Language Inclusion for Boundedly-Ambiguous Vector Addition Systems Is Decidable

Wojciech Czerwiński
University of Warsaw, Poland

Piotr Hofman
University of Warsaw, Poland

Abstract

We consider the problems of language inclusion and language equivalence for Vector Addition Systems with States (VASSes) 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 VASSes, thus to get decidability we investigate restricted subclasses. On 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 Ackermann-hard already for deterministic VASSes. 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.

2012 ACM Subject Classification Theory of computation → Parallel computing models

Keywords and phrases vector addition systems, language inclusion, language equivalence, determinism, unambiguity, bounded ambiguity, Petri nets, well-structured transition systems

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2022.16

Related Version *Full Version*: `https://arxiv.org/abs/2202.08033` [8]

Funding *Wojciech Czerwiński*: Supported by the ERC grant INFSYS, agreement no. 950398.
Piotr Hofman: Supported by the ERC grant INFSYS, agreement no. 950398.

Acknowledgements We thank Filip Mazowiecki for asking the question for boundedly-ambiguous VASSes and formulating the conjecture that control automata of boundedly-ambiguous VASSes 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 25, Mahsa Shirmohammadi for pointing us to the undecidability result [20] and Lorenzo Clemente for inspiring discussions on weighted models.

1 Introduction

Vector Addition Systems (VASes) together with almost equivalent Petri Nets and Vector Addition Systems with States (VASSes) are one of the most fundamental computational models with a lot of applications in practice for modelling concurrent behaviour. There is also an active field of theoretical research on VASes, with a prominent example being the reachability problem whose complexity was established recently to be Ackermann-complete [23, 11] and [24]. 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 [1] (Theorem 3). They also have shown that the language equivalence (acceptance by configuration) for deterministic VASes 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 [25]. The equality of the reachability sets of two given VASes was also shown

undecidable in the 70-ties by Hack [16]. Jančar has proven in 1995 that the most natural behavioural equivalence, namely the bisimilarity equivalence is undecidable for VASSes [19]. His proof works for only two dimensions (improving the previous results [1]) 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 [20] 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 [17] (Theorem 20) it was shown that already for one-dimensional VASSes 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 VASSes (even in very small dimensions) it is natural to search for subclasses in which the problem is decidable. Decidability of the problem for deterministic VASSes [1, 25] suggests that restricting nondeterminism might be a good idea. Recently a lot of attention was drawn to unambiguous systems [6], 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 [5]. 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 [32] (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 [26, 2, 10] culminating in the work of Bojańczyk, Klin and Moerman [3]. 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 [27] or one register automata with guessing (the proof can be obtained following the lines of [12] as explained in [10]). Recently it was also shown in [7] that the language universality problem for VASSes accepting with states is ExpSpace-complete in the unambiguous case in contrast to Ackermann-hardness in the nondeterministic case (even for one-dimensional VASSes) [18].

Our contribution. In this article we follow the line of [7] and consider problems of language equivalence and inclusion for unambiguous VASSes and also for their generalisations k-ambiguous VASSes (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 [7]. 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 main lower bound result is the following one.

▶ **Theorem 1.** *The language equivalence problem for deterministic VASSes is* Ackermann*-hard.*

Our first important upper bound result is the following one.

▶ **Theorem 2.** *The inclusion problem of a nondeterministic VASS language in an unambiguous VASS language is in* Ackermann*.*

The proof of Theorem 2 is quite simple, but it uses a novel technique. We add a regular lookahead to a VASS and use results about regular-separability of VASSes from [9] to reduce the problem, roughly speaking, to the deterministic case. This technique can be applied to more general systems namely well-structures transition-systems [14]. We believe that it might be interesting on its own and reveal some connection between separability problems and the notion of unambiguity.

Our main technical result concerns VASSes with bounded ambiguity.

▶ **Theorem 3.** *For each $k \in \mathbb{N}$ the language inclusion problem of a VASS in a k-ambiguous VASS is in* Ackermann*.*

Notice that Theorem 3 generalises Theorem 2. We however decided to present separately the proof of Theorem 2 because it presents a different technique of independent interest, which can be applied more generally. Additionally it is a good introduction to a more technically challenging proof of Theorem 3. The proof of Theorem 3 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 be even made k-deterministic (roughly speaking for each word there are at most k runs). Finally we show that the problem of inclusion of a VASS language in a k-deterministic VASS can be reduced to the reachability problem for VASSes which is in Ackermann [24].

On a way to show Theorem 3 we also prove several other lemmas and theorems, which we believe may be interesting on their own. Theorems 1 and 3 together easily imply the following corollary.

▶ **Corollary 4.** *The language equivalence problem is* Ackermann*-complete for:*

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

Organisation of the paper. In Section 2 we introduce the needed notions. Then in Section 3 we present results concerning deterministic VASSes. First we show Theorem 1. Next, we prove that the inclusion problem of a VASS language in a language of a deterministic VASS, a k-deterministic VASS or a VASS with holes (to be defined) is in Ackermann. This is achieved by a reduction to the VASS reachability problem. In Section 4 we define adding a regular lookahead to VASSes. 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 language of deterministic VASS with holes. This latter one is in Ackermann due to Section 3 so the former one is also in Ackermann. In Section 5 we present the proof of Theorem 3 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-deterministic VASSes. Many of the technically involved proofs are moved to the appendix.

2 Preliminaries

Basic notions. 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 zero except of 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 a 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 5** ([9] Lemma 17, [21], [13]). *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, numbers are encoded in binary. The size of representation of upward- or 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 6.** *Let $U \subseteq \mathbb{N}^d$ be an upward-closed set and $D \subseteq \mathbb{N}^d$ be downward-closed set. Then the size of representation of their complements $\overline{U} = \mathbb{N}^d \setminus U$ and $\overline{D} = \mathbb{N}^d \setminus D$ is at most exponential wrt. the sizes $||U||$ and $||D||$, respectively and can be computed in exponential time.*

We prove the Proposition 6 in the appendix. For a more general study (for arbitrary well-quasi orders) see [15].

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. *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 the 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 a configuration $r(v)$ if $p = r$ and $u + v \in \mathbb{N}^d$. We write then $p(v) \xrightarrow{t} q(u + v)$. 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 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 the *labelling* to runs, labelling of a run ρ is

the concatenation of labels of its transitions. Such a run ρ is *from* the configuration c_1 *to* the configuration c'_n and configuration c'_n is *reachable* from 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, we say that the run ρ is *over* the word w.

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 VASSes is a strict superset of the languages accepted by deterministic VASSes. To see that unambiguous VASSes can indeed accept more consider a language $(a^*b)^*a^nc^m$ where $n \geq m$. On 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, it counts down reading c's. As there is exactly one correct guess this VASS is indeed 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.

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

Inclusion problem for VASSes	**Equivalence problem for VASSes**
Input Two VASSes V_1 and V_2.	**Input** Two VASSes V_1 and V_2.
Question Does $L(V_1) \subseteq L(V_2)$?	**Question** Does $L(V_1) = L(V_2)$?

In the sequel, we are mostly interested in VASSes with the set of final configurations F of some special form. We extend the order $\preceq$ on vectors from $\mathbb{N}^d$ 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 5 holds for any well quasi-order, it applies also to $Q \times \mathbb{N}^d$. Proposition 6 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 downward-closed 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*. As in this paper we mostly work with upward-VASSes 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 VASSes. The following emptiness problem is the central problem for VASSes.

Emptiness problem for VASSes
Input A VASS $V = (\Sigma, Q, T, c_I, F)$
Question Does $c_I \longrightarrow c_F$ 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 labels of transitions 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 VASSes 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-VASSes is in Ackermann [24] and actually Ackermann-complete [23, 11].

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

▶ **Lemma 7.** *The emptiness problem for VASSes 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.

We prove Lemma 7 in the appendix. The following is a simple and useful corollary of Lemma 7.

▶ **Corollary 8.** *The emptiness problem for updown-VASSes is in* Ackermann.

Proof. Recall that for updown-VASSes 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 7. Notice that the number of queries is not bigger than the size of the representation of F thus the emptiness problem for updown-VASSes is also in Ackermann. ◀

By Proposition 5 each downward-VASS is also an updown-VASS, thus Corollary 8 implies the following one.

▶ **Corollary 9.** *The emptiness problem for downward-VASSes is in* Ackermann.

Recall that the coverability problem in VASSes is in ExpSpace [29], and the coverability problem is equivalent to the emptiness problem for the set of final configurations being an up-atom. By Proposition 5 we have the following simple corollary which creates an elegant duality for the emptiness problems in VASSes.

▶ **Corollary 10.** *The emptiness problem for upward-VASSes is in* ExpSpace.

Actually, even the following stronger fact is true and helpful for us in the remaining part of the paper, it is shown in [22].

▶ **Proposition 11.** *For each upward-VASS the representation of the downward-closed set of configurations with the empty language can be computed in doubly-exponential time.*

3 Deterministic VASSes

3.1 Lower bound

First we prove a lemma, which easily implies Theorem 1.

▶ **Lemma 12.** *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*

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

The sketch of the proof. To prove the lemma we take V and we 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. Now it is easy to see that $L(V_1) \neq L(V_2)$ if and only if c_F can be reached. Detailed proof is in the appendix.

Notice that Lemma 12 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 VASSes. This proves Theorem 1 as the emptiness problem, even with zero counter values of the final configuration is Ackermann-hard [23, 11].

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-VASSes simply VASSes and for VASSes with other acceptance conditions we use their full name (like downward-VASSes or updown-VASSes) to distinguish them from upward-VASSes. The following lemma is very useful in our strategy of deciding the inclusion problem for VASS languages.

▶ **Lemma 13.** *For a VASS V_1 and a downward-VASS V_2 one can construct in polynomial time an updown-VASS V such that $L(V) = L(V_1) \cap L(V_2)$.*

The proof is in the appendix.

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

▶ **Theorem 14.** *For a deterministic VASS one can build in exponential time a downward-VASS which recognises the complement of its language.*

Sketch of the proof. A word may be in the complement of our VASS language for the following reasons: (1) the run reaches a configuration that is not accepted, (2) the run does not exist as one of the counters would drop below zero, (3) the run is not possible due to the structure of the control automaton. For each case we separately design a part of a downward-VASS accepting it. Cases (1) and (3) are simple. For the 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. Detailed proof is in the appendix.

The following theorem is a simple corollary of Theorem 14, Lemma 13 and Corollary 8.

▶ **Theorem 15.** *The inclusion problem of a VASS language in a deterministic VASS language is in* Ackermann.

Deterministic VASSes with holes. We define here VASSes with holes, which are a useful tool to obtain our results about unambiguous VASSes 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 HVASSes 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 VASSes having ε-labelled transitions, we call such VASSes the ε-VASSes. 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 HVASSes in dimension zero are exactly VASSes in dimension zero, so finite automata. However, for higher dimensions the notions of HVASSes and VASSes differ.

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

▶ **Lemma 16.** *For each HVASS one can compute in exponential time a language equivalent ε-VASS.*

Sketch of the proof. 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 every 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.

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 14. Thus in order to prove the decidability of the inclusion problem for HVASSes we need to generalise Theorem 14 to HVASSes.

▶ **Theorem 17.** *For a deterministic HVASS one can compute in exponential time a downward-ε-VASS which recognises the complement of its language.*

Sketch of the proof. The proof is very similar to the proof of Theorem 14. In the case (1) we have to check if the accepting run stays above the holes, do perform it we use the trick from Lemma 16. In the 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 14.

Now the following theorem is an easy consequence of the shown facts. We need only to observe that proofs of Lemma 13 and Corollary 8 work as well for ε-VASSes.

▶ **Theorem 18.** *The inclusion problem of an HVASS language in a deterministic HVASS language is in* Ackermann.

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

A VASS $V = (\Sigma, Q, T, c_I, F)$ 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 which have ever been started during the word. To illustrate the difference better let us consider an example 0-VASS (a finite automaton) V over $\Sigma = \{a, b\}$ with two states p, q and with three transitions: (p, a, p), (p, a, q) and (q, b, q). Then V 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 V is not 2-deterministic even though for each input word it has at most two runs.

▶ **Theorem 19.** *For a k-deterministic d-VASS one can build in exponential time a $(k \cdot d)$-dimensional downward-VASS which recognises the complement of its language.*

Sketch of the proof. In the construction $(k \cdot d)$-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 the copies there is a reason that the simulated maximal runs do not accept. The reasons why each individual copy do not accepts are the same as in Theorem 14.

Theorem 19 together with Lemma 13 and Corollary 8 easily implies (analogously as in the proof of Theorem 18) the following theorem.

▶ **Theorem 20.** *The inclusion problem of a VASS language in a k-deterministic VASS language is in* Ackermann.

4 Unambiguous VASSes

In this section we aim to prove Theorem 2. However, possibly a more valuable contribution of this section is a novel technique which we introduce in order to show Theorem 2. 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 the use of finite monoids.

The high level intuition behind the proof of Theorem 2 is the following. We first introduce the notion of (M, h)-decoration of words, languages and VASSes, where M is a monoid and $h : \Sigma^* \to M$ is a homomorphism. Proposition 23 states that language inclusion of

two VASSes can be reduced to language inclusion of its decorations. On the other hand Theorem 26 shows that for appropriately chosen pair (M, h) the decorations of unambiguous VASSes are deterministic HVASSes. Theorem 25 states that such an appropriate pair can be computed quickly enough. Thus language inclusion of unambiguous VASSes reduces to language inclusion of deterministic HVASSes, which is in Ackermann due to Theorem 18.

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 [28] (Proposition 3.12).

▶ **Proposition 21.** *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 21 implies that for each finite family of regular languages there is a monoid, which recognises all of them, this fact is useful in Theorem 26. Let us 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 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 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 22.** *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

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

As the (M, h)-decoration is a function from 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 23.** *For each finite alphabet Σ, two languages $K, L \subseteq \Sigma^*$, monoid M and homomorphism $h : \Sigma^* \to M$ we have*

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

Recall now that HVASS (VASS with holes) is a VASS with some downward-closed set H of prohibited configurations (see Section 3, paragraph Deterministic VASSes 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 $L(V_{(M,h)}) = L(V)_{(M,h)}$. Till 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, namely $(q,m)(v)$ 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 11.

By Proposition 23 we know that for two VASSes 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 say than 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 [9].

▶ **Theorem 24** (Theorem 24 in [9]). *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 the regular separator in elementary time.*

Proof. Theorem 24 in [9] 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 10 one can check in doubly-exponential time whether they nonempty intersect. ◀

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 25.** *For any VASS one can compute in an elementary time a finite family of regular languages which 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 11 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 finish 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 $V_1 = (\Sigma, Q, T_1, c_I, F_1)$ 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 $u_1[i] = \omega$,
- $(q, v_1) \in F_1$ 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 24 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 26.** *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 some two 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 with the second component equal 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 $V_{(M,h)}$. This means however that there are at least two accepting runs over the projection of $u \cdot (a, m') \cdot w$ in V, which contradicts 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 the languages in $\mathcal{F}$ are recognisable by (M, h) thus words from $L(c_1)$ should be mapped by the homomorphism h to different elements of 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 2. 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 25 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 23 we get that $L(V_1) \subseteq L(V_2)$ if and only if $L_{(M,h)}(V_1) \subseteq L_{(M,h)}(V_2)$. We now compute HVASSes $V_1' = V_{1_{(M,h)}}$ and $V_2' = V_{2_{(M,h)}}$. By Theorem 26 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 18.

▶ Remark 27. We remark that our technique can be applied not only to VASSes but also in a more general setting of well-structured transition systems. In [9] it was shown that for any well-structured transition systems fulfilling some mild conditions (finite branching is enough) disjointness of two languages implies regular separability of these languages. We claim that an analogue of our Theorem 25 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 14 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 VASSes with one reset. The emptiness problem for VASSes with one zero-test (and thus also for VASSes with one reset) is decidable due to [30, 4]. 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 VASSes

In this section we aim to prove Theorem 3. It is an easy consequence of the following theorem.

▶ **Theorem 28.** *For any $k \in \mathbb{N}$ and a k-ambiguous VASS one can build in elementary time a downward-VASS which recognises the complement of its language.*

Let us show how Theorem 28 implies Theorem 3. Let V_1 be a VASS and V_2 be a k-ambiguous VASS. By Theorem 28 one can compute in elementary time a downward-VASS V_2' such that $L(V_2') = \Sigma^* \setminus L(V_2)$. By Lemma 13 one can construct in time polynomial wrt.

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 8 emptiness of V is decidable in Ackermann which in consequence proves Theorem 3.

Thus the rest of this section focuses on the proof of Theorem 28.

Proof of Theorem 28. We prove now Theorem 28 using Lemmas 29 and 30 stated below. Then in Sections D and D in the appendix we prove the formulated lemmas. Let V be a k-ambiguous VASS over an alphabet Σ. In the proof we construct a sequence of VASSes $V^1, V^2, \ldots, V^6$ related in various ways to V with the property that V^6 is a downward-VASS and $L(V^6)$ is exactly the complement of $L(V)$. More concretely $L(V^1)$ equals $L(V)$, $L(V^2)$ is a decoration of $L(V)$, $L(V^3)$ is the complement of $L(V^2)$, while V^4, V^5 recognise more sophisticated languages related to $L(V^3)$.

First due to Lemma 29 proved in Section D we construct a VASS V^1 which is language equivalent to V and additionally has the control automaton being k-ambiguous.

▶ **Lemma 29.** *For each k-ambiguous VASS V one can construct in doubly-exponential time a language equivalent VASS V' with the property that its control automaton is k-ambiguous.*

Now our aim is to get a k-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-deterministic VASS V^2. We use the following lemma which is proved in Section D.

▶ **Lemma 30.** *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 $h : \Sigma^* \to M$ which 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 30 $\mathcal{A}_{(M,h)}$ is k-deterministic which immediately implies that V^2 is k-deterministic as well.

Now by Theorem 19 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 not well-formed words and all the well-formed words which project into the complement of $L(V)$. By Proposition 22 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 $L(V)$. It is easy now 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 just 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 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 28. 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 3 along the same lines as it is proved now. ◀

6 Future research

VASSes accepting by configuration. In our work we prove Theorem 28 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-VASSes. The most natural question which can be asked in this context is whether Theorem 28 or some of its consequences generalises to singleton-VASSes (so VASSes accepting by a single configuration) or more generally to downward-VASSes. Our results about complementing deterministic VASSes apply also to downward-VASSes. However generalising our results for nondeterministic (but k-ambiguous or unambiguous) VASSes encounter essential barriers. Techniques from Section 4 do not work as the regular-separability result from [9] applies only to upward-VASSes. Techniques from Section 5 break as the proof of Lemma 29 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 VASSes accepting by configuration.

Weighted models. Efficient decidability procedures for language equivalence were obtained for finite automata and for register automata with the use of weighted models [31, 3]. For many kinds of systems one can naturally define weighted models by adding weights and computing value of a word in the field $(\mathbb{Q}, +, \cdot)$. Decidability of equivalence for weighted models easily implies language equivalence for unambiguous models as accepted words always have the output equal 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 VASSes are a counterexample to this conjecture. In the case of upward-VASSes language equivalence for unambiguous models is decidable. However equivalence for weighted VASSes 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 VASSes [20].

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

References

1 Toshiro Araki and Tadao Kasami. Some decision problems related to the reachability problem for Petri nets. *Theor. Comput. Sci.*, 3(1):85–104, 1976.

2 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.

3 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.

4 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.

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

6 Thomas Colcombet. Unambiguity in automata theory. In *Proceedings of DCFS 2015*, pages 3–18, 2015.

7 Wojciech Czerwinski, Diego Figueira, and Piotr Hofman. Universality problem for unambiguous VASS. In *Proceedings of CONCUR 2020*, pages 36:1–36:15, 2020.

8 Wojciech Czerwinski and Piotr Hofman. Language inclusion for boundedly-ambiguous vector addition systems is decidable. *CoRR*, abs/2202.08033, 2022. arXiv:2202.08033.

9 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.

10 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.

11 Wojciech Czerwinski and Lukasz Orlikowski. Reachability in vector addition systems is Ackermann-complete. In *Proceedings of FOCS 2021*, pages 1229–1240, 2021.

12 Stéphane Demri and Ranko Lazic. LTL with the freeze quantifier and register automata. *ACM Trans. Comput. Log.*, 10(3):16:1–16:30, 2009.

13 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.

14 Alain Finkel and Philippe Schnoebelen. Well-structured transition systems everywhere! *Theor. Comput. Sci.*, 256(1-2):63–92, 2001.

15 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.

16 Michel Hack. The equality problem for vector addition systems is undecidable. *Theor. Comput. Sci.*, 2(1):77–95, 1976.

17 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.

18 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.

19 Petr Jancar. Undecidability of bisimilarity for Petri nets and some related problems. *Theor. Comput. Sci.*, 148(2):281–301, 1995.

20 Petr Jancar. Nonprimitive recursive complexity and undecidability for petri net equivalences. *Theor. Comput. Sci.*, 256(1-2):23–30, 2001.

21 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.

22 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.

23 Jérôme Leroux. The reachability problem for petri nets is not primitive recursive. In *Proceedings of FOCS 2021*, pages 1241–1252, 2021.

24 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.

25 Ernst W. Mayr. An algorithm for the general Petri net reachability problem. In *Proceedings of STOC 1981*, pages 238–246, 1981.

26 Antoine Mottet and Karin Quaas. The containment problem for unambiguous register automata. In *Proceedings of STACS 2019*, pages 53:1–53:15, 2019.

27 Frank Neven, Thomas Schwentick, and Victor Vianu. Finite state machines for strings over infinite alphabets. *ACM Trans. Comput. Log.*, 5(3):403–435, 2004.

28 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.

29 Charles Rackoff. The covering and boundedness problems for vector addition systems. *Theor. Comput. Sci.*, 6:223–231, 1978.

30 Klaus Reinhardt. Reachability in Petri nets with inhibitor arcs. *Electron. Notes Theor. Comput. Sci.*, 223:239–264, 2008.

31 Marcel Paul Schützenberger. On the definition of a family of automata. *Inf. Control.*, 4(2-3):245–270, 1961.

32 Wen-Guey Tzeng. On path equivalence of nondeterministic finite automata. *Inf. Process. Lett.*, 58(1):43–46, 1996.

A Missing proofs from Section 2

We recall the statement of Proposition 6.

Proposition 6. Let $U \subseteq \mathbb{N}^d$ be an upward-closed set and $D \subseteq \mathbb{N}^d$ be downward-closed set. Then the size of representation of their complements $\overline{U} = \mathbb{N}^d \setminus U$ and $\overline{D} = \mathbb{N}^d \setminus D$ is at most exponential wrt. the sizes $||U||$ and $||D||$, respectively and can be computed in exponential time.

Proof of Proposition 6. 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 $U = u_1{\uparrow} \cup u_2{\uparrow} \cup \ldots \cup u_n{\uparrow}$. Then

$$\begin{aligned}\overline{U} &= \mathbb{N}^d \setminus U = \mathbb{N}^d \setminus (u_1{\uparrow} \cup u_2{\uparrow} \cup \ldots \cup u_n{\uparrow}) \\ &= (\mathbb{N}^d \setminus u_1{\uparrow}) \cap (\mathbb{N}^d \setminus u_2{\uparrow}) \cap \ldots \cap (\mathbb{N}^d \setminus u_n{\uparrow}).\end{aligned}$$

Thus in order to show that $||\overline{U}||$ is at most exponential wrt. $||U||$ we need to face two challenges. The first one is to show that representation of $(\mathbb{N}^d \setminus u{\uparrow})$ for $u \in \mathbb{N}^d$ is not too big wrt. size of u and the second one is to show that the intersection of sets $(\mathbb{N}^d \setminus u{\uparrow})$ does not introduce too big blowup.

Let us first focus on the first challenge. Let $|u|$ be the biggest value that appear in u i.e. $|u| = \max\{u[i] : i \in [1,d]\}$. We claim that if $v \in \mathbb{N}^d \setminus u{\uparrow}$ and $v[i] > |u|$ for $i \in [1,d]$ then $v + e_i \in \mathbb{N}^d \setminus u{\uparrow}$. Indeed, if $v \in \mathbb{N}^d \setminus u{\uparrow}$ then there is $j \in [1,d]$ such that $v[j] < u[j]$. Of course $i \neq j$, so $v + e_i \not\geq u$ and thus $v + e_i \in \mathbb{N}^d \setminus u{\uparrow}$. But this means that if $\hat{v} \in (\mathbb{N} \cup \{\omega\})^d$ such that $\hat{v}{\downarrow} \subseteq \mathbb{N}^d \setminus u{\uparrow}$ and $\hat{v}$ is maximal (namely its entries cannot be increased without violating $\hat{v}{\downarrow} \subseteq \mathbb{N}^d \setminus u{\uparrow}$) then $\hat{v} \in ([0,|u|] \cup \{\omega\})^d$. Thus there are only exponentially many possibilities for $\hat{v}$ and the representation of $\mathbb{N}^d \setminus u{\uparrow}$ is at most exponentially bigger than the representation of u.

Let us face now the second challenge. Let $\hat{v}_1, \hat{v}_2 \in ([1,|u|] \cup \{\omega\})^d$. Observe that $v \in \hat{v}_1{\downarrow} \cap \hat{v}_2{\downarrow}$ if and only if $v[i] \leq \hat{v}_1[i]$ and $v[i] \leq \hat{v}_2[i]$ for all $i \in [1,d]$. But this means that if $\hat{v} \in (\mathbb{N} \cup \{\omega\})^d$ and $\hat{v}{\downarrow} \subseteq \hat{v}_1{\downarrow} \cap \hat{v}_2{\downarrow}$ is maximal then $\hat{v} \in ([0,|u|] \cup \{\omega\})^d$. Thus the representation of $\mathbb{N}^d \setminus U$ is also only at most exponentially bigger than the representation of U.

In order to compute the representation of U one can simply check for all $\hat{v} \in ([0,|u|] \cup \{\omega\})^d$ whether $\hat{v}{\downarrow} \subseteq \mathbb{N}^d \setminus U$. ◀

We recall the statement of Lemma 7.

Lemma 7. The emptiness problem for VASSes 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 of Lemma 7. We provide a polynomial reduction of the problem to the emptiness problem in singleton-VASSes 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 dimension $i \in [1, d_1]$ 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 dimension i, these are the dimensions corresponding to the up-atom U. Similarly for each dimension $i \in [d_1 + 1, d_1 + d_U]$ we add in q'_F a loop with the effect $-e_i$, these are the unbounded dimensions corresponding to the down-atom D. Finally for each dimension $i \in [d_1 + d_U + 1, d]$ 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 dimensions corresponding to the 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 problem in V and in V' are equivalent which finishes the proof. ◀

B Missing proofs from Section 3.1

We recall the statement of Lemma 12.

Lemma 12. 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

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

Proof of Lemma 12. 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, 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 dimensions and on the last $(d+1)$-th dimension it keeps the sum of all the others. Notice that 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 equal 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 $c'_I = q_I(x_1, x_2, \ldots x_d, \sum_{i=1}^{d} x_i)$ where $c_I = q_I(x_1, x_2, \ldots x_d)$. 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-VASSes, 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. ◀

C Missing proofs from Section 3.2

We recall the statement of Lemma 13.

Lemma 13. For a VASS V_1 and a downward-VASS V_2 one can construct in polynomial time an updown-VASS V such that $L(V) = L(V_1) \cap L(V_2)$.

Proof of Lemma 13. 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 5 a finite union of $q(U \times D)$ for a state q of V, an up-atom U and a down-atom D. ◀

We recall the statement of Theorem 14.

Theorem 14. For a deterministic VASS one can build in exponential time a downward-VASS which recognises the complement of its language.

Proof of Theorem 14. 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) a possible transition over a would decrease some of the counters below zero, (3) there is no such transition possible in V in the state of c.

We are ready to describe VASS $V' = (\Sigma, Q', T', c'_I, F')$. Roughly speaking it consists of $|T| + |\Sigma| + 1$ copies of V. Concretely the set of states Q' is the set of pairs $Q \times (T \cup \Sigma \cup \{-\})$. 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 because of 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 wrt. the size of the description of F by Proposition 6.

Now we describe the second part of V' which is responsible for words rejected by V because of the scenario (2). The idea is that we guess when the run over w is finished. For each transition $t = (p, a, v, q) \in T$ we add $(p', a, 0^d, q')$ to T' where $p' = (p, -)$ and $q' = (q, t)$. The idea is that the run reaches the configuration in which the transition t cannot be fired. Now we have to check that our guess is correct. In the state (q, t) for $t \in T$ no transition changes the configuration. Namely for each $q' = (q, t) \in Q \times T$ and each $a \in \Sigma$ we add to T' transition $(q', a, 0^d, q')$. We add now to F' the set $F_2 = \{(q, t)(v) \mid v + \mathrm{eff}(t) \notin \mathbb{N}^d\}$. 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), namely 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 $v \in \mathbb{N}^d$ and $q \in Q$ we add to T' transition $((p, -), a, 0^d, (p, a))$. In each state $p' = (p, a) \in Q \times \Sigma$ we have a transition $(p', b, 0^d, p')$ for each $b \in \Sigma$. We also add to F' the set $F_3 = \{(p, a)(v) \mid v \in \mathbb{N}^d$ and there is no $(p, a, u, q) \in T$ for $u \in \mathbb{N}^d$ and $q \in Q\}$. Size of F_3 is polynomial wrt. 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. ◀

We recall the statement of Lemma 16.

Lemma 16. For each HVASS one can compute in exponential time a language equivalent ε-VASS.

Proof of Lemma 16. 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 6 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. In order 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 the ones before the test and the states in $Q \times \{1\}$ are the ones after the test. 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 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. ◀

We recall the statement of Theorem 17.

Theorem 17. For a deterministic HVASS one can compute in exponential time a downward-ε-VASS which recognises the complement of its language.

Proof of Theorem 17. The proof of Theorem 17 is very similar to the proof of Theorem 14 so 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 6 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 wrt. the size $||H||$.

The construction of V' recognising the complement of $L(V)$ is almost the same as in the proof of Theorem 14, we need to introduce only small changes. The biggest changes are in the part of V' recognising 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 16. Namely we guess to which $q_i(u_i{\uparrow})$ the current configuration belongs and check it by simple VASS modifications (for details look to the proof of Lemma 16). The size of this part of V' can have a blowup of at most size of U times, namely the size can be multiplied by some number, which is at most exponential wrt. the size $||H||$.

In the part recognising words rejected by V because of scenario (2), we need only 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 14). This change does not introduce any new superlinear blowup.

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 wrt. the size V as explained above, which finishes the proof. ◀

We recall the statement of Theorem 18.

Theorem 18. The inclusion problem of an HVASS language in a deterministic HVASS language is in Ackermann.

Proof of Theorem 18. 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 16 an ε-VASS V_1' equivalent to V_1 can be computed in exponential time. By Theorem 17 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 13 (extended to ε-VASSes) one can compute an updown-ε-VASS V such that $L(V) = L(V_1') \cap L(V_2')$. Finally by Corollary 8 (also extended to ε-VASSes) the emptiness problem for updown-ε-VASSes is in Ackermann which finishes the proof. ◀

We recall the statement of Theorem 19.

Theorem 19. For a k-deterministic d-VASS one can build in exponential time a $(k \cdot d)$-dimensional downward-VASS which recognises the complement of its language.

Proof of Theorem 19. Before starting the proof let us remark that it would seem natural to first build a $(k \cdot d)$-VASS equivalent to the input k-deterministic d-VASS and then apply construction from the proof of Theorem 14 to recognise the its complement. However, it is not clear how to construct a $(k \cdot d)$-VASS equivalent to k-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-deterministic d-VASS. We aim to construct $(k \cdot d)$-dimensional downward-VASS $V' = (\Sigma, Q', T', c'_I, F')$ such that $L(V') = \Sigma^* \setminus L(V)$. Also in this proof we strongly rely on the ideas introduced in the proof of Theorem 14. 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 the copies there is a reason that 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 14 it was shown how a VASS can handle all the 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 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 till the first moment 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 are following the same maximal run and which have already split. Each copy is exactly as in the proof of Theorem 14, it realises one of the scenarios (1), (2) or (3). As we know that V is k-deterministic we are sure that all the possible runs of V can be simulated by V' under the condition the V' correctly guesses which copies should simulate which runs. If guesses of V' are wrong and at some point it cannot send to each branch a copy then the run of V' rejects. At the end of the run over the input word w VASS V' checks using the accepting condition F' that indeed all the copies have simulated all the possible maximal runs and that all of them reject. It is easy to see that F' is a downward-closed set, as roughly speaking it is a product of k downward-closed accepting conditions, which finishes the proof. ◀

D Missing proofs from Section 5

The proofs from this section are available only in the arxiv version of this paper because of the space limitation. Please check `https://arxiv.org/pdf/2202.08033.pdf`.