Timed pushdown automata and branching vector addition systems

Lorenzo Clemente*, Sławomir Lasota*, Ranko Lazić†, and Filip Mazowiecki†
*Faculty of Mathematics, Informatics and Mechanics, University of Warsaw, Poland
†DIMAP, Department of Computer Science, University of Warwick, UK

Abstract—**We prove that non-emptiness of timed register pushdown automata is decidable in doubly exponential time. This is a very expressive class of automata, whose transitions may involve state and top-of-stack clocks with unbounded differences. It strictly subsumes pushdown timed automata of Bouajjani et al., dense-timed pushdown automata of Abdulla et al., and orbit-finite timed register pushdown automata of Clemente and Lasota. Along the way, we prove two further decidability results of independent interest: for non-emptiness of least solutions to systems of equations over sets of integers with addition, union and intersections with $\mathbb{N}$ and $-\mathbb{N}$, and for reachability in one-dimensional branching vector addition systems with states and subtraction, both in exponential time.**

I. Introduction

Background: Timed automata [3] are one of the most studied and used models of reactive timed systems. Motivated by verification of programs with both procedural and timed features, several extensions of timed automata by a pushdown stack have been proposed, including pushdown timed automata (PDTA) [6], recursive timed automata (RTA) [4], [19], dense-timed pushdown automata (dtPDA) [1], and timed register pushdown automata (trPDA) [9].

While PDTA simply add an untimed stack to a timed automaton, dtPDA are allegedly more powerful since they allow to store clocks on the stack evolving at the same rate as clocks in the finite control. Surprisingly, Clemente and Lasota showed that, as a consequence of the interplay of the stack discipline and the monotone elapsing of time, dtPDA is in fact not more expressive than PDTA, and the two models are strictly subsumed by *orbit-finite* trPDA [9]. Moreover, subsumption still holds if trPDA are restricted to timeless stack, and in this case there is nothing to pay in terms of the complexity of non-emptiness, which is the central decision problem for model checking: it is ExpTime-complete for both PDTA, dtPDA, and trPDA with timeless stack; for orbit-finite trPDA, the best known upper bound rises to NExpTime (ibid.). The main question posed in the latter work is whether the heavy restriction of orbit finiteness, which bounds the differences between state and top-of-stack clocks, can be lifted while keeping non-emptiness decidable.

Partially supported by the European Research Council (ERC) project *Lipa* under the EU's Horizon 2020 research and innovation programme (grant agreement No. 683080), by the Polish NCN grant 2016/21/B/ST6/01505, and by the Engineering and Physical Sciences Research Council (grant EP/M011801/1).

The proofs of the NExpTime and ExpTime upper bounds for orbit-finite and timeless-stack trPDA (respectively) [9] involved translations to systems of equations in which variables range over sets of integers, and available operations include addition, union, and intersection with the singleton set $\{0\}$. Similar systems have been studied in a variety of contexts, and extensions quickly lead to undecidability: e.g., already over the naturals, when arbitrary intersections are permitted, decidability is lost since this model subsumes unary conjunctive grammars [14].

Contributions: Our headline result answers positively the question raised by Clemente and Lasota [9]: we prove that non-emptiness remains decidable when the assumption of orbit-finiteness of trPDA is dropped. The resulting class of automata strictly subsumes all pushdown extensions of timed automata mentioned above (with the exception of RTA [4], [19][1]), and is the first one to allow timed stacks without bounding the differences of state and top-of-stack clocks.[2] For example, it is able to recognise the language of all *timed* palindromes over $\{a, b\}$ containing the same number of a's and b's (cf. Example 2).

The decidability proof proceeds in three stages. The first one is a sequence of transformations that translates, in exponential time, trPDA to systems of equations over sets of integers with addition, union, and intersections with $\mathbb{N}$ and $-\mathbb{N}$, and is considerably more involved than the corresponding translation for orbit-finite trPDA [9].

Secondly, we show how such systems of equations can be translated in polynomial time to one-dimensional branching vector addition systems with states extended with a *subtraction operation* (1-BVASS$^{\pm}$). Branching vector addition systems with states (BVASS) have been studied extensively in recent years with motivations coming from computational linguistics, linear logic, and verification of recursively parallel programs amongst others; cf. Lazić and Schmitz [18] and references therein. In particular, while the reachability problem for BVASS is not known to be decidable in general, it was recently proved to be PTime-complete in dimension one for unary encoding of constants [13], and PSpace-complete for binary

[1]The model of RTA differs significantly from the other models since the stack contains clock *values* which are constant w.r.t. the elapsing of time.

[2]Note that Clemente and Lasota denoted by trPDA an undecidable class in which many stack symbols can be popped and pushed in one step, like in prefix-rewriting. For simplicity, we use the same name for the new largest decidable subclass.

encoding [12]; however, the model extended with subtractions is undecidable already in dimension six [17].

In the final stage of the decidability proof, we argue by an ingenious surgery of reachability trees that 1-BVASS$^{\pm}$ have a small-model property, i.e., that reachability in dimension one has witnesses with values bounded exponentially. We thus deduce that 1-BVASS$^{\pm}$ reachability is decidable in EXPTIME.

The former upper bound on 1-BVASS$^{\pm}$ combined with our exponential reduction from trPDA yields a 2-EXPTIME upper bound for the trPDA non-emptiness problem. To supplement the two main translations, we exhibit also their reverses: from 1-BVASS$^{\pm}$ to the systems of equations, and from the latter to trPDA, both in polynomial time.

Finally, we mention the analysis of dtPDA based on tree automata of [2]. It is shown there that runs of dtPDA can be represented as graphs of bounded split-width, and one can construct a finite tree automaton recognizing precisely those decompositions corresponding to timed runs of the dtPDA. Upon a closer inspection of our approach for trPDA (cf. the reduction to BVASS outlined below), it can be argued that we also perform a similar reduction to a kind of tree automaton, albeit not a finite one, but one *with a counter* taking values in the nonnegative integers. This extra counter is needed to keep track of possibly unbounded differences between register values for matching push/pop pairs. The fact that a finite tree automaton suffices when analyzing dtPDA should be contrasted to our previous semantic collapse result of dtPDA to the variant with timeless stack [9]: For the latter model, since the stack is timeless, there are no long push/pop timing dependencies and a finite tree automaton clearly suffices.

Organisation: After defining timed register pushdown automata in Sec. II and systems of equations over sets of integers in Sec. III, we develop the translations between them in Sec. IV and V. In Sec. VI we define 1-BVASS$^{\pm}$, in Sec. VII we establish PTIME interreducibility between 1-BVASS$^{\pm}$ and equations, and in Sec. VIII we prove that their reachability problem is in EXPTIME. Some closing remarks can be found in Sec. IX.

II. TIMED REGISTER PUSHDOWN AUTOMATA

We are interested in an extension of pushdown automata where control states and stack symbols are additionally equipped with tuples of values from an infinite *time domain* $\mathbb{T}$, which can be either the *dense time* structure $(\mathbb{Q}, \leqslant, +1)$, or the simpler *discrete time* structure $(\mathbb{Z}, \leqslant, +1)$. Our results are valid for both structures; in the sequel, we focus on the former.

A. Constraints

We describe subsets of $\mathbb{Q}^k$, for $k > 0$, by formulas. A *constraint* $\phi(x_1, \ldots, x_k)$ of *dimension* k is a Boolean combination[3] of atomic formulas using variables $x_1, \ldots, x_k$, the binary predicate $\leqslant$, and the unary function $+1$. We denote by $\mathrm{var}(\phi)$ the set of free variables appearing in ϕ. For example, the following is a constraint with free variables $\mathrm{var}(\phi) = \{x_1, x_2, x_3\}$:

$$\begin{aligned}\phi(x_1, x_2, x_3) \;\equiv\; & \big(x_1 \leqslant x_2 \wedge x_2 \leqslant x_1{+}1 \wedge \neg(x_1{+}1 \leqslant x_2)\big) \\ & \vee \big((x_1{+}1){+}1 \leqslant x_3 \wedge x_3 \leqslant (x_1+1){+}1\big).\end{aligned}$$

By using syntactic sugar we can rewrite the above constraint as $x_1 \leqslant x_2 < x_1 + 1 \vee x_1 + 2 = x_3$. We assume a binary representation of integer constants in constraints.

A constraint ϕ defines the set $[\![\phi]\!] \subseteq \mathbb{Q}^k$ of all valuations satisfying ϕ. For example, the above constraint denotes the set $[\![\phi]\!] = \big\{(a, b, c) \in \mathbb{Q}^3 \;\big|\; a \leqslant b < a + 1 \text{ or } a + 2 = c\big\}$. Sets of the form $[\![\phi]\!]$ for a constraint ϕ are called *definable*.

An *automorphism* of the dense time structure is a bijective function $f : \mathbb{Q} \to \mathbb{Q}$ which is monotonic (i.e., $x \leqslant y$ implies $f(x) \leqslant f(y)$) and preserves integer distances (i.e., $f(x+1) = f(x) + 1$). Thus, rigid translations $f(x) = x + k$ are dense time automorphisms for every $k \in \mathbb{Q}$, as well as functions of the form $g(x) = \lfloor x \rfloor + h(x - \lfloor x \rfloor)$ where h is a monotonic bijection of the interval $[0, 1)$ to itself.[4] The *orbit* of a tuple $x \in \mathbb{Q}^k$ is the set of all tuples of the form $f(x)$, where f is a dense time automorphism (extended point-wise in the natural way). In other words, an orbit is an equivalence class of tuples carrying the same information in terms of ordering and integer differences. For example, the orbit of the tuple $(0, 0.5, 1)$ is the set of tuples (x, y, z) s.t. $x < y < z$ and $z - x = 1$; thus, $(2, 2.2, 3)$ is in the same orbit as $(0, 0.5, 1)$ (i.e., they are equivalent modulo timed automorphisms), but neither is $(2, 2, 3)$ nor $(2, 2.2, 3.1)$. Every definable set $[\![\phi]\!]$ is invariant under automorphisms, and thus is partitioned into (necessarily disjoint) orbits. When the partition is finite, we call $[\![\phi]\!]$ *orbit-finite*. For instance, the set defined by the constraint $x_1 \leqslant x_2 < x_1 + 1$ is partitioned into two orbits, which are defined, respectively, by the constraints:

$$x_1 = x_2 \quad \text{and} \quad x_1 < x_2 < x_1 + 1.$$

Notice that a definable set needs not be orbit-finite in general. For instance the constraint $x_1 > x_2{+}1$ defines an orbit-infinite set, since for every two integer distances $i, j > 1$ and $x \in \mathbb{Q}$, the two tuples $(x + i, x)$ and $(x + j, x)$ cannot be in the same orbit unless $i = j$. A constraint defining an orbit-finite set is itself called orbit-finite.

Let the *span* of a tuple $a \in \mathbb{Q}^k$ be $\max a - \min a$, i.e., the difference between the maximal and the minimal values in a. A set has *bounded span* if it admits a common bound on the spans of all its elements. If the span is bounded, then it is exponentially bounded since constants are encoded in binary. We obtain the following characterization of orbit-finiteness:

Proposition 1 (cf. [9, Lemma III.1])**.** *A definable set $[\![\phi]\!]$ is orbit-finite if, and only if, it has bounded span K. Moreover, K is at most exponential.*

In dimension $k = 1$ there are only trivial constraints, and hence the only definable sets are $\mathbb{Q}$ and $\varnothing$.

[3]Allowing first order quantifiers does not increase the expressive power, as both dense and discrete time structures admit quantifier elimination.

[4]While we are usually concerned with dense time automorphisms in the paper, we will occasionally use also automorphisms of the structure $(\mathbb{Q}, \leqslant)$, which is a weaker variant requiring only monotonicity.

B. Timed register PDA

Our model is obtained by extending classical PDA with additional *registers* holding timestamps from the dense time structure. Registers can be compared with the binary predicate $\leqslant$ and the unary function $+1$. We allow both registers in the finite control (i.e., global registers) and in the stack; the latter registers can be pushed, popped, and checked against global registers. Most importantly, we allow unbounded differences between global registers and registers on top of the stack.

While classical timed models (e.g. [3], [7], [1]) use *clocks*, which measure differences between timestamps, we use registers, which record the timestamps themselves. Clock resets are simulated by assigning the current input timestamp to a register, and comparing a clock against a constant k is simulated by comparing the difference between a register and the current input timestamp against k. While a clock increases its value monotonically with the elapse of time, the value of a register is preserved until the next assignment. In particular, our model can in general read non-monotonic words, i.e., words with no relationship between timestamps of subsequent symbols; if necessary, monotonic input can be enforced *within the model* by adding an extra control register recording the previous input timestamp and constraining the next one.

A *timed register PDA* $\mathcal{P}$ (trPDA) of *dimension* k is a tuple

$$\langle(\phi_q)_{q\in Q}, (\phi_a)_{a\in A}, (\phi_\gamma)_{\gamma\in\Gamma}, I, F, (\mathsf{push}_\delta)_{\delta\in\Delta_{\mathsf{push}}}, (\mathsf{pop}_\delta)_{\delta\in\Delta_{\mathsf{pop}}}\rangle$$

where:

(1) Q is a finite set of of control states and, for each $q \in Q$, ϕ_q is an *orbit-finite* constraint of dimension k; variables appearing in ϕ_q can be understood as register names in state q, and $[\![\phi_q]\!] \subseteq \mathbb{Q}^k$ describes all admissible values of registers in control state q;
(2) A is a finite alphabet of input symbols; for each $a \in A$, ϕ_a is an *orbit-finite* constraint of dimension k describing admissible tuples of time values that can accompany the input symbol a;
(3) Γ is a finite set of stack symbols; for each $\gamma \in \Gamma$, ϕ_γ is an *orbit-finite* constraint of dimension k that describes those tuples of time values that can be stored on the stack together with the stack symbol γ;
(4) $I, F \subseteq Q$ are the sets of initial and accepting control states, resp., and
(5) $\Delta_{\mathsf{push}}, \Delta_{\mathsf{pop}} \subseteq Q \times A_\varepsilon \times Q \times \Gamma$ are the set of push and pop transitions, resp., where A_ε is A extended with the empty word $\varepsilon \notin A$; moreover, for every transition $\delta = (p, a, q, \gamma) \in \Delta_{\mathsf{push}}$, $\mathsf{push}_{paq\gamma}$ is a *not necessarily orbit-finite* constraint of dimension $4k$, and similarly for Δ_{pop}.

A push constraint $\mathsf{push}_{paq\gamma}(\vec{x}_p, \vec{x}_a, \vec{x}_q, \vec{x}_\gamma)$ has $4k$ free variables $\vec{x}_p, \vec{x}_a, \vec{x}_q, \vec{x}_\gamma$ (each of size k), where $\vec{x}_p = (x_{p,1}, \ldots, x_{p,k})$ represents registers in the current control state p, $\vec{x}_a$ represents the timestamps in the input symbol a, $\vec{x}_q$ represents the registers in the next control state q, and $\vec{x}_\gamma$ represents the registers in the stack symbol γ (which is in this case pushed on the stack); similarly for Δ_{pop}.

We consider a trPDA $\mathcal{P}$ as a symbolic representation for the infinite-state pushdown automaton $\mathcal{P}' = \left\langle Q', A', \Gamma', I', F', \Delta'_{\mathsf{push}}, \Delta'_{\mathsf{pop}} \right\rangle$ where

$$\begin{aligned} Q' &= \bigcup_{q\in Q} \{q\} \times [\![\phi_q]\!] & A' &= \bigcup_{a\in A} \{a\} \times [\![\phi_a]\!] \\ I' &= \bigcup_{q\in I} \{q\} \times [\![\phi_q]\!] & \Gamma' &= \bigcup_{\gamma\in \Gamma} \{\gamma\} \times [\![\phi_\gamma]\!] \\ F' &= \bigcup_{q\in F} \{q\} \times [\![\phi_q]\!] \end{aligned}$$

and $\Delta'_{\mathsf{push}} \subseteq Q' \times A' \times Q' \times \Gamma'$ is defined as the union, over all $(p, a, q, \gamma) \in \Delta_{\mathsf{push}}$, of relations of the form

$$\left\{((p, t), (a, u), (q, v), (\gamma, w)) \;\middle|\; (t, u, v, w) \in [\![\mathsf{push}_{paq\gamma}]\!]\right\},$$

and similarly for Δ'_{pop}. To the PDA $\mathcal{P}'$ we can apply all the classical definitions for PDAs, namely the notion of run, accepting run, language recognized, etc., and thus all these notions transfer to the trPDA $\mathcal{P}$. As an immediate consequence, classical closure properties of languages recognized by PDAs (i.e., context-free languages) directly transfer to trPDA, such as closure under union, reversal, concatenation, iteration, images and inverse images of (timed) homomorphisms. For technical convenience, in the sequel we assume w.l.o.g. that a trPDA starts in an initial control state with the empty stack, and accepts when it enters an accepting control state *and* its stack is empty. The *non-emptiness problem* for a given trPDA $\mathcal{P}$ is to decide whether there exists an accepting run.

Observe that we do not include "nop" transitions, i.e., transitions not accessing the stack, since they can be simulated by pushing a dummy symbol (with any time value) and then immediately popping it. An alternative definition yielding an essentially equivalent model could be considered where each control state, input symbol, or stack symbol, has its own dimension, including dimension 0, but we avoid this for simplicity. In dimension $k = 1$ the orbit-finite constraints on states, input alphabet, and stack alphabet can be omitted. Examples demonstrating the syntax and expressive power of trPDA can be found in Example 1 and 2 below.

Remark 1. The model of trPDA is not ad hoc: it is an instantiation of a general model of *definable* PDA [8] where atoms are $(\mathbb{Q}, \leqslant, +1)$, resp. $(\mathbb{Z}, \leqslant, +1)$, along the general lines of [8], [15], [16].

C. State of the art

Since trPDA generalise pushdown automata, all problems which are undecidable for pushdown automata (such as universality, equivalence, inclusion, disjointness, etc.) remain undecidable for trPDA. The challenge then, is to show which decidable problems on pushdown automata are still decidable for trPDA. In this paper, we focus on the non-emptiness problem. In the rest of the section, we discuss the relationship of trPDA with related models in the literature.

First of all, trPDA without stack is the same as timed register finite automata [5], a generalization of timed automata [3] *with* *uninitialized clocks*. For this reason, we require that the state constraints ϕ_q are orbit-finite, otherwise non-emptiness would be undecidable already in dimension $k = 3$ and without a stack [5]. Indeed, by dropping orbit-finiteness of the control state, it becames possible to simulate a 2-counter machine with zero test (i.e., a Minsky machine) using only three registers x_1, x_2, x_3: The two counters are represented as the differences $x_1 - x_3$ and $x_2 - x_3$, respectively; incrementation of the first counter is $x_1' = x_1+1$, decrementation is $x_1 = x_1'+1$, and zero test is $x_1 = x_3$; similarly for the second counter. On the other hand, orbit-finiteness of ϕ_a, for input symbols a's, is inessential for the non-emptiness problem (we existentially quantify it away; cf. the end of Sec. II-D), and thus can be dropped. We do not know whether dropping the orbit-finiteness of ϕ_γ, for stack symbols γ, would lead to undecidability.

If we add a (classical, untimed) stack to timed register finite automata, then we obtain trPDA with timeless stack, which already subsume several models from the literature, such as *pushdown timed automata* [7] with uninitialized clocks, and *dense-timed PDA* [1] with uninitialized clocks. (The latter model has been shown to be expressively equivalent to pushdown timed automata in [9].) Non-emptiness of trPDA with timeless stack is EXPTIME-complete [9].

Timed register context-free grammars (trCFG) correspond to trPDA (thus with a timed stack) with a single control state of dimension zero, plus "long rules" reading one stack symbol and pushing possibly many stack symbols. We show in [9] that non-emptiness of trCFG is decidable in EXPTIME. Intuitively, for grammars one can just orbitize separately the stack symbols; while this operation in general provides an overapproximation of the set of accepting runs for trPDA, in the case of grammars it is precise. Unlike the models mentioned so far, trPDA with untimed control states, and also trCFG, can express truly timed context-free properties such as timed palindromes, as shown below.Notice that in this model push and pop transitions have $2k$ free variables, where k is the dimension of input and stack alphabets.

Example 1. Let the input alphabet $A = \{a, b\}$ contain two input symbols of dimension one, and consider the language L of *timed palindromes* of even length, $L = \{ww^R \mid w \in (A \times \mathbb{Q})^*\}$. Notice how palindromicity is required also in the timestamps, which makes it impossible for L for be recognized without a truly timed stack. We construct trPDA $\mathcal{P}$ recognizing L with just two control states $Q = \{p, q\}$ of dimension zero, of which p is initial and q is final, and a stack alphabet $\Gamma = \{\bar{a}, \bar{b}\}$ of dimension one. (Alternatively, one can construct a trCFG recognizing the same language; but we stick to the trPDA syntax here.) In the constraints below, the variable y refers to the timed value on the top of the stack (pushed or popped, depending on the transition), and z refers to the timed value of the input symbol. In control state p, upon reading an input symbol $(c, z) \in A \times \mathbb{Q}$, the automaton pushes $(\bar{c}, z) \in \Gamma \times \mathbb{Q}$ to the stack, and it decides nondeterministically whether to stay in p, or move to control state q: $(p, c, r, \bar{c}) \in \Delta_{\mathsf{push}}$ for $r \in \{p, q\}$ and $c \in \{a, b\}$, where

$$\mathsf{push}_{pcr\bar{c}}(z, y) \equiv y = z.$$

From control state q, the automaton attempts to pop the symbols appearing in the input: $(q, c, q, \bar{c}) \in \Delta_{\mathsf{pop}}$ for $c \in \{a, b\}$, where $\mathsf{pop}_{qcq\bar{c}}(z, y) \equiv y = z$.

Let's now consider trPDA with time both in the control state *and* in the stack. *Orbit-finite trPDA* are trPDA where we additionally demand orbit-finiteness of the following two projections consisting of control state and topmost stack symbol [9]:

$$\left\{(t, w) \in \mathbb{Q}^{2k} \mid \exists u, v \in \mathbb{Q}^{2k} \cdot (t, u, v, w) \in [\![\mathsf{push}_{paq\gamma}]\!]\right\}, \text{ and}$$
$$\left\{(t, w) \in \mathbb{Q}^{2k} \mid \exists u, v \in \mathbb{Q}^{2k} \cdot (t, u, v, w) \in [\![\mathsf{pop}_{paq\gamma}]\!]\right\}.$$

Non-emptiness for orbit-finite trPDA is in NEXPTIME [9], while the decidability status of trPDA without the orbit-finite restriction above has been left open. Orbit-finite trPDA are more expressive than trCFG. In fact, while the projection to the untimed component of languages recognized by trCFG are the (classical) context-free languages [9, Lemma IV.2], orbit-finite trPDA can recognize (untimed) non-context-free languages. One such language is the set of *untimed* palindromes over $\{a, b\}$ containing the same number of a's and b's [9, Example IV.3]. Notice that the orbit-finiteness constraint prevents us from recognizing timed palindromes and at the same time carrying unrelated timing information in the control state; this will be fixed in Example 2 below.

In trPDA studied in this paper, *we do not require* that the push/pop constraints $\mathsf{push}_{paq\gamma}$ and $\mathsf{pop}_{paq\gamma}$ are orbit-finite. Therefore, in trPDA we allow values of registers in the stack to be arbitrarily far from the values of registers in the state. We show in this paper that non-emptiness for trPDA is decidable (in 2-EXPTIME; cf. Theorem 1), even dropping such orbit-finiteness requirement. This closes the problem left open in [9]. Moreover, the example below presents a language which can be recognized by trPDA but not by an orbit-finite trPDA, thus showing that orbit-finiteness is a true semantic constraint.

Example 2. With $A = \{a, b\}$ as in Example 1 above, consider the language L of timed palindromes of even length over $A \times \mathbb{Q}$ *with the same number of a's and b's*. Thus, L combines the language from Examples 1 and [9, Example IV.3]. We construct below a trPDA $\mathcal{P}$ of dimension $k = 1$ recognizing L. Notice that (1) palindromicity applies not only to the finite part A, but also to time values, and (2) the projection of L to A is not a context-free language. Thus trPDA can express nontrivial counting properties on top of a given context-free language.

In the constraints below, the variable x represents the value of the register in the current state, x' refers to its value in the next state, y refers to the timed value on the top of the stack (pushed or popped, depending on the transition), and z refers to the timed value of the input symbol. The stack alphabet is $\Gamma = \{\bar{a}, \bar{b}, \bot\}$, and we have four control states $Q = \{p, q, r, s\}$, of which $I = \{p\}$ is initial, and $F = \{s\}$ is final.

The sets of push and pop transitions Δ_{push} and Δ_{pop} contain precisely the transitions mentioned below. From the initial control state p, the automaton moves to the control state q by nondeterministically initializing its register and pushing it on the stack with bottom-of-stack symbol $\perp$: $(p, \varepsilon, q, \perp) \in \Delta_{\mathsf{push}}$, where $\mathsf{push}_{p\varepsilon q\perp}(x, z, x', y) \equiv (x = x' = y)$. In control state q, the automaton pushes the input symbol $(c, z) \in A \times \mathbb{Q}$ to the stack, and at the same time increments or decrements its state register depending on whether $c = a$ or $c = b$; moreover, it stays nondeterministically in q, or moves to r: $(q, c, q', \bar{c}) \in \Delta_{\mathsf{push}}$ for $q' \in \{q, r\}$ and $c \in \{a, b\}$, where

$$\mathsf{push}_{qaq'\bar{a}}(x, z, x', y) \equiv (x' = x + 1 \wedge y = z), \text{ and}$$
$$\mathsf{push}_{qbq'\bar{b}}(x, z, x,' y) \equiv (x = x' + 1 \wedge y = z).$$

From control state r, the automaton attempts to pop the symbols appearing in the input, while keeping incrementing or decrementing the state register depending on whether it sees a or b. Thus, $(r, c, r, \bar{c}) \in \Delta_{\mathsf{pop}}$ for $c \in \{a, b\}$, where the constraint $\mathsf{pop}_{rcr\bar{c}}$ is the same as $\mathsf{push}_{qcq\bar{c}}$ above. Finally, on seeing the bottom-of-stack symbol $\perp$, the automaton can pop it only if its current register equals the time value on the stack, and go to the accepting state s: $(r, \varepsilon, s, \perp) \in \Delta_{\mathsf{pop}}$, where $\mathsf{pop}_{r\varepsilon s\perp}(x, z, x', y) \equiv (x = y)$.

We conclude this section by mentioning that more general models than trPDA can be considered, for which however the non-emptiness problem becomes undecidable. One such example is obtained if we allow push transitions *to read the top of the stack*, for which we have shown that non-emptiness is undecidable, already for dimension $k = 1$ [9] [5]. Intuitively, one can simulate a 2-counter automaton by encoding one counter as the difference between the register in the state and the register on the top of the stack, and the other counter as the height of the stack. It is not clear whether allowing "rewrite" transitions of the form (p, a, q, α, β) (the topmost stack symbol α is rewritten into β) would lead to undecidability.

D. Outline

Our main result is a decision procedure for testing non-emptiness of trPDA. For complexity-theoretic considerations, we assume a binary representation of all numeric constants appearing in constraints.

Theorem 1. *The non-emptiness problem for trPDA is decidable in 2-*EXPTIME.

We apply the following strategy in the decidability proof. As the first step, we translate a trPDA into a system of equations over sets of integers, with an exponential blowup (cf. Sections III and IV); for the sake of completeness, a reverse translation from systems of equations to trPDA is provided in Section V. Then, we show that the non-emptiness problem for systems of equations is polynomially interreducible with the analogous problem for an extension of branching vector addition systems in dimension one (cf. Sections VI and VII). Finally, we obtain an EXPTIME decision procedure for the latter problem, by bounding exponentially the counter values in an accepting run (cf. Section VIII).

In the following, we are interested exclusively in non-emptiness of trPDA, and thus the actual recognized language will not be relevant. Therefore, we drop the input alphabet, and consider the projected transition relations $\Delta'_{\mathsf{push}}, \Delta'_{\mathsf{pop}} \subseteq Q \times Q \times \Gamma$ instead, where $(p, q, \gamma) \in \Delta'_{\mathsf{push}}$ if there exists $a \in A$ s.t. $(p, a, q, \gamma) \in \Delta_{\mathsf{push}}$ with constraint[6] (the constraint for Δ'_{pop} is analogous):

$$\mathsf{push}_{pq\gamma} \equiv \bigvee_{a \in A} \exists \mathrm{var}(\phi_a) \cdot \mathsf{push}_{paq\gamma}.$$

III. SYSTEMS OF EQUATIONS OVER SETS OF INTEGERS

We consider systems of equations of the form

$$\Delta : \begin{cases} X_1 &= t_1 \\ &\vdots \\ X_n &= t_n, \end{cases}$$

where $X_1, \dots, X_n$ are pairwise-distinct variables to be interpreted as sets of integers, and right-hand side expressions $t_1, \dots, t_n$ are built according to the following abstract syntax:

$$t ::= \{1\} \mid \{-1\} \mid X_i \mid t + t \mid t \cup t \mid t \cap \mathbb{N} \mid t \cap -\mathbb{N}$$

where "+" is interpreted element-wise as $A + B := \{a + b \mid a \in A \text{ and } b \in B\}$ for $A, B \subseteq \mathbb{Z}$, and intersection can be taken only w.r.t. $\mathbb{N}$ and $-\mathbb{N}$. A *variable assignment* μ is a function assigning to each variable X_i a subset $\mu(X_i)$ of integers. This is extended to terms in the natural way. A *solution* is a variable assignment μ satisfying every equation, i.e., $\mu(X_i) = \mu(t_i)$ for every equation $X_i = t_i$.

We are interested in the *least solution* of systems of equations, which exists since we can only build monotonic terms. Thus, equivalently we can consider systems of *inclusions*, by replacing equalities $X_i = t_i$ by inclusions $X_i \supseteq t_i$. In this case there is no need to require the variables X_i to be pairwise distinct, i.e., there may be multiple inclusions for the same left-hand side variable. By introducing extra variables, one can transform inclusions into the following restricted form:

$$\begin{array}{ll} X \supseteq \{1\} & X \supseteq Y \cap \mathbb{N} \\ X \supseteq \{-1\} & X \supseteq Y \cap -\mathbb{N} \end{array} \quad X \supseteq Y + Z. \qquad (1)$$

The *non-emptiness problem* asks, for a given system Δ of equations and a distinguished variable X therein, whether the least solution μ of Δ assigns to X a non-empty set of integers. The *membership problem* asks, given an additional integer $k \in \mathbb{Z}$ (coded in binary), whether $k \in \mu(X)$.

Example 3. We can compactly represent a singleton $\{k\}$ using only constants $\{1\}, \{-1\}$ as the least solution of Z_k in

$$Z_0 \supseteq \{1\} + \{-1\}, \quad \begin{array}{ll} Z_{2m} & \supseteq Z_m + Z_m, \\ Z_{2m+1} & \supseteq Z_m + Z_m + \{1\}. \end{array}$$

[5]This more general model, allowing popping and pushing many stack symbols in one step, was called trPDA in [9].

[6]Due to quantifier elimination, both $\mathsf{push}_{pq\gamma}$ and $\mathsf{pop}_{pq\gamma}$ can be rewritten as constraints.

Thus, we can represent a finite set $P = \{p_1, \ldots, p_n\}$ as $Z_P \supseteq Z_{p_1} \cup \cdots \cup Z_{p_n}$, a linear set $L = b + P^*$ (with bases and periods given in binary) as $X_L \supseteq Z_b \cup (X_L + Z_P)$, and a semilinear set $S = \bigcup_{i=1}^{n} L_i$ as $X_S \supseteq X_{L_1} \cup \cdots \cup X_{L_n}$.

The use of intersection is assumed to be very limited. Unrestricted intersections of the form $X \cap Y$ immediately lead to undecidability of non-emptiness. Indeed, already equations over $\mathbb{N}$ with unrestricted intersection subsume unary conjunctive grammars by replacing terminals with $\{1\}$ and concatenation with $+$, and the latter model has undecidable non-emptiness [14]. On the other hand, non-emptiness of system of equations without intersections can be solved in PTIME by constructing a context-free grammar, which is obtained by interpreting 1 and -1 as terminal symbols, and by replacing sum with concatenation [9]. However, the membership problem—which reduces to check whether 0 belongs to the least solution—is already NP-complete even without intersections, since now one really needs to count 1's and -1's [9]. Finally, allowing intersections with singleton constants makes both non-emptiness and membership NP-complete [9].

We show that decidability of system of equations is preserved even if we allow intersections with $\mathbb{N}$ and $-\mathbb{N}$.

Theorem 2. *Non-emptiness of systems of equations with intersections with $\mathbb{N}$ and $-\mathbb{N}$ is decidable in* EXPTIME.

The proof of the theorem follows from the PTIME reduction to reachability in 1-BVASS$^{\pm}$ shown in Sec. VII, and the fact that the latter problem is in EXPTIME by Theorem 6.

IV. FROM TRPDA TO SYSTEMS OF EQUATIONS

This section is devoted entirely to the proof of the following theorem (recall that we assume binary representations of integer constants appearing in the constraints):

Theorem 3. *The non-emptiness problem of trPDA reduces, in exponential time, to the same problem of systems of equations.*

This is achieved in several steps, which we now outline. First, we decompose rational time values into the (discrete) integer part and the (continuous) fractional part (Sec. IV-A). This is beneficial, since control state registers, as well as registers on top of the stack, have bounded span (cf. Proposition 1), and thus we can represent many integer values by a *single* integer plus a bounded distance, which is encoded in the state. In this way, we reduce the integer part to dimension one (Sec. IV-B). Since we have now only one integer in the current control location, one in the top of the stack, and one in the next control location, it follows that transitions are in dimension *three* w.r.t. the integer part. We define a convenient normal form for these transitions in dimension three, called *triangles* (Sec. IV-C), which we can further simplify (Sec. IV-D and IV-E). Finally, once the structure of triangles is at its simplest, we can write a system of equations encoding reachability in the original trPDA (Sec. IV-F).

A. Product structure

Instead of dense time structure $(\mathbb{Q}, \leqslant, +1)$, we prefer to work in the product structure $\mathbb{T}$ defined below. This allows us, roughly speaking, to deal with the integer and fractional parts separately. The factorization of a rational number into the integer and fractional part gives rise to a bijection between $\mathbb{Q}$ and $\mathbb{Z} \times \mathbb{I}$, for $\mathbb{I} = [0, 1) \cap \mathbb{Q}$. Consider the product of the integer part structure $(\mathbb{Z}, \leqslant, +1)$ with the fractional part structure $(\mathbb{I}, \leqslant)$:

$$(\mathbb{Z} \times \mathbb{I}, \leqslant_{\mathbb{Z}}, \leqslant_{\mathbb{I}}, +1)\,, \text{ where} \tag{2}$$

$$\begin{aligned} (z, q) \leqslant_{\mathbb{Z}} (z', q') \quad &\text{iff} \quad z \leqslant z', \\ (z, q) \leqslant_{\mathbb{I}} (z', q') \quad &\text{iff} \quad q \leqslant q', \text{ and} \\ (z, q) + 1 \quad &= \quad (z + 1, q), \end{aligned} \tag{3}$$

Using relations (3) one can define an isomorphic copy of $(\mathbb{Q}, \leqslant, +1)$ inside $\mathbb{Z} \times \mathbb{I}$, i.e., dense time is a *reduct* of (2): the $+1$ function is already present in (3), and the copy $\leqslant'$ of the order $\leqslant$ can be defined lexicographically:[7]

$$(z, q) \leqslant' (z', q') \iff z <_{\mathbb{Z}} z' \vee (z = z' \wedge q \leqslant_{\mathbb{I}} q'). \tag{4}$$

Nothing changes if we replace the half-open interval by the open one $\mathbb{I} = (0, 1)$. Indeed, up to isomorphism the formula (4) defines the same order as before, namely the unique countable infinite dense total order without end points. Finally, up to isomorphism nothing changes if $(\mathbb{I}, \leqslant)$ is replaced by $(\mathbb{Q}, \leqslant)$.

Thus, from now on we work in the product structure

$$\mathbb{T} := (\mathbb{Z}, \leqslant, +1) \times (\mathbb{Q}, \leqslant) = (\mathbb{Z} \times \mathbb{Q}, \leqslant_{\mathbb{Z}}, \leqslant_{\mathbb{Q}}, +1) \tag{5}$$

Modulo isomorphism, the subset of $\mathbb{Q}^k$ defined by a constraint $\phi(x_1, \ldots, x_k)$ is the same as the subset of $(\mathbb{Z} \times \mathbb{Q})^k$ defined by the new formula $\widehat{\phi}$, obtained by replacing every atomic formula $x \leqslant y + k$ in ϕ by $(x <_{\mathbb{Z}} y + k) \vee ((x =_{\mathbb{Z}} y + k) \wedge x \leqslant_{\mathbb{Q}} y)$. We can thus assume that every constraint ϕ in the input trPDA has been rewritten to $\widehat{\phi}$. In the sequel, we write $\leqslant$ instead of $\leqslant_{\mathbb{Q}}$ and $\leqslant_{\mathbb{Z}}$, when this does not lead to confusion.

B. Reducing the integer part to dimension one

When interpreted over $\mathbb{T}$, registers appearing in states and stack symbols are members of $(\mathbb{Z} \times \mathbb{Q})^k$, subject to the local legality constraints $\widehat{\phi_p}$ and $\widehat{\phi_\gamma}$, respectively. The sets defined by the constraints ϕ_p and ϕ_γ are orbit-finite by definition of trPDA, and thus by Proposition 1 they have at most exponential span K. We construct an equivalent (w.r.t. non-emptiness) trPDA whose integer part has dimension one by fixing for each control state (resp. stack symbol) a reference integer register x_i and by recording in the control state (resp. stack symbol) for every other integer register x_j, the difference $x_j - x_i \in \{-K, \ldots, K\}$. Fractional values are not modified and thus remain of dimension k. Therefore, states and stack symbols instead of being members of $(\mathbb{Z} \times \mathbb{Q})^k$ are from now on members of $\mathbb{Z} \times \mathbb{Q}^k$, i.e., the integer part

[7]On the other hand, the product structure (2) is not a reduct of $(\mathbb{Q}, \leqslant, +1)$; indeed, the predicate $x =_{\mathbb{I}} y$ in the product structure says that the difference between x and y is an integer, and is not definable in $(\mathbb{Q}, \leqslant, +1)$.

is reduced to dimension 1. The number of controls states and stack symbols in the resulting automaton is increased by an exponential multiplicative factor.

With this simple first step we achieve a fundamental gain: except for the one-dimensional integer part which will be treated separately, we have moved from the dense time domain $(\mathbb{Q}, \leqslant, +1)$ to the simpler (homogeneous) structure $(\mathbb{Q}, \leqslant)$. The fundamental difference between these two structures is that while $\mathbb{Q}^k$ is orbit-infinite with respect to automorphisms preserving $\leqslant$ and $+1$, it becomes orbit-finite when only $\leqslant$ is considered. Indeed, an orbit of $\mathbb{Q}^k$ with respect to automorphisms of $(\mathbb{Q}, \leqslant)$ is fully determined by consistently choosing for each pair of variables x_i, x_j whether or not $x_i \leqslant x_j$ holds. The set of these orbits, denoted $\mathsf{orbits}(\mathbb{Q}^k)$ below, is thus of exponential size with respect to k, and its finiteness will be crucial for constructing equations later.

C. Triangles

Now that the integer part is one-dimensional, we can further simplify the structure of transitions. Since values are now in $\mathbb{Z} \times \mathbb{Q}^k$, push and pop operations are described by formulas of the form $\mathsf{push}_{pq\gamma}((x_p, \vec{x}_p), (x_q, \vec{x}_q), (x_\gamma, \vec{x}_\gamma))$, which can be transformed into a disjunction of formulas of the form

$$\varphi^{\mathbb{Z}}(x_p, x_q, x_\gamma) \ \wedge \ \varphi^{\mathbb{Q}}(\vec{x}_p, \vec{x}_q, \vec{x}_\gamma),$$

for a formula $\varphi^{\mathbb{Z}}(x_p, x_q, x_\gamma)$ of discrete time atoms $(\mathbb{Z}, \leqslant, +1)$ and a formula $\varphi^{\mathbb{Q}}(\vec{x}_p, \vec{x}_q, \vec{x}_\gamma)$ of dense total order atoms $(\mathbb{Q}, \leqslant)$, both being conjunctions of atomic statements. Atomic statements appearing in $\varphi^{\mathbb{Z}}$ are upper/lower bounds on differences $x_\alpha - x_\beta$ of variables $x_\alpha, x_\beta \in \{x_p, x_q, x_\gamma\}$, $\alpha \neq \beta$, and thus $\varphi^{\mathbb{Z}}$ is equivalent to a conjunction of statements of the form $x_\alpha - x_\beta \in I$ for $I \subseteq \mathbb{Z}$ an interval. Since there are only three possible types of differences to consider ($x_q - x_p$, $x_\gamma - x_q$, and $x_p - x_\gamma$), it suffices to consider triples of intervals. A *triangle* is a triple $\langle I, J, K\rangle$ of (not necessarily bounded) intervals $I, J, K \subseteq \mathbb{Z}$, denoting the following set $[\![\langle I, J, K\rangle]\!] \subseteq \mathbb{Z}^3$:

$$\left\{(x, y, z) \in \mathbb{Z}^3 \;\middle|\; (y - x) \in I \wedge (z - y) \in J \wedge (x - z) \in K\right\}.$$

From now on push and pop operations will be described by disjunctions of *clauses* of the form

$$(\langle I, J, K\rangle, \varphi^{\mathbb{Q}}), \text{ where } [\![\varphi^{\mathbb{Q}}]\!] \subseteq \mathbb{Q}^{3k}. \tag{6}$$

The interval I specifies the difference between the new control state and the previous one, J specifies the difference between the topmost stack symbol and the new control state, and K specifies the difference between the previous control state and the stack symbol. Transforming push and pop formulas into disjunction of clauses increases the size by an exponential multiplicative factor.

D. Redundant triangles

We now proceed to show that triangles of a very special form suffice. A triangle $\langle I, J, K\rangle$ is *redundant* if at least one of the intervals I, J, K equals $\mathbb{Z}$. We show that *every* triangle is equivalent to a finite union of redundant triangles, in fact linearly many w.r.t. the maximal (absolute values of) integers defining I, J, K, thus at most exponential.

First, we notice that we can always apply the following "strengthening" of intervals, by intersecting each interval with the opposite of the sum of the other two: A triangle $\langle I, J, K\rangle$ is equivalent to the triangle $\langle I', J', K'\rangle$, i.e., $[\![\langle I, J, K\rangle]\!] = [\![\langle I', J', K'\rangle]\!]$, where $I' = I \cap (-(J + K))$, $J' = J \cap (-(I + K))$, and $K' = K \cap (-(I + J))$. Strengthening results in a triangle $\langle I, J, K\rangle$ which satisfies

$$I \subseteq -(J + K), J \subseteq -(I + K), K \subseteq -(I + J). \tag{7}$$

When any of the inclusions above is an equality, the corresponding interval can be deduced from the other two, and we can replace this interval with $\mathbb{Z}$ to obtain an equivalent redundant triangle. For example, if $I = -(J + K)$, then $\langle I, J, K\rangle$ is equivalent to $\langle \mathbb{Z}, J, K\rangle$. We call this the "redundancy principle". It is possible that all inclusions in (7) are strict: For example, take $I = [2, 4]$, $J = [6, \infty)$, and $K = (-\infty, -9]$, and we immediately have $-(I + J) = (-\infty, -8]$, $-(I + K) = [5, \infty)$, and $-(J + K) = \mathbb{Z}$. However, when all inclusions in (7) are strict, we can nonetheless split one of the three intervals into a finite disjoint union of intervals for which some inclusions in (7) become equalities.

Fact 1. *A non-empty triangle is equivalent to a finite union of redundant triangles.*

E. Restricted triangles

A triangle $\langle I, J, K\rangle$ is *restricted* if at most two vertices change at a time, i.e., if one of the following conditions holds:

- $I = \{0\}$ (and consequently $J = -K$ by strengthening),
- $J = \{0\}$ (and consequently $K = -I$ by strengthening),
- $K = \{0\}$ (and consequently $I = -J$ by strengthening).

A trPDA is *restricted* if its push and pop transitions use constraints containing only restricted triangles. While triangles (even redundant ones) are in general *not* equivalent to boolean combinations of restricted triangles, we can nonetheless show that push and pop operations with redundant triangles can be simulated by short sequences of operations with restricted triangles. We show it for push operations; a similar construction can be given to simulate pop operations by restricted pop operations. We use auxiliary intermediate control states written $\bullet$ (different for each simulated transition). We also use nop transitions of the form nop_{pq} consisting of a single clause $(I, \varphi^{\mathbb{Q}})$ where the semantics is that the register $(x, \vec{x}) \in \mathbb{T}$ in the previous control state is related to the new one $(y, \vec{y}) \in \mathbb{T}$ iff $y - x \in I$ and $(\vec{x}, \vec{y}) \in [\![\varphi^{\mathbb{Q}}]\!]$; these additional nop operations can later be removed by pushing and immediately popping a dummy symbol on the stack (with any time value), which can clearly be achieved by restricted operations. Consider one of the clauses $(\langle I, J, K\rangle, \varphi^{\mathbb{Q}})$ of $\mathsf{push}_{pq\gamma}$ with redundant triangle $\langle I, J, K\rangle$. There are three cases to consider:

a) $\langle I, J, \mathbb{Z}\rangle$: The difference between the register in the previous control state and in the stack symbol is determined by the other two differences. Execute a nop $\mathsf{nop}_{p\bullet} \equiv (I, (\vec{x} = \vec{y}))$ (only the integer register in the control state changes) followed by a restricted push $\mathsf{push}_{\bullet q\gamma} \equiv (\langle\{0\}, -J, J\rangle, \varphi^{\mathbb{Q}})$ (the integer register in the control state does not change).

b) $\langle I, \mathbb{Z}, K\rangle$: The difference between the register in the new control state and in the stack symbol is determined by the other two differences. Execute a restricted push $\mathsf{push}_{p\bullet\gamma} \equiv (\langle\{0\}, -K, K\rangle, \varphi^{\mathbb{Q}})$ (the integer register in the control state does not change) followed by a nop $\mathsf{nop}_{\bullet q} \equiv (I, (\vec{x} = \vec{y}))$ (only the integer register in the control state changes).

c) $\langle\mathbb{Z}, J, K\rangle$: The difference between the register in the previous state and the new one is determined by the other two differences. Execute a restricted push $\mathsf{push}_{p\bullet\gamma} \equiv (\langle -K, \{0\}, K\rangle, \varphi^{\mathbb{Q}})$ (the integer register in the next control state is the same as the integer register pushed on the stack) followed by a nop $\mathsf{nop}_{\bullet q} \equiv (-J, (\vec{x} = \vec{y}))$.

We have thus transformed a trPDA into a restricted one; this will be useful in the proof of soundness of Lemma 2 below.

Remark 2. The transformations described by now are all effective. The first three of them, namely reducing the integer part to dimension one, transforming push and pop formulas into triangles, and finally decomposing the triangles into redundant ones, can all be achieved at the total cost of a single exponential blowup. The last transformation, namely reducing the triangles to restricted ones, is polynomial.

F. System of equations

We are now ready to complete the reduction by defining a system of equations Δ (actually, we will define inclusions). In order to do this, we define the following convenient reachability relation: For two states $(p, \vec{x}), (q, \vec{y})$, let

$$(p, \vec{x}) \rightsquigarrow (q, \vec{y})$$

if there is a run from $(p, \vec{x})$ to $(q, \vec{y})$ starting and ending with empty stack. The following lemma characterizes $\rightsquigarrow$.

Lemma 1. *The relation $\rightsquigarrow$ is the least relation satisfying*

$$\text{(base)} \qquad \frac{}{(p, \vec{x}) \rightsquigarrow (p, \vec{x})}$$

$$\text{(transitivity)} \qquad \frac{(p, \vec{x}) \rightsquigarrow (r, \vec{z}) \quad (r, \vec{z}) \rightsquigarrow (q, \vec{y})}{(p, \vec{x}) \rightsquigarrow (q, \vec{y})}$$

$$\text{(push-pop)} \qquad \frac{(r, \vec{z}) \rightsquigarrow (s, \vec{t})}{(p, \vec{x}) \rightsquigarrow (q, \vec{y})} \quad (\vec{x}, \vec{z}, \vec{t}, \vec{y}) \in [\![\mathsf{push\text{-}pop}_{prsq}]\!]$$

where $\mathsf{push\text{-}pop}_{prsq} \equiv \bigvee_{\gamma\in\Gamma} \exists \mathsf{var}(\phi_\gamma) \cdot \mathsf{push}_{pr\gamma} \wedge \mathsf{pop}_{sq\gamma}$.

We are now ready to define the system Δ. There is a variable X_{pqo} for each pair p, q of control states and for each of the finitely many orbits $o \in \mathsf{orbits}(\mathbb{Q}^{2k})$. This variable represents the difference between the integer value of the starting state p and the ending state q along runs starting and ending with empty stack; moreover, the fractional values at p and q are related as specified by o (cf. Lemma 2 below). Recall that the orbits are with respect to the automorphisms of $(\mathbb{Q}, \leqslant)$. As the number of states has only grown exponentially during the previous transformations, and the number of orbits is exponential in the dimension k, the total number of variables

$$\begin{tikzcd}
& r \arrow[dr, "J"] & & s \arrow[ddr, "L"] & \\
& & \gamma \arrow[dll, "K"] \arrow[ur, "N"] & & \\
p \arrow[uur, "I"] & & & & q \arrow[ull, "M"]
\end{tikzcd}$$

Fig. 1: Illustration for push-pop inclusions.

X_{pqo} is exponential as well. Following the characterization of $\rightsquigarrow$ by Lemma 1, for every control state p and for every diagonal orbit $o \in \mathsf{orbits}(\{(\vec{x}, \vec{x}) \mid \vec{x} \in \mathbb{Q}^k\})$, the system Δ contains the following inclusion:

$$\text{(base)} \qquad X_{ppo} \supseteq \{0\}\,.$$

For every control states p, r, q and for every orbit $o \in \mathsf{orbits}(\mathbb{Q}^{3k})$, the system Δ contains the following inclusion:

$$\text{(transitivity)} \quad X_{pqo_{13}} \supseteq X_{pro_{12}} + X_{rqo_{23}},$$

where o_{ij} is the projection to components i, j of the orbit o.

The push-pop inclusions are the most interesting. Recall that push and pop formulas $\mathsf{push}_{pq\gamma}$ and $\mathsf{pop}_{pq\gamma}$ have been transformed into a disjunction of clauses of the form $(\langle I, J, K\rangle, \varphi^{\mathbb{Q}})$; cf. (6). For any control states p, q, r, s and a stack symbol γ, consider one clause of $\mathsf{push}_{pr\gamma}$ and one clause of $\mathsf{pop}_{sq\gamma}$ (cf. Fig. 1),

$$(\langle I, J, K\rangle, \varphi^{\mathbb{Q}}_{\mathsf{push}}) \quad \text{and} \quad (\langle L, M, N\rangle, \varphi^{\mathbb{Q}}_{\mathsf{pop}}), \tag{8}$$

and consider also one of the finitely many orbits $o \in \mathsf{orbits}([\![\exists \vec{u} \cdot \varphi^{\mathbb{Q}}_{\mathsf{push}} \wedge \varphi^{\mathbb{Q}}_{\mathsf{pop}}]\!])$. For each of these choices, Δ contains the following inclusion (push-pop):

$$X_{pqo_{14}} \supseteq (I + (X_{rso_{23}} \cap (J+N)) + L) \cap -(K+M). \tag{9}$$

The correctness of the construction of Δ follows by the lemma below. Its proof strongly relies on the fact that the trPDA is restricted (cf. Sec IV-E).

Lemma 2. *Let $\mathcal{P}$ a restricted trPDA, and let μ be the least solution of the system Δ. For every X_{pqo}, let*

$$\widehat{X}^{\exists}_{pqo} := \{y - x \in \mathbb{Z} \mid \exists (\vec{x}, \vec{y}) \in o \cdot (p, (x, \vec{x})) \rightsquigarrow (q, (y, \vec{y}))\}\,,$$
$$\widehat{X}^{\forall}_{pqo} := \{y - x \in \mathbb{Z} \mid \forall (\vec{x}, \vec{y}) \in o \cdot (p, (x, \vec{x})) \rightsquigarrow (q, (y, \vec{y}))\}\,.$$

Then, $\widehat{X}^{\exists}_{pqo} = \widehat{X}^{\forall}_{pqo} = \mu(X_{pqo})$.

Thus, we immediately deduce that the trPDA is non-empty if, and only if, some variable X_{pqo} of Δ with $p \in I$ and $q \in F$ is non-empty (recall that a trPDA starts and accepts with the empty stack). This proves the correctness of our reduction.

V. From systems of equations to trPDA

In this section we provide a PTime reduction in the direction opposite than in the previous section. trPDA of dimension 1 already capture least solutions of system of equations.

Theorem 4. *The non-emptiness problem of systems of equations reduces, in polynomial time, to the same problem for trPDA of dimension 1.*

Consider a restricted system of inclusions (1) with a distinguished variable X_0 and μ its least solution. We reduce non-emptiness of $\mu(X_0)$ to non-emptiness of a 1-dimensional trPDA $\mathcal{P}$ of polynomial size over discrete time $(\mathbb{Z}, \leqslant, +1)$. Since dense time is more general than discrete time, the same construction also yields a trPDA over dense time.

For every variable X of Δ there is a corresponding stack symbol X of dimension 0 and we have two additional stack symbols $\{\mathbb{N}, -\mathbb{N}\}$ of dimension 0 in order to simulate tests. For convenience assume $\mathcal{P}$ starts its execution in control state p (which is also final), with stack containing one symbol X_0, thus the initial configuration is $((p, x), X_0)$ for some $x \in \mathbb{Z}$. $\mathcal{P}$ simulates the least solution μ in the following sense: for every $x, y \in \mathbb{Z}$ and variable (stack symbol) X, $((p, y), \varepsilon)$ is reachable from $((p, x), X)$ iff $(y - x) \in \mu(X)$. Therefore, non-emptiness of Δ reduces to non-emptiness of $\mathcal{P}$ (recall that $\mathcal{P}$ accepts when the stack is empty).

We now describe the construction of the transitions of $\mathcal{P}$. The evolution of the automaton is driven by the contents of the topmost stack symbol. For every inclusion of the form $X \supseteq \{k\}$ we pop the topmost untimed stack symbol X and we increment the local register by k:

$$\mathsf{pop}_{ppX}(x, y) \equiv (y = x + k).$$

For every inclusion of the form $X \supseteq Y \cap N$ with $N \in \{\mathbb{N}, -\mathbb{N}\}$ we pop X, we push the constraint obligation (N, x) where x is the current value of the local register, and we then push Y; the register does not change:

$$\mathsf{pop}_{pp'X}(x, y) \equiv (y = x)$$
$$\mathsf{push}_{p'p''N}(x, y, z) \equiv (z = y = x)$$
$$\mathsf{push}_{p''pY}(x, y) \equiv (y = x).$$

For every inclusion of the form $X \supseteq Y + Z$ we first pop X, and we then push Y and Z, without changing the value of the register in the control:

$$\mathsf{pop}_{pp'X}(x, y) \equiv \mathsf{push}_{p'p''Y}(x, y) \equiv \mathsf{push}_{p''pZ}(x, y) \equiv (y = x).$$

Finally, if we encounter a test obligation (N, z) with $N \in \{\mathbb{N}, -\mathbb{N}\}$ on top of the stack, we pop (N, z) and we check that $x - z \in N$, where x is the current value of the local register; again, the local register does not change:

$$\mathsf{pop}_{pp\mathbb{N}}(x, y, z) \equiv (y = x \wedge x \geqslant z)$$
$$\mathsf{pop}_{pp(-\mathbb{N})}(x, y, z) \equiv (y = x \wedge x \leqslant z).$$

In the description above, the states p', p'' are fresh and depend on the inclusion being simulated. Consequently, the number of control states in $\mathcal{P}$ is polynomial w.r.t. the size of Δ. and thus all together the size of $\mathcal{P}$ is polynomial as well.

VI. Extension of 1-dimensional branching VASS

In this section we define the last model used in this paper, namely 1-dimensional branching vector addition systems with states, addition, and *subtraction*, denoted shortly 1-BVASS$^{\pm}$. They are triples $B = (Q, q_0, \Delta)$, where Q is a finite set of states, q_0 is a distinguished *leaf state*, and $\Delta \subseteq Q^3$ is the transition relation. The intended meaning of a transition $(q, q_l, q_r) \in \Delta$ is that q is the parent and q_l, q_r are respectively the left and the right child. For simplicity we assume that q_0 is never a parent, i.e., $q \neq q_0$. We assume that the set of non-leaf states is partitioned $Q \backslash \{q_0\} = Q^+ \uplus Q^-$ into addition states Q^+ and subtraction states Q^-.

Such systems give rise to binary ordered trees whose nodes are labeled with pairs (q, n), where $q \in Q$ and $n \in \mathbb{N}$. We shall refer to q and n as the state and natural label of a tree node, respectively. A node with state label $q \in Q^+$ is an addition nodes, if $q \in Q^-$ a subtraction node. We only consider finite trees t, where every internal (non-leaf) node v has two children; we shall refer to them as the left child and the right child of v, respectively. The right child of a subtraction node is called a *subtrahend node*, and the subtree of t rooted at subtrahend node is called *subtrahend subtree*. Similarly, on the left we have the *minuend node* and the *minuend subtree*. For a node v in a tree t, by t_v we denote the subtree of t rooted at the node v.

A labelled tree is a *witness* provided that:

- every leaf is labeled by $(q_0, 1)$[8];
- for every internal node v, let (q, n), (q_l, n_l) and (q_r, n_r) be labels of v, its left child and its right child, respectively; then $q \neq q_0$ and $(q, q_l, q_r) \in \Delta$ and $n = n_l + n_r$ if $q \in Q^+$ and $n = n_l - n_r$ if $q \in Q^-$.

It is convenient to think that the direction of the computation is bottom-up because it reflects the values of parent nodes. Notice that if $q \in Q^-$ then $n_l - n_r \in \mathbb{N}$ and thus $n_l \geqslant n_r$. If the root node of t is labeled by (q, n) then we say that t is a (q, n)-*witness* and (q, n) is the *root label* of t; we also define *the value of* t as the natural label n of the root node. The reachability problem asks, for a given 1-BVASS$^{\pm}$ B and a pair $(q, n) \in Q \times \mathbb{N}$, whether B admits a (q, n)-witness. As we argue below, w.l.o.g. one may assume $n = 0$.

Remark 3. In the standard model of 1-dimensional branching VASS (cf. e.g. [13]) there are no subtraction states. On the other hand, our model does not have unary transitions that add a fixed integer value to the counter, as such transitions can be easily simulated (see Example 4 below). It is also not hard to see that the encoding of the target counter value (binary vs unary) does not make a difference in terms of complexity of the reachability problem for 1-BVASS$^{\pm}$.

Example 4. We show how 1-BVASS$^{\pm}$ can succinctly encode natural numbers. For $m \geqslant 0$, we define a 1-BVASS$^{\pm}$ B_m of size logarithmic in m, and a distinguished state p_m, such that B_m admits exactly one witnessing tree with root node labeled by state p_m, and the value of this tree is m. For $m = 0$ we define a state $p_0 \in Q^-$ with one transition $(p_0, q_0, q_0) \in \Delta$. For any other $m > 0$, the construction relies on binary encoding of m, similarly as in Example 3. Formally, we define a state

[8]Usually leaves in BVASS models are labeled with 0. Since this model does not have explicit unary transitions (see Remark 3) this change is important.

$p_m \in Q^+$ and the following transitions:

$$\begin{array}{ll} (p_{2k}, p_k, p_k) & \text{if } m = 2k; \\ (p_{2k+1}, p_k, \bullet), (\bullet, p_k, q_0) & \text{if } m = 2k+1, \end{array}$$

where $\bullet$ is a fresh intermediate state.

Using B_m as a gadget one can simulate unary transitions by adding or subtracting m, even when m is encoded in binary. Moreover, we can also simulate unary tests "$\geqslant m$" and "$\leqslant m$" (and thus "$= m$") with m encoded in binary by removing m and then adding it back (for "$\geqslant m$"), or by taking the complement to m two times in a row (for "$\leqslant m$"). Finally, introducing an auxiliary subtraction step at the very end, w.l.o.g. we may assume the target counter value in the reachability problem to be 0.

Remark 4. Another closely related formalism are bounded one-counter automata [11], which are essentially 1-dimensional VASS (without branching), where the values in the counter are bounded by some fixed value N (encoded in binary). We show that these model can be simulated by 1-BVASS$^\pm$. Adding and subtracting values (encoded in binary) can be simulated as already described in Example 4. Moreover, to guarantee that the value in the counter does not exceed N one can add after every transition a test "$\leqslant N$"; cf. Example 4. Using this reduction, one can easily simulate bounded one-counter automata that are known to be PSPACE-complete [11], which yields a direct proof that the reachability problem is PSPACE-hard for 1-BVASS$^\pm$.

State of the art. As already mentioned in the introduction, the reachability problem for the standard BVASS model in dimension one is known to be PTIME-complete for unary encoding [13], and was recently shown to be PSPACE-complete for binary encoding [12]. Therefore, according to Remark 3, the reachability problem is PSPACE-hard for 1-BVASS$^\pm$ as well. When BVASS is extended with subtraction reachability becomes undecidable already in dimension six [17]. In the next section we show that systems of equations and 1-BVASS$^\pm$ are PTIME-interreducible, and then we establish in Sec. VIII that 1-BVASS$^\pm$ are decidable in EXPTIME.

VII. Systems of equations and 1-BVASS$^\pm$

Theorem 5. *The non-emptiness problem for systems of equations and the reachability problem for 1-BVASS$^\pm$ are* PTIME-*interreducible.*

While equations manipulate integers, 1-BVASS$^\pm$ manipulate natural numbers. A 1-BVASS$^\pm$ $B = (Q, q_0, \Delta)$ is simulated by a system of equations as follows. For each state $q \in Q$, we have a variable X_q storing the values in state q and their opposites. For the leaf state q_0 we have the inclusion $X_{q_0} \supseteq \{1, -1\}$, and for every transition $(q, r, s) \in \Delta$ we have

$$q \in Q^+ : X_q \supseteq (X_r \cap \mathbb{N} + X_s \cap \mathbb{N}) \cup (X_r \cap (-\mathbb{N}) + X_s \cap (-\mathbb{N})),$$
$$q \in Q^- : X_q \supseteq (X_r + X_s \cap (-\mathbb{N})) \cap \mathbb{N} \cup (X_r + X_s \cap \mathbb{N}) \cap (-\mathbb{N}).$$

It is immediate to verify that, if μ is the least solution of the system thus constructed, then (q, n) is reachable in B if, and only if, $0 \leqslant n \in \mu(X_q)$. Moreover, the size of the system of inclusions is polynomial in the size of the 1-BVASS$^\pm$.

Consider now a system of equations using restricted rules as in (1). It will be more convenient to work with a different definition of 1-BVASS$^\pm$. In the original definition the operations (sum and subtraction) were encoded in the states. We consider *transition* 1-BVASS$^\pm$ $B = (Q, q_0, \Delta^=, \Delta^+, \Delta^-)$, with unary rules $(p, q) \in \Delta^=$ just copying the value from p to q, and binary rules Δ^+ and Δ^- computing the sum and difference of values. Notice that in this definition the same state can be used in both addition and subtraction rules. In witnesses the internal nodes have a single child if a rule from $\Delta^=$ is used, and the leaf state q_0 has value 1. It is immediate to see that a transition 1-BVASS$^\pm$ as above can be put in the conventional form (Q, q_0, Δ) by doubling control states, but we use the transition form to ease the construction below.

For each variable X we have two states $q_X^+, q_X^- \in Q$ representing, intuitively, the positive and the opposite of the negative part of X. For each inclusion $X \supseteq \{1\}$ we have a transition $(q_X^+, q_0) \in \Delta^=$, and for each inclusion $X \supseteq \{-1\}$ we have a transition $(q_X^-, q_0) \in \Delta^=$. For each inclusion $X \supseteq Y \cap \mathbb{N}$ we have a transition $(q_X^+, q_Y^+) \in \Delta^=$, and for each inclusion $X \supseteq Y \cap (-\mathbb{N})$ we have a transition $(q_X^-, q_Y^-) \in \Delta^=$. Finally, for each inclusion $X \supseteq Y + Z$ we have the following three transitions rooted in q_X^+:

$$(q_X^+, q_Y^+, q_Z^+) \in \Delta^+ \text{ and } (q_X^+, q_Y^+, q_Z^-), (q_X^+, q_Z^+, q_Y^-) \in \Delta^-,$$

as well as the following three transitions rooted in q_X^-:

$$(q_X^-, q_Y^-, q_Z^-) \in \Delta^+ \text{ and } (q_X^-, q_Y^-, q_Z^+), (q_X^-, q_Z^-, q_Y^+) \in \Delta^-.$$

It is easy to verify that $x \in \mu(X)$ if, and only if, either $x \geqslant 0$ and (q_X^+, x) is reachable, or $x \leqslant 0$ and $(q_X^-, -x)$ is reachable. Moreover, the size of the transition 1-BVASS$^\pm$ thus constructed is polynomial in the size of the system of equations, and the same complexity is preserved even when translating it into a 1-BVASS$^\pm$.

VIII. ExpTime decision procedure for 1-BVASS$^\pm$

This section is devoted to proving our last result.

Theorem 6. *Reachability of 1-BVASS$^\pm$ is in* EXPTIME.

The complexity is w.r.t. the size of a 1-BVASS$^\pm$, understood as the number of its transitions. Theorem 6 is easily inferred from the following small witness property:

Lemma 3. *If a 1-BVASS$^\pm$ B admits a $(q, 0)$-witness, then it admits also a $(q, 0)$-witness with all natural labels bounded by a constant N exponential in the size of B.*

Indeed, Lemma 3 implies Theorem 6: the existence of a bounded $(q, 0)$-witness is reducible to non-emptiness of a tree automaton with states $Q \times \{0 \dots N\}$, and with $(q_0, 1)$ and $(q, 0)$ as the initial and the final state, respectively. The size of the tree automaton is exponential, thus the polynomial-time complexity of non-emptiness of tree automata [10] yields the desired upper bound.

In the rest of this section we prove Lemma 3. The general idea is to show that if a $(q, 0)$-witness contains a node with natural label larger than N then one can remove some parts thereof to obtain a $(q, 0)$-witness of smaller size (i.e., having less nodes). Fix a 1-BVASS$^{\pm}$ $B = (Q, q_0, \Delta)$. We will use the following constants:

$$M_1 = |Q| \cdot 2^{|Q|}, \text{ and}$$
$$M_2 = M_1 + 2^{3|Q|}.$$

Our proof is based on two cornerstone Propositions 2 and 3.

Proposition 2. *Let t be a (q, m)-witness, for $m > M_1$. There exists a multiset $\{m_1, \ldots, m_k\}$ of natural numbers such that*

(1) $k \geqslant (m - M_1)/2^{|Q|}$;
(2) $m_i \leqslant 2^{|Q|}$ for all i;
(3) for every non-empty subset of indices $X \subseteq \{1, \ldots, k\}$, there is a (q, m')-witness t', strictly smaller than t, where $m' = m - \sum_{i \in X} m_i$;

Consequently, $m_1 + \ldots + m_k \leqslant m$. We call the multiset $\{m_1, \ldots, m_k\}$ *a downward decomposition of* t. Note that we work with multisets, where each element has its cardinality; in particular, when writing $\{m_1, \ldots, m_k\}$ we allow for $m_i = m_j$. Intuitively, the numbers m_i are values of some subtrees t_i in t that are pairwise disjoint. The conditions in Proposition 2 correspond to: (1) the number of subtrees is big enough; (2) the values m_i are small enough; (3) every subset of subtrees t_i's can be safely removed from the tree and the value of the resulting witness has decreased by the sum of all the removed values.

Proposition 2 allows us to bound the values of subtrahend subtrees by M_2. Intuitively, when the subtrahend is larger than M_2—and thus the same holds for the minuend—, we can decrease both these two subtrees without changing the value of the subtraction node.

Lemma 4. *Suppose that some of subtrahend subtrees of a witness t has value larger that M_2. Then, there exists a strictly smaller witness t' with the same root label as t.*

Proof. Suppose that a subtree of t rooted in a subtrahend node v_r has value larger than M_2. Let v be the parent of v_r, and v_l its sibling (i.e., left child of v). Let (q_l, n_l) and (q_r, n_r) be the labels of v_l and v_r, respectively, and let t_l and t_r denote the subtrees of t rooted at v_l and v_r, respectively. Since v is a subtraction node, its natural label is $n_l - n_r$ with $n_l \geqslant n_r$. Thus $n_l, n_r > M_2$.

We apply Proposition 2 simultaneously to the two subtrees rooted in v_l and v_r, and obtain their respective downward decompositions $\{m_1^l, \ldots, m_{k_l}^l\}$ and $\{m_1^r, \ldots, m_{k_r}^r\}$. Since $n_l, n_r > M_2$, we know that

$$k_l, k_r > (M_2 - M_1)/2^{|Q|} \geqslant 2^{2|Q|}.$$

By the pigeonhole principle, some value $c \leqslant 2^{|Q|}$ appears in the multiset $\{m_1^l, \ldots, m_{k_l}^l\}$ at least $2^{|Q|}$ times. Similarly, some value $d \leqslant 2^{|Q|}$ appears in the multiset $\{m_1^r, \ldots, m_{k_r}^r\}$ at least $2^{|Q|}$ times. Consider d appearances of c in the former multiset, and symmetrically c appearances of d in the latter one. By Proposition 2(3), there is a $(q_l, n_l - c \cdot d)$-witness u_l strictly smaller than t_l, and similarly there is a $(q_r, n_r - c \cdot d)$-witness u_r, strictly smaller than t_r. The tree t' obtained from t by replacing t_l with u_l and t_r with u_r is thus strictly smaller than t but with the same value; indeed, the value of the subtree rooted in v remains unchanged. This completes the proof. $\square$

In order to state the second cornerstone (Proposition 3 below) we need to introduce some notation. For a witness t, let $\mathrm{ss}(t)$ denote the set of all subtrahend subtrees of t. Being subtrees of t, two subtrahend subtrees are either contained one in another, or disjoint. A node of t is *positive* if it does not belong to any subtrahend subtree of t. In other words, on the path from the root to a positive node we never pass through a subtrahend node, and consequently, if we increase the value at a positive node, then the value of the root increases too; moreover, these increases can be arbitrary. The *positive part* of t, denoted t^+, contains the positive nodes of t; it is thus obtained by removing all (maximal) subtrahend subtrees $s \in \mathrm{ss}(t)$. Notice that we remove at most one of the subtrees of an internal node, therefore a leaf in t^+ is a leaf in t. Consider the positive part t^+ of t and the positive parts of all subtrahend subtrees of t. Their sets of nodes are disjoint and sum up to the set of nodes of t. We may thus say that $\{t^+\} \cup \{s^+ \mid s \in \mathrm{ss}(t)\}$ forms a partition of t.

Consider a (p, m)-witness t and a node $v \in t^+$ in its positive part, labelled by (q, n). The $(q, n), (p, m)$*-witness-context*, denoted $t.v$, is obtained from t by replacing the subtree rooted at v by a single leaf labelled by (q, n). Thus the leaf v of a witness-context $t.v$ may have label (q, n) different from the label $(q_0, 1)$ of all other leaves. The following is the second cornerstone of our proof.

Proposition 3. *Let $t.v$ be a $(q, n), (p, m)$-witness-context, for $n > M_2 \geqslant m$. Suppose that all subtrahend subtrees of t have value at most M_2. There exists a multiset $\{n_1, \ldots, n_k\}$ of natural numbers such that*

(1) $k \geqslant (n - M_2)/(M_2 \cdot (|Q| + 1))$;
(2) $n_i \leqslant |Q| \cdot M_2$ for all i;
(3) for every non-empty subset $X \subseteq \{1, \ldots, k\}$, there is a $(q, n'), (p, m)$-witness-context $t'.v$ strictly smaller than $t.v$, where $n' = n - \sum_{i \in X} n_i$.

Implicitly $n_1 + \ldots + n_k \leqslant n$, as above. The multiset $\{n_1, \ldots, n_k\}$ we call *an upward decomposition of $t.v$*. We now use the two cornerstone propositions to prove Lemma 3.

Proof of Lemma 3. Fix a witness t with value 0, of minimal size. We aim at showing that all natural labels in t are bounded by N, for a sufficiently large constant N exponential in the size of B, which will be made explicit below. Towards contradiction, suppose that some node v in t labelled with (q, n) has $n > N$.

Let $s \in \{t\} \cup \mathrm{ss}(t)$ be the unique tree such that $v \in s^+$, i.e., v belongs to the positive part of s. Let (p, m) be the root label of s. Note that $m \leqslant M_2$; indeed, if $s \in \mathrm{ss}(t)$ this follows by Lemma 4 and minimality of t, and if $s = t$ we have $m = 0$. We will arrive at contradiction if we show that s can be replaced with a strictly smaller (p, m)-witness. We will work with two separate parts of s: the $(q, n), (p, m)$-witness-context $s.v$, and the (q, n)-witness s_v rooted at v.

We apply Proposition 3 to the witness-context $s.v$, and obtain its upward decomposition $\{n_1, \ldots, n_k\}$. As we want to derive

$$k \geqslant 2^{|Q|} \cdot |Q| \cdot M_2 \qquad (10)$$

from Proposition 3(1), knowing that $n > N$, we stipulate

$$N \geqslant M_2 + 2^{|Q|} \cdot |Q| \cdot (M_2)^2 \cdot (|Q| + 1). \qquad (11)$$

Similarly, we apply Proposition 2 to the (q, n)-witness s_v, and obtain its downward decomposition $\{m_1, \ldots, m_l\}$. Again, aiming at obtaining the same inequality (10) as before for the cardinality l of $\{m_1, \ldots, m_l\}$, but this time from Proposition 2(1), we stipulate

$$N \geqslant M_1 + 2^{2|Q|} \cdot |Q| \cdot M_2. \qquad (12)$$

Now we are eventually able to reveal the value of N: it can be any value satisfying the constraints (11) and (12).

The following pigeonhole argument is similar as in the proof of Lemma 4. We know that both k and l satisfy inequality (10). Thus some value $c \leqslant |Q| \cdot M_2$ necessarily appears at least $2^{|Q|}$ times in $\{n_1, \ldots, n_k\}$, and similarly some value $d \leqslant 2^{|Q|}$ appears at least $|Q| \cdot M_2$ times in $\{m_1, \ldots, m_l\}$. Consider d appearances of c in the former multiset, and c appearances of d in the latter one. Using simultaneously Propositions 2(3) and 3(3), we know that there is a $(q, n-c\cdot d)$-witness s' strictly smaller that s_v, and a $(q, n-c\cdot d), (p, m)$-witness-context s'' of the same root value (p, m) as $s.v$, but strictly smaller in size. By inserting s' at node v in s'', we obtain a (p, m)-witness strictly smaller than s, and by substituting it into t instead of s we obtain a witness strictly smaller than t. This contradicts minimality of t; the proof of Lemma 3 is thus completed. □

IX. Conclusions

We have shown decidability of the non-emptiness problem for trPDA in 2-ExpTime, via an exponential reduction to equations and then to 1-BVASS$^{\pm}$, both shown solvable in ExpTime. This answers a question raised in [9] about whether orbit-finiteness restriction on push and pop rules could be lifted while maintaining decidability; moreover, our reduction to equations is considerably more involved than in [9]. The latter question is directly related to whether in equations one can have intersections with arbitrary intervalswhile preserving decidability. We have thus answered positively both questions.

Several directions for future work can be identified. The most urgent issue is complexity. While we provide a 2-ExpTime upper-bound for trPDA, the only known lower-bound is ExpTime, already for the less expressive orbit-finite and grammar classes (cf. [9]). Regarding 1-BVASS$^{\pm}$—or, equivalently, equations—we have provided an ExpTime upper-bound, while a PSpace lower-bound can be immediately inferred by simulating bounded one-counter automata [11]. Moreover, there is a gap between our decidability result of BVASS$^{\pm}$ in dimension one, and the known undecidability in dimension six [17].

We conclude by mentioning that both solutions of equations and 1-BVASS$^{\pm}$ reachability sets are semilinear subsets of $\mathbb{Z}$ and $\mathbb{N}$, respectively. We omit the proof of this fact, which can be found in the full version of the paper.

References

[1] P. A. Abdulla, M. F. Atig, and J. Stenman. Dense-timed pushdown automata. In *Proc. LICS'12*, pages 35–44. IEEE, 2012.

[2] S. Akshay, P. Gastin, and S. N. Krishna. Analyzing timed systems using tree automata. In *Proc. CONCUR'16*, pages 27:1–27:14, 2016.

[3] R. Alur and D. L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126:183–235, 1994.

[4] M. Benerecetti, S. Minopoli, and A. Peron. Analysis of timed recursive state machines. In *Proc. TIME'10*, pages 61–68. IEEE, sept. 2010.

[5] M. Bojańczyk and S. Lasota. A machine-independent characterization of timed languages. In *Proc. ICALP'12*, volume 7392 of *LNCS*, pages 92–103. Springer, 2012.

[6] A. Bouajjani, R. Echahed, and R. Robbana. Verification of context-free timed systems using linear hybrid observers. In *Proc. CAV'94*, volume 818 of *LNCS*, pages 118–131. Springer, 1994.

[7] A. Bouajjani, R. Echahed, and R. Robbana. On the automatic verification of systems with continuous variables and unbounded discrete data structures. In *Proc. Hybrid Systems '94*, volume 999 of *LNCS*, pages 64–85. Springer, 1995.

[8] L. Clemente and S. Lasota. Reachability analysis of first-order definable pushdown systems. In *Proc. CSL'15*, volume 41 of *LIPIcs*, pages 244–259. Dagstuhl, 2015.

[9] L. Clemente and S. Lasota. Timed pushdown automata revisited. In *Proc. LICS'15*, pages 738–749. IEEE, July 2015.

[10] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: http://www.grappa.univ-lille3.fr/tata, 2007.

[11] J. Fearnley and M. Jurdziński. Reachability in two-clock timed automata is PSPACE-complete. *Inf. Comput.*, 243:26–36, 2015.

[12] D. Figueira, R. Lazić, J. Leroux, F. Mazowiecki, and G. Sutre. Polynomial-space completeness of reachability for succinct branching vass in dimension one. *In preparation*, 2017.

[13] S. Göller, C. Haase, R. Lazić, and P. Totzke. A polynomial-time algorithm for reachability in branching VASS in dimension one. In *Proc. ICALP'16*, volume 55 of *LIPIcs*. Dagstuhl, 2016.

[14] A. Jeż and A. Okhotin. Conjunctive grammars over a unary alphabet: Undecidability and unbounded growth. *Theory Comput. Syst.*, 46(1):27–58, 2010.

[15] B. Klin, E. Kopczyński, J. Ochremiak, and S. Toruńczyk. Locally finite constraint satisfaction problems. In *Proc. LICS'15*, pages 475–486. IEEE, 2015.

[16] B. Klin, S. Lasota, J. Ochremiak, and S. Toruńczyk. Homomorphism problems for first-order definable structures. In *Proc. FSTTCS'16*, volume 65 of *LIPIcs*, pages 14:1–14:15. Dagstuhl, 2016.

[17] R. Lazić. The reachability problem for branching vector addition systems requires doubly-exponential space. *Inf. Process. Lett.*, 110(17):740–745, 2010.

[18] R. Lazić and S. Schmitz. Non-elementary complexities for branching VASS, MELL, and extensions. In *Proc. CSL-LICS'14*. ACM, 2014.

[19] A. Trivedi and D. Wojtczak. Recursive timed automata. In *Proc. ATVA'10*, volume 6252 of *LNCS*, pages 306–324. Springer, 2010.