Reachability for Bounded Branching VASS

Filip Mazowiecki
LaBRI, Université de Bordeaux, France
filip.mazowiecki@u-bordeaux.fr

Michał Pilipczuk
University of Warsaw, Poland
michal.pilipczuk@mimuw.edu.pl

──── **Abstract** ────

In this paper we consider the reachability problem for bounded branching VASS. Bounded VASS are a variant of the classic VASS model where all values in all configurations are upper bounded by a fixed natural number, encoded in binary in the input. This model gained a lot of attention in 2012 when Haase et al. showed its connections with timed automata. Later in 2013 Fearnley and Jurdziński proved that the reachability problem in this model is PSPACE-complete even in dimension 1. Here, we investigate the complexity of the reachability problem when the model is extended with branching transitions, and we prove that the problem is EXPTIME-complete when the dimension is 2 or larger.

1 Introduction

Vector addition systems with states (VASS or d-VASS when the dimension is d), also known as Petri nets, are a long established model for concurrent systems. The branching generalisation of VASS (BrVASS or d-BrVASS[1]) is among the most popular extensions due to its multiple connections with database theory [4], recursively parallel programs [5], timed pushdown systems [6], and many others. A configuration of a d-VASS (or a d-BrVASS) is a pair consisting of a state and a vector over $\mathbb{N}^d$. In d-VASS, the transitions update the configurations by changing the state and updating the configuration vector using vectors over $\mathbb{Z}^d$. In d-BrVASS, in addition there can be branching transitions, which create two independent copies of the system splitting the configuration vector among them.

---

[1] The standard abbreviation for this model appearing in the literature is BVASS. Since in this work we work with both bounded and branching models of VASS, we prefer to disambiguate these variants by using prefixes Bo and Br, respectively.

■ **Table 1** Complexities in dimension 1. In the last entry only a PSPACE lower bound is known.

	unary	binary	binary with a bound
1-VASS	NL-complete [20]	NP-complete [14]	PSPACE-complete [10]
1-BrVASS	PTIME-complete [13]	PSPACE-complete [11]	in EXPTIME

The central decision problem for both VASS and BrVASS is the *reachability problem*, i.e., whether there exists a computation between given input configurations. For VASS we know that reachability is decidable [18], moreover, recently the upper bound on the complexity was improved to Ackermann [17]. Whether reachability is decidable for BrVASS is one of the most intriguing open problems in formal verification, highlighted for example in the survey of Bojańczyk [3].

To understand better the source of hardness, the reachability problem was studied when the dimension of the vectors is fixed to a constant. Then it matters how the vectors of the models are encoded and one must consider two cases: unary and binary encodings. Particularly, when the dimension is 1 it is known that 1-VASS with vectors encoded in unary is NL-complete [20], while for the binary encoding it becomes NP-complete [14]. For 1-BrVASS the complexities are also known, namely for the unary encoding the reachability is PTIME-complete [13], while for the binary encoding the problem is PSPACE-complete [11]. We present these results in Table 1. For these models the complexities are easy to remember as in both cases the complexity of the branching variant adds alternation to the complexity, as expected. However, branching is not alternation and this intuition is possibly misleading. Already in dimension 2 reachability for 2-VASS is known to be NL-complete [9] and PSPACE-complete [2] for unary and binary encodings, respectively; while for 2-BrVASS it is not even known whether reachability is decidable.

In this paper we are interested in the variants of VASS and BrVASS with vectors encoded in binary, where all values in all configurations are upper bounded by some natural number, given on input. We shall denote these classes BoVASS and BoBrVASS, respectively. The model BoVASS was popularised due to its connections with timed automata [15], where it was shown that reachability for timed automata is interreducible in logarithmic space with reachability for BoVASS. At the time it was known that for timed automata this problem is PSPACE-complete [1], even when there are only 3 clocks available [7], and NL-complete for the case with 1 clock; leaving open the complexity for timed automata with 2 clocks. The problem for two clocks was proved later to be PSPACE-complete [10], where the main technical contribution was proving that reachability for 1-BoVASS is PSPACE-complete. Then the final result for timed automata with two clocks followed from the reductions presented in [15][2]. Regarding the branching extension a more powerful version of the model 1-BoBrVASS was studied in [6], where connections between BrVASS and pushdown timed automata were explored.

The reachability for 1-BoVASS has become one of the standard problems used in reductions proving PSPACE-hardness of various decision problems. Apart from the already mentioned application to prove PSPACE-hardness of reachability for two clock automata, 1-BoVASS was used e.g., to prove: PSPACE-completeness of reachability for two dimensional VASS [2]; and PSPACE-completeness of regular separability of one counter automata [8].

---

[2] Both papers [15, 10] write *bounded counter automata* instead of BoVASS. We write BoVASS to have a uniform terminology for all models in this paper.

Our contribution. In this paper we study the complexity of the reachability problem for BoBrVASS. This problem can be easily proved to be in EXPTIME, but only PSPACE-hardness is known; e.g., due to the mentioned PSPACE-hardness of 1-BoVASS. Our main contribution is that the reachability problem for 2-BoBrVASS is EXPTIME-complete and we leave the complexity of 1-BoBrVASS as an open problem (see Table 1). We believe that if there exists an EXPTIME lower bound for reachability for 1-BoBrVASS, then this would give a new, interesting starting problem for EXPTIME-hardness reductions. As an example application, we prove that reachability for 1-BoBrVASS is reducible in polynomial time to reachability in 2-BrVASS, which slightly improves the state of art, as currently only a PSPACE lower bound is known for 2-BrVASS [2].

Organisation. After presenting the main definitions and problems in Section 2 and Section 3, we prove the EXPTIME-completeness of reachability for 2-BoBrVASS in Section 4. Then in Section 5 we give some indications that using our techniques it would be hard to lift the EXPTIME-hardness result to 1-BoBrVASS. We conclude in Section 6 providing a polynomial time reduction of reachability for 1-BoBrVASS to reachability for 2-BrVASS.

2 Preliminaries

VASS model. A *Vector Addition System with States* (VASS) in dimension d (in short d-VASS) is $\mathcal{V} = (Q, \Delta)$, where: Q is a finite set of states and $\Delta \subseteq Q \times \mathbb{Z}^d \times Q$ is a finite set of transitions. The size of $\mathcal{V}$ is $|Q| + |\Delta| \cdot r$, where r is the maximal representation size of vectors in Δ (written in binary).

A configuration is a pair $(q, \boldsymbol{n})$, denoted $q(\boldsymbol{n})$, where $\boldsymbol{n} \in \mathbb{N}^d$. Whenever $d = 1$, we identify vectors in dimension 1 with natural numbers and in particular we denote configurations by $q(n)$ for $n \in \mathbb{N}$. We will often refer to values of vector components as counter values and when convenient we will denote them $c_1, \ldots, c_d$. We write that q is the state of the configuration and $\boldsymbol{n}$ is the vector of the configuration. We write $p(\boldsymbol{n}) \xrightarrow{t} q(\boldsymbol{m})$ if $t = (p, \boldsymbol{z}, q) \in \Delta$ and $\boldsymbol{n} + \boldsymbol{z} = \boldsymbol{m}$. More generally, a *run* is a sequence of configurations

$$q_0(\boldsymbol{n}_0) \xrightarrow{t_0} q_1(\boldsymbol{n}_1) \xrightarrow{t_1} \ldots \xrightarrow{t_k} q_{k+1}(\boldsymbol{n}_{k+1}), \tag{1}$$

where $t_i = (q_i, \boldsymbol{z}_i, q_{i+1}) \in \Delta$ and $\boldsymbol{n}_i + \boldsymbol{z}_i = \boldsymbol{n}_{i+1}$ for all $0 \leqslant i \leqslant k$. Notice that configurations have vectors over natural numbers and transitions have vectors over integers. We write $p(\boldsymbol{n}) \xrightarrow{t_0 \ldots t_k} q(\boldsymbol{m})$ or $p(\boldsymbol{n}) \rightarrow^* q(\boldsymbol{m})$ if there exists a run like (1) such that $p(\boldsymbol{n}) = q_0(\boldsymbol{n}_0)$ and $q(\boldsymbol{m}) = q_{k+1}(\boldsymbol{n}_{k+1})$. We write $p(\boldsymbol{n}) \not\rightarrow^* q(\boldsymbol{m})$ if no such run exists.

A bounded d-VASS (d-BoVASS) is a d-VASS equipped with a bound B such that numbers in all configurations cannot exceed B. Formally, a BoVASS $\mathcal{V}$ is a tuple (Q, Δ, B) such that (Q, Δ) is a VASS and $B \in \mathbb{N}$. We will assume that B is given in binary. A run is defined in the same way like for d-VASS but in (1) we additionally assume that $\boldsymbol{n}_i \in \{0, \ldots, B\}^d$ for all $0 \leqslant i \leqslant k$. That is, all components in all vectors are bounded by B. Whenever speaking about bounded models of vector addition systems, we add the length of the bit representation of B to the size of the system.

In the d-VASS model one can simulate lower bound inequality tests as follows. We write that $t = (p, n, i, q) \in Q \times \mathbb{N} \times \{1, \ldots, d\} \times Q$ is an inequality test, and we allow to write $p(\boldsymbol{n}) \xrightarrow{t} q(\boldsymbol{m})$ only if $\boldsymbol{n} = \boldsymbol{m}$ and $c_i \geqslant n$ in both configurations. We use the notation $t = (p, q)_{c_i \geqslant n}$ for simplicity. In a d-VASS $\mathcal{V}$ such a test can be simulated with one extra state r and two transitions: $t_1 = (p, -\boldsymbol{v}_i, r)$ and $t_2 = (r, \boldsymbol{v}_i, q)$, where $\boldsymbol{v}_i[j] = n$ for $i = j$ and $\boldsymbol{v}_i[j] = 0$ otherwise. Then it is easy to see that $p(\boldsymbol{n}) \xrightarrow{t} q(\boldsymbol{m})$ iff $p(\boldsymbol{n}) \xrightarrow{t_1 t_2} q(\boldsymbol{m})$ for all pairs

of configurations $p(\boldsymbol{n})$ and $q(\boldsymbol{m})$. If $\mathcal{V}$ is additionally a BoVASS with a bound $B \geqslant n$, then we can also implement analogously defined upper bound inequality tests $(p,q)_{c_i \leqslant n}$. Indeed, it suffices to add one extra state r and two transitions: $t_1 = (p, \boldsymbol{w}_i, r)$ and $t_2 = (r, -\boldsymbol{w}_i, q)$, where $\boldsymbol{w}_i[j] = B - n$ for $i = j$ and $\boldsymbol{w}_i[j] = 0$ otherwise. Finally, equality tests $(p,q)_{c_i = n}$ can be encoded by a lower bound test followed by an upper bound test. Slightly abusing the notation, we will assume that such inequality and equality tests are allowed in the transition set of BoVASS models, as they can be simulated as above at the cost of increasing the size of the system by a constant multiplicative factor.

▶ **Definition 1.** *Fix some function* $f \colon \{0, \ldots, M\}^d \to \{0, \ldots, N\}^d$ *for some* $M, N \in \mathbb{N}$. *We say that a d-BoVASS computes f if there exist states $p, q \in Q$ such that for all* $\boldsymbol{n} \in \{0, \ldots, M\}^d$ *and* $\boldsymbol{m} \in \mathbb{N}^d$, *we have* $p(\boldsymbol{n}) \to^* q(\boldsymbol{m})$ *if and only if* $\boldsymbol{m} = f(\boldsymbol{n})$. *The bound B of the BoVASS does not have to be equal to M or N.*

Note that in the above definition, we do not specify how the BoVASS computing f behaves starting in a configuration outside of the domain of f. However, it is easy to restrict the system to block on such configurations, e.g. by adding tests $c_i \leqslant M$ in the beginning.

▶ **Example 2.** Consider the copying function $f \colon \{0, \ldots, M\}^2 \to \{0, \ldots, M\}^2$ defined by $f(n, m) = (n, n)$ for all $0 \leqslant n, m \leqslant M$. We define a 2-BoVASS $\mathcal{V} = (Q, \Delta, B)$ computing f as follows. The bound is $B = M(M + 2)$; the set of states is $Q = \{p, r_1, r_2, q\}$; the transitions are depicted in Figure 1. To prove that this is the right function it suffices to observe that:
- $p(n, m) \to^* r_1(k, l)$ iff $k = n$ and $l = 0$;
- $r_1(n, 0) \to^* r_2(k, l)$ iff $k = 0$ and $l = (M + 2)n$;
- and $r_2(0, (M + 2)n) \to^* q(k, l)$ iff $k = l = n$.
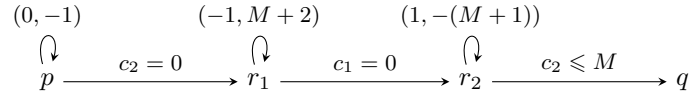
Figure 1 The BoVASS computing $f(n, m) = (n, n)$.

Note that in Example 2, the size of $\mathcal{V}$ is polynomial in the representation of M. A 2-BoVASS of exponential size is trivial to define. In fact, for every function f there exists a BoVASS computing f of size exponential in the representation of the sizes of domain and codomain: one can simply use an equality test for every element of the domain to specify the corresponding value.

Branching VASS model. A *Branching Vector Addition System with States* (BrVASS) in dimension d (in short d-BrVASS) is $\mathcal{B} = (Q, \Delta_1, \Delta_2, q_0)$, where: Q is a finite set of states, $\Delta_1 \subseteq Q \times \mathbb{Z}^d \times Q$ is the set of unary transitions, $\Delta_2 \subseteq Q^3$ is the set of binary transitions, and $q_0 \in Q$ is an initial state. The notions regarding configurations are the same as for VASS. A *run* of a BrVASS is a tree labelled with configurations such that internal nodes have either one or two children, and the following conditions hold.
- For every internal node with one child, let $p(\boldsymbol{n})$ be its label and let $q(\boldsymbol{m})$ be the label of its child. Then there exists $(p, \boldsymbol{z}, q) \in \Delta_1$ such that $\boldsymbol{n} + \boldsymbol{z} = \boldsymbol{m}$;
- For every internal node with two children, let $p(\boldsymbol{n})$ be its label and let $q_1(\boldsymbol{m}_1)$, $q_2(\boldsymbol{m}_2)$ be the labels of its children. Then $(p, q_1, q_2) \in \Delta_2$ and $\boldsymbol{n} = \boldsymbol{m}_1 + \boldsymbol{m}_2$.
- The leaves are labelled $q_0(\boldsymbol{0})$, where $\boldsymbol{0}$ is the zero vector.
Notice that a BrVASS with $\Delta_2 = \emptyset$ is essentially a VASS with a distinguished state q_0. A *partial run* is a run that does not have to satisfy the last condition.

A *bounded d-*BrVASS (*d*-BoBrVASS) is a *d*-BrVASS equipped with a bound $B \in \mathbb{N}$ that bounds all numbers in all configurations, similarly as for *d*-BoVASS. As in the case of BoVASS, for every BoBrVASS we abuse the notation allowing for inequality and equality test transitions in Δ_1.

A $(p(\boldsymbol{n}), q(\boldsymbol{m}))$-context (or just context if the configurations are not relevant) of a BrVASS is a partial run such that the root of the tree is labelled $p(\boldsymbol{n})$ and one of the leaves is labelled $q(\boldsymbol{m})$; all other leaves are labelled $q_0(\boldsymbol{0})$. We write $p(\boldsymbol{n}) \rightarrow^* q(\boldsymbol{m})$ if there exists a $(p(\boldsymbol{n}), q(\boldsymbol{m}))$-context and we write $p(\boldsymbol{n}) \not\rightarrow^* q(\boldsymbol{m})$ otherwise. Using this notation, the definition of computing a function $f \colon \{0, \dots, M\}^d \rightarrow \{0, \dots, N\}^d$ by a *d*-BoBrVASS generalises from the definition for *d*-BoVASS in the expected way.

For future reference, we state and prove the following easy claim.

▶ **Lemma 3.** *Given a d-*BoBrVASS *$\mathcal{B}$ with state set Q and upper bound B together with an integer $B' \geqslant B$, one can compute in polynomial time a d-*BoBrVASS *$\mathcal{B}'$ with state set $Q' \supseteq Q$ and upper bound B' such that the following holds: for any $p, q \in Q$ and $n, m \in \{0, \dots, B\}$, we have $p(n) \rightarrow^* q(m)$ in $\mathcal{B}$ if and only if $p(n) \rightarrow^* q(m)$ in $\mathcal{B}'$.*

Proof. It suffices to modify $\mathcal{B}$ as follows: after each transition of $\mathcal{B}$, add an additional sequence of inequality tests checking that all counter values are bounded by B.   ◀

The reachability problem.   The reachability question for all mentioned models is, given a system within the model (say, a VASS or a BoBrVASS etc.) and two configurations $p(\boldsymbol{n})$, $q(\boldsymbol{m})$, whether $p(\boldsymbol{n}) \rightarrow^* q(\boldsymbol{m})$[3].

In this paper we are interested in the reachability problem for bounded models with vectors encoded in binary. For *d*-BoVASS (i.e. without branching), the problem is easily contained in PSPACE for any *d*, because configurations can be described in polynomial space and solving the reachability problem amounts to nondeterministically guessing a run. Fearnley and Jurdziński proved in [10] that, in fact, the problem is PSPACE-complete even for $d = 1$.

For the BoBrVASS model the situation is somewhat similar, however there is a gap in our understanding. On one hand, the problem is trivially in EXPTIME for any *d*, because the total number of configurations is exponential in the size of the input system. Note here, that even though again every configuration can be described in polynomial space, due to branching we cannot apply the same reasoning as for BoVASS to argue PSPACE membership. For lower bounds, the result of [10] shows PSPACE-hardness. This leaves a gap between PSPACE and EXPTIME, which is the starting point of our work.

In this paper we show that the reachability problem for *d*-BoBrVASS model is EXPTIME-complete for any $d \geqslant 2$, leaving as an open problem the case of $d = 1$.

▶ **Theorem 4.** *For all $d \geqslant 2$, the reachability problem for d-*BoBrVASS *is EXPTIME-complete.*

The proof of Theorem 4 is split into two parts. In Section 3 we introduce auxiliary models that lie between 1-BoBrVASS and 2-BoBrVASS. Later in Section 4 we prove that these models are EXPTIME-hard.

---

[3] For BrVASS usually this question is phrased asking for the existence of a run not a context. In this scenario it means that in the input of the problem one requires $q(\boldsymbol{m}) = q_0(\boldsymbol{0})$. It is easy to see that the two questions are equivalent in terms of complexity for all variants of BrVASS in this paper.

3 Extensions of 1-BoBrVASS with doubling and halving

We introduce convenient extensions of the models 1-BoVASS and 1-BoBrVASS, where we add the ability of multiplying and dividing the counter value by 2. Formally, a *doubling transition* is a transition of the form $t = (p,q)_{\times 2}$ for some states p, q that has the following semantics: we can write $p(n) \xrightarrow{t} q(m)$ if and only if $m = 2n$. Similarly, a *halving transition* is a transition of the form $t = (p,q)_{\div 2}$ for some states p, q that has the following semantics: we may write $p(n) \xrightarrow{t} q(m)$ if and only if $m = n/2$. Note that in particular, if n is odd then the transition cannot be applied. Models 1-BoVASS$^{\times 2}$, 1-BoVASS$^{\div 2}$, and 1-BoVASS$^{\times 2, \div 2}$ extend 1-BoVASS by respectively allowing doubling transitions, halving transitions, and both doubling and halving transitions. Models 1-BoBrVASS$^{\div 2}$, 1-BoBrVASS$^{\times 2}$ and 1-BoBrVASS$^{\times 2, \div 2}$ extend 1-BoBrVASS in the same way.

It is not hard to see that doubling and halving transitions can be simulated using an additional counter, as explained next.

▶ **Lemma 5.** *For every 1-BoVASS$^{\times 2, \div 2}$ $\mathcal{V} = (Q, \Delta)$ there exists a 2-BoVASS $\mathcal{V}' = (Q', \Delta')$ of polynomial size in $\mathcal{V}$ such that $Q \subseteq Q'$ and $p(n) \rightarrow^* q(m)$ iff $p(n,0) \rightarrow^* q(m,0)$, for all $p, q \in Q$. Similarly, for every 1-BoBrVASS$^{\times 2, \div 2}$ there exists a 2-BoBrVASS that is analogous as above.*

Proof. Every doubling transition $(p,q)_{\times 2}$ can be simulated using the extra counter, two additional states $r_1, r_2 \notin Q$ and five transitions: $t_1 = (p, (0,0), r_1)$, $t_2 = (r_1, (-1, 2), r_1)$, $t_3 = (r_1, r_2)_{c_1 = 0}$, $t_4 = (r_2, (1, -1), r_2)$, $t_5 = (r_2, q)_{c_2 = 0}$. Similarly, every halving transition $(p,q)_{\times 2}$ can be simulated using the extra counter, two additional states $s_1, s_2 \notin Q$ and five transitions: $t_1 = (p, (0,0), s_1)$, $t_2 = (s_1, (-2, 1), s_1)$, $t_3 = (s_1, s_2)_{c_1 = 0}$, $t_4 = (s_2, (1, -1), s_2)$, $t_5 = (s_2, q)_{c_2 = 0}$. ◀

Therefore, 1-BoBrVASS$^{\times 2, \div 2}$ is a model in between 1-BoBrVASS and 2-BoBrVASS. This model was inspired by the *more powerful one register machine* (MP1RM) [19], used to prove the limitations of expressiveness for two-counter machines[4]. It is a one counter machine (without bounds) extended with two operations: multiplying by constants and dividing by constants. A more general model with polynomial updates was also studied in [12].

By Lemma 5, the reachability problem for d-BoBrVASS for any $d \geqslant 2$ is at least as hard as the reachability problem for 1-BoBrVASS$^{\times 2, \div 2}$. In the next section we will prove that the latter problem is already EXPTIME-hard, which will conclude the proof of Theorem 4. Before this, we show a side result which essentially says that doubling can be implemented using halving and vice versa.

▶ **Proposition 6.** *The reachability problems for models 1-BoVASS$^{\times 2, \div 2}$, 1-BoVASS$^{\times 2}$ and 1-BoVASS$^{\div 2}$ are equivalent with respect to polynomial time reductions. Similarly, the reachability problems for models 1-BoBrVASS$^{\times 2, \div 2}$, 1-BoBrVASS$^{\times 2}$ and 1-BoBrVASS$^{\div 2}$ are equivalent with respect to polynomial time reductions.*

Proof. We will give polynomial-time reductions from model 1-BoVASS$^{\times 2, \div 2}$ to models 1-BoVASS$^{\times 2}$ and 1-BoVASS$^{\div 2}$. The converse reductions are trivial, and the result for models with branching (1-BoBrVASS$^{\times 2, \div 2}$, 1-BoBrVASS$^{\times 2}$ and 1-BoBrVASS$^{\div 2}$) follows by applying the same construction.

[4] The amazing name and abbreviation of the model come from [19].

We start with the reduction from 1-BoVASS$^{\times 2, \div 2}$ to 1-BoVASS$^{\times 2}$. Consider a 1-BoVASS$^{\times 2, \div 2}$ $\mathcal{B}$, say with an upper bound B. By applying Lemma 3, we may assume that $B + 1$ is a power of 2, say $B = 2^n - 1$ for some positive integer n.

The idea is to simulate every halving transition $(u, v)_{\div 2}$ in $\mathcal{B}$ with a sequence of $O(n)$ transitions (standard or doubling); since n is polynomial in the size of $\mathcal{B}$, this is fine for a polynomial-time reduction. We explain this sequence in words; translating this description into a formal definition of a system and adding appropriate states and transitions to $\mathcal{B}$ is straightforward. Apply $n - 1$ times the following operation (depicted on Figure 2): if the counter value is smaller than 2^{n-1} then just double the counter; and otherwise subtract 2^{n-1} from the counter, double it and then add 1. Finally, at the end test whether the counter value is smaller than 2^{n-1}.
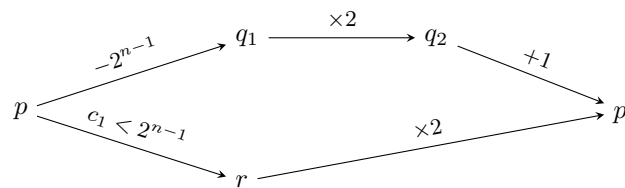
Figure 2 Shifting the bits cyclically to the left.

To see that these operations correctly implement halving the counter value, consider its bit encoding. Each single operation described above amounts to cyclically shifting the bit encoding to the left by 1: every bit is moved to a position one higher, apart from the bit from the highest (nth) position which is moved to the lowest position. Thus, after $n - 1$ applications the initial bit encoding is cyclically shifted by $n-1$ to the left, which is equivalent to shifting it by 1 to the right. So now it suffices to verify that the highest bit (which was the lowest in the encoding of the initial counter value) is 0, and if this is the case, then the final counter value is equal to half of the initial counter value.
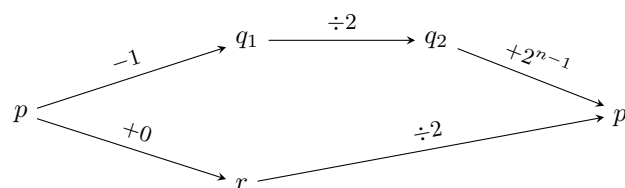
Figure 3 Shifting the bits cyclically to the right.

The reduction from 1-BoVASS$^{\times 2, \div 2}$ to 1-BoVASS$^{\div 2}$ follows by the same reasoning, but reversed. That is, we have to implement every doubling transition $(u, v)_{\times 2}$ in $\mathcal{B}$ using a sequence of $O(n)$ transitions (standard or halving). First, we verify that the counter value is smaller than 2^{n-1}, for otherwise the doubling transition cannot be applied. If this is the case, we cyclically shift the binary encoding of the counter value by 1 to the right $n - 1$ times. As Figure 3 shows, each such shift is performed by first nondeterministically guessing whether the counter value is odd, subtracting 1 if this is the case, halving, and adding 2^{n-1} to the counter value if the counter value was odd.   ◀
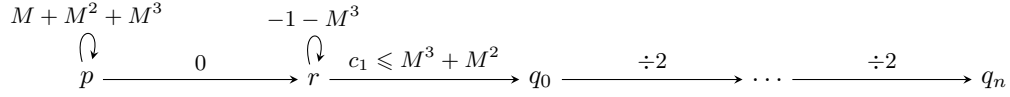
4   Exptime-hardness

This section is devoted to prove EXPTIME-completness of 1-BoBrVASS$^{\times 2, \div 2}$. Note that Theorem 7 below together with Lemma 5 immediately give Theorem 4.

▶ **Theorem 7.** *The reachability problem for 1-BoBrVASS$^{\times 2, \div 2}$ is EXPTIME-complete.*

We start by proving two lemmas that will be helpful in the reduction. Afterwards we define countdown games, an EXPTIME-complete problem from which we reduce. In the next two claims we write M for some power of 2, i.e. $M = 2^n$ for some $n \in \mathbb{N}$.

▶ **Lemma 8.** *Let $f \colon \{0, 1, \ldots, M-1\} \to \{0, 1, \ldots, M-1 + M(M-1)\}$ be defined as $f(x) = x + Mx$. Then f is computable by a 1-BoVASS$^{\times 2, \div 2}$ of size polynomial in the representation of M, with an upper bound $B = M + M(M-1) + M^2(M-1) + M^3(M-1)$.*

Proof. The set of states is $Q = \{p, r, q_0, q_1, \ldots, q_n\}$. The transitions are depicted in Figure 4.

$$M + M^2 + M^3 \qquad\qquad -1 - M^3$$

$$\begin{array}{ccccccc} & \circlearrowright & & & \circlearrowright & & \\ p & \xrightarrow{\quad 0 \quad} & r & & q_0 & \xrightarrow{\quad \div 2 \quad} & \cdots \xrightarrow{\quad \div 2 \quad} q_n \\ & & & c_1 \leqslant M^3 + M^2 & & & \end{array}$$

Figure 4 The 1-BoVASS$^{\times 2, \div 2}$ computing $f(x) = x + Mx$.

We now prove that the system defined above indeed computes f. Consider any $x \in \{0, 1, \ldots, M-1\}$. Observe that $p(x) \to^* r(y)$ if and only if $y = x + Ma + M^2 a + M^3 a - (M^3 + 1)b$ for some $0 \leqslant b \leqslant a \leqslant M-1$ (a is the number of loops taken in p and b is the number of loops taken in r). Then observe that $r(x + Ma + M^2 a + M^3 a) \to^* q_0(y)$ if and only if $y = x - a + Ma + M^2 a$. Finally, note that it is possible to go from q_0 to q_n only if the initial counter value is divisible by M. Since $|x - a| < M$, this is possible only if $x = a$. Then we get $q_0(Mx + M^2 x) \to^* q_n(y)$ if and only if $y = x + Mx$, so in total $p(x) \to^* q_n(y)$ if and only if $y = x + Mx$, as required.   ◀

We now combine the system provided by Lemma 8 with the branching feature of the BoBrVASS model to implement the key functionality: copying the counter value into two separate branches.

▶ **Lemma 9.** *There is a 1-BoBrVASS$^{\times 2, \div 2}$ of size polynomial in the representation of M, with an upper bound as in Lemma 8, and distinguished states p, q_1, q_2, with the following property. For every $x \in \{0, 1, \ldots, M-1\}$ there is a unique partial run ρ_x of the system whose root is labelled with configuration $p(x)$ and whose all leaves have states q_1 or q_2. Moreover, ρ_x has exactly two leaves: one with configuration $q_1(x)$ and second with configuration $q_2(x)$.*

Proof. We start the construction with the 1-BoVASS$^{\times 2, \div 2}$ provided by Lemma 8, hence we can assume that there is a state p' such that for all $x \in \{0, 1, \ldots, M-1\}$, $p(x) \to^* p'(y)$ if and only if $y = x + Mx$. Next, add states $\{r, q_1, s_0, \ldots, s_{n-1}, q_2\}$ to the system together with the following transitions:

- branching transition (p', r, s_0);
- inequality test $(r, q_1)_{c_1 \leqslant M-1}$; and
- halving transitions $(s_i, s_{i+1})_{\div 2}$ for all $i \in \{0, 1, \ldots, n-1\}$, where $s_n = q_2$.

We now argue that this system has the required property. Observe that starting from configuration $p(x)$, the system first has to move to configuration $p'(x + Mx)$, where it branches into configurations $r(y)$ and $s_0(z)$ with $y + z = x + Mx$. In one branch we test

that $y \leqslant M - 1$ and end in configuration $q_1(y)$. In the other branch we divide z by 2 exactly n times, which amounts to testing that z is divisible by M and finishing in configuration $q_2(z/M)$. Since $y \leqslant M - 1$, z is divisible by M, and $x \in \{0, 1, \ldots, M - 1\}$, we conclude that the split (y, z) of $x + Mx$ has to be equal to (x, Mx). Hence, the only two leaves are labelled with configurations $q_1(x)$ and $q_2(x)$, respectively. ◀

We will now show a reduction from a classic EXPTIME-complete problem – determining a winner of a countdown game – to reachability in 1-BoBrVASS$^{\times 2, \div 2}$. A *countdown game* is a pair (S, T) such that S is a set of nodes partitioned into two subsets $S = S_\exists \uplus S_\forall$, and $T \subseteq S \times (\mathbb{N} \setminus \{0\}) \times S$ are weighted transitions, with weights encoded in binary. We can assume that for every node s there exists exactly two outgoing transitions (s, n, s') for some integer n and node s'. A configuration of a game is $s(c)$, where $s \in S$ and $c \in \mathbb{N}$.

The game has two players, each owning her set of nodes (Player q owns S_q, for $q \in \{\exists, \forall\}$). The game proceeds as follows: for every configuration $s(c)$ the owner of the node s chooses one of the two transitions (s, n, s') such that $n \leqslant c$ and moves to the configuration $s'(c - n)$. If it is not possible to choose such a transition, then the game ends and Player $\forall$ wins the game. Otherwise, if at some point a configuration with value 0 is reached, then the game ends and Player $\exists$ wins the game.

Given a countdown game (S, T) and a starting configuration $s(c)$ the problem whether Player $\exists$ has a winning strategy is known to be EXPTIME-complete [16][5]. Without loss of generality we can assume that $c = 2^n - 1$ for some integer n.

Proof of Theorem 7. We give a reduction from the problem of determining the winner of a countdown game. Let then (S, T) be the given countdown game and $s(c)$ be its starting configuration, where $c = 2^n - 1$ for some integer n. We are going to construct, in polynomial time, a 1-BoBrVASS$^{\times 2, \div 2}$ $\mathcal{B}$ together with a root configuration such that $\mathcal{B}$ has an accepting run from the root configuration (i.e. one where all leaves are labelled with the initial configuration $q_0(0)$) if and only if Player $\exists$ wins the game.

We start by including all nodes of S in the state set of $\mathcal{B}$. For every node $r \in S_\exists$, let (r, n_1, r_1') and (r, n_2, r_2') be the two transitions in T that have r as the origin. Then add transitions $(r, -n_1, r_1')$ and $(r, -n_2, r_2')$ to the transition set of $\mathcal{B}$.

Next, for every node $r \in S_\forall$, let (r, n_1, r_1') and (r, n_2, r_2') be the two transitions in T that have r as the origin. Then add a copy of the system provided by Lemma 9 for $M = c + 1$, and identify its distinguished state p with r. Denoting the other two distinguished states of the copy by q_1^r and q_2^r, add transitions $(q_1^r, -n_1, r_1')$ and $(q_2^r, -n_2, r_2')$ to $\mathcal{B}$.

Finally, we add an initial state q_0 together with a transition $(r, 0, q_0)$ for each $r \in S$. Thus, $\mathcal{B}$ may accept whenever it reaches any state $r \in S$ with counter value 0. This finishes the construction of $\mathcal{B}$. We set the root configuration to be $s(c)$.

To see that the reduction is correct it suffices to observe that accepting runs of $\mathcal{B}$ with root configuration $s(c)$ are in one-to-one correspondence with winning strategies of Player $\exists$ in the game (S, T) starting from $s(c)$. Indeed, in nodes belonging to Player $\exists$, she may choose any of the two transitions originating there, which corresponds to the possibility of taking any of the two transitions in an accepting run of $\mathcal{B}$. In nodes belonging to Player $\forall$, Player $\exists$ has to be able to win for both possible moves of Player $\forall$, which corresponds to requiring acceptance in both subtrees obtained as a result of running the respective copy of the system provided by Lemma 9. ◀

---

[5] The game considered in [16] is slightly different but it is easy to show that both definitions are equivalent.

5   Limitations of 1-BoBrVASS

The natural approach to show EXPTIME hardness for 1-BoBrVASS is to ask whether they can efficiently compute the function $f : \{0, \ldots, M\} \to \{0, \ldots, 2M\}$ defined by $f(x) = 2x$. Then 1-BoBrVASS could simulate 1-BoBrVASS$^{\times 2}$ and EXPTIME-hardness would follow from Theorem 7 and Proposition 6. Unfortunately, we show that this is not the case. In the following proposition, we assume the original definition of 1-BoBrVASS, without inequality or equality tests as single transitions.

▶ **Proposition 10.** *Consider the function* $f \colon \{0, \ldots, M\} \to \{0, \ldots, 2M\}$ *defined as* $f(x) = 2x$, *for some* $M \in \mathbb{N}$. *Then every* 1-BoBrVASS *computing the function* f *has at least* M *states.*

Proof. Suppose $\mathcal{B}$ is a 1-BoBrVASS that computes f: there are $p, q \in Q$ such that for all $x \in \{0, \ldots, M\}$, $p(x) \to^* q(y)$ if and only if $y = 2x$. Here, Q denotes the set of states of $\mathcal{B}$.

For every $x \in \{0, \ldots, M\}$, fix any run ρ_x that witnesses $p(x) \to^* q(2x)$. The *spine* of ρ_x is the path from the root (labelled $p(x)$) to the leaf labelled $q(2x)$ (in the special case when $q(2x) = q_0(0)$ this path might be not unique and the spine is defined as any path).

▷ Claim 11. For every $x \in \{1, \ldots, M\}$, the spine of ρ_x contains at least one configuration with counter value 0.

Proof. Suppose every configuration on the spine of ρ_x has a positive counter value. Consider run ρ' obtained from ρ_x by decrementing all the counter value in each configuration on the spine by 1. Then ρ' witnesses that $p(x-1) \to^* q(2x-1)$, contradicting the assumption that $\mathcal{B}$ computes f.   ◁

▷ Claim 12. For all different $x, x' \in \{0, \ldots, M\}$, the sets of configurations on the spines of ρ_x and $\rho_{x'}$ are disjoint.

Proof. Suppose some configuration $r(z)$ appears both on the spine of ρ_x and of $\rho_{x'}$. Then replacing the context rooted at $r(z)$ in ρ_x with the context rooted at $r(z)$ in $\rho_{x'}$ yields a run witnessing that $p(x) \to^* q(2x')$. As $x \neq x'$, this contradicts the assumption that $\mathcal{B}$ computes f.   ◁

The proposition now follows immediately from the two claims above: For every $x \in \{1, \ldots, M\}$ the spine of ρ_x has to contain some configuration with counter value 0, and these configuration have to be pairwise distinct for different x. This implies that $\mathcal{B}$ has to have at least M states, as claimed.   ◀

6   Conclusion

We have proved EXPTIME-completeness of reachability for 2-BoBrVASS leaving the complexity for 1-BoBrVASS between PSPACE and EXPTIME. Our EXPTIME-hardness proof for 2-BoBrVASS heavily relied on the ability of manipulating a counter by multiplying and dividing it by 2. As shown by Proposition 10, one counter alone is not enough to implement this functionality in the BoBrVASS model.

At this moment we do not have any clear indication on whether reachability for 1-BoBrVASS is EXPTIME-hard, or it rather leans towards PSPACE. On one hand, EXPTIME-hardness would give us a natural problem to reduce from for other infinite-state systems with branching, similarly to the role served by reachability for 1-BoVASS. On the other hand,

PSPACE-membership would be even more interesting, as intuitively it would show that in the 1-BoBrVASS model the branching transitions are too weak to emulate alternation, contrary to the situation in multiple other models of similar kind.

To motivate the study of this problem even further, we highlight the connections between 1-BoBrVASS and 2-BrVASS. It is known that reachability for 2-VASS is PSPACE-hard, however the only proof of hardness we are aware of uses the reduction from 1-BoVASS. We show that similarly one can reduce 1-BoBrVASS to 2-BrVASS.

▶ **Proposition 13.** *The reachability problem for 1-BoBrVASS is reducible to the reachability problem for 2-BrVASS in polynomial time.*

Proof. Consider a 1-BoBrVASS $\mathcal{B}$, say with state set Q and upper bound $B \in \mathbb{N}$. The idea is to construct a 2-BrVASS $\mathcal{V}$ where every configuration $p(x)$ of $\mathcal{B}$ will be simulated by a configuration of the form $p(x, 2B - x)$ of $\mathcal{V}$.

We start by including Q in the state set of $\mathcal{V}$.

Next, we model every unary transition $t = (u, k, v)$ of $\mathcal{B}$ using two unary transitions in $\mathcal{V}$: $(u, (k, -B - k), r_t)$ followed by $(r_t, (0, B), v)$, where r_t is a new state added for the transition t. Note that this sequence of two transitions can be fired in configuration of the form $u(x, 2B - x)$ if and only if $x + k \in \{0, \ldots, B\}$, and then the resulting configuration is $v(x + k, 2B - (x + k))$.

Every branching transition $t = (u, v_1, v_2)$ of $\mathcal{B}$ is modelled in $\mathcal{V}$ as follows. We add to $\mathcal{V}$ two new states $s_{t,1}$ and $s_{t,2}$, a branching transition $(u, s_{t,1}, s_{t,2})$, and unary transitions $(s_{t,1}, (0, B), v_1)$ and $(s_{t,2}, (0, B), v_2)$. Observe that thus, after firing transition $(u, s_{t,1}, s_{t,2})$ in configuration of the form $u(x, 2B - x)$ we obtain two configurations of the form $s_{t,1}(x_1, y_1')$ and $s_{t,2}(x_2, y_2')$ satisfying $x_1 + x_2 = x$ and $y_1' + y_2' = 2B - x$, in which we have to apply the transitions that add B to the second counter, obtaining configurations $v_1(x_1, y_1)$ and $v_2(x_2, y_2)$ satisfying $x_1 + x_2 = x$ and $y_1 + y_2 = 4B - x$. Hence, provided the initial split of the second counter was $(y_1', y_2') = (B - x_1, B - x_2)$, we indeed obtain configurations $v_1(x_1, 2B - x_1)$ and $v_2(x_2, 2B - x_2)$, as was our initial intention. We will later argue that in runs in which we are interested, in all branching transitions the second counter has to be split as above.

Finally, if q_0 is the initial state of $\mathcal{B}$, we add a new initial state q_0' to $\mathcal{V}$ together with a transition $(q_0, (0, -2B), q_0')$. Thus, the only way to arrive at configuration $q_0'(0, 0)$ is to apply this transition in configuration $q_0(0, 2B)$.

Now consider any pair of configurations $p(n), q(m)$ of $\mathcal{B}$, where $n, m \in \{0, \ldots, B\}$. We claim that $p(n) \to^* q(m)$ in $\mathcal{B}$ if and only if $p(n, 2B - n) \to^* q(m, 2B - m)$ in $\mathcal{V}$.

For the forward direction, consider a partial run π of $\mathcal{B}$ witnessing $p(n) \to^* q(m)$ and modify it to a run ρ of $\mathcal{V}$ as follows. First, replace every configuration of the form $u(x)$ in π with configuration $u(x, 2B - x)$. Next, replace the transitions of $\mathcal{B}$ in π with sequences of transitions of $\mathcal{V}$ in the expected way, as described above. In particular, whenever in π a branching transition (u, v_1, v_2) is applied to configuration $u(x)$ and results in configurations $v_1(x_1)$ and $v_2(x_2)$, in ρ we model this by applying a branching transition to $u(x, 2B - x)$ and obtaining $s_{t,1}(x_1, B - x_1)$ and $s_{t,2}(x_2, B - x_2)$, which are subsequently modified to $v_1(x_1, 2B - x_1)$ and $v_2(x_2, 2B - x_2)$. Then ρ witnesses that $p(n, 2B - n) \to^* q(m, 2B - m)$ in $\mathcal{V}$.

For the backward direction, consider a partial run ρ of $\mathcal{V}$ witnessing $p(n, 2B - n) \to^* q(m, 2B - m)$. A node of ρ will be called *principal* if the state of its configuration belongs to the original state set Q; other nodes will be called *auxiliary*. Note that in ρ, every second level consists entirely of principal nodes, and every other level consists entirely of auxiliary nodes. More specifically, every principal node a, say labelled with a configuration $u(x, y)$, is of one of the following four kinds:

- a has one child and one grandchild, labelled respectively with configurations $r_t(x + k, y - k - B)$ and $v(x + k, y - k)$ for some unary transition (u, k, v) of $\mathcal{B}$;
- a has two children, each having one child of its own, and these are respectively labelled with configurations $s_{t,1}(x_1, y_1'), s_{t,2}(x_2, y_2'), v_1(x_1, y_1' + B), v_2(x_2, y_2' + B)$ for some branching transition (u, v_1, v_2) of $\mathcal{B}$;
- a has one leaf child labelled with $q_0'(0, 0)$, so in particular the configuration at a is $q(x, z) = q_0(0, 2B)$; or
- a is the unique leaf labelled with $q(m, 2B - m)$.

These types of principal nodes will be called *unary*, *branching*, *leaf*, and *final*, respectively.

We first prove by a bottom-up induction over ρ that in all principal nodes, the sum of counter values is equal to $2B$. This trivially holds for all principal nodes that are leaf or final. Next, whenever a is a unary principal node with configuration $u(x, y)$, then by the induction assumption for its grandchild we infer that $(x + k) + (y - k) = 2B$ for some integer k, implying $x + y = 2B$ as required. The case of branching principal nodes is analogous.

We now prove by a top-down induction over ρ that in all principal nodes, the value of the first counter is upper bounded by B. This is certainly true in the root, since we assumed that $n \leqslant B$. Suppose a is a unary principal node with configuration $u(x, 2B - x)$ for which we already know that $x \leqslant B$, and its child and grandchild are respectively labelled with $r_t(x + k, B - x - k)$ and $v(x + k, 2B - x - k)$ for some unary transition (u, k, v) of $\mathcal{B}$. Then the existence of the child's configuration implies that $B - x - k \geqslant 0$, implying that $x + k \leqslant B$, as required. Finally, observe that if a is a branching principal node, then the sum of the first counter values in its grandchildren is equal to the first counter value in a, hence an upper bound of B on the latter entails an upper bound of B on the former.

All in all, we argued that all principal nodes in ρ have configurations of the form $u(x, 2B - x)$, where $u \in Q$ and $x \in \{0, \dots B\}$. It now remains to observe that taking all principal nodes with tree order induced from ρ and forgetting the second counter value yields a run of $\mathcal{B}$ witnessing that $p(n) \to^* q(m)$ in $\mathcal{B}$. ◀

───── **References** ─────

1   Rajeev Alur and David L. Dill. A Theory of Timed Automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994. `doi:10.1016/0304-3975(94)90010-8`.

2   Michael Blondin, Alain Finkel, Stefan Göller, Christoph Haase, and Pierre McKenzie. Reachability in Two-Dimensional Vector Addition Systems with States Is PSPACE-Complete. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*, pages 32–43, 2015. `doi:10.1109/LICS.2015.14`.

3   Mikolaj Bojańczyk. Automata column. *SIGLOG News*, 1(2):3–12, 2014. URL: `https://dl.acm.org/citation.cfm?id=2677163`.

4   Mikolaj Bojańczyk, Anca Muscholl, Thomas Schwentick, and Luc Segoufin. Two-variable logic on data trees and XML reasoning. *J. ACM*, 56(3):13:1–13:48, 2009. `doi:10.1145/1516512.1516515`.

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 Lazi', and Filip Mazowiecki. Timed pushdown automata and branching vector addition systems. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12, 2017. `doi:10.1109/LICS.2017.8005083`.

7   Costas Courcoubetis and Mihalis Yannakakis. Minimum and Maximum Delay Problems in Real-Time Systems. *Formal Methods in System Design*, 1(4):385–415, 1992. `doi:10.1007/BF00709157`.

8 Wojciech Czerwinski and Slawomir Lasota. Regular separability of one counter automata. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12, 2017. `doi:10.1109/LICS.2017.8005079`.

9 Matthias Englert, Ranko Lazic, and Patrick Totzke. Reachability in Two-Dimensional Unary Vector Addition Systems with States is NL-Complete. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 477–484, 2016. `doi:10.1145/2933575.2933577`.

10 John Fearnley and Marcin Jurdzinski. Reachability in two-clock timed automata is PSPACE-complete. *Inf. Comput.*, 243:26–36, 2015. `doi:10.1016/j.ic.2014.12.004`.

11 Diego Figueira, Ranko Lazic, Jérôme Leroux, Filip Mazowiecki, and Grégoire Sutre. Polynomial-Space Completeness of Reachability for Succinct Branching VASS in Dimension One. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 119:1–119:14, 2017. `doi:10.4230/LIPIcs.ICALP.2017.119`.

12 Alain Finkel, Stefan Göller, and Christoph Haase. Reachability in Register Machines with Polynomial Updates. In *Mathematical Foundations of Computer Science 2013 - 38th International Symposium, MFCS 2013, Klosterneuburg, Austria, August 26-30, 2013. Proceedings*, pages 409–420, 2013. `doi:10.1007/978-3-642-40313-2_37`.

13 Stefan Göller, Christoph Haase, Ranko Lazic, and Patrick Totzke. A Polynomial-Time Algorithm for Reachability in Branching VASS in Dimension One. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 105:1–105:13, 2016. `doi:10.4230/LIPIcs.ICALP.2016.105`.

14 Christoph Haase, Stephan Kreutzer, Joël Ouaknine, and James Worrell. Reachability in Succinct and Parametric One-Counter Automata. In *CONCUR 2009 - Concurrency Theory, 20th International Conference, CONCUR 2009, Bologna, Italy, September 1-4, 2009. Proceedings*, pages 369–383, 2009. `doi:10.1007/978-3-642-04081-8_25`.

15 Christoph Haase, Joël Ouaknine, and James Worrell. On the Relationship between Reachability Problems in Timed and Counter Automata. In *Reachability Problems - 6th International Workshop, RP 2012, Bordeaux, France, September 17-19, 2012. Proceedings*, pages 54–65, 2012. `doi:10.1007/978-3-642-33512-9_6`.

16 Marcin Jurdziński, Jeremy Sproston, and François Laroussinie. Model Checking Probabilistic Timed Automata with One or Two Clocks. *Logical Methods in Computer Science*, 4(3), 2008. `doi:10.2168/LMCS-4(3:12)2008`.

17 Jérôme Leroux and Sylvain Schmitz. Reachability in Vector Addition Systems is Primitive-Recursive in Fixed Dimension. In *LICS*, 2019. `arXiv:1903.08575`.

18 Ernst W. Mayr. An Algorithm for the General Petri Net Reachability Problem. *SIAM J. Comput.*, 13(3):441–460, 1984. `doi:10.1137/0213029`.

19 Rich Schroeppel. A two counter machine cannot calculate $2N$. *Artificial Intelligence Memo No. 257*, 1972.

20 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`.