Polynomial-Space Completeness of Reachability for Succinct Branching VASS in Dimension One*

Diego Figueira[1], Ranko Lazić[2], Jérôme Leroux[3], Filip Mazowiecki[4], and Grégoire Sutre[5]

1 **LaBRI, CNRS, Bordeaux, France**
diego.figueira@labri.fr
2 **DIMAP, University of Warwick, Warwick, UK**
R.S.Lazic@warwick.ac.uk
3 **LaBRI, CNRS, Bordeaux, France**
jerome.leroux@labri.fr
4 **DIMAP, University of Warwick, Warwick, UK**
f.mazowiecki@warwick.ac.uk
5 **LaBRI, CNRS, Bordeaux, France**
gregoire.sutre@labri.fr

Abstract

Whether the reachability problem for branching vector addition systems, or equivalently the provability problem for multiplicative exponential linear logic, is decidable has been a long-standing open question. The one-dimensional case is a generalisation of the extensively studied one-counter nets, and it was recently established polynomial-time complete provided counter updates are given in unary. Our main contribution is to determine the complexity when the encoding is binary: polynomial-space complete.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases branching vector addition systems, reachability problem

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.119

1 Introduction

Background. Vector addition systems, also known as Petri nets (cf., e.g., Reisig's book [21]), are one of the longest established, most extensively studied, and most widely applied models of concurrent computing systems. Their branching generalisation has attracted considerable attention in recent years from the research community on logic in computer science. In addition to the simplicity and elegance of the model, this popularity is due to remarkably close connections with computational linguistics [20, 22], cryptographic protocols [25], linear logic [8, 16], semi-structured databases [13, 1], recursively parallel programs [5], game semantics [7], and timed pushdown systems [6].

A central decision problem for branching vector addition systems is reachability: whether a computation tree exists that has the given root and leaves. Similarly to the simpler setting of Petri nets, this problem has turned out to be very challenging. However, in contrast to Petri nets where the challenge is determining the complexity of reachability below a currently best cubic-Ackermann bound [17], even decidability is still open for the branching vector addition systems reachability problem. For reasons indicated above, the latter question was

* Partially supported by EPSRC grant EP/M011801/1 and Royal Society grant IE150122.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 119; pp. 119:1–119:14

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Table 1 Complexity of reachability for one-dimensional vector addition systems with states, depending on the presence of branching and the encoding of counter updates.

	unary	binary
1VASS	NL-complete [24, 15]	NP-complete [11]
1BVASS	P-complete [10]	PSPACE-complete

recently highlighted by Bojańczyk as one of a handful of most interesting open problems in computer science logic [4].[1]

The decidability of the branching reachability problem is in fact open already in two dimensions. However, in one dimension, i.e. when there is only one counter, Göller et al. [10] established decidability, and more precisely polynomial-time completeness provided the numbers that specify the counter updates in the system are given in unary. The precise complexity with the encoding in binary remained undetermined.

From another point of view, our investigation builds on the voluminous literature on decision problems for one-counter automata, a ubiquitous class obtained by either dropping one counter from Minsky (two-counter) machines or restricting pushdown automata to one stack symbol. In particular, the complexity of the reachability problem for one-counter systems is known: NL-completeness with the updates given in unary is a classical result [24, 15], and NP-completeness for succinct systems is due to Haase et al. [11] (cf. the latter paper for further references on the subject).

Contributions. Our main result is the closure of the complexity gap for the reachability problem on succinct one-dimensional branching vector addition systems with states (1BVASS), which was between NP hardness inherited from 1VASS [11] and EXPTIME membership that follows from the P membership for unary 1BVASS [10].[2] We show that the problem is in fact PSPACE-complete, which fills the little Table 1.

The fact that the complexities for 1BVASS correspond exactly to 'adding alternation' to the complexities for 1VASS makes them easy to remember. However, it is quite misleading in terms of proofs, at least as far as we can see. The branchings in computations of BVASS are not alternations: counter valuations at child nodes are summed, not compared for equality.[3] Already in the unary case, the proof of P-completeness for 1BVASS [10] is considerably more involved than of NL-completeness for 1VASS [24, 15]. In our proof of PSPACE-completeness for binary 1BVASS, there are several substantial new insights in comparison to both unary 1BVASS and binary 1VASS [11]:

- we introduce a novel notion of implicit reachability witnesses, show that such a witness of at most an exponential size always exists, and hence argue that it can be guessed and checked in polynomial space;
- for the exponential bound on the size of witnesses, a polynomial bound on their counter valuations as for unary 1BVASS [10] is not sufficient because trees with exponentially long branches may be doubly exponentially large;

[1] Although decidability has been stated in a published journal article [2], we believe that claim has not been accepted by the community due to lack of proof, cf. [23, Footnote 4].

[2] We remark that we write 'with states' because stateless (B)VAS are sometimes considered in higher dimensions since states can be encoded at the expense of three additional counters; and that 1VASS, i.e. one-counter nets, are as hard as one-counter systems in this context since the ability to zero-test the counter does not make reachability significantly more complex.

[3] Reachability for alternating VASS is actually undecidable, for relatively trivial reasons [19].

- one of the techniques we employ for establishing the exponential bound involves a novel rewriting strategy, which may be of wider interest since it transforms fragments of computation trees to a normal form that features principal branches, whereas the lack of such a structure has hitherto been an obstacle to generalising Kosaraju's approach [14, 17] to BVASS;
- in contrast to the other three hardness results summarised in Table 1, our lower bound proof is highly intricate, resting on a system of encodings and checks through which alternation can be simulated by additive branching up to a linear depth.

Organisation. After the next section in which we define the systems we consider and observe some of their basic properties, the two sections that follow contain the PSPACE-membership proof. In the penultimate section, we present the PSPACE-hardness construction, and then finish with some concluding remarks.

2 Preliminaries

1BVASS. A *one-dimensional branching vector addition system with states* (*1BVASS* for short) is a triple $B = (Q, \Delta, I)$ where Q is a non-empty finite set of *states*, Δ is a non-empty finite subset of $Q \times Q \times \mathbb{Z} \times Q$, and $I \subseteq Q$ is a finite set of *initial states*. An element $\delta = (q_L, q_R, z, q)$ in Δ is called a *transition*, and the integer z is called the *displacement* of the transition. In the sequel, the maximal absolute displacement is denoted by M. A *configuration* is a pair in $Q \times \mathbb{N}$, and a configuration in $I \times \{0\}$ is called an *initial configuration*. Since the displacements are given in binary, we define the size of B as $|B| = |Q| + |\Delta| \log_2(M+1)$.

Trees. We write $u \preceq v$ if u is a prefix of v and $u \prec v$ if u is a strict prefix. A *tree* is a non-empty finite prefix-closed subset T of $\{L, R\}^*$ satisfying the property that $tL \in T$ if, and only if, $tR \in T$ for every $t \in T$. Elements of T are called *nodes*. Its *root* is the empty word ε. An *ancestor* s of a node t is a prefix of t. In that case t is called a *descendant* of s. By writing *strict* descendants and ancestors we exclude $s = t$. A *child* of a node t is a node tL or tR in T. A node is called a *leaf* if it has no child, and it is said *internal* otherwise. The *sibling* of a node $t \neq \varepsilon$ in the tree T is the node obtained by swapping the last letter. The *height* of a node t is $|t|$. The size of a tree T is its cardinality $|T|$. The *height* of T is the maximal height of any of its nodes. The *subtree* of T rooted at a node t in T is the tree $t^{-1}T = \{t' \in \{L, R\}^* \mid tt' \in T\}$. The *truncation* of T at a node t is the tree $T \backslash t\{L, R\}^+$. Notice that t becomes a leaf of that truncated tree.

Runs and Reachability. We consider labeled trees T where each node t is labeled by a state $q_t \in Q$ and a value $n_t \in \mathbb{N}$ defining a configuration (q_t, n_t). A *run* ρ is a labeled tree such that for every internal node t, there exists an integer z such that (q_{tL}, q_{tR}, z, q_t) is a transition in Δ and such that $n_t = n_{tL} + n_{tR} + z$. The notions of height, size, subtree (called subrun in that context), and truncation are extended from trees to runs as one would expect. Notice that the labels are ignored in the size of a run.

A run is said to be *complete* if every leaf is labeled by an initial configuration. We write *partial run*, instead of run, when we want to emphasize that the run could be not complete. A configuration is said to be *reachable* if it is the root configuration of a complete run. We are mostly interested in the *reachability problem*: given a 1BVASS B and a configuration (q, n) decide whether (q, n) is reachable. The size of the input is $|B| + \log_2(n+1)$.

▶ **Example 1.** Fix numbers n, b such that $0 \leq b \leq 2^n$. We define the 1BVASS $B = (Q, \Delta, I)$, where $Q = \{q_1 \dots q_n\} \cup \{q_I, q_F\}$ and $I = \{q_I\}$. There are three types of transitions:

$$(q_I, q_I, 0, q_1), (q_I, q_I, 1, q_1); \quad (q_i, q_i, 0, q_{i+1}) \text{ for all } i < n; \quad (q_n, q_n, -b, q_F).$$

The first two transitions initialize q_1 with 0 or 1; the next n transitions build a full binary tree below each state q_i; and the last transition decreases the value of the counter by b. Consider the reachability problem of $(q_F, 0)$. The complete runs with $(q_F, 0)$ in the root are full binary trees of height $n+1$ such that the number of nodes with state q_1 is 2^n and exactly b of them have value 1.

Contexts and Concatenation. A *context* $\pi = (\rho, t)$ is a run ρ equipped with a distinguished leaf t called the *source* of π. The label of t is called the *source configuration* of π. Such a context is also called a context from the source configuration up to the root configuration of ρ. Given a node t in a run ρ and an ancestor s of t, i.e. such that $t = su$ for some word u, we define the context between (s, t) as the subrun rooted at s of the truncation at t of ρ, equipped with u as the source node. An ancestor of the source t is called a *main node* of π. The set of main nodes of π is called the *main branch*. A *dangling node* in π is a node that is a sibling of a main node. A *dangling configuration* is a configuration of such a node.

The concatenation $\pi\rho$ of a context π with a run ρ is defined if the source configuration (p, m) of π and the root configuration (q, n) of ρ satisfy $p = q$ and if the natural numbers labeling the main nodes of π are larger than or equal to $m - n$. Then $\pi\rho$ is defined by adding to the main nodes of π the integer $n - m$, and replacing the leaf node t of that context with ρ. Notice that $\pi\rho$ is a run. Contexts can be *concatenated* a similar way. The concatenation $\pi\pi'$ of a context $\pi = (\rho, t)$ with a context $\pi' = (\rho', t')$ is defined if $\pi\rho'$ is defined. In that case $\pi\pi'$ is the context $(\pi\rho', tt')$.

Cycles and Minimal Nodes. A context from (p, m) up to (q, n) is called a *cycle* if the source is distinct from the root node and $p = q$. The cycle is said to be simple if on the main branch only the source and the root have the same states. A main node v is said to be *minimal* in a cycle if its value is minimal on the main branch, i.e., $n_v \leq n_{v'}$ for any other main node v'. We write *d-cycle* do emphasize the growth of the cycle, where $d = n - m$. The cycle is said to be *increasing* if $d > 0$, *zero* if $d = 0$, and *decreasing* if $d < 0$.

Let π be a d-cycle, and let p be the state of its source node. Let $\rho = \rho_1\pi\rho_2$ be a context or a run. By *removing* π from ρ we obtain $\rho' = \rho_1\rho_2$ (provided there is no drop below 0). Similarly, let $\rho = \rho_1\rho_2$ be a context or a run such that the source of ρ_1 has state label p. By *inserting* π into v we obtain $\rho_1\pi\rho_2$ (provided that there is no drop below 0). Notice that it is always safe to remove decreasing cycles and to add increasing cycles.

3 d-Coverability

A key role in our polynomial-space algorithm for the reachability problem is played by a more relaxed notion of 'coverability modulo d': instead of reaching a value x exactly, it is allowed to reach any value which is at least x, provided the difference with x is a multiple of d. More formally, a configuration (q, n) is said to be *d-coverable* where $d > 0$ is a natural number if there exists a reachable configuration (q, x) with $x \in n + \mathbb{N}.d$. A *d-coverability run* of a configuration (q, n) is a complete run rooted by a configuration (q, x) with $x \in n + \mathbb{N}.d$. Note that a similar notion of d-reachability was used to show P-completeness for reachability of unary 1BVASS [10].

This section culminates by establishing that every d-coverable configuration (q, n) admits a 'small' d-coverability run, namely of size bounded by $(n+5).(d2^{|B|})^6$. We present most of the proof, which is an orchestration of pigeonhole arguments and safe collapses, as a sequence of lemmas. Let us start with a simple observation about divisibility of subset sums.

▶ **Lemma 2.** *Let $z_1, \ldots, z_d$ be a non-empty sequence of integers. There exists a non-empty finite set $J \subseteq \{1, \ldots, d\}$ such that d divides $\sum_{j \in J} z_j$.*

▶ **Lemma 3.** *Let (q, n) be a configuration and let ρ be a d-coverability run of (q, n) of minimal size. If the size of ρ is larger than $|Q|.2^{|Q|}.d^2$ then ρ contains an increasing cycle.*

Proof. Let ρ be a d-coverability run of (q, n). Suppose: (1) the height of ρ is smaller than $|Q|.d$; and (2) for every height $\ell \geq 1$ the number of nodes in ρ of height ℓ is smaller than $2^{|Q|}.d$. Then it follows that the number of nodes of ρ is bounded by 1 (the root) plus $|Q|.d$ times $2^{|Q|}.d - 1$ (the remaining nodes). Hence the size is bounded by $|Q|.2^{|Q|}.d^2$. It remains to show that if (1) or (2) does not hold then ρ is not minimal or contains an increasing cycle.

In the first case (1) assume that the height of ρ is at least $|Q|.d$. In that case, there exists a node t such that $|t| = |Q|.d$. The nodes on the branch from the root to t are the prefixes of t. It follows that the number of nodes on that branch is equal to $|Q|.d+1$. Notice that if every state of Q occurs at most d times on that branch, then the number of nodes of that branch is bounded by $|Q|.d$ and we get a contradiction. It follows that some state q occurs at least $d+1$ times as a label of a node in that branch. If ρ does not contain any increasing cycle, then all repetitions induce zero or decreasing cycles. Lemma 2 shows that by removing at most d such cycles in that branch we get another d-coverability run of (q, n), smaller than ρ.

In the second case (2) assume that there exists a level $\ell \geq 1$ such that the number of nodes in ρ of height ℓ is at least $2^{|Q|}.d$. It follows that $\ell \geq |Q|$. Notice that these nodes have ancestors in level $\ell - |Q|$. Since the number of elements in level ℓ that have the same ancestors in level $\ell - |Q|$ is bounded by $2^{|Q|}$, it follows that the level $\ell - |Q|$ contains at least d distinct nodes $t_1, \ldots, t_d$ such that, for some words $u_1, \ldots, u_d$ of length $|Q|$, we have $t_i u_i \in \rho$ for every $1 \leq i \leq d$. The pigeon-hole principle shows that we can extract a cycle in the context between $(t_i, t_i u_i)$ for every i. If ρ contains no increasing cycles, then these cycles are zero or decreasing. Lemma 2 shows that by removing at most d such cycles we get another d-coverability run of (q, n), smaller than ρ. ◀

Small d-coverability runs are obtained thanks to the class of *witnesses of d-coverability* defined as follows. A *witness of d-coverability* of a configuration (q, n) is a partial run ψ with the root labeled (q, x), where $x \in n + \mathbb{N}.d$. Every leaf labeled by a configuration (p, m) that is not initial is equipped with a complete run with root label (p, y) with $y \equiv m \mod d$ and containing an increasing cycle. A node of ψ is said to be *modular* when it is an ancestor of such a leaf. We show in the sequel that the existence of d-coverability runs implies the existence of small witnesses of d-coverability. Moreover we provide a way to forge small d-coverability runs from small witnesses of d-coverability.

▶ **Lemma 4.** *Every d-coverable configuration (q, n) has a witness ψ such that:*

- *the subruns of ψ rooted at non-modular nodes have size at most $|Q|.2^{|Q|}.d^2$, and*
- *the complete runs attached to modular leaves have size at most $2|Q|.2^{|Q|}.d^2 + 1$.*

Proof Sketch. We truncate a minimal d-coverability run, bottom-up, at the first increasing cycles. The bounds follow from Lemma 3. ◀

To reduce the number of modular nodes, we introduce an operation on d-coverability witnesses that collapses cycles between modular nodes, as follows. Given a modular leaf ℓ and two ancestors u, v satisfying $u \prec v \preceq \ell$ such that $q_u = q_v$, we transform the witness as follows. First, we introduce the minimal $k \geq 0$ such that $r = kd - n_u + n_v$ is non-negative. Second, we relabel the branch from the root to the leaf ℓ by adding r on nodes s such that $\varepsilon \preceq s \preceq u$ and by adding kd on nodes s such that $v \preceq s \preceq \ell$. It is readily seen that the new labels of u and v are equal. Third, we remove the cycle between u and v by collapsing[4] the nodes $u \prec v$. Notice that after this transformation we get a witness of d-coverability for $(q, n + r)$ where (q, n) was the root label of the original witness of d-coverability. We obtain the following lemma whose proof is along the same lines as that of Lemma 3.

▶ **Lemma 5.** *Let (q, n) be a configuration. By iteratively collapsing cycles, every witness of d-coverability of (q, n) can be simplified into a witness with at most $|Q|.2^{|Q|}.d^2$ modular nodes and where the height of each modular node is smaller than $|Q|.d$.*

▶ **Lemma 6.** *We may relabel modular nodes of any witness of d-coverability of (q, n) in such a way that $n_\ell < n + d + |Q|.d.M$ for every modular leaf ℓ.*

▶ **Theorem 7.** *Every d-coverable configuration (q, n) admits a d-coverability run of size at most $(n + 5).(d2^{|B|})^6$.*

Proof. By applying Lemmas 4, 5, and 6 in succession, we get a witness of d-coverability for (q, n) satisfying the bounds in these lemmas. Assume first that the root node of that witness is not a modular node. In that case the witness of d-coverability of (q, n) is in fact a d-coverability run of (q, n) and by Lemma 4 the size of this run is bounded by $|Q|.2^{|Q|}.d^2$.

Now, suppose that the root node of the witness is a modular node. Let μ be the number of all modular nodes, and ζ the number of all non-modular nodes. By Lemma 5 we get $\mu \leq |Q|.2^{|Q|}.d^2$. We bound ζ as follows. A non-modular node t is called a side node if it is the sibling of a modular node. Observe that non-modular nodes are descendant of side nodes and by Lemma 4 subruns rooted in side nodes have sizes bounded by $|Q|.2^{|Q|}.d^2$. Since the number of side nodes is bounded by μ, we derive that $\zeta \leq \mu.|Q|.2^{|Q|}.d^2$.

To build a d-coverability run from the witness we iterate the following process for each modular leaf ℓ in the witness. As a first step, we transform the attached complete run ρ_ℓ of ℓ in such a way its root label (q_ℓ, x) satisfies $x \in n_\ell + \mathbb{N}d$. Recall that ρ_ℓ contains an increasing cycle π. The above-mentioned transformation simply amounts to iterating this cycle $d.n_\ell$ times. By Lemma 4 the size of ρ_ℓ is bounded by $2.|Q|.2^{|Q|}.d^2 + 1$, which also bounds the size of π. The size of the resulting complete run ρ'_ℓ is bounded by $2.|Q|.2^{|Q|}.d^2 + 1$ (the size of ρ_ℓ) plus $d.n_\ell.2.|Q|.2^{|Q|}.d^2$ (the result of iterating the increasing cycle). It follows that the size of $\rho_{\ell'}$ is bounded by $2.(d.n_\ell + 1).|Q|.2^{|Q|}.d^2 + 1$. By Lemma 6 we have $n_\ell < n + d + |Q|.d.M$, so we get that the size of $\rho_{\ell'}$ is bounded by $2(n + 3).M.|Q|^2.2^{|Q|}.d^4 + 1$.

Let (q_ℓ, m_ℓ) be the configuration of the root of ρ'_ℓ. Observe that $m_\ell \in n_\ell + \mathbb{N}d$. In the second step we add $m_\ell - n_\ell$ to each node on the branch from ℓ to the root of the witness. After this step, the new label of ℓ is equal to the root label of ρ'_ℓ. As a third step, we simply replace the leaf ℓ by the complete run ρ'_ℓ.

We obtain a d-coverability run ρ for (q, n) of size $\mu + \zeta$, plus the sum of the sizes of the complete runs ρ'_ℓ for each modular leaf ℓ. This is bounded by

$$\mu + \mu.|Q|.2^{|Q|}.d^2 + \mu.(2(n + 3).M.|Q|^2.2^{|Q|}.d^4 + 1) \leq (2n + 9).M.|Q|^3.4^{|Q|}.d^6.$$

Since $M.2^{|Q|} \leq 2^{|B|}$, we get that the size of ρ is at most $(n + 5).(d2^{|B|})^6$. ◀

[4] Collapsing two nodes $u \prec v$ consists in replacing the labeled subtree rooted in u by the labeled subtree rooted in v.

4 Reachability

This section is devoted to the proof of the following theorem.

▶ **Theorem 8.** *The reachability problem for 1BVASS is in* PSPACE.

The complexity is w.r.t. the sizes of the input 1BVASS and root configuration, both encoded in binary. Our proof relies on the following small witness property.

▶ **Lemma 9.** *If c is a reachable configuration with a value bounded by $2^{|B|}$ in a given 1BVASS B then there exists a complete run of size at most $2^{60|B|^3}$ with root configuration c.*

Indeed, Lemma 9 implies Theorem 8. First, notice that for configurations with value bigger than $2^{|B|}$ it suffices to solve the problem for value 0 with an auxiliary step in the given BVASS. More precisely, if we ask for reachability of (q, n) we can add two states r, r' such that r is initial, a new transition $(q, r, -n, r')$, and change the question to reachability of $(r', 0)$. To verify if a configuration c with a value bounded by $2^{|B|}$ is reachable, we guess a complete run for c in nondeterministic polynomial space. Since it is impossible to maintain, in polynomial space, all nodes of the run in the memory, we only maintain the ancestors of the currently processed node whose other child was not processed yet. For every processed node v if it is an initial configuration then we go back to the closest ancestor a whose other child was not verified and proceed with that child. In this case we remove the ancestor a from the memory. Otherwise, we guess nondeterministically two children of v such that their triple satisfies some transition in Δ and continue with one of the children. The procedure nondeterministically guesses to proceed with the child whose subtree contains at most half of the leaves in the complete run. It remains to observe that the procedure does not need to remember more than $60|B|^3$ ancestors, otherwise the complete run would require more than $2^{60|B|^3}$ nodes. Notice that it is possible that a node has an exponential number of ancestors, but the procedure does not need to remember them all. The rest of this section is devoted to prove Lemma 9.

Small complete runs are forged from the so-called witnesses of reachability. Formally, the class of *witnesses of reachability* is defined inductively as follows. A witness of reachability w of a configuration c is a partial run with root labeled by c and such that every leaf labeled by (p, m) that is not an initial configuration is a reachable configuration equipped with an *implicit* decreasing simple cycle up to the configuration $(p, 0)$. Implicit means that only the main branch and the dangling nodes of the decreasing cycle are given explicitly. Each dangling configuration is equipped with a witness of reachability. Notice that every configuration admitting a witness is reachable since the leaves of the top most partial run of that witness are labeled by reachable configurations. The *depth* of a witness of reachability is defined as follows. The depth of a complete run is zero, and the depth of a witness of reachability that is not a complete run is one plus the maximal depth of the witnesses of reachability defining the dangling configurations of the decreasing cycles. The depth of a witness of reachability w is denoted by $\mathrm{depth}(w)$. Figure 1 shows an example witness of reachability, suggesting how we turn it into a complete run. To bound the sizes of complete runs obtained from reachability witnesses we introduce the value $\mathrm{maxsize}(w)$ denoting the size of the biggest partial runs occurring in a witness of reachability w. The following lemma shows that $\mathrm{maxsize}(w)$ provides a simple way to bound values occurring in w.

▶ **Lemma 10.** *For every witness of reachability w of a configuration with a value bounded by $2^{|B|}$, the root values of the partial runs used by w are bounded by $2^{|B|}$. Moreover, any value occuring in w is bounded by $2^{2|B|} \cdot \mathrm{maxsize}(w)$.*

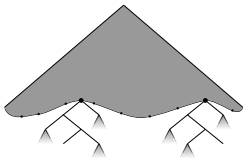

Figure 1 A witness whose topmost partial run has two leaves that are not initial. Decreasing cycles are attached, and the dangling configurations are provided with their subwitnesses.

Proof. The maximal root values can be bounded by observing that, except for the top most partial run of w, partial runs provide root configurations that are dangling configurations of simple decreasing cycles. It follows that these values cannot exceed $|Q|.M \leq 2^{|B|}$. We bound the other values as follows. Observe that the total sum of displacements plus the leave values of a partial run is equal to its root value. It follows that every value of w is bounded by $2^{|B|} + \text{maxsize}(w).M \leq 2^{2|B|}.\,\text{maxsize}(w)$. ◀

▶ **Lemma 11.** *Let w be a witness of reachability of a configuration (q,n) satisfying $n \leq 2^{|B|}$ in a 1BVASS B. There exists a complete run ρ_w with root label (q,n) and size bounded by $(2^{10|B|}.\,\text{maxsize}(w))^{2(\text{depth}(w)+1)}$.*

Proof. We associate to $s, \ell \in \mathbb{N}$ the set $C_{s,\ell}$ of configurations (q,n) such that $n \leq 2^{|B|}$ and such that there exists a witness of reachability w of (q,n) such that $\text{maxsize}(w) \leq s$ and $\text{depth}(w) \leq \ell$. We also define $f(s,\ell) = \max_{c \in C_{s,\ell}}(|\rho_c|)$, where $|\rho_c|$ is the minimal size of a complete run rooted at c. Such a run always exist since c is reachable. Notice that $f(s,0) \leq s$ since a witness of reachability of depth 0 is a complete run.

We provide a bound for $f(s,\ell+1)$ using $f(s,\ell)$. Consider a configuration $c \in C_{s,\ell+1}$. There exists a witness of reachability of c with depth bounded by $\ell+1$ such that $\text{maxsize}(w) \leq s$. Let ρ be the top most partial run of w. Suppose there is a leaf labeled by a non-initial configuration (p,m) that is provided with an implicit simple decreasing cycle up to $(p,0)$. Let us denote by $-d$ the effect of that cycle. As the cycle is simple, it follows that $d \leq |Q|.M \leq 2^{|B|}$. The dangling configurations $c_1, \ldots, c_k$ of that cycle are given by witnesses of reachability $w_1, \ldots, w_k$ such that $\text{depth}(w_j) \leq \ell$ and $\text{maxsize}(w_j) \leq s$. It follows that $c_1, \ldots, c_k \in C_{\ell,s}$. By induction, the dangling configurations $c_1, \ldots, c_k$ can be replaced by complete runs of size bounded by $f(s,\ell)$. Since the cycle is simple, $k \leq |Q|$. After these replacements we obtain an (explicit) simple cycle π of size at most $|Q| + |Q|.f(s,\ell)$. Moreover, since (p,m) is reachable, it is d-coverable. Theorem 7 shows that there exists a d-coverability run for (p,m) of size at most $(m+5).(d.2^{|B|})^6$. Lemma 10 shows that $m \leq 2^{2|B|}.\,\text{maxsize}(w)$. Since $5 \leq 2^{3|B|}.\,\text{maxsize}(w)$ we get $m + 5 \leq 2^{4|B|}.\,\text{maxsize}(w)$. We derive that there is a d-coverability run ρ of (p,m) of size bounded by $\lambda = 2^{16|B|}.\,\text{maxsize}(w) \leq 2^{16|B|}.s$.

There exists $k \in \mathbb{N}$ such that $(p, m + k.d)$ is the root configuration of ρ. In order to obtain a complete run with root configuration (p,m), we just have to consider $\pi^k\rho$. Notice that we can never reach a value below zero since π is a decreasing cycle up to $(p,0)$. As $m + k.d \leq \lambda.M$, it follows that $k \leq \lambda.M$. We have proved that there exists a complete run with root configuration (p,m) and of size bounded by $\lambda.M.(|Q| + |Q|f(s,\ell)) + \lambda \leq 2^{19|B|}.s.f(s,\ell)$. Finally, by replacing every non-terminal leaf by a complete run as performed previously, we get a complete run with root configuration c and size bounded by $2^{19|B|}.s^2.f(s,\ell)$. We have proved that $f(s,\ell+1)$ is bounded by that value. An immediate induction shows that

$$f(s,\ell) \leq (2^{19|B|}.s^2)^\ell.f(s,0) \leq (2^{19|B|}.s^2)^\ell.s \leq (2^{10|B|}.s)^{2(\ell+1)}.$$ ◀

By Lemma 11, to prove Lemma 9 it suffices to find a witness w such that maxsize(w) is bounded exponentially and depth(w) is bounded polynomially in the size of the given 1BVASS. Before we prove that we introduce some notation and two auxiliary lemmas.

Before the next lemma we introduce an operation that intuitively moves increasing cycles from left branches to right branches. By applying that operation as many times as possible, we obtain a so-called *saturated* partial run. Formally, a partial run ρ is said to be *reducible* in a node s if there exists an increasing cycle π between (sL, t) for some node $t \succeq sL$, and a minimal node v in π such that q_v is the state of some descendant of sR. A partial run that is not reducible is said to be *saturated*.

▶ **Lemma 12.** *For every partial run ρ there exists a saturated partial run ρ' with the same root configuration, the same number of nodes, and the same mutiset of leaf configurations.*

Proof. Assume that a partial run ρ is reducible on a node s. Let us denote by π an increasing d-cycle between (sL, t) in ρ for some node $t \succeq sL$ and a minimal node v in π such that $q_v = q_{v'}$ for some v', a descendant of sR. Let $\pi = \pi_1\pi_2$ be such that π_1 is the fragment of π with the source node v. We define $\pi' = \pi_2\pi'_1$, where π'_1 is obtained from π_1 by decreasing all values on the main branch by n_v. Since v is minimal, notice that π' is an increasing d-cycle from $(q_v, 0)$ up to (q_v, d) such that the multiset of configurations of nodes not on the main branch in π and π' are equal. By removing from ρ the increasing cycle π, and inserting the increasing cycle π' into v', we get a partial run ρ' such that the root configuration, the number of nodes, and the mutiset of leaf configurations remain the same as in ρ. Notice that by removing π we decrease the value of s by d, but by inserting π' its value is increased by d, therefore, these operations do not cause a drop below 0. Since this transformation can be performed only a finite number of times, at some point, we get a saturated run satisfying the lemma. ◀

▶ **Lemma 13.** *The number of nodes of a saturated run with root labeled (q, n) with $n \leq 2^{|B|}$ that does not contain any decreasing or zero cycles is bounded by $2^{5|B|^2}$.*

To prove Lemma 9 we will decompose partial runs that contain a decreasing cycle. Since such cycles are not necessarily simple, we provide the following lemma.

▶ **Lemma 14.** *For every decreasing cycle π there exists a state p that labels a main node of π and a simple decreasing cycle π' up to $(p, 0)$ such that the set of dangling configurations of π' is included in the set of dangling configurations of π.*

Proof of Lemma 9. We consider a reachable configuration c with a value bounded by $2^{|B|}$. Obviously c admits a witness of reachability because every complete run is a witness. By Lemma 11 we need to find a witness w of c such that maxsize(w), depth(w) have proper bounds. To do so, we associate to every witness w the sequence of natural numbers $s(w) = (s_j)_{j\geq 1}$, where s_j is the number of partial runs of size j in w, called the *rank* of w. These sequences are ordered colexicographically by the total order $\sqsubseteq$ defined by $(s_j)_{j\geq 1} \sqsubseteq (s'_j)_{j\geq 1}$ if the two sequences are equal or there exists $j \geq 1$ such that $s_j < s'_j$ and $s_i = s'_i$ for every $i > j$. Notice that sequences $s(w)$ have finite support, i.e., $\{j \mid s_j \neq 0\}$ is finite. When restricted to sequences with finite support the order $\sqsubseteq$ is well-founded, i.e., there are no infinite decreasing sequences. We consider for the remainder of the proof a reachability witness w with a minimal rank (for $\sqsubseteq$).

Suppose w has depth $\ell > |Q|$. Then there exists a sequence $\pi_1 \dots \pi_\ell$ of implicit decreasing simple cycles such that π_{i+1} is a cycle equipped to the partial run of a dangling node in π_i. Then there exist $i < j$ such that π_i and π_j have the same state in the root. We replace π_i with

π_j. In particular we remove all cycles $\pi_i \dots \pi_{j-1}$ and all partial runs that were associated to them. The resulting witness has a smaller rank which contradicts our minimality assumption on w.

Now, suppose that w contains partial runs of size bigger than $2^{5|B|^2}$. Let σ be a partial run having the maximal size and such that its depth is maximal (with respect to others of the same size). Notice that all partial runs of larger depth are smaller than σ. Using Lemma 12 we turn σ into a saturated run without changing the multiset of configurations of the leaves. Lemma 10 shows that the run σ has root value at most $2^{|B|}$ and thus by Lemma 13 there exists a decreasing or a zero cycle π in σ. If π is a zero cycle then we just remove it obtaining a smaller rank which contradicts our minimality assumption on w. Otherwise, π is a decreasing cycle. By Lemma 14 there exists a state p that labels a main node u of π and a simple decreasing cycle π' up to $(p, 0)$ such that the set of dangling configurations of π' is included in the set of dangling configurations of π. Since π is a cycle, we can assume w.l.o.g. that u is distinct from the source of π. Let v be the original node of u in σ. By assuption on u, notice that v is an internal node of σ. We define σ' as the partial run truncated at v and equip it with the simple decreasing cycle π'. Since v is an internal node, σ' is smaller from σ. The configurations of dangling nodes in π' are also configurations of dangling nodes in π, which come from σ. We use the partial subruns of σ as partial runs for dangling nodes in π'. These subruns come with implicit simple decreasing cycles and additional partial runs of smaller depth. Notice that, possibly, we have introduced double copies of partial runs of smaller depth. Let us show that the resulting witness w' has a smaller rank which will contradict our minimality assumption on w. Recall that all partial runs of bigger depth are smaller than σ. Since we have decreased the size of σ, and all introduced partial runs are of smaller size than σ it follows that $s(w') \sqsubset s(w)$.

We have proved that $\mathrm{depth}(w) \leq |Q|$ and $\mathrm{maxsize}(w) \leq 2^{5|B|^2}$. From Lemma 11, we get a complete run rooted by (q, n) with size bounded by $2^{60|B|^3}$. ◀

5 Hardness

We prove that the reachability problem for 1BVASS is PSpace-hard. Intuitively, one would like to encode runs of an alternating PTime Turing machine: the tape configuration is maintained as the binary representation of the counter value, and alternation is represented by the branching structure of the run. At first sight, binary rules of BVASS are not compatible with any sort of alternation: the value ℓ of a node may come from two arbitrary values ℓ_1, ℓ_2 from children with the sole restriction that $\ell = \ell_1 + \ell_2$. It thus seems pointless to pretend to replicate information encoded in ℓ into both ℓ_1 and ℓ_2. However, we show that one can enforce —in a highly restricted setting— that transitions behave in a regular way, where $\ell = 2 \cdot \ell_1 = 2 \cdot \ell_2$. Using this, child nodes can recover information from the parent node: the i-th bit of the child has a 1 iff the $i+1$-th bit of the parent has a 1. In this way information can be 'copied' into different branches, and we can benefit from some form of alternation.

▶ **Theorem 15.** *The reachability problem for 1BVASS is* PSpace*-hard.*

Proof Idea. The proof goes by reduction from the PSpace-complete problem of validity for Quantified Boolean Formulas. Given a QBF sentence, such as

$$\varphi = \forall P_1\ \exists P_2\ \forall P_3\ (P_1 \vee \neg P_2 \vee P_3) \wedge (\neg P_1 \vee P_2),$$

we define a polynomial size 1BVASS $\mathcal{B}$, in such a way that the configuration $(q_1, 0)$ is reachable if, and only if, φ is valid. Transitions in $\mathcal{B}$ enforce that any complete run for $(q_1, 0)$ encodes a 'certificate' of the validity of φ. In particular, this certificate contains

- nondeterministic choices for the valuation of existentially quantified variables such as P_2;
- one branch for each of the exponentially-many valuations for universally quantified variables, such as P_1 and P_3;
- for each branch encoding a valuation, a sub-branch for each disjunctive clause, certifying that the clause is true under that valuation.

Levels. For this reduction, it is natural to think of runs as proceeding top-down instead of bottom-up as done hitherto. That is, we start with a configuration $(q_1, 0)$ at the root, and we build valuations going downward until eventually finding an initial configuration on every branch. From this perspective, transitions of $\mathcal{B}$ 'increment' a value $c > 0$ before 'splitting' the value into children with states q', q'' with a transition of the form $(q', q'', -c, q)$. The behaviour of $\mathcal{B}$ ensure that any complete run for $(q_1, 0)$ can be divided into 'levels', so that the i-th level of the tree contains encodings for the choices of valuations for the first i variables of φ. For the purpose of this sketch, the level i is the set of all nodes of height $2 \cdot i$ in the run (*e.g.*, in the run of Figure 2, nodes at level i are those labelled q_{i+1}).

Valuation encoding. Variable valuations are encoded in the counter value by exploiting its compact binary representation, which throughout the run remains always a bitstring of quadratic length in the size of the sentence φ. The counter value bitstring can be split into equal length *segments*, one for each variable, so that the i-th segment is a $2m$ substring encoding the valuation of the i-th variable of φ, where m is the number of universally quantified variables plus the $\log_2$ of the number of conjuncts of φ —in our running example, $m = 3$. The encoding of a valuation for a variable will evolve along the run, for example the encoding for P_1 being true at nodes at different levels may differ. This is because branchings change the counter value and thus its binary representation. For a node at level j, the encoding for a *true* ($\top$) valuation of a variable P_i with $i \leq j$ is through a bitstring $0^{u(j)-1}10^{2m-u(j)}$ at the i-th segment, where $u(j)$ is the number of universally quantified variables P_i with $i \leq j$ in the input sentence —in our example, $u = \{(1,1),(2,1),(3,2)\}$. Similarly, the way to encode a *false* ($\bot$) valuation is through the bitstring $0^{m+u(j)-1}10^{m-u(j)}$. For φ as above, where $m = 3$, the valuation $\{(P_1, \top), (P_2, \bot), (P_3, \bot)\}$ at level 3 (*i.e.*, $j = 3$, $u(j) = 2$) is represented by the bitstring $z = (010|000)(000|010)(000|010)$ (parentheses and pipes are only to improve readability). Let us call the '(i, j)-bit' the j-th most significant bit of the i-th most significant segment in the bitstring, and let $c_{i,j} \in \mathbb{N}$ be the number whose sole (i, j)-bit is 1 in its binary representation (*e.g.*, for z as above, $z = c_{1,2} + c_{2,5} + c_{3,5}$).

As discussed before, for this reduction to work we need that already defined valuation are somehow 'replicated' in all the subtrees, that is, when a configuration branches, information on the valuations is preserved in both children configurations and remains uncorrupted. For this, we enforce that, for every internal node t inside a complete run for $(q_1, 0)$ of $\mathcal{B}$, either:

1. t has a right child with the initial configuration $(q_I, 0)$, or, otherwise,
2. both children of t have the same value, that is, $n_t = 2 \cdot n_{tL} = 2 \cdot n_{tR}$.

Assuming such a property (as verified by the run of Fig. 2), information can be 'spread' along branches of a run: at any node t of type (2) the i-th least significant bit of n_t is 1 iff the $(i-1)$-th least significant bit of n_{tL} and n_{tR} are 1. We use transitions of type (1) to generate a new valuation for the i-th variable, and transitions of type (2) to split the current valuation into two branches. For example, a configuration containing a *true* segment with bitstring $0^{i-1}10^{2m-i}$ is split into two children whose segment value is now 0^i10^{2m-i-1}, which still codes a *true* value for the next level $i + 1$. The choice of m is such that it corresponds

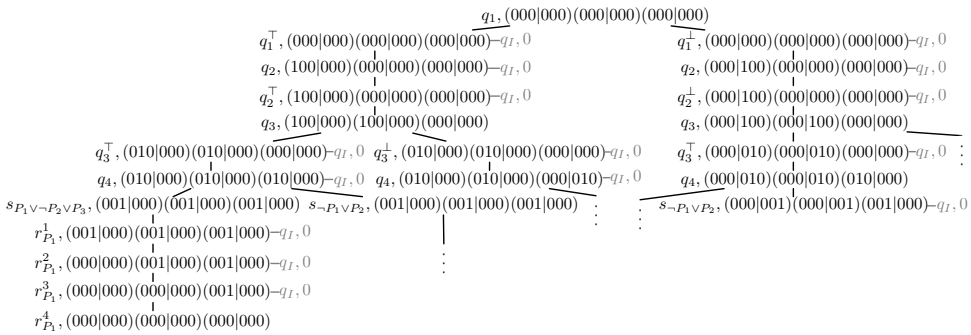

Figure 2 Clipping of a complete run for $(q_1, 0)$. Values are represented by 3 segments of 6 bits.

to the maximum number of transitions of type (2) in any root-to-leaf branch of the run. In other words, m is the maximum distance that a 1-bit can 'travel' along the run.

Here we only show how to build $\mathcal{B}$ for our running example φ. We use the state space $Q = \{q_j, q_i^\top, q_i^\bot, s_\psi, r_A^j, q_I \mid 1 \leq i \leq 3, 1 \leq j \leq 4, \psi : \text{clause}, A : \text{atom}\}$. For each universally quantified variable P_i (*i.e.*, for $i = 1, 3$) we include a transition $(q_i^\top, q_i^\bot, 0, q_i)$, which splits the run into a subtree where P_i is true ($q_i^\top$) and another where it is false ($q_i^\bot$). On the other hand, for each existentially quantified variable P_i (*i.e.*, for $i = 2$), we include non-deterministic transitions $(q_i^\top, q_I, 0, q_i)$ and $(q_i^\bot, q_I, 0, q_i)$, which choose one valuation for P_i. Each state $q_i^\top$ and $q_i^\bot$ has a transition incrementing the corresponding bit in the encoding: $(q_{i+1}, q_I, -c, q_i^\top)$ for c having its $(i, u(i))$-bit in 1, and 0's elsewhere; and $(q_{i+1}, q_I, -\bar{c}, q_i^\bot)$ for $\bar{c}$ having its $(i, m + u(i))$-bit in 1, and 0's elsewhere. Finally, $\mathcal{B}$ checks for the satisfaction of both clauses by splitting the computation through the transition $(s_{P_1 \vee \neg P_2 \vee P_3}, s_{\neg P_1 \vee P_2}, 0, q_4)$. For each clause, $\mathcal{B}$ chooses the atom which will witness its satisfaction, with transitions $(r_A^1, q_I, 0, s_\psi)$ for every disjunctive clause ψ of φ and atom A of ψ (*e.g.*, for $\psi = \neg P_1 \vee P_2$ and $A = \neg P_1$). Finally, the job of r_A^1 is to decrement the (i, m)-bit for verifying that P_i holds true, or the $(i, 2m)$-bit otherwise. However, this choice between the (i, m)- and the $(i, 2m)$-bit must be consistent with the choice of the atom A (*e.g.*, if $A = \neg P_2$ then we must verify that P_2 is false and thus we shouldn't allow the decrement of the (i, m)-bit). Concretely, we include a transition $(r_A^{i+1}, q_I, c_{i,m}, r_A^i)$ if and only if A is not $\neg P_i$; and we include $(r_A^{i+1}, q_I, c_{i,2m}, r_A^i)$ iff A is not P_i. Finally, the initial states I is defined as all states r_A^4 for an atom A, as well as q_I.

Figure 2 contains a depiction of a complete run witnessing the validity of φ. ◀

6 Conclusion

An interesting next question is the complexity of the reachability problem for two-dimensional BVASS, which we conjecture decidable. One approach to establishing decidability could be by generalising the classical algorithm of Hopcroft and Pansiot for two-dimensional VASS [12]. To determine the precise complexity, investigating branching extensions of the flatness notion (cf. [18, 3]) and the cutting technique (cf. [9]) seem like promising directions.

References

1 Sergio Abriola, Diego Figueira, and Santiago Figueira. Logics of repeating values on data trees and branching counter systems. In *FoSSACS*, volume 10203 of *LNCS*. Springer, 2017, to appear.

2 Katalin Bimbó. The decidability of the intensional fragment of classical linear logic. *Theor. Comput. Sci.*, 597:1–17, 2015. doi:10.1016/j.tcs.2015.06.019.

3 Michael Blondin, Alain Finkel, Stefan Göller, Christoph Haase, and Pierre McKenzie. Reachability in two-dimensional vector addition systems with states is PSPACE-complete. In *LICS*, pages 32–43. IEEE Computer Society, 2015. doi:10.1109/LICS.2015.14.

4 Mikołaj Bojańczyk. Automata column. *SIGLOG News*, 1(2):3–12, 2014. doi:10.1145/2677161.2677163.

5 Ahmed Bouajjani and Michael Emmi. Analysis of recursively parallel programs. *ACM Trans. Program. Lang. Syst.*, 35(3):10:1–10:49, 2013. doi:10.1145/2518188.

6 Lorenzo Clemente, Sławomir Lasota, Ranko Lazić, and Filip Mazowiecki. Timed pushdown automata and branching vector addition systems. Submitted, 2017.

7 Conrad Cotton-Barratt, Andrzej S. Murawski, and C.-H. Luke Ong. ML and extended BVASS. In *ESOP*, 2017. To appear.

8 Philippe de Groote, Bruno Guillaume, and Sylvain Salvati. Vector addition tree automata. In *LICS*, pages 64–73. IEEE Computer Society, 2004. doi:10.1109/LICS.2004.1319601.

9 Matthias Englert, Ranko Lazić, and Patrick Totzke. Reachability in two-dimensional unary vector addition systems with states is NL-complete. In *LICS*, pages 477–484. ACM, 2016. doi:10.1145/2933575.2933577.

10 Stefan Göller, Christoph Haase, Ranko Lazić, and Patrick Totzke. A polynomial-time algorithm for reachability in branching VASS in dimension one. In *ICALP*, volume 55 of *LIPIcs*, pages 105:1–105:13. Schloss Dagstuhl, 2016. doi:10.4230/LIPIcs.ICALP.2016.105.

11 Christoph Haase, Stephan Kreutzer, Joël Ouaknine, and James Worrell. Reachability in succinct and parametric one-counter automata. In *CONCUR*, volume 5710 of *LNCS*, pages 369–383. Springer, 2009. doi:10.1007/978-3-642-04081-8_25.

12 John E. Hopcroft and Jean-Jacques Pansiot. On the reachability problem for 5-dimensional vector addition systems. *Theor. Comput. Sci.*, 8:135–159, 1979. doi:10.1016/0304-3975(79)90041-0.

13 Florent Jacquemard, Luc Segoufin, and Jerémie Dimino. FO2($<$, $+1$, $\sim$) on data trees, data tree automata and branching vector addition systems. *Log. Meth. Comput. Sci.*, 12(2), 2016. doi:10.2168/LMCS-12(2:3)2016.

14 S. Rao Kosaraju. Decidability of reachability in vector addition systems (preliminary version). In *STOC*, pages 267–281. ACM, 1982. doi:10.1145/800070.802201.

15 Pascal Lafourcade, Denis Lugiez, and Ralf Treinen. Intruder deduction for AC-like equational theories with homomorphisms. Research Report LSV-04-16, ENS de Cachan, 2004.

16 Ranko Lazić and Sylvain Schmitz. Nonelementary complexities for branching VASS, MELL, and extensions. *ACM Trans. Comput. Log.*, 16(3):20:1–20:30, 2015. doi:10.1145/2733375.

17 Jérôme Leroux and Sylvain Schmitz. Ideal decompositions for vector addition systems (invited talk). In *STACS*, volume 47 of *LIPIcs*, pages 1:1–1:13. Schloss Dagstuhl, 2016. doi:10.4230/LIPIcs.STACS.2016.1.

18 Jérôme Leroux and Grégoire Sutre. On flatness for 2-dimensional vector addition systems with states. In *CONCUR*, volume 3170 of *LNCS*, pages 402–416. Springer, 2004. doi:10.1007/978-3-540-28644-8_26.

19 Patrick Lincoln, John C. Mitchell, Andre Scedrov, and Natarajan Shankar. Decision problems for propositional linear logic. *Ann. Pure Appl. Logic*, 56(1-3):239–311, 1992. doi:10.1016/0168-0072(92)90075-B.

20 Owen Rambow. Multiset-valued linear index grammars: Imposing dominance constraints on derivations. In *ACL*, pages 263–270. Morgan Kaufmann Publishers / ACL, 1994. URL: http://aclweb.org/anthology/P/P94/P94-1036.pdf.

21 Wolfgang Reisig. *Understanding Petri Nets – Modeling Techniques, Analysis Methods, Case Studies*. Springer, 2013. doi:10.1007/978-3-642-33278-4.

22 Sylvain Schmitz. On the computational complexity of dominance links in grammatical formalisms. In *ACL*, pages 514–524. The Association for Computer Linguistics, 2010. URL: http://www.aclweb.org/anthology/P10-1053.

23 Sylvain Schmitz. The complexity of reachability in vector addition systems. *SIGLOG News*, 3(1):4–21, 2016. doi:10.1145/2893582.2893585.

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

25 Kumar Neeraj Verma and Jean Goubault-Larrecq. Karp-Miller trees for a branching extension of VASS. *Discr. Math. & Theor. Comput. Sci.*, 7(1):217–230, 2005. URL: http://www.dmtcs.org/volumes/abstracts/dm070113.abs.html.