449–482, 1992.

[Ler21a] Jérôme Leroux. The Reachability Problem for Petri Nets is Not Primitive Recursive. In *Proceedings of FOCS 2021*, pages 1241–1252, 2021. doi:10.1109/FOCS52979.2021.00121.

[Ler21b] Jérôme Leroux. The Reachability Problem for Petri Nets is Not Primitive Recursive. *CoRR*, abs/2104.12695, 2021.

[LS19] 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, 2019. doi:10.1109/LICS.2019.8785796.

[LS21] Ranko Lazic and Sylvain Schmitz. The ideal view on Rackoff's coverability technique. *Inf. Comput.*, 277:104582, 2021. doi:10.1016/j.ic.2020.104582.

[May81] Ernst W. Mayr. An Algorithm for the General Petri Net Reachability Problem. In *Proceedings of STOC 1981*, pages 238–246, 1981. doi:10.1145/800076.802477.

[MQ19] Antoine Mottet and Karin Quaas. The Containment Problem for Unambiguous Register Automata. In *Proceedings of STACS 2019*, pages 53:1–53:15, 2019. doi:10.4230/LIPICS.STACS.2019.53.

[NSV04] Frank Neven, Thomas Schwentick, and Victor Vianu. Finite state machines for strings over infinite alphabets. *ACM Trans. Comput. Log.*, 5(3):403–435, 2004. doi:10.1145/1013560.1013562.

[Pin97] Jean-Eric Pin. Syntactic Semigroups. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages, Volume 1: Word, Language, Grammar*, pages 679–746. Springer, 1997. doi:10.1007/978-3-642-59136-5_10.

[Rac78] Charles Rackoff. The Covering and Boundedness Problems for Vector Addition Systems. *Theor. Comput. Sci.*, 6:223–231, 1978. doi:10.1016/0304-3975(78)90036-1.

[Rei08] Klaus Reinhardt. Reachability in Petri Nets with Inhibitor Arcs. *Electron. Notes Theor. Comput. Sci.*, 223:239–264, 2008. doi:10.1016/J.ENTCS.2008.12.042.

[Sch61] Marcel Paul Schützenberger. On the Definition of a Family of Automata. *Inf. Control.*, 4(2-3):245–270, 1961. doi:10.1016/S0019-9958(61)80020-X.

[Tze96] Wen-Guey Tzeng. On Path Equivalence of Nondeterministic Finite Automata. *Inf. Process. Lett.*, 58(1):43–46, 1996. doi:10.1016/0020-0190(96)00039-7.

This work is licensed under the Creative Commons Attribution License. To view a copy of this license, visit https://creativecommons.org/licenses/by/4.0/ or send a letter to Creative Commons, 171 Second St, Suite 300, San Francisco, CA 94105, USA, or Eisenacher Strasse 2, 10777 Berlin, Germany