One-Parametric Presburger Arithmetic Has Quantifier Elimination

Alessio Mansutti
IMDEA Software Institute, Madrid, Spain

Mikhail R. Starchak
St. Petersburg State University, Russia

Abstract

We give a quantifier elimination procedure for *one-parametric Presburger arithmetic*, the extension of Presburger arithmetic with the function $x \mapsto t \cdot x$, where t is a fixed free variable ranging over the integers. This resolves an open problem proposed in *[Bogart et al., Discrete Analysis, 2017]*. As conjectured in *[Goodrick, Arch. Math. Logic, 2018]*, quantifier elimination is obtained for the extended structure featuring all integer division functions $x \mapsto \left\lfloor \frac{x}{f(t)} \right\rfloor$, one for each integer polynomial f.

Our algorithm works by iteratively eliminating blocks of existential quantifiers. The elimination of a block builds on two sub-procedures, both running in non-deterministic polynomial time. The first one is an adaptation of a recently developed and efficient quantifier elimination procedure for Presburger arithmetic, modified to handle formulae with coefficients over the ring $\mathbb{Z}[t]$ of univariate polynomials. The second is reminiscent of the so-called "base t division method" used by Bogart *et al.* As a result, we deduce that the satisfiability problem for the existential fragment of one-parametric Presburger arithmetic (which encompasses a broad class of non-linear integer programs) is in NP, and that the smallest solution to a satisfiable formula in this fragment is of polynomial bit size.

2012 ACM Subject Classification Computing methodologies → Symbolic and algebraic algorithms; Theory of computation → Logic

Keywords and phrases decision procedures, quantifier elimination, non-linear integer arithmetic

Digital Object Identifier 10.4230/LIPIcs.MFCS.2025.72

Related Version *Full Version*: https://doi.org/10.48550/arXiv.2506.23730

Funding *Alessio Mansutti*: Funded by the Madrid Regional Government (César Nombela grant 2023-T1/COM-29001), and by MCIN/AEI (grant PID2022-138072OB-I00).
Mikhail R. Starchak: Supported by the Russian Science Foundation, project 23-71-01041.

1 Introduction

The first-order theory of the integers $\mathbb{Z}$ with addition and order, which is also known as *Presburger arithmetic* (PrA) or linear integer arithmetic, has been intensively studied during almost a century [26]. It is a textbook fact that Presburger arithmetic admits quantifier elimination when the structure $\langle \mathbb{Z}; 0, 1, +, \leq \rangle$ is extended with the predicates $(d \mid (\cdot))_{d \in \mathbb{Z}}$ for divisibilities by fixed integers d: in the theory of this extended structure, for every quantifier-free formula $\varphi(x, \boldsymbol{y})$ there is a quantifier-free formula $\psi(\boldsymbol{y})$ that is equivalent to $\exists x : \varphi(x, \boldsymbol{y})$. The construction of ψ is effective, which implies the decidability of Presburger arithmetic. The algorithm to decide PrA is the canonical example for the notion of *quantifier elimination procedure*. The computational complexity of the many variants of this procedure has a long history, beginning with Oppen's proof [25] that Cooper's procedure [12] runs in triply exponential time, and followed by refinements from Reddy and Loveland [27], and later Weispfenning [30], which enable handling formulae with fixed quantifier alternations in doubly exponential time. Recent research on quantifier elimination aims at narrowing the complexity gap for the existential fragment (only last year it was discovered that quantifier elimination can be performed in exponential time in this case [11, 20]) and on extending

the procedure to handle additional predicates and functions [3, 11, 22, 28], or other forms of quantification [4, 10, 21]. The reader can find an extensive bibliography on Presburger arithmetic, and quantifier elimination, in the survey papers by Haase [19] and Chistikov [9].

This paper addresses the open problem raised in the papers [6] and [15] regarding the existence of a quantifier elimination procedure for the theory $\mathrm{Th}\langle\mathbb{Z};\, 0,\, 1,\, +,\, x \mapsto t \cdot x,\, \leq\rangle$, known as *one-parametric Presburger arithmetic* (PrA[t]). In this theory, the structure of Presburger arithmetic is extended with the function $x \mapsto t \cdot x$ for multiplication by a single parameter t ranging over $\mathbb{Z}$. Every PrA[t] formula $\varphi(\boldsymbol{x})$ defines a *parametric Presburger family* $\mathbb{S}(\varphi) \coloneqq \{[\![\varphi]\!]_k : k \in \mathbb{Z}\}$, where $[\![\varphi]\!]_k$ is the set of (integer) solutions of the Presburger arithmetic formula obtained from φ by replacing the parameter t with the integer k.

▶ **Example 1.** Consider the statement "for every two successive positive integers t and $t+1$, and for all integers a and b, there is an integer x in the interval $[0, t(t+1)-1]$ that is congruent to a modulo t, and to b modulo $t+1$". The truth of this sentence follows from the Chinese remainder theorem together with the fact that successive positive integers are always coprime. We can encode this statement in PrA[t] with the formula $\forall a \forall b : \chi(a, b)$, where

$$\chi \;\coloneqq\; t \geq 1 \implies \exists x :\; 0 \leq x \wedge x \leq t^2 + t - 1 \wedge (\exists y : x - a = t \cdot y) \wedge (\exists z : x - b = (t+1) \cdot z).$$

From the validity of the statement, we find $[\![\chi]\!]_k = \mathbb{Z}^2$ for every $k \in \mathbb{Z}$. That is to say, for every instantiation of t, both φ and χ are tautologies of PrA. ◀

There are many natural decision problems regarding PrA[t] (all taking a formula φ as input):

- *satisfiability*: Is φ satisfiable for an instantiation of the parameter (i.e., $\mathbb{S}(\varphi) \neq \{\emptyset\}$)?
- *universality*: Is φ satisfiable for all instantiations of the parameter (i.e., $\emptyset \notin \mathbb{S}(\varphi)$)?
- *finiteness*: Is φ satisfiable for only finitely many instantiations of the parameter t?

In [6], Bogart, Goodrick, and Woods consider a search problem that generalises all the problems above: they show how to compute, from an input formula φ, a closed expression for the function $f(k) \coloneqq \#[\![\varphi]\!]_k$, where $\#S$ stands for the cardinality of a set S. By relying on properties of this function, one can solve satisfiability, universality, and finiteness. In their proof, the first ingredient is given by Goodrick's bounded quantifier elimination procedure [15]. In contrast to the quantifier elimination procedures for PrA, in this procedure every quantified variable x is not completely eliminated from the formula φ, but acquires instead a bound $0 \leq x \leq f(t)$, for some univariate polynomial $f(t)$. An example of this is given by the variable x in Example 1, which is bounded in $[0, t(t+1)-1]$. Closely related bounded quantifier elimination procedures were also developed in [23, 31]. The second ingredient of the construction is given by a method developed by Chen, Li, and Sam for the study of parametric polytopes [8], and dubbed "base t division method" in [6]. This method produces a quantifier-free PrA[t] formula ψ satisfying $\#[\![\psi]\!]_k = \#[\![\varphi]\!]_k$ for every $k \in \mathbb{Z}$.

The combination of the two ingredients has a drawback: the equivalence of the initial formula φ with the quantifier-free formula ψ over $\mathbb{Z}$ is not preserved. In [6, p. 13], the authors ask whether this issue can be fixed: "*one might try to show that any formula* [of PrA[t]] *is logically equivalent to a quantifier-free formula in a slightly larger language with additional "well-behaved" function and relation symbols* [...] *But we already know that quantifier elimination in the original language* [of the structure $\langle\mathbb{Z};\, 0,\, 1,\, +,\, x \mapsto t \cdot x,\, \leq\rangle$] *is impossible, and finding a reasonable language for quantifier elimination seems difficult".* A candidate for the extended structure was suggested by Goodrick in [15, Conjecture 2.6]: PrA[t] must be extended with all *integer division functions* $x \mapsto \left\lfloor \frac{x}{|f(t)|} \right\rfloor$, one for each integer polynomial $f(t)$

(these functions are assumed to occur in a formula only under the proviso that $f(t) \neq 0$). This is a rather tight conjecture, as all added functions are trivially definable in PrA[t]: the equality $y = \left\lfloor \frac{x}{|f(t)|} \right\rfloor$ holds if and only if $f(t) \neq 0 \wedge |f(t)| \cdot y \leq x \wedge x < |f(t)| \cdot y + |f(t)|$.

We give a positive answer to Goodrick's conjecture:

▶ **Theorem 2.** *One-parametric Presburger arithmetic admits effective quantifier elimination in the extended structure* $\langle \mathbb{Z}; 0, 1, +, x \mapsto t \cdot x, x \mapsto \left\lfloor \frac{x}{|f(t)|} \right\rfloor, \leq \rangle$.

Above, the adjective "effective" reflects the fact that there is a quantifier elimination procedure for constructing, given an input formula φ, an equivalent quantifier-free formula ψ. The main contribution towards the proof of Theorem 2 is a procedure for removing bounded quantifiers while preserving formula equivalence; hence obtaining $[\![\psi]\!]_k = [\![\varphi]\!]_k$ for all $k \in \mathbb{Z}$, instead of the weaker $\#[\![\psi]\!]_k = \#[\![\varphi]\!]_k$ obtained with the "base t division method" from [6].

As mentioned above, an active area of research in quantifier elimination focuses on existential Presburger arithmetic (∃PrA) and its extensions. This interest is motivated, on the one hand, by the goal of improving both the performance and expressiveness of SMT solvers, mostly targeting existential theories [2, 14]. On the other hand, recent work has revisited the computational complexity of quantifier elimination procedures. For many years, these procedures were regarded as inefficient when applied to ∃PrA. Notably, Weispfenning's classic approach [30] yields only a NExpTime upper bound for satisfiability, despite fundamental results from integer programming [7, 29] establishing that ∃PrA is in NP. It was not until 2024 that two independent works [11, 20] provided quantifier elimination procedures with matching NP upper bounds. The approach in [20] builds on the geometric ideas from [29], while [11] adapts Bareiss' fraction-free Gaussian elimination procedure [1] to ∃PrA. Starting from the fact that Bareiss' algorithm works in any integral domain, we show that the second approach extends naturally to PrA[t]. By analysing the runtime of our procedure, we derive:

▶ **Theorem 3.** *For the class of all existential formulae of* PrA[t]*, the following holds:*

Satisfiability	*Universality*	*Finiteness*
NP-*complete*	coNExp-*complete*	coNP-*complete*

Our result on the satisfiability problem generalises the feasibility in NP of non-linear integer programs $A \cdot \boldsymbol{x} \leq \boldsymbol{b}(t)$, where $\boldsymbol{b}(t)$ is a vector of quotients of integer polynomials in t, proved by Gurari and Ibarra [17]. We remark that both problems are NP-hard in fixed dimension: solvability of systems $x \geq 0 \wedge a \cdot t^2 + b \cdot x = c$ is a well-known NP-complete problem [24].

Future work. This paper does not provide a complexity analysis for *full* PrA[t]. A back-of-the-envelope calculation of the runtime of the procedures in [6] and [15] suggests that the satisfiability problem for PrA[t] is in elementary time (potentially in 3ExpTime). However, these procedures do not yield an NP upper bound for the existential fragment. In contrast, the procedure we introduce shows ∃PrA[t] in NP, but it may in principle run in non-elementary time on arbitrary formulae. Unifying these procedures into a single "optimal" one seems possible and will be addressed in a forthcoming extended version of this paper. This would also provide an extension to the 3ExpTime quantifier-elimination procedure for *almost linear arithmetic* proposed by Weispfenning in [30].

Most of the literature on PrA[t] focuses on computing the function $f(k) \coloneqq \#[\![\varphi]\!]_k$, as introduced in [6]. Not much is known regarding the complexity of computing this function. To our knowledge, the most significant result in this direction is the one in [5], where Bogart, Goodrick, Nguyen, and Woods show that $f(k)$ can be computed in polynomial time (in the bit-length of k given as part of the input) whenever φ is a *fixed* PrA[t]-formula. We believe Theorem 3 to be a good starting point for further research in this direction.

While our work shows that the feasibility problem for integer programs in which a single variable occurs non-linearly is in NP, the paper does not discuss related optimisation problems of minimisation/maximisation. From our quantifier elimination procedure, we can show that if there are optimal solutions to a linear polynomial with coefficients in $\mathbb{Z}[t]$ subject to a formula in $\exists\mathrm{PrA}[t]$, then one is of polynomial bit size. Generalising this result to other non-convex objectives is an interesting avenue for future research.

2 Preliminaries

We write $\mathbb{N}$ for the non-negative integers, and $\mathbb{Z}[t]$ for the set of univariate polynomials $f(t) = \sum_{i=0}^{d} a_i \cdot t^i$, where the coefficients $a_1, \ldots, a_d$ and the constant a_0 are over the integers $\mathbb{Z}$. The *height* $h(f)$, *degree* $\deg(f)$ and *bit size* $\langle f \rangle$ of f are defined as $h(f) \coloneqq \max\{|a_i| : i \in [0,d]\}$, $\deg(f) \coloneqq \max\{0, i \in [0,d] : a_i \neq 0\}$, and $\langle f \rangle \coloneqq (\deg(f)+1) \cdot (\lceil \log_2(h(f)+1) \rceil + 1)$, respectively. For example, $f(t) = 2 \cdot t^2 - 3$ has degree 2, height 3 and bit size 9. Vectors of variables are denoted by $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$, *etc.*; we write $\lfloor \cdot \rfloor$ for the floor function.

On extending the structure of PrA$[t]$. As discussed in Section 1, the paper concerns the extension of the first-order theory of $\langle \mathbb{Z};\, 0,\, 1,\, +,\, x \mapsto t \cdot x,\, \leq \rangle$ by all *integer division functions* $x \mapsto \lfloor \frac{x}{|f|} \rfloor$, where $f \in \mathbb{Z}[t]$. However, in the context of our quantifier elimination procedure, it is more natural to work within the (equivalent) first-order theory of the structure:

$$\left\langle \mathbb{Z};\, 0,\, 1,\, +,\, x \mapsto t \cdot x,\, \left\{ x \mapsto \left\lfloor \tfrac{x}{t^d} \right\rfloor \right\}_{d \in \mathbb{N}},\, \left\{ x \mapsto (x \bmod f) \right\}_{f \in \mathbb{Z}[t]},\, \left\{ f \mid x \right\}_{f \in \mathbb{Z}[t]},\, =,\, \leq \right\rangle$$

where:

- The integer division $x \mapsto \lfloor \frac{x}{t^d} \rfloor$ is only defined for $t \neq 0$, with the obvious interpretation.
- The *integer remainder function* $x \mapsto (x \bmod f(t))$ is defined following the equivalence

  $$(x \bmod f(t) = y) \iff (f(t) = 0 \wedge y = x) \vee (f(t) \neq 0 \wedge y = x - |f(t)| \cdot \lfloor \tfrac{x}{|f(t)|} \rfloor).$$

  We remark that whenever $f(t) \neq 0$ the result of $(x \bmod f(t))$ belongs to $[0, |f(t)| - 1]$. (Also note that the absolute value function $|.|$ is easily definable in Presburger arithmetic.)
- The *divisibility relation* $f(t) \mid x$ is a unary relation, and is defined following the equivalence

  $$(f(t) \mid x) \iff (f(t) = 0 \wedge x = 0) \vee (f(t) \neq 0 \wedge (x \bmod f(t) = 0)).$$

  We remark that the divisibility relations and integer remainder functions are defined to satisfy the equivalence $f(t) \mid x \iff f(t) \mid (x \bmod f(t))$ also when $f(t)$ evaluates to 0.

For simplicity, *we still denote this first-order theory with* PrA$[t]$. Observe that we have ultimately defined $(f(t) \mid \cdot)$ and $x \mapsto (x \bmod f(t))$ in terms of $x \mapsto \lfloor \frac{x}{|f|} \rfloor$. As a result, every formula in this first-order theory can be translated into a formula from the theory in Theorem 2. This translation can be performed in polynomial time by introducing new existential quantifiers, or in exponential time without adding quantifiers (the blow-up is only due to the disjunctions in the definitions of $(f(t) \mid \cdot)$ and $x \mapsto (x \bmod f(t))$).

The *terms* of PrA$[t]$ are built from the constants $0, 1$, integer variables, and the functions of the structure. Without loss of generality, we restrict ourselves to (finite) terms of the form

$$\tau \;\coloneqq\; f_0(t) + \textstyle\sum_{i=1}^{n} f_i(t) \cdot x_i + \sum_{i=n+1}^{m} f_i(t) \cdot \lfloor \frac{\tau_i}{t^{d_i}} \rfloor + \sum_{i=m+1}^{k} f_i(t) \cdot (\tau_i \bmod g_i(t)), \qquad (1)$$

where all f_i and g_i belong to $\mathbb{Z}[t]$, all d_i belong to $\mathbb{N}$, and each τ_i is another term of this form. The term τ is said to be *linear*, if $f_i(t) = 0$ for $i \in [n+1, k]$ (i.e., it does not contain integer

division, nor integer remainder functions), and *non-shifted* whenever $f_0(t) = 0$. Above, the terms $\left\lfloor \frac{\tau_i}{t^{d_i}} \right\rfloor$ and $(\tau_i \bmod g_i(t))$ are *linear occurrences* of the integer division functions and integer remainder functions, and are said to *occur linearly* in τ. When every f_i is an integer (a degree 0 polynomial), we define 1-norm of τ as $\|\tau\|_1 \coloneqq \sum_{i=0}^{k} |f_i|$.

Moving to the atomic formulae of the theory, it is easy to see that equalities, inequalities and divisibility relations can be rewritten (in polynomial time) to be of the form $\tau = 0$, $\tau \leq 0$ and $f(t) \mid \tau$, respectively, where τ is a term of the form given in Equation (1). Syntactically, we will only work with atomic formulae of these forms, which we call PrA[t] *constraints*. However, for readability, we will still sometimes write (in)equalities featuring non-zero terms on both sides (i.e., $\tau_1 \leq \tau_2$), and strict inequalities $\tau_1 < \tau_2$ and $\tau_2 > \tau_1$; both are short for $\tau_1 - \tau_2 + 1 \leq 0$. A PrA[$t$] constraint is said to be *linear* if the term τ featured in it is linear.

We restrict ourselves to formulae in prenex normal form $\exists \boldsymbol{x}_1 \forall \boldsymbol{x}_2 \dots \exists \boldsymbol{x}_n : \varphi$ where φ is a *positive* Boolean combination of PrA[t] constraints; that is, the only Boolean connectives featured in φ are conjunctions $\wedge$ and disjunctions $\vee$. This restriction is without loss of generality, as De Morgan's laws allow to push all negations at the level of literals, which can then be removed with the equivalences $\neg(\tau = 0) \iff \tau < 0 \vee \tau > 0$, $\neg(\tau \leq 0) \iff \tau > 0$ and $\neg(f(t) \mid \tau) \iff (f(t) = 0 \wedge (\tau < 0 \vee \tau > 0)) \vee (\tau \bmod f(t) > 0)$.

For two terms τ_1 and τ_2, we write $[\tau_2 \,/\, \tau_1]$ for a term *substitution*. We see these substitutions as functions from terms to terms or from formulae to formulae: $\tau[\tau_2 \,/\, \tau_1]$ is the term obtained by replacing, in the term τ, every occurrence of τ_1 with τ_2. Analogously, $\varphi[\tau_2 \,/\, \tau_1]$ is the formula obtained from φ by replacing the term τ with $\tau[\tau_2 \,/\, \tau_1]$ in every atomic formula $\tau = 0$, $\tau \leq 0$, or $f(t) \mid \tau$. In Section 4 we will also need a stronger notion of substitution, called *vigorous substitution* in [11]; we defer its definition to that section.

3 Outline of the quantifier elimination procedure

In this section, we provide a high-level overview of our quantifier elimination procedure, highlighting the interactions among its various components. A detailed analysis of the two main components will be provided in the subsequent sections of the paper.

Let us consider a formula $\psi(\boldsymbol{x}_0) \coloneqq \exists \boldsymbol{x}_1 \forall \boldsymbol{x}_2 \dots \exists \boldsymbol{x}_n : \varphi(\boldsymbol{x}_0, \dots, \boldsymbol{x}_n)$, where φ is a positive Boolean combination of PrA[t] constraints. Our quantifier elimination procedure will work under the assumption that the parameter t is greater than or equal to 2. This simplifying assumption is without loss of generality, as the general problem is then solved as follows:

- For every $k \in \{-1, 0, 1\}$, call a quantifier elimination procedure for Presburger arithmetic (e.g., the one in [30]) on the formula $\psi[k \,/\, t]$, obtaining ψ'_k. Let $\psi_k \coloneqq \psi'_k \wedge (t = k)$.
- Call our procedure on ψ, obtaining a formula ψ^+. Let $\psi_{\geq 2} \coloneqq \psi^+ \wedge t \geq 2$.
- Call our procedure on $\psi[-t \,/\, t]$, obtaining ψ^-. Let $\psi_{\leq -2} \coloneqq \psi^-[-t \,/\, t] \wedge t \leq -2$.

Then, the formula $\psi_{\leq -2} \vee \psi_{-1} \vee \psi_0 \vee \psi_1 \vee \psi_{\geq 2}$ is quantifier-free, and equivalent to ψ.

As a second assumption, we only consider the case where ψ has a single block of existential quantifiers: $\psi(\boldsymbol{x}_0) = \exists \boldsymbol{x}_1 : \varphi(\boldsymbol{x}_0, \boldsymbol{x}_1)$. A procedure for this type of formulae can be iterated bottom-up to eliminate arbitrarily many blocks of quantifiers (rewriting $\forall \boldsymbol{x}$ as $\neg \exists \boldsymbol{x} \neg$).

The pseudocode of our procedure is given in Algorithm 1 (PrA[t]-QE). It describes a non-deterministic algorithm: for an input $\exists \boldsymbol{x} : \varphi$, each non-deterministic execution of PrA[t]-QE returns a positive Boolean combination of PrA[t] constraints $\psi(\boldsymbol{z})$. The disjunction obtained by aggregating all output formulae is equivalent to $\exists \boldsymbol{x} : \varphi$; so it is this disjunction that must ultimately be used to perform quantifier elimination. The choice to present the procedure in this manner is not merely stylistic: it automatically implements Reddy and Loveland's optimisation for Presburger arithmetic [27]. In quantifier elimination procedures for PrA,

Algorithm 1 PrA[t]-QE: A quantifier elimination procedure for PrA[t].

Input: $\exists \boldsymbol{x} : \varphi(\boldsymbol{x}, \boldsymbol{z})$ where φ is a positive Boolean combination of PrA[t] constraints.
Output of each branch (β): positive Boolean combination $\psi_\beta(\boldsymbol{z})$ of PrA[t] constraints.
Ensuring: $\bigvee_\beta \psi_\beta$ is equivalent to $\exists \boldsymbol{x} : \varphi$.

1: **while** φ contains a subterm of the form $\left\lfloor \frac{\tau}{t^d} \right\rfloor$ **do**
2: append a fresh variable x to $\boldsymbol{x}$
3: $\varphi \leftarrow \varphi[x \,/\, \left\lfloor \frac{\tau}{t^d} \right\rfloor] \wedge (t^d \cdot x \leq \tau) \wedge (\tau < t^d \cdot x + t^d)$
4: $\boldsymbol{y} \leftarrow \varnothing$; $B \leftarrow \varnothing$ ▷ *variables and map used to remove occurrences of* $(\cdot \bmod f(t))$
5: **while** φ contains a subterm of the form $(\tau \bmod f(t))$ **do**
6: **if** $*$ **then** ▷ *non-deterministic choice: skip or execute*
7: $\varphi \leftarrow \varphi[\tau \,/\, \tau \bmod f(t)] \wedge f(t) = 0$
8: **continue**
9: **guess** $\pm \leftarrow$ symbol in $\{+, -\}$
10: append a fresh variable y to $\boldsymbol{y}$ and update B : add the key-value pair $(y, \pm f(t) - 1)$
11: $\varphi \leftarrow \varphi[y \,/\, \tau \bmod f(t)] \wedge (f(t) \mid \tau - y)$
12: **return** ElimBounded(ElimDiv($\exists \boldsymbol{y} \leq B$: BoundedQE($\exists \boldsymbol{x} : \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$)))

eliminating a single variable x from an existential block $\exists \boldsymbol{y} \exists x$ produces a formula $\bigvee_i \gamma_i$ with a DNF-like structure. Reddy and Loveland observed that pushing the remaining existential quantifiers $\exists \boldsymbol{y}$ inside the scope of the disjunctions, i.e., rewriting $\exists \boldsymbol{y} \bigvee_i \gamma_i$ into $\bigvee_i \exists \boldsymbol{y} \gamma_i$, and then performing quantifier elimination locally to each disjunct leads to a faster procedure. By keeping variables local to a non-deterministic branch, one achieves the same effect.

We now describe the four components that make up PrA[t]-QE, which can be summarized under the following titles: pre-processing (lines 1–11), bounded quantifier elimination (call to BoundedQE), elimination of divisibility constraints (call to ElimDiv), and elimination of all bounded quantifiers (call to ElimBounded). For the rest of the section, let $\exists \boldsymbol{x} : \varphi(\boldsymbol{x}, \boldsymbol{z})$ be the input to PrA[t]-QE, where φ is a positive Boolean combination of PrA[t] constraints.

Pre-processing (lines 1–11). These lines remove all occurrences of the integer division functions $x \mapsto \left\lfloor \frac{x}{t^d} \right\rfloor$ and of the integer remainder functions $x \mapsto (x \bmod f(t))$, at the expense of adding new existentially quantified variables that are later eliminated. After this step, the formula is a positive Boolean combination of *linear* constraints. For the integer division function, the algorithm simply adds to the sequence of variables $\boldsymbol{x}$ to be eliminated a fresh variable x to proxy a term $\left\lfloor \frac{\tau}{t^d} \right\rfloor$ (line 2). It then relies on the equivalence $x = \left\lfloor \frac{\tau}{t^d} \right\rfloor \iff t^d \cdot x \leq \tau \wedge \tau < t^d \cdot (x+1)$ to replace $\left\lfloor \frac{\tau}{t^d} \right\rfloor$ with x (line 3). The removal of integer remainder functions is performed differently. First, let us observe that the following equivalence holds:

$$y = (\tau \bmod f(t)) \iff (f(t) = 0 \wedge y = \tau) \vee (0 \leq y \wedge y < |f(t)| - 1 \wedge f(t) \mid \tau - y).$$

The formula $f(t) = 0 \wedge y = \tau$ on the right-hand side of the equivalence is considered in line 8. This line is executed conditionally to a non-deterministic branching (line 6). If it is not executed, then lines 9–11 are executed instead; these correspond to the formula $0 \leq y \wedge y < |f(t)| - 1 \wedge f(t) \mid \tau - y$. The interesting property of this formula is that the variable y appears *bounded* by 0 from below, and by either $f(t) - 1$ or $-f(t) - 1$ from above (following the sign of $f(t)$). Instead of quantifying y using standard existential quantifiers (as done for the variables replacing $\left\lfloor \frac{\tau}{t^d} \right\rfloor$), in line 10 we use a bounded quantifier:

Algorithm 2 ElimDiv: Elimination of divisibility constraints.

Input: $\exists \boldsymbol{w} \leq B : \psi(\boldsymbol{w}, \boldsymbol{z})$, with ψ positive Boolean combination of linear PrA[t] constraints.
Output of each branch (β): a formula $\exists \boldsymbol{w}_\beta \leq B_\beta : \psi_\beta(\boldsymbol{w}_\beta, \boldsymbol{z})$ in PrA[t], where ψ_β is a positive Boolean combination of linear equalities and inequalities, and equalities of the form $\sigma(\boldsymbol{w}) + (\tau(\boldsymbol{z}) \bmod f(t)) = 0$, with σ linear, and τ linear and non-shifted.
Ensuring: $\bigvee_\beta (\exists \boldsymbol{w}_\beta \leq B_\beta : \psi_\beta)$ is equivalent to $\exists \boldsymbol{w} \leq B : \psi$.

1: **foreach** divisibility $f(t) \mid \sigma(\boldsymbol{w}) + \tau(\boldsymbol{z})$ in ψ, where τ is non-shifted **do**
2: **let** $\sigma(\boldsymbol{w})$ be the term $f_0(t) + \sum_{i=1}^{n} f_i(t) \cdot w_i$, where $\boldsymbol{w} = (w_1, \ldots, w_n)$
3: $d \leftarrow (n+3) \cdot \max\{\langle f \rangle, \langle f_0 \rangle, \langle f_i \rangle \cdot \langle B(w_i) \rangle \ : i \in [1, n]\}$
4: append a fresh variable y to $\boldsymbol{w}$ and update B : add the key-value pair (y, t^d)
5: **guess** $\pm \leftarrow$ symbol in $\{+, -\}$
6: update ψ : replace $(f(t) \mid \sigma(\boldsymbol{w}) + \tau(\boldsymbol{z}))$ with $\pm f(t) \cdot y + \sigma(\boldsymbol{w}) + (\tau \bmod f(t)) = 0$
7: **return** $\exists \boldsymbol{w} \leq B : \psi$

▶ **Definition 4.** *A block of bounded quantifiers $\exists \boldsymbol{w} \leq B$ is given by a sequence of variables $\boldsymbol{w} = (w_1, \ldots, w_m)$ and a map B assigning to each variable in $\boldsymbol{w}$ a polynomial in $\mathbb{Z}[t]$. Its semantics is given by the equivalence $\exists \boldsymbol{w} \leq B : \psi \iff \exists \boldsymbol{w} : \bigwedge_{i=1}^{m} (0 \leq w_i \leq B(w_i)) \wedge \psi$.*

Let us write $\boldsymbol{x}_\beta, \boldsymbol{y}_\beta, B_\beta$ and φ_β for the values taken by $\boldsymbol{x}$, $\boldsymbol{y}$, B and φ in the non-deterministic branch β, when the control flow of the program reaches line 12. The following equivalence holds, where the disjunction $\bigvee_\beta$ ranges across all non-deterministic branches:

$$\exists \boldsymbol{x} : \varphi(\boldsymbol{x}, \boldsymbol{z}) \iff \bigvee_\beta \exists \boldsymbol{y}_\beta \leq B_\beta \, \exists \boldsymbol{x}_\beta : \varphi_\beta(\boldsymbol{x}_\beta, \boldsymbol{y}_\beta, \boldsymbol{z}). \tag{2}$$

Bounded quantifier elimination. Once reaching line 12, the algorithm proceeds by calling the procedure BoundedQE. We will discuss this procedure in Section 4. In a nutshell, its role is to replace the quantifiers $\exists \boldsymbol{x}_\beta$ on the right-hand side of Equation (2) with bounded quantifiers, that are merged with the already existing block of bounded quantifiers $\exists \boldsymbol{y}_\beta \leq B_\beta$. The formal specification of BoundedQE is given in the next lemma.

▶ **Lemma 5.** *There is a non-deterministic procedure with the following specification:*

Input: *$\exists \boldsymbol{x} : \varphi(\boldsymbol{x}, \boldsymbol{z})$, with φ positive Boolean combination of linear PrA[t] constraints.*
Output of each branch (β): *a formula $\exists \boldsymbol{w}_\beta \leq B_\beta : \psi_\beta(\boldsymbol{w}_\beta, \boldsymbol{z})$, where ψ_β is a positive Boolean combination of linear PrA[t] constraints.*

The algorithm ensures that the disjunction $\bigvee_\beta \exists \boldsymbol{w}_\beta \leq B_\beta : \psi_\beta$ of output formulae ranging over all non-deterministic branches is equivalent to $\exists \boldsymbol{x} : \varphi$.

As stated, the lemma above is also proved by Goodrick in [15], who introduced the first bounded quantifier elimination procedure specifically for PrA[t]. When applied to existential formulae, that procedure requires doubly-exponential time, making it unsuitable for establishing Theorem 3. In contrast, BoundedQE runs in non-deterministic polynomial time. Due to this difference, we cannot rely directly on [15] and must thus re-establish Lemma 5.

Removing divisibility constraints: more bounded quantifiers. BoundedQE introduces divisibility constraints $f(t) \mid \tau$. The next step, detailed in Algorithm 2 (ElimDiv), eliminates all divisibility constraints in favour, once more, of bounded quantifiers.

The idea behind Algorithm 2 is as follows. Let $f(t) \mid \sigma(\boldsymbol{w}) + \tau(\boldsymbol{z})$ be a constraint from the input formula, where $\boldsymbol{w}$ are the variables in the block of bounded quantifiers (these correspond to the variables $\boldsymbol{y}_\beta$ from Equation (2) and those introduced by BoundedQE), and $\boldsymbol{z}$ are

the free variables. Notice that this constraint is equivalent to $f(t) \mid \sigma(\boldsymbol{w}) + (\tau(\boldsymbol{z}) \bmod f(t))$, which is in turn equivalent to the existential formula $\exists y : f(t) \cdot y + \sigma(\boldsymbol{w}) + (\tau(\boldsymbol{z}) \bmod f(t)) = 0$, where y is a fresh variable ranging over $\mathbb{Z}$. Since $\boldsymbol{w}$ is constrained by bounded quantifiers, we can upper-bound the number of digits in the base t encoding of the linear term $\sigma(\boldsymbol{w})$ (recall: $t \geq 2$). When $f(t) \neq 0$, the same applies to $(\tau(\boldsymbol{z}) \bmod f(t))$, which ranges between 0 and $f(t) - 1$; and this in turn imposes a bound on the base t representation of y. When $f(t) = 0$ instead, the truth of $f(t) \cdot y + \sigma(\boldsymbol{w}) + (\tau(\boldsymbol{z}) \bmod f(t)) = 0$ only depends on whether $\sigma(\boldsymbol{w}) + (\tau(\boldsymbol{z}) \bmod f(t)) = 0$, and we can thus restrict y to any non-empty interval. This allows us to replace the quantifier $\exists y$ with a bounded quantifier (lines 3 and 4). Since y ranges over $\mathbb{Z}$, whereas bounded quantifiers use non-negative ranges, the algorithm explicitly guesses the sign of y in line 5, allowing y to only range over $\mathbb{N}$ instead. Formalising these arguments yields the following lemma.

▶ **Lemma 6.** *Algorithm 2 (ElimDiv) complies with its specification.*

Elimination of all bounded quantifiers. From the output of ElimDiv, the final operation by PrA[t]-QE is a call to ElimBounded, which removes all bounded quantifiers. This algorithm is detailed in Section 5. Its specification is given in the next lemma.

▶ **Lemma 7.** *There is a non-deterministic procedure with the following specification:*

Input: *$\exists \boldsymbol{w} \leq B : \varphi(\boldsymbol{w}, \boldsymbol{z})$, with φ positive Boolean combination of linear* PrA[t] *(in)equalities and constraints $\sigma(\boldsymbol{w}) + (\tau(\boldsymbol{z}) \bmod f(t)) = 0$, with σ linear, and τ linear and non-shifted.*

Output of each branch (β): *a positive Boolean combination $\psi_\beta(\boldsymbol{z})$ of* PrA[t] *constraints. In all divisibility constraints $f(t) \mid \tau$, the divisor $f(t)$ is an integer.*

The algorithm ensures that the disjunction $\bigvee_\beta \psi_\beta$ of output formulae ranging over all non-deterministic branches is equivalent to $\exists \boldsymbol{w} \leq B : \varphi$.

Together, Equation (2) and Lemmas 5–7 show that PrA[t]-QE meets its specification; thus showing Theorem 2 conditionally to the correctness of BoundedQE and ElimBounded.

4 Efficient bounded quantifier elimination in PrA[t]

This section outlines the arguments leading to the procedure BoundedQE. Its pseudocode is given in Algorithm 3. We start with an example demonstrating the key idea used to develop a version of bounded quantifier elimination in PrA[t]. These arguments are sufficient for establishing Lemma 5; although they do not result in an optimal procedure complexity-wise. We will then recall the main arguments used in [11] to obtain an optimal procedure, which, when implemented, yield BoundedQE.

Let us consider a formula $\exists x : \varphi(x, \boldsymbol{z})$ where, for simplicity, φ is of the form:

$$\tau(\boldsymbol{z}) \leq a \cdot x \wedge b \cdot x \leq \rho(\boldsymbol{z}) \wedge \big(m \mid c \cdot x + \sigma(\boldsymbol{z})\big) \wedge a > 0 \wedge b > 0 \wedge m > 0,$$

where a, b, c and m are polynomials from $\mathbb{Z}[t]$, and τ, ρ and σ are *linear* PrA[t] terms. For the time being, we invite the reader to pick some values for t and the free variables $\boldsymbol{z}$, so that the formula φ becomes a formula from Presburger arithmetic in a single variable x. The standard argument for eliminating x in PrA goes as follows (see, e.g., [30]). We first update the inequalities to ensure that all coefficients of x are equal; this results in the inequalities $b \cdot \tau \leq a \cdot b \cdot x$ and $a \cdot b \cdot x \leq a \cdot \rho$. The quantification $\exists x$ expresses that there is $g \in \mathbb{Z}$ such that **(i)** g is a multiple of $a \cdot b$ that belongs to the interval $[b \cdot \tau, a \cdot \rho]$; and **(ii)** m divides $c \cdot \frac{g}{a \cdot b} + \sigma$. The key observation is that such an integer (if it exists) can be found by only looking at

elements of $[b \cdot \tau, a \cdot \rho]$ that are "close" to $b \cdot \tau$. More precisely, the properties **(i)** and **(ii)** must be simultaneously satisfied by $b \cdot \tau + r$, for some $r \in [0, a \cdot b \cdot m]$. We can thus restrict x to satisfy an additional constraint $a \cdot b \cdot x = b \cdot \tau + r$. A small refinement: since this equality is unsatisfiable when the shift r is not a multiple of $b > 0$, we can rewrite it as $a \cdot x = \tau + s$, where the shift s now ranges in $[0, a \cdot m]$. Observe that s lies in an interval that is independent of the values picked for $\boldsymbol{z}$; as a and m were originally polynomials in t.

Let us keep assigning a value to the parameter t (so, a and m are still integers), but reinstate the variables $\boldsymbol{z}$. From the above argument, the formula $\exists x : \varphi(x, \boldsymbol{z})$ is equivalent to $\bigvee_{s=0}^{a \cdot m} \exists x \, (\varphi(x, \boldsymbol{z}) \wedge a \cdot x = \tau + s)$. It is now straightforward to eliminate x from each disjunct $\exists x \, (\varphi(x, \boldsymbol{z}) \wedge a \cdot x = \tau + s)$: we simply "apply" the equality $a \cdot x = \tau + s$, substituting x for $\frac{\tau + r}{a}$, and add a divisibility constraint forcing $\tau + s$ to be a multiple of a. After this substitution, both $a \cdot x = \tau + s$ and $\tau(\boldsymbol{z}) \leq a \cdot x$ become $\top$. The resulting disjunct is

$$\psi(s, \boldsymbol{z}) \coloneqq b \cdot (\tau + s) \leq a \cdot \rho \wedge (m \cdot a \mid c \cdot (\tau + s) + a \cdot \sigma) \wedge (a \mid \tau + s) \wedge a > 0 \wedge b > 0 \wedge m > 0,$$

and $\bigvee_{s=0}^{a \cdot m} \psi(s, \boldsymbol{z})$ is equivalent to $\exists x : \varphi(x, \boldsymbol{z})$. (For PrA, this concludes the quantifier elimination procedure.) When restoring the parameter t, these two formulae are still equivalent, but the number of disjunctions $\bigvee_{s=0}^{a \cdot m}$ now depends on t. We replace them with a bounded quantifier, rewriting $\bigvee_{s=0}^{a \cdot m} \psi(s, \boldsymbol{z})$ as $\exists s \leq B : \psi(s, \boldsymbol{z})$, where $B(s) \coloneqq a(t) \cdot m(t)$. This is, in a nutshell, the *bounded quantifier elimination procedure* from [15, 31].

When the signs of a, b, and m are unknown (i.e., φ does not feature the constraints $a > 0$, $b > 0$ and $m > 0$), we must perform a "sign analysis": we write a disjunction (or guess) over all possible signs of the three polynomials. In Algorithm 3, the lines marked in yellow are related to this analysis; e.g., line 16 guesses the sign of the (non-zero) coefficient a of x.

Taming the complexity of the procedure. Problems arise when looking at the complexity of the procedure outlined above. To understand this point, consider the inequality $b \cdot (\tau + s) \leq a \cdot \rho$, which was derived by substituting $\frac{\tau + s}{a}$ for x in $b \cdot x \leq \rho$, and suppose $\tau = c \cdot y + \tau'$ and $\rho = d \cdot y + \rho'$, for some variable y. This inequality can be rewritten as $(b \cdot c - a \cdot d) \cdot y + b \cdot \tau' - a \cdot \rho' + b \cdot s \leq 0$. When looking at the coefficient $(b \cdot c - a \cdot d)$ of y one deduces that, if quantifier elimination is performed carelessly on a block $\exists \boldsymbol{x}$ of multiple existential quantifiers, the coefficients of the variables in the formula will grow quadratically with each eliminated variable. Then, by the end of the procedure, their binary bit size will be exponential in the number of variables in $\boldsymbol{x}$. However, this explosion can be avoided by noticing that coefficients are updated following the same pattern as in Bareiss' polynomial-time Gaussian elimination procedure [1]. This insight was highlighted in [11], building upon an earlier observation from [31]. In Bareiss' algorithm, the key to keeping coefficients polynomially bounded is given by the Desnanot–Jacobi identity. Consider an $m \times d$ matrix A. Let us write $A[i_1, \ldots, i_r; j_1, \ldots, j_\ell]$ for the $r \times \ell$ sub-matrix of A made of the rows with indices $i_1, \ldots, i_r \in [1, m]$ and columns with indices $j_1, \ldots, j_\ell \in [1, d]$. For $i, j, \ell \in \mathbb{N}$ with $0 \leq \ell \leq \min(m, d)$, $1 \leq i \leq m$ and $1 \leq j \leq d$, we define $a_{i,j}^{(\ell)} \coloneqq \det A[1, \ldots, \ell, i; 1, \ldots, \ell, j]$.

▶ **Proposition 8** (Desnanot–Jacobi identity). *For every $i, j, \ell \in \mathbb{N}$ with $\ell \geq 2$, $\ell < i \leq m$ and $\ell < j \leq d$, we have $(a_{\ell,\ell}^{(\ell-1)} \cdot a_{i,j}^{(\ell-1)} - a_{\ell,j}^{(\ell-1)} \cdot a_{i,\ell}^{(\ell-1)}) = a_{\ell,\ell}^{(\ell-2)} \cdot a_{i,j}^{(\ell)}$.*

The Desnanot–Jacobi identity is true for all matrices with entries over an integral domain (a non-zero commutative ring in which the product of non-zero elements is non-zero), and therefore we can take the entries of A to be polynomials in $\mathbb{Z}[t]$.

Returning to our informal discussion, we now see that the coefficient $(b \cdot c - a \cdot d)$ of y is oddly similar to the left-hand side of the Desnanot–Jacobi identity. Suppose that the elements $a_{i,j}^{(\ell-1)}$ are the coefficients of the variables in the formula currently being processed by the

Algorithm 3 BoundedQE: A bounded quantifier elimination procedure for PrA[t].

Input: $\exists \boldsymbol{x} : \varphi(\boldsymbol{x}, \boldsymbol{z})$ where φ is a positive Boolean combination of linear PrA[t] constraints.
Output of each branch (β): a formula $\exists \boldsymbol{w}_\beta \le B_\beta : \psi_\beta(\boldsymbol{w}_\beta, \boldsymbol{z})$ where ψ_β is a positive Boolean combination of linear PrA[t] constraints.
Ensuring: $\bigvee_\beta \exists \boldsymbol{w}_\beta \le B_\beta : \psi_\beta$ is equivalent to $\exists \boldsymbol{x} : \varphi$.

1: **guess** $Z \leftarrow$ subset of $\{f(t) :$ the relation $(f(t) \mid \cdot)$ occurs in $\varphi\}$
2: **foreach** $f(t)$ in Z **do**
3: update φ : replace each divisibility $f(t) \mid \tau$ with $\tau = 0$
4: $\varphi \leftarrow \varphi \wedge (f(t) = 0)$
5: **guess** $\pm \leftarrow$ symbol in $\{-, +\}$ ▷ *sign required to make $m(t)$ below positive*
6: $m(t) \leftarrow \pm \prod\{f(t) :$ the relation $(f(t) \mid \cdot)$ occurs in $\varphi\}$
7: $\chi \leftarrow (m(t) > 0)$
8: $(\pm, \ell(t)) \leftarrow (+, 1)$; $B \leftarrow \varnothing$ ▷ *B: map from variables to upper bounds*
9: update φ : replace each inequality $\tau \le 0$ with $\tau + y = 0$, where y is a fresh slack variable
10: **foreach** x in $\boldsymbol{x}$ **do**
11: **if** $*$ **then** ▷ *non-deterministic choice: skip or execute*
12: update B : add the key-value pair $(x, m(t) - 1)$
13: **continue**
14: **guess** $f(t) \cdot x + \tau = 0$ $\leftarrow$ equality in φ that contains x
15: $p(t) \leftarrow \ell(t)$; $\ell(t) \leftarrow f(t)$ ▷ *previous and current leading coefficients*
16: $\pm \leftarrow$ **guess** a symbol in $\{-, +\}$ ▷ *sign of $f(t)$*
17: $\chi \leftarrow \chi \wedge (\pm f(t) > 0)$
18: $m(t) \leftarrow \pm f(t) \cdot m(t)$
19: **if** τ contains a slack variable y such that $B(y)$ is undefined **then**
20: update B : add the key-value pair $(y,\ m(t) - 1)$
21: $\varphi \leftarrow \varphi[[\frac{-\tau}{f(t)} \ / \ x]]$
22: update φ : divide all constraints by $p(t)$ ▷ *both sides for divisibility constraints*
23: $\varphi \leftarrow \varphi \wedge (f(t) \mid \tau)$
24: **foreach** equality $\eta = 0$ of φ with a slack variable y such that $B(y)$ is undefined **do**
25: update φ : replace $\eta = 0$ with $\eta[0/y] \le 0$ **if** the sign $\pm$ is plus **else** with $\eta[0/y] \ge 0$
26: **return** $\exists \boldsymbol{w} \le B : \varphi \wedge \chi$ **where** $\boldsymbol{w}$ is the sequence of keys of the map B

quantifier elimination procedure, and we are eliminating the ℓ-th quantifier. Proposition 8 tells us that all coefficients produced by the naïve elimination (left-hand side of the identity) have $a_{\ell,\ell}^{(\ell-2)}$ as a common factor. By dividing through by this common factor, we obtain smaller coefficients for the next step of variable elimination – namely, $a_{i,j}^{(\ell)}$. When eliminating the first variable ($\ell = 1$), the common factor is 1. Otherwise, it is the coefficient a that the $(\ell - 1)$-th eliminated variable x has in the equality $a \cdot x = \tau + s$ used for the elimination. In Algorithm 3, the lines marked in blue implement Bareiss' optimisation: line 8 initialises the common factor $\ell(t)$ and its sign, line 15 updates it, and line 22 performs the division.

Some details on BoundedQE. Lines 1–4 handle the divisibility constraints $f(t) \mid \tau$ with the divisor $f(t)$ equal to 0. Such constraints are equalities in disguise, and the procedure replaces them with $\tau = 0$. When the procedure reaches line 5, all divisors in the divisibility constraints are assumed non-zero. Following the example from the previous paragraph, recall

that the shifts s belong to intervals that depend on these divisors; when multiple divisors occur, the procedure for PrA takes their lcm (instead of just m as in our example). For simplicity, instead of lcm, BoundedQE considers the absolute value $m(t)$ of their product (line 6). After guessing the sign $\pm$ of this product, the procedure enforces $m(t) > 0$ in line 7. This information is stored in the formula χ, which accumulates all sign guesses made by the algorithm; these are conjoined to φ when the procedure returns.

Line 9 replaces all inequalities with equalities by introducing slack variables ranging over $\mathbb{N}$. (This step is inherited from [11].) Slack variables represent the shifts s from the quantifier elimination procedure for PrA. Line 12 covers the corner cases of x not appearing in equalities, or t being such that all the coefficients $f(t)$ of x evaluate to zero. After guessing an equality $f(t) \cdot x + \tau = 0$ to perform the substitution (line 14), line 19 checks whether τ features a slack variable y (i.e., the equality was originally an inequality). If so, the procedure generates a bounded quantifier for y. The elimination of x (line 21) is performed with the *vigorous substitution* $\varphi[\![\frac{-\tau}{f(t)} \,/\, x]\!]$ which works as follows: **1:** Replace every equality $\rho = 0$ with $f(t) \cdot \rho = 0$, and every divisibility $g(t) \mid \rho$ with $f(t) \cdot g(t) \mid f(t) \cdot \rho$; this is done also for constraints where x does not occur. **2:** Replace every occurrence of $f(t) \cdot x$ with τ (from step **1**, each coefficient of x in the system can be factored as $f(t) \cdot h(t)$ for some $h \in \mathbb{Z}[t]$).

After applying the vigorous substitution, the procedure divides all coefficients of the inequalities and divisibility constraints in φ by the common factor of the Bareiss' optimisation (line 22). In the case of divisibility constraints, divisors are also affected. Proposition 8 ensures that these divisions are all without remainder. In practice, the traditional Euclidean algorithm for polynomial division can be used to construct the quotient in polynomial time. As a result of these divisions, throughout the procedure all polynomials $f(t)$ guessed in line 14 have polynomial bit sizes in the size of the input formula.

After the **foreach** loop of line 10 completes, all variables from $\boldsymbol{x}$ have been eliminated (line 21) or bounded (line 12). A benefit of translating inequalities into equalities in line 9 is that x can be eliminated independently of the sign of its coefficient $f(t)$; inequalities would need to flip for $f(t)$ negative instead. The final step (lines 24 and 25) drops all slack variables for which no bound was assigned in line 20, reintroducing the inequalities (the sign stored in $\pm$ tells us the direction of these inequalities). This step is also inherited from [11].

By fully developing the arguments above, one shows that BoundedQE is correct:

▶ **Lemma 9.** *Algorithm 3 (BoundedQE) complies with its specification.*

This lemma implies Lemma 5. In the sequel we will also need the next lemma, discussing properties of the outputs of BoundedQE for "Presburger-arithmetic-like" inputs.

▶ **Lemma 10.** *Let $\exists \boldsymbol{x} : \varphi(\boldsymbol{x}, \boldsymbol{z})$ be a formula input of Algorithm 3, in which all coefficients of the variables in $\boldsymbol{x}$, and all divisors $f(t)$ in relations $(f(t) \mid \cdot)$, are integers. The map B_β in the output of each non-deterministic branch β ranges over the integers.*

5 Elimination of polynomially bounded quantifiers

We move to Algorithm 4 (ElimBounded), which eliminates the bounded quantifiers in three steps: replacement of bounded variables by their t-ary expansions; "divisions by t" until all t-digits have integer coefficients; elimination of t-digits via BoundedQE.

Base t expansion (lines 1–6). Following the semantics of bounded quantifiers, line 1 adds to φ the bounds $0 \leq w \wedge w \leq B(w)$, for each bounded variable w. The subsequent lines "bit blast" w into its t-ary expansion $t^M \cdot y_M + \cdots + t \cdot y_1 + y_0$, where M is the largest bit size of the bounds in B (line 2). All added variables $\boldsymbol{y}$ are t-digits, i.e., they range in $[0, t-1]$.

Algorithm 4 ElimBounded: Elimination of polynomially bounded quantifiers.

Input: $\exists \boldsymbol{w} \leq B : \varphi(\boldsymbol{w}, \boldsymbol{z})$, with φ positive Boolean combination of linear PrA$[t]$ (in)equalities, and constraints $\sigma(\boldsymbol{w}) + (\tau(\boldsymbol{z}) \bmod f(t)) = 0$, with σ linear, and τ linear and non-shifted.
Output of each branch (β): a positive Boolean combination $\psi_\beta(\boldsymbol{z})$ of PrA$[t]$ constraints.
Ensuring: $\bigvee_\beta \psi_\beta$ is equivalent to $\exists \boldsymbol{w} \leq B : \varphi$.

1: $\varphi \leftarrow \varphi \wedge \bigwedge_{w \in \boldsymbol{w}} (0 \leq w) \wedge (w \leq B(w))$
2: $M \leftarrow \max\{\langle B(w) \rangle \ : \ w \in \boldsymbol{w}\}$
3: $\boldsymbol{y} \leftarrow \varnothing$ ▷ *$\boldsymbol{y}$ is a vector of variables used to "bit blast" bounded variables*
4: **foreach** w in $\boldsymbol{w}$ **do**
5: append fresh variables $y_0, \ldots, y_M$ to $\boldsymbol{y}$
6: $\varphi \leftarrow \varphi[(t^M \cdot y_M + \cdots + t \cdot y_1 + y_0) \,/\, w] \wedge \bigwedge_{i=0}^{M} ((0 \leq y_i) \wedge (y_i \leq t - 1))$
7: **while** a variable from $\boldsymbol{y}$ has a non-integer coefficient in a constraint $(\eta \sim 0)$ of φ **do**
8: **if** the symbol $\sim$ is $\leq$ **then** $\eta \leftarrow \eta - 1$ ▷ *we work with $\eta - 1 < 0$; else $\sim$ is $=$*
9: **let** η be $(\sigma(\boldsymbol{y}) \cdot t + \rho(\boldsymbol{y}) + \tau(\boldsymbol{z}))$, where ρ does not contain t, and τ is non-shifted
10: $\rho \leftarrow \rho(\boldsymbol{y}) + (\tau(\boldsymbol{z}) \bmod t)$ ▷ *add to ρ the unbounded part modulo t*
11: **guess** $r \leftarrow$ integer in $[-\|\rho\|_1, \|\rho\|_1]$ ▷ *quotient of the division of ρ by t*
12: **if** the symbol $\sim$ is $=$ **then**
13: $\gamma \leftarrow (t \cdot r = \rho)$
14: **else**
15: $\gamma \leftarrow (t \cdot r \leq \rho) \wedge (\rho \leq t \cdot (r + 1) - 1)$
16: $r \leftarrow r + 1$
17: update φ : replace $(\eta \sim 0)$ with $\gamma \wedge \left(\sigma + r + \left\lfloor \frac{\tau}{t} \right\rfloor \sim 0\right)$
18: $\boldsymbol{z}' \leftarrow \varnothing$; $S \leftarrow \varnothing$ ▷ *$\boldsymbol{z}'$ are used to rewrite φ as a combination of linear constraints*
19: **foreach** constraint $(\rho(\boldsymbol{y}) + \tau(\boldsymbol{z}) \sim 0)$ of φ, where τ is non-shifted **do**
20: append a fresh variable z' to $\boldsymbol{z}'$ and update S : add the key-value pair $(z', \tau(\boldsymbol{z}))$
21: $\varphi \leftarrow \varphi[z' \,/\, \tau(\boldsymbol{z})]$
22: $\exists \boldsymbol{w}' \leq B' : \psi(\boldsymbol{w}', \boldsymbol{z}') \leftarrow \text{BoundedQE}(\boldsymbol{y}, \varphi(\boldsymbol{y}, \boldsymbol{z}'))$
23: **foreach** w in $\boldsymbol{w}'$ **do** ▷ *now every $B'(w)$ is an integer*
24: **guess** $g \leftarrow$ integer in $[0, B'(w)]$
25: $\psi \leftarrow \psi[g \,/\, w]$
26: **return** $\psi[[S(z') \,/\, z'] : z' \in \boldsymbol{z}']$

▶ **Example 11.** Let us see this step in action on a bounded version of the formula in Example 1:

$$\exists (x, y, z) \leq B \ : \ \big(t \cdot y = x + (-a \bmod t)\big) \wedge \big((t + 1) \cdot z = x + (-b \bmod t + 1)\big),$$

where $B(x) = t^2 + t - 1$, and $B(y) = B(z) = t + 2$. By "bit blasting" the bounded variables into t-digits $\boldsymbol{x} = (x_0, x_1, x_2)$, $\boldsymbol{y} = (y_0, y_1, y_2)$, and $\boldsymbol{z} = (z_0, z_1, z_2)$, we obtain the formula

$$\begin{aligned} \exists \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \ : \ & \bigwedge\nolimits_{w \in \{x,y,z\}} \Big(0 \leq (w_2 \cdot t^2 + w_1 \cdot t + w_0) \leq B(w) \wedge \bigwedge\nolimits_{i \in \{0,1,2\}} 0 \leq w_i < t\Big) \\ & \wedge \ t \cdot (y_2 \cdot t^2 + y_1 \cdot t + y_0) = (x_2 \cdot t^2 + x_1 \cdot t + x_0) + (-a \bmod t) \\ & \wedge \ (t + 1) \cdot (z_2 \cdot t^2 + z_1 \cdot t + z_0) = (x_2 \cdot t^2 + x_1 \cdot t + x_0) + (-b \bmod t + 1). \end{aligned}$$

◀

▶ **Lemma 12.** *Let $\varphi_\varnothing(\boldsymbol{y}, \boldsymbol{z})$ be the formula obtained from φ by performing lines 1–6 of Algorithm 4. Then, the input formula $\exists \boldsymbol{w} \leq B : \varphi(\boldsymbol{w}, \boldsymbol{z})$ is equivalent to $\exists \boldsymbol{y} : \varphi_\varnothing(\boldsymbol{y}, \boldsymbol{z})$ and every (t-digit) variable in $\boldsymbol{y}$ has only linear occurrences in $\varphi_\varnothing$.*

The coefficients of the t-digits become integers (lines 7–17). This step is defined by the **while** loop of line 7, whose goal is to transform the formula $\varphi_\varnothing$ into an equivalent positive Boolean combination of equalities and inequalities in which all coefficients of $\boldsymbol{y}$ are integers.

▶ **Example 13.** Before delving into the details, let us illustrate the transformation on the equality $(t+1)\cdot(z_2\cdot t^2+z_1\cdot t+z_0)=(x_2\cdot t^2+x_1\cdot t+x_0)+(-b \bmod t+1)$ from Example 11. Grouping terms according to powers of t, we obtain:

$$-z_2\cdot t^3+(x_2-z_1-z_2)\cdot t^2+(x_1-z_0-z_1)\cdot t+(x_0-z_0)+(-b \bmod t+1)=0.$$

We symbolically perform a division with remainder on the sub-term $(-b \bmod t+1)$ concerning the free variables, rewriting it as $\left\lfloor\frac{-b \bmod t+1}{t}\right\rfloor\cdot t+((-b \bmod t+1) \bmod t)$. In the resulting equality, we notice that $(x_0-z_0)+((-b \bmod t+1) \bmod t)$ must be divisible by t. Since both x_0 and z_0 belong to $[0,t-1]$, only two multiples of t are possible: 0 and t. Consider the latter case: we can rewrite the equality as the conjunction of $t=(x_0-z_0)+((-b \bmod t+1) \bmod t)$ and $-z_2\cdot t^3+(x_2-z_1-z_2)\cdot t^2+(x_1-z_0-z_1)\cdot t+\left\lfloor\frac{-b \bmod t+1}{t}\right\rfloor\cdot t+t=0$. By dividing the second equality by t, the variables x_0, x_1 and z_0 end up appearing with integer coefficients only. Repeating this process guarantees that all quantified variables satisfy this property: the second iteration "frees" x_2 and z_1, and the third iteration handles the variable z_2. ◀

The **while** loop guesses some integers in line 11. Let R_i be the (finite) set of all sequences $\boldsymbol{s}$ of guesses from the first i iterations of the loop (so, $\boldsymbol{s}$ has length i), and let $\varphi_{\boldsymbol{s}}$ be the unique formula obtained from $\varphi_\varnothing$ after iterating the loop i times, using $\boldsymbol{s}$ as the sequence of guesses. (The subscript $\varnothing$ corresponds to the empty sequence of guesses; the only element in R_0.)

Together with proving that the **while** loop preserves formula equivalence (across non-deterministic branches), the critical parameter to track during the execution of the loop is the degrees of all coefficients of the t-digits $\boldsymbol{y}$. Showing that this parameter reaches 0 implies loop termination, and correctness of this step of the procedure. More formally, we inductively define the *$\boldsymbol{y}$-degree* $\deg(\boldsymbol{y},\varphi)$ of a positive Boolean combination of PrA$[t]$ constraints $\varphi(\boldsymbol{y},\boldsymbol{z})$, where the variables $\boldsymbol{y}=(y_1,\ldots,y_\ell)$ occur only linearly, as follows (below, $\sim\in\{\leq,=\}$):

- $\deg(\boldsymbol{y},\varphi)=\max\{\deg(f_i):i\in[1,\ell]\}$ if φ is an (in)equality $\sum_{i=1}^{\ell}f_i(t)\cdot y_i+\tau(\boldsymbol{z})\sim 0$;
- $\deg(\boldsymbol{y},\varphi)=\deg(\boldsymbol{y},\varphi_1)+\deg(\boldsymbol{y},\varphi_2)$ if φ is either $(\varphi_1\wedge\varphi_2)$ or $(\varphi_1\vee\varphi_2)$.

Then, the defining property of the **while** loop of line 7 can be stated as follows:

▶ **Lemma 14.** *Consider $\boldsymbol{s}\in R_i$ with $\deg(\boldsymbol{y},\varphi_{\boldsymbol{s}})>0$, and the set $G:=\{\boldsymbol{s}r\in R_{i+1}:r\in\mathbb{Z}\}$. Then,* **(i)** *$\varphi_{\boldsymbol{s}}$ is equivalent to $\bigvee_{\boldsymbol{r}\in G}\varphi_{\boldsymbol{r}}$, and* **(ii)** *$\deg(\boldsymbol{y},\varphi_{\boldsymbol{s}})>\deg(\boldsymbol{y},\varphi_{\boldsymbol{r}})$ for every $\boldsymbol{r}\in G$.*

Proof sketch. The **while** loop considers an (in)equality $\eta\sim 0$ from $\varphi_{\boldsymbol{s}}$ (line 7); inequalities are transformed into strict inequalities $\eta-1<0$ in line 8, as in this case the latter are easier to work with. Line 9 represents the term of the (in)equality as $\sigma(\boldsymbol{y})\cdot t+\rho(\boldsymbol{y})+\tau(\boldsymbol{z})$, where σ is linear, ρ is a linear term with coefficients in $\mathbb{Z}$, and $\tau(\boldsymbol{z})$ is non-shifted.

▶ Remark. Observe that line 17 will later replace $\eta\sim 0$ with $\sigma+r+\left\lfloor\frac{\tau}{t}\right\rfloor\sim 0$. If the latter (in)equality is considered again by the **while** loop at a later iteration, this replacement will produce the term $\left\lfloor\frac{\left\lfloor\frac{\tau}{t}\right\rfloor}{t}\right\rfloor$, which can be rewritten as $\left\lfloor\frac{\tau}{t^2}\right\rfloor$. We assume the algorithm to implicitly perform this rewriting, so that the term above can in fact be written as

$$\sigma(\boldsymbol{y})\cdot t+\rho(\boldsymbol{y})+\left\lfloor\frac{\tau(\boldsymbol{z})}{t^k}\right\rfloor,\qquad\text{where } k\geq 0,\text{ such that }\left\lfloor\frac{\tau}{t^0}\right\rfloor:=\tau. \tag{3}$$

Let us define $\eta'=\sigma(\boldsymbol{y})+\left\lfloor\frac{\tau(\boldsymbol{z})}{t^{k+1}}\right\rfloor$ and $\rho'=\rho(\boldsymbol{y})+\left(\left\lfloor\frac{\tau(\boldsymbol{z})}{t^k}\right\rfloor \bmod t\right)$, so that the term in Equation (3) is then equal to $\eta'\cdot t+\rho'$. (Note: ρ' is exactly the term in line 10.) We are now ready to perform the symbolic division by t. Indeed, since all variables in $\boldsymbol{y}$, as well as the term

$\left(\left\lfloor \frac{\tau(\boldsymbol{z})}{t^k} \right\rfloor \bmod t\right)$, belong to $[0, t-1]$, we conclude that $\rho' \in [-t \cdot N, t \cdot N]$ where $N := \|\rho'\|_1$. Line 11 guesses an integer r from the segment $[-N, N]$, which stands for the quotient of the division of ρ' by t. Each guess corresponds to a disjunct from the following two equivalences:

$$\begin{array}{lcl} \eta' \cdot t + \rho' = 0 & \iff & \bigvee_{r=-N}^{N} \big(\underbrace{\eta' + r = 0}_{\text{quotient of the division}} \wedge \underbrace{r \cdot t = \rho'}_{\text{formula } \gamma \text{ in the pseudocode}} \big), \\ \eta' \cdot t + \rho' < 0 & \iff & \bigvee_{r=-N}^{N} \big(\underbrace{\eta' + r + 1 \leq 0}_{\text{quotient of the division}} \wedge \underbrace{t \cdot r \leq \rho' \wedge \rho' < t \cdot (r+1)}_{\text{formula } \gamma \text{ in the pseudocode}} \big). \end{array}$$

These equivalences "perform" the symbolic division by t. In line 17, the algorithm substitutes the constraint $\eta \sim 0$ with the result of the division, that is, the conjunction of the quotient and the remaining part that is stored the formula γ (lines 12–16). Observe a key property of γ: in it, all the coefficients of the t-digits $\boldsymbol{y}$ only have integer coefficients, i.e., $\deg(\boldsymbol{y}, \gamma) = 0$.

Let χ_r be the formula obtained from φ by performing the replacement in line 17. This formula belongs to R_{i+1} and, moreover, $\varphi_{\boldsymbol{s}} \iff \bigvee_{r=-N}^{N} \chi_r$. We have $G = \{\boldsymbol{s}r : r \in [-N, N]\}$, and Item **(i)** is proved. To prove Item **(ii)**, observe that

$$\begin{aligned} \deg(\boldsymbol{y}, \chi_r) &= \deg(\boldsymbol{y}, \varphi_{\boldsymbol{s}}) - \deg(\boldsymbol{y}, \eta \sim 0) + \deg(\boldsymbol{y}, \eta' \sim 0) + \deg(\boldsymbol{y}, \gamma) \\ &= \deg(\boldsymbol{y}, \varphi_{\boldsymbol{s}}) - \deg(\boldsymbol{y}, \sigma \cdot t \sim 0) + \deg(\boldsymbol{y}, \sigma \sim 0) = \deg(\boldsymbol{y}, \varphi_{\boldsymbol{s}}) - 1. \end{aligned}$$ ◀

Elimination of t-digits (lines 18–26). By inductively applying Lemma 14, we deduce that the **while** loop performs at most $\deg(\boldsymbol{y}, \varphi_\varnothing)$ iterations, and that the disjunction (over all non-deterministic branches) of formulae $\varphi_{\boldsymbol{r}}$ obtained at the end of this loop is equivalent to $\varphi_\varnothing$. The constraints in each $\varphi_{\boldsymbol{r}}$ are (in)equalities $f(t) + \tau(\boldsymbol{z}) + \sum_{i=1}^{\ell} a_i \cdot y_i \sim 0$, where $a_1, \ldots, a_\ell \in \mathbb{Z}$, $f \in \mathbb{Z}[t]$, all $y_1, \ldots, y_\ell$ are t-digits, and τ is a non-shifted term of $\mathrm{PrA}[t]$.

The last step is to remove the t-digits $\boldsymbol{y}$ by appealing to BoundedQE (line 22). Recall that this algorithm requires all $\mathrm{PrA}[t]$ constraints in the input to be linear, while terms $\tau(\boldsymbol{z})$ in $\varphi_{\boldsymbol{r}}$ may contain (nested) occurrences of the functions $\left\lfloor \frac{\cdot}{t^d} \right\rfloor$ and $(\cdot \bmod f(t))$. In order to respect this specification, ElimBounded first replaces each non-shifted term $\tau(\boldsymbol{z})$ in $\varphi_{\boldsymbol{r}}$ with a fresh variable z', storing the substitution $[\tau(\boldsymbol{z})/z']$ in the map S (line 20). These terms are restored at the end of the procedure (line 26). Since the variables $\boldsymbol{z}'$ occur free in the formula in input to BoundedQE, and this procedure preserves formula equivalence (Lemma 9), the overall process remains sound.

The formula in input of BoundedQE has no divisibility constraints, and the eliminated variables $\boldsymbol{y}$ have integer coefficients. By Lemma 10 the output of each non-deterministic branch is a formula $\exists \boldsymbol{w}' \leq B' : \psi(\boldsymbol{w}', \boldsymbol{z}')$ where, for every variable w in $\boldsymbol{w}'$, the bound $B'(w)$ is an integer. We can thus replace w with a (guessed) integer $g \in [0, B'(w)]$ (lines 23–25). After restoring the terms stored in S, the resulting formula is quantifier-free and the disjunction over all outputs of ElimBounded is equivalent to the input formula.

▶ **Lemma 15.** *Algorithm 4 (ElimBounded) complies with its specification.*

This lemma implies Lemma 7, which was the last missing piece required to complete the proof of Theorem 2. To simplify the complexity arguments in the next section, we make use of the two observations in the following lemma, concerning the output of ElimBounded.

▶ **Lemma 16.** *In every output of ElimBounded, all functions $\left\lfloor \frac{\cdot}{t^d} \right\rfloor$ and $(\cdot \bmod f(t))$ are applied to non-shifted terms. In divisibility relations $(f(t) \mid \cdot)$, the divisor $f(t)$ is an integer.*

6 Solving satisfiability, universality and finiteness

Now that we have established that $\mathrm{PrA}[t]$ admits quantifier elimination, let us discuss the decision problems of *satisfiability*, *universality* and *finiteness* defined in Section 1. Without loss of generality, we add to the formula φ in input to these problems a prefix of existential quantifiers over all its free variables; thus assuming that φ is a sentence.

Applying our quantifier elimination procedure to the sentence φ results in a variable-free formula ψ whose truth only depends on the value taken by the parameter t. Furthermore, from Lemma 16, all occurrences of the functions $\left\lfloor \frac{\cdot}{t^d} \right\rfloor$ and $(\cdot \bmod f(t))$ are applied to the constant 0 (the only variable-free non-shifted term) and can thus be replaced with 0; and all divisibility relations $(f(t) \mid \cdot)$ are such that $f(t)$ is an integer. That is to say, ψ is a positive Boolean combination of univariate polynomial inequalities $g(t) \leq 0$, equalities $g(t) = 0$ and divisibility constraints $d \mid g(t)$, where $g \in \mathbb{Z}[t]$ and $d \in \mathbb{Z}$.

We study the solutions to such a univariate formula $\psi(t)$. First, recall that computing the set of all integer roots of a polynomial in $\mathbb{Z}[t]$ can be done in polynomial time:

▶ **Theorem 17** ([13, Theorem 1]). *There is a polynomial time algorithm that returns the set of integer roots of an input polynomial $f \in \mathbb{Z}[t]$.*

This theorem implies that the every integer root of $f \in \mathbb{Z}[t]$ has bit size polynomial in $\langle f \rangle$. Let $r_1 < \cdots < r_n$ be the roots of all the polynomials occurring in (in)equalities of ψ. These roots partition $\mathbb{Z}$ in $2n+1$ regions $(-\infty, r_1 - 1], \{r_1\}, [r_1 + 1, r_2 - 1], \{r_2\}, \ldots, \{r_n\}, [r_n + 1, \infty)$. The truth of all (in)equalities in ψ remains invariant for integers within the same region. Furthermore, the solutions to the divisibility constraints in ψ are periodic with period $p \coloneqq \operatorname{lcm}\{d : (d \mid \cdot) \text{ occurs in } \psi\} > 0$, i.e., setting $t = b$ satisfies the same divisibility constraints as $t = b + p$, for every integer b. Then, a solution to ψ (if it exists) can be found in the interval $[r_1 - p, r_n + p]$. Moreover, ψ has infinitely many solutions if and only if one solution lies in intervals $[r_1 - p, r_1 - 1]$ or $[r_n + 1, r_n + p]$. To recap:

▶ **Lemma 18.** *Let ψ be a positive Boolean combination of polynomial inequalities $g \leq 0$, equalities $g = 0$ and divisibility constraints $d \mid g$, where $g \in \mathbb{Z}[t]$ and $d \in \mathbb{Z}$. Then,*

1. *If ψ has a solution, then it has one of bit size polynomial in the size of ψ.*
2. *If ψ has a polynomial bit size solution that is either larger or smaller than all roots of the polynomials in (in)equalities of ψ, then ψ has infinitely many solutions, and vice versa.*

The complexity of the existential fragment. Lemma 18 implies decidability of all the decision problems of *satisfiability*, *universality* and *finiteness*. We now analyse their complexity for the existential fragment of $\mathrm{PrA}[t]$, establishing Theorem 3. For simplicity of the exposition, we keep assuming $t \geq 2$. Our reasoning can be extended in a straightforward way to all $t \in \mathbb{Z}$, following the discussion given at the beginning of Section 3.

First and foremost, we study the complexity of our quantifier elimination procedure.

▶ **Lemma 19.** *Algorithm 1 (PrA[t]-QE) runs in non-deterministic polynomial time.*

Proof idea. For the proof we track the evolution of the following parameters as Algorithm 1 executes, where φ is a formula, and B is a map used for the bounded quantification:

- $\mathit{atom}(\varphi) \coloneqq \big($number of occurrences of atomic formulae in $\varphi\big)$,
- $\mathit{vars}(\varphi) \coloneqq \big($number of variables in $\varphi\big)$,
- $\mathit{func}(\varphi) \coloneqq \big($number of occurrences of $\left\lfloor \frac{\cdot}{t^d} \right\rfloor$ and $(\cdot \bmod f(t))$ in an atomic formula of $\varphi\big)$,
- $\langle \mathit{const} \rangle(\varphi) \coloneqq \max\{\langle f \rangle : f \in \mathbb{Z}[t] \text{ occurs in } \varphi\}$,
- $\langle B \rangle \coloneqq \max\{0, \langle B(w) \rangle : w \text{ is in the domain of } B\}$.

For example, for the formula $\gamma \coloneqq (g \mid f_1 \cdot x + \left\lfloor \frac{f_2 \cdot x + f_3 \cdot \left\lfloor \frac{y}{t} \right\rfloor + f_4}{t} \right\rfloor + f_5)$, we have $\mathit{atom}(\gamma) = 1$, $\mathit{vars}(\gamma) = 2$, $\mathit{func}(\gamma) = 2$ (two occurrences of $\left\lfloor \frac{\cdot}{t} \right\rfloor$), $\langle \mathit{const} \rangle(\gamma) = \max\{\langle g \rangle, \langle f_1 \rangle, \ldots, \langle f_5 \rangle, \langle t \rangle\}$. The bit size of φ is polynomial in the values of these parameters.

One can show that throughout the procedure each of these parameters remain polynomially bounded with respect to all the parameters of the input formula. For instance, during the first two steps of ElimBounded, the number of variables in the manipulated formulae (φ

and γ) increases at most polynomially in $\langle B \rangle$, due to the bit blasting of the bounded variables. However, by the end of the procedure, it reduces to the number of free variables – as expected by a quantifier elimination procedure.

While tedious, this proof is not dissimilar to the other complexity analyses of quantifier elimination procedures for PrA; see, e.g., [30]. Once the evolution of the parameters is known, it is simple to show that Algorithm 1 runs in non-deterministic polynomial time. ◀

Our results on decision problems for existential formulae of PrA$[t]$ follow.

Proof of Theorem 3. Satisfiability. By Lemma 19 and Lemma 18.1, if the input sentence $\gamma \coloneqq \exists \boldsymbol{x} : \varphi(\boldsymbol{x})$ is satisfiable (equivalently, valid), then $[\![\gamma]\!]_k \neq \emptyset$ for some k of bit size polynomial in the size of γ. Observe that replacing t for k in γ yields an existential sentence of Presburger arithmetic of size polynomial in the size of γ. Checking satisfiability for ∃PrA is a well-known NP-complete problem [29]. We conclude that the satisfiability problem for existential formulae of PrA$[t]$ is also NP-complete.

Universality. By Lemma 19, Algorithm 1 can be implemented by a deterministic exponential time Turing machine T: given an input sentence γ, T computes an equivalent disjunction $\psi \coloneqq \bigvee_\beta \psi_\beta$ of exponentially many polynomial-size formulae ψ_β. By Lemma 16, each ψ_β is a positive Boolean combination of (in)equalities $g(t) \sim 0$ and divisibility constraints $d \mid g(t)$, with $g \in \mathbb{Z}[t]$ and $d \in \mathbb{N} \setminus \{0\}$. From Theorem 17, all roots $r_1 < \cdots < r_n$ of polynomials in (in)equalities of ψ are of bit size polynomial in the size of γ. However, the period $p \coloneqq \operatorname{lcm}\{d : (d \mid \cdot) \text{ occurs in } \psi\}$ may be of exponential bit size.

As a certificate asserting that γ is a *negative* instance, we can take ψ, the sequence of configurations reached by T when computing ψ from γ, and a number $k \in [r_1 - p, r_n + p]$. The certificate is verified in polynomial time (in its size) by checking that the sequence of configurations is a valid run of T computing ψ from γ, and that k is not a solution to $\psi(t)$. Since the certificate has exponential size, universality is in coNExp. (coNExp-hardness follows from the coNExp-hardness of the $\forall\exists^*$ fragment of PrA [16, 18].)

Finiteness. Equivalently, we show that the problem of checking whether a sentence γ is satisfiable for *infinitely* many instantiations of t is NP-complete. The proof of NP-hardness is trivial: for sentences without t, this problem is equivalent to the satisfiability problem for ∃PrA. For the NP membership, let us consider the formula $\bigvee_\beta \psi_\beta$ computed from an input sentence γ via Algorithm 1. If γ is satisfied by infinitely many values of t, then the same holds for least one of the formulae ψ_β. By Lemma 18.2, ψ_β has infinitely many solutions if and only if it has a solution k of bit size polynomial in the size of γ, such that k is either larger or smaller than all roots of polynomials appearing in (in)equalities of ψ_β. Then, as a certificate asserting that γ has infinitely many solutions we can provide ψ_β, the sequence of non-deterministic guesses made by Algorithm 1 to compute ψ_β from γ, and the value k. This certificate can be verified in polynomial time: first, run Algorithm 1 using the provided sequence of guesses, and check that the output is ψ_β. Then, compute (in polynomial time) all roots of the polynomials appearing in the (in)equalities of ψ_β, and verify that k is a solution to ψ_β that is either larger or smaller than all of them. ◀

References

1 Erwin H. Bareiss. Sylvester's identity and multistep integer-preserving Gaussian elimination. *Math. Comput.*, 1968. doi:10.1090/S0025-5718-1968-0226829-0.

2 Clark W. Barrett and Cesare Tinelli. Satisfiability modulo theories. In *Handbook of Model Checking*. 2018. doi:10.1007/978-3-319-10575-8_11.

3 Michael Benedikt, Dmitry Chistikov, and Alessio Mansutti. The complexity of Presburger arithmetic with power or powers. In *ICALP*, 2023. doi:10.4230/LIPICS.ICALP.2023.112.

4 Pascal Bergsträßer, Moses Ganardi, Anthony W. Lin, and Georg Zetzsche. Ramsey quantifiers in linear arithmetics. In *POPL*, 2024. doi:10.1145/3632843.

5 Tristram Bogart, John Goodrick, Danny Nguyen, and Kevin Woods. Parametric Presburger arithmetic: complexity of counting and quantifier elimination. *Math. Logic Quart.*, 2019. doi:10.1002/malq.201800068.

6 Tristram Bogart, John Goodrick, and Kevin Woods. Parametric Presburger arithmetic: logic, combinatorics, and quasi-polynomial behavior. *Discrete Analysis*, 2017. doi:10.19086/da.1254.

7 Itshak Borosh and Leon B. Treybig. Bounds on positive integral solutions of linear Diophantine equations. *Proc. Am. Math. Soc.*, 55(2):299–304, 1976. doi:10.2307/2041711.

8 Sheng Chen, Nan Li, and Steven V. Sam. Generalized Ehrhart polynomials. *Trans. Amer. Math. Soc.*, 2012. doi:10.1090/S0002-9947-2011-05494-2.

9 Dmitry Chistikov. An introduction to the theory of linear integer arithmetic. In *FSTTCS*, 2024. doi:10.4230/LIPIcs.FSTTCS.2024.1.

10 Dmitry Chistikov, Christoph Haase, and Alessio Mansutti. Quantifier elimination for counting extensions of Presburger arithmetic. In *FoSSaCS*, 2022. doi:10.1007/978-3-030-99253-8_12.

11 Dmitry Chistikov, Alessio Mansutti, and Mikhail R. Starchak. Integer linear-exponential programming in NP by quantifier elimination. In *ICALP*, 2024. (Full version of this paper: arXiv:2407.07083). doi:10.4230/LIPICS.ICALP.2024.132.

12 David C. Cooper. Theorem proving in arithmetic without multiplication. *Machine Intelligence*, 1972.

13 Felipe Cucker, Pascal Koiran, and Steve Smale. A polynomial time algorithm for diophantine equations in one variable. *J. Symb. Comput.*, 1999. doi:10.1006/jsco.1998.0242.

14 Florian Frohn and Jürgen Giesl. Satisfiability modulo exponential integer arithmetic. In *IJCAR*, 2024. doi:10.1007/978-3-031-63498-7_21.

15 John Goodrick. Bounding quantification in parametric expansions of Presburger arithmetic. *Arch. Math. Logic*, 2018. doi:10.1007/s00153-017-0593-0.

16 Erich Grädel. Dominoes and the complexity of subclasses of logical theories. *Ann. Pure Appl. Log.*, 1989. doi:10.1016/0168-0072(89)90023-7.

17 Eitan M. Gurari and Oscar H. Ibarra. An NP-complete number-theoretic problem. *J. ACM*, 26(3):567–581, 1979. doi:10.1145/322139.322152.

18 Christoph Haase. Subclasses of Presburger arithmetic and the weak EXP hierarchy. In *CSL-LICS*, pages 47:1–47:10, 2014. doi:10.1145/2603088.2603092.

19 Christoph Haase. A survival guide to Presburger arithmetic. *ACM SIGLOG News*, 2018. doi:10.1145/3242953.3242964.

20 Christoph Haase, Shankara Narayanan Krishna, Khushraj Madnani, Om Swostik Mishra, and Georg Zetzsche. An efficient quantifier elimination procedure for Presburger arithmetic. In *ICALP*, 2024. doi:10.4230/LIPIcs.ICALP.2024.142.

21 Peter Habermehl and Dietrich Kuske. On Presburger arithmetic extended with non-unary counting quantifiers. *Log. Methods Comput. Sci.*, 2023. doi:10.46298/lmcs-19(3:4)2023.

22 Toghrul Karimov, Florian Luca, Joris Nieuwveld, Joël Ouaknine, and James Worrell. On the decidability of presburger arithmetic expanded with powers. In *SODA*, 2025. doi:10.1137/1.9781611978322.89.

23 Aless Lasaruk and Thomas Sturm. Effective quantifier elimination for Presburger arithmetic with infinity. In *CASC*, 2009. doi:10.1007/978-3-642-04103-7_18.

24 Kenneth L. Manders and Leonard Adleman. NP-complete decision problems for binary quadratics. *Journal of Computer and System Sciences*, 16(2):168–184, 1978. doi:10.1016/0022-0000(78)90044-2.

25 Derek C. Oppen. A $2^{2^{2^{pn}}}$ upper bound on the complexity of Presburger arithmetic. *J. Comput. Syst. Sci.*, 1978. doi:10.1016/0022-0000(78)90021-1.

26 Mojżesz Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In *Comptes Rendus du I Congrès des Mathématiciens des Pays Slaves*, pages 92–101. 1929.

27 Cattamanchi R. Reddy and Donald W. Loveland. Presburger arithmetic with bounded quantifier alternation. In *STOC*, 1978. doi:10.1145/800133.804361.

28 Mikhail R. Starchak. Positive existential definability with unit, addition and coprimeness. In *ISSAC*, 2021. doi:10.1145/3452143.3465515.

29 Joachim von zur Gathen and Malte Sieveking. A bound on solutions of linear integer equalities and inequalities. *Proc. Am. Math. Soc.*, 1978. doi:10.1090/S0002-9939-1978-0500555-0.

30 Volker Weispfenning. The complexity of almost linear Diophantine problems. *J. Symb. Comput.*, 1990. doi:10.1016/S0747-7171(08)80051-X.

31 Volker Weispfenning. Complexity and uniformity of elimination in Presburger arithmetic. In *ISSAC*, pages 48–53, 1997. doi:10.1145/258726.258746.