One-Parametric Presburger Arithmetic has Quantifier Elimination

Alessio Mansutti
IMDEA Software Institute, Madrid, Spain

Mikhail R. Starchak
St. Petersburg State University, St. Petersburg, 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

Related Version 10.4230/LIPIcs.MFCS.2025.58

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* ($\mathrm{PrA}[t]$). 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 $\mathrm{PrA}[t]$ 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 $\mathrm{PrA}[t]$ 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 $\mathrm{PrA}[t]$ (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 $\mathrm{PrA}[t]$ 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 $\mathrm{PrA}[t]$] *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]: $\mathrm{PrA}[t]$ 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 $y = \lfloor \frac{x}{|f(t)|} \rfloor$ 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* $\langle \mathbb{Z}; 0, 1, +, x \mapsto t \cdot x, x \mapsto \lfloor \frac{x}{|f(t)|} \rfloor, \leq \rangle$.

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:

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

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, eliminating a single variable x from an existential block $\exists \boldsymbol{y} \exists x$ produces a formula $\bigvee_i \gamma_i$ with

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($\exists \boldsymbol{x} : \varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$)))

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:

▶ **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 $\exists \boldsymbol{w} \leq B : \psi \iff \exists \boldsymbol{w} : \bigwedge_{i=1}^m (0 \leq w_i \leq B(w_i)) \wedge \psi$.*

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** $\exists \boldsymbol{w} \leq B : \psi$

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, and technical details can be found in Appendix A. 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 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)

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 $\exists \boldsymbol{w}_\beta \leq B_\beta : \psi_\beta(\boldsymbol{w}_\beta, \boldsymbol{z})$ where ψ_β is a positive Boolean combination of linear PrA$[t]$ constraints.
Ensuring: $\bigvee_\beta \exists \boldsymbol{w}_\beta \leq B_\beta : \psi_\beta$ 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 $\tau \leq 0$ 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 $\eta[0/y] \leq 0$ **if** the sign $\pm$ is plus **else** with $\eta[0/y] \geq 0$
26: **return** $\exists \boldsymbol{w} \leq B : \varphi \wedge \chi$ **where** $\boldsymbol{w}$ is the sequence of keys of the map B

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: $\exists \boldsymbol{w} \le B : \varphi(\boldsymbol{w}, \boldsymbol{z})$, 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 $\exists \boldsymbol{w} \le B : \varphi$.

1: $\varphi \leftarrow \varphi \wedge \bigwedge_{w \in \boldsymbol{w}} (0 \le w) \wedge (w \le B(w))$
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 $y_0, \dots, y_M$ to $\boldsymbol{y}$
6: $\varphi \leftarrow \varphi[(t^M \cdot y_M + \dots + t \cdot y_1 + y_0) \,/\, w] \wedge \bigwedge_{i=0}^{M} ((0 \le y_i) \wedge (y_i \le t - 1))$
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 $\le$ **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: $\gamma \leftarrow (t \cdot r \le \rho) \wedge (\rho \le t \cdot (r + 1) - 1)$
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: $\exists \boldsymbol{w'} \le B' : \psi(\boldsymbol{w'}, \boldsymbol{z'}) \leftarrow \text{BoundedQE}(\boldsymbol{y}, \varphi(\boldsymbol{y}, \boldsymbol{z'}))$
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:

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

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

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

◀

▶ **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 $\exists \boldsymbol{w} \le B : \varphi(\boldsymbol{w}, \boldsymbol{z})$ 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 $\boldsymbol{y}=(y_1,...,y_\ell)$ 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 $N \coloneqq \|\rho'\|_1$. 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:

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

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

We prove (in Appendices A.3, B.4, and C) 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. ◀

Using this lemma, we can now obtain our results on the decision problems for existential formulae of PrA[t].

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. doi: 10.4230/LIPICS.ICALP.2024.132. Full version of this paper: arXiv:2407.07083.

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.

A Extended material for Section 4

The core of the arguments required for the correctness and complexity of BoundedQE stems directly from Appendix B in the full version of [11]. (The key being that the arguments in that appendix work on any integral domain, not only the integers.) Throughout the appendix, let $\exists \boldsymbol{x} : \varphi(\boldsymbol{x}, \boldsymbol{z})$ be the formula in input of BoundedQE, where φ is a positive Boolean combination of linear PrA[t] constraints, $\boldsymbol{x} = (x_1, \dots, x_n)$ and $\boldsymbol{z} = (x_{n+1}, \dots, x_g)$.

A.1 How coefficients evolve throughout the procedure

We start by studying the evolution of the coefficients in the formula φ throughout the procedure. For simplicity, fix a non-deterministic branch β of the algorithm, and let $\exists \boldsymbol{w}_\beta \leq B_\beta : \psi_\beta(\boldsymbol{w}_\beta, \boldsymbol{z})$ be its output. Observe that the branch β can be identified by at most $3n + 2$ guesses:

- The guess of a set $Z \subseteq \{f(t) : \text{the relation } (f(t) \mid \cdot) \text{ occurs in } \varphi\}$ (line 1).
- The guess of a sign $\pm$ among $-$ or $+$ (line 5).
- For every $i \in [0, n-1]$, consider the $(i+1)$th iteration of the **foreach** loop of line 10. This loop performs one guess in line 13, followed (if the execution reaches line 14) by two guesses: one for the equality $f_i(t) \cdot x_{i+1} + \tau_i = 0$ (line 14), and one for the sign $\pm_{i+1}$ among $-$ and $+$ (line 16).

We can thus represent β pictorially as follows:

$$\varphi \xrightarrow{(Z,\pm)} (\varphi_0, m_0, \ell_0, \pm_0, B_0, \chi_0) \xrightarrow{\delta_0} (\varphi_1, m_1, \ell_1, \pm_1, B_1, \chi_1) \xrightarrow{\delta_1} \dots$$
$$\dots \xrightarrow{\delta_{n-1}} (\varphi_n, \chi_n, m_n, \ell_n, \pm_n, B_n, \chi_n) \to \psi_\beta.$$

Here, each δ_i is $\top$ if the $(i+1)$th iteration of the **foreach** loop executes the **continue** statement of line 13, and otherwise it is the pair $(f_i(t) \cdot x_{i+1} + \tau_i = 0, \pm_{i+1})$. The tuple $(\varphi_0, m_0, \ell_0, \pm_0, \chi_0, B_0)$ refers to the state of the program after the execution of line 9 —hence after inserting the slack variables. In particular:

- φ_0 is the formula φ before the first iteration of the **foreach** loop of line 10.
- $m_0(t)$ is the polynomial $m(t)$ constructed in line 6.
- $\ell_0(t)$ is the polynomial $\ell(t)$ initialized as 1 in line 8. Following the discussion in Section 4, is the current "common factor" implementing the optimization of Bareiss's algorithm.
- $\pm_0$ is the sign $+$, i.e., the value of the "sign variable" $\pm$ from line 8. Throughout the procedure this variable matches the sign of the polynomial $\ell(t)$.
- B_0 is the empty map (line 8). Recall that B is the map used for the bounded quantifiers.
- χ_0 is the formula $m(t) > 0$ defined in line 7. As explained in Section 4, χ contains all signs guessed by the algorithm.

For every $i \in [1, n]$, the tuple $(\varphi_i, m_i, \ell_i, \pm_i, B_i, \chi_i)$ contains (in order) the formula φ, the polynomial $m(t)$, the polynomial $\ell(t)$, the value of the variable $\pm$, the map B, and the formula χ, that are obtained just *after* the ith iteration of the body of the **foreach** loop of line 10. Some observations:

- If $\delta_i = (f_i(t) \cdot x_{i+1} + \tau_i = 0, \pm_{i+1})$, then $\ell_{i+1}(t)$ is the polynomial $f_i(t)$ (see line 15), and $(\pm_{i+1} f_i(t)) > 0$ is one of the conjuncts in χ_{i+1} (see line 17).
- If $\delta_{i-1} = \top$, then $(\varphi_i, m_i, \ell_i, \pm_i, \chi_i) = (\varphi_{i-1}, m_{i-1}, \ell_{i-1}, \pm_{i-1}, \chi_{i+1})$, that is, only the map B is updated.

Below, let $\boldsymbol{y} = (y_1, \dots, y_r)$ be the slack variables added in line 9. For simplicity, we work under the following assumption:

(∗) The variables $x_1, \dots, x_n$ are rearranged in such a way that the δ_i that are equal to $\top$ all follow those of the form $(f_i(t) \cdot x_{i+1} + \tau_i = 0, \pm_i)$. That is to say, we assume that, for some $q \in [0, n]$, the **continue** statement of line 13 is only executed in the last $n - q$ iterations of the **foreach** loop of line 10.

Since for the moment we are only interested in how the formula φ is updated during the procedure, and $\delta_{i-1} = \top$ implies that $(\varphi_i, m_i(t), \ell_i(t), \pm_i, \chi_i) = (\varphi_{i-1}, m_{i-1}(t), \ell_{i-1}(t), \pm_{i-1}, \chi_{i-1})$, the assumption **(∗)** is without loss of generality. Therefore, $\delta_i = (f_i(t) \cdot x_{i+1} + \tau_i = 0, \pm_i)$ for every $i \in [0, q-1]$, and $\varphi_q = \varphi_{q+1} = \cdots = \varphi_n$. Observe also that, then, the variables $x_1, \dots, x_q$ are eliminated via substitution in line 21 (opportunely bounding a slack variable if necessary, see line 19), whereas variables $x_{q+1}, \dots, x_n$ will be bounded in line 12.

Suppose that φ_0 has e equalities and d divisibility constraints. Our aim is to associate to each formula φ_i among $\varphi_0, \dots, \varphi_q$ a matrix M_i with entries in $\mathbb{Z}[t]$. Intuitively, M_i will be storing all the atomic formulae of φ_i (with repetitions) in some specific order. First, let us note some straightforward observations on the updates performed in lines 21–22:

1. For every equality $f(t) \cdot x + \tau = 0$ the equality $(f(t) \cdot x + \tau = 0)[\![\frac{-\tau}{f(t)} \,/\, x]\!]$ is $(0 = 0)$.
2. There is a one-to-one mapping between the equalities and divisibility constraints in φ and those in $\varphi[\![\frac{-\tau}{f(t)} \,/\, x]\!]$ (i.e., the substitutions are applied locally to each atomic formula).
3. The divisions performed in line 22 preserves the one-to-one mapping of the substitutions (they are also applied locally to each atomic formula).

Because of these properties, we can fix an enumeration

$$(\rho_1 = 0),\ \dots,\ (\rho_e = 0),\ (h_{e+1}(t) \mid \rho_{e+1}),\ \dots,\ (h_{e+d}(t) \mid \rho_{e+d}), \tag{4}$$

of all the atomic formulae of φ_0. The last d elements of the enumeration correspond to the divisibility constraints in φ_0, in any order. The first e elements of the enumeration correspond to the equalities in φ_0, picked in such a way that, for every $i \in [1, k]$, $\rho_i = 0$ is in one-to-one correspondence with the equality $f_{i-1}(t) \cdot x_i + \tau_{i-1} = 0$ from φ_{i-1} that is guessed in line 14. In other words, executing the following program on $\rho_i = 0$ yields $f_{i-1}(t) \cdot x_i + \tau_{i-1} = 0$:

for j from 1 to $i - 1$ **do**
  apply $[\![\frac{-\tau_{j-1}}{f_{j-1}(t)} \,/\, x_j]\!]$ to $\rho_i = 0$
  divide $\rho_i = 0$ by the common factor $\ell_{j-1}(t)$.

In particular, $\rho_1 = 0$ is equal to $f_0(t) \cdot x_1 + \tau_0 = 0$. We denote the enumeration in Equation (4) by $E^{(0)}$, and write $E^{(0)}_j$ for its jth entry ($j \in [1, e + d]$).

Starting from the enumeration in Equation (4), we build enumerations $E^{(1)}, \dots, E^{(q)}$ for all the constraints in $\varphi_1, \dots, \varphi_q$, perhaps excluding some trivial equalities $0 = 0$. Let us first explicitly derive $E^{(1)}$ as an example. An enumeration for the atomic formulae of φ_1 can be obtained by applying to each element of the enumeration in Equation (4) the substitution $[\![\frac{-\tau_0}{f_0(t)} \,/\, x_1]\!]$ (in general we also need to divide by the common factor $\ell_{i-1}(t)$, but for $i = 1$ we have $\ell_0(t) = 1$), except that the resulting enumeration misses the divisibility $f_0(t) \mid \tau_0$ added in line 23. However, from Item 1 above we know that $(\rho_1 = 0)[\![\frac{-\tau_0}{f_0(t)} \,/\, x_1]\!]$ is equal to $(0 = 0)$. Hence, instead of adding a constraint to the enumeration, let us simply replace $\rho_1 = 0$ with $f_0(t) \mid \tau_0$. (This makes perfect sense in relation with Gaussian Elimination: we are using $\rho_1 = 0$ to remove x_1 from *all other* constraint; except for $\rho_1 = 0$ itself which we recast in terms of the divisibility $f_0(t) \mid \tau_0$.) The enumeration for φ_1 is then

$$(f_0(t) \mid \tau_0),\ E^{(0)}_2[\![\tfrac{-\tau_0}{f_0(t)} \,/\, x_1]\!],\ \dots,\ E^{(0)}_{e+d}[\![\tfrac{-\tau_0}{f_0(t)} \,/\, x_1]\!].$$

In general, for $i \in [1, q]$, let $E^{(i-1)} \coloneqq (E^{(i-1)}_1, \dots, E^{(i-1)}_{e+d})$ be the enumeration associated to φ_{i-1}. The enumeration $E^{(i)} \coloneqq (E^{(i)}_1, \dots, E^{(i)}_{e+d})$ associated to φ_i is given by the program:

for j from 1 to $e+d$, except for $j=i$ **do**
$E_j^{(i)} \leftarrow E_j^{(i-1)}[[\frac{-\tau_{j-1}}{f_{j-1}(t)} \,/\, x_j]]$
divide $E_j^{(i)}$ by $\ell_{i-1}(t)$
$E_i^{(i)} \leftarrow (f_{i-1}(t) \mid \tau_{i-1})$

The following properties follows from the definition of $E^{(i)}$ and the updates done in lines 21–23:

4. Every constraint from φ_i, except for trivial constraints $0=0$, occur in $E^{(i)}$.
5. Each constraint in $E^{(i)}$ occurs in φ_i.
   (Therefore, $E^{(i)}$ only features variables from $x_{i+1}, \dots, x_g$ and $\boldsymbol{y}$.)
6. The first i entries of $E^{(i)}$, as well as the last d entries, are divisibility constraints. All remaining entries are equalities.

We now move from the "enumeration view" of the constraints in $\varphi_0, \dots, \varphi_q$ to a "matrix view" from which we will be able to directly appeal to the results in [11]. For $i \in [0,q]$, we associate to φ_i the matrix $M_i = \left[G_i \mid A_i \mid \boldsymbol{c}_i \mid S_i \mid D_i\right]$ (with entries over $\mathbb{Z}[t]$), where

- G_i is a $(e+d) \times i$ (rectangular) diagonal matrix having in position $(j,j) \in [1,i] \times [1,i]$ the divisor of the divisibility constraint $E_j^{(i)}$.
- A_i is a $(e+d) \times (g-i)$ matrix. For every $j \in [1, e+d]$ and $k \in [1, g-i]$, the entry of A_i in position (j,k) is the coefficient of the variable x_{i+k} in the constraint $E_j^{(i)}$.
- $\boldsymbol{c}_i$ is a $(e+d)$ column vector. For every $j \in [1, e+d]$, the jth entry of $\boldsymbol{c}_i$ correspond to the constant (i.e., the element in $\mathbb{Z}[t]$ that does not appear as a coefficient of any variable) of the term in the constraint $E_j^{(i)}$. This term is τ for divisibility constraints $h(t) \mid \tau$.
- S_i is a $(e+d) \times r$ matrix. For every $j \in [1, e+d]$ and $k \in [1, r]$, the entry of S_i in position (j,k) is the coefficient of the slack variable y_k in the constraint $E_j^{(i)}$.
- D_i is a $(e+d) \times d$ matrix. Its first e rows are all zero. The last d rows form a diagonal matrix: the entry in position $(e+j, j)$ with $j \in [1,d]$ is the divisor of the divisibility constraint $E_{e+j}^{(i)}$.

The matrices $M_0, \dots, M_q$ are just another way of encoding the constraints in the enumerations $E^{(0)}, \dots, E^{(q)}$. The key observation from [11] is that $M_1, \dots, M_q$ also corresponds to the sequence of matrices obtained by performing Bareiss algorithm on the first q columns of the matrix, diagonalizing them (equivalently, removing $x_1, \dots, x_q$). Following [1], [11, Appendix B] characterizes the entries of $M_1, \dots, M_q$ from the entries of M_0. This in turn characterizes all polynomials in $\mathbb{Z}[t]$ occurring in the formulae $\varphi_1, \dots, \varphi_q$.

▶ **Proposition 20** ([11, Appendix B]). *For every $i \in [1,q]$, the matrix M_i satisfies:*

- **I.** *For every row $j \in [1,i]$ and column $k \in [1, g+1+r+d]$, the entry in position (j,k) of M_i is equal to the determinant of the matrix $M_0[1, \dots, i;\ 1, \dots, j-1, k, j+1, \dots, i]$. (Hence, G_i is a diagonal matrix whose non-zero entries are all $\det(M_0[1, \dots, i; 1, \dots, i])$.)*
- **II.** *For row $j \in [i+1, e+d]$ and column $k \in [1,i]$, the entry in position (j,k) of M_i is 0.*
- **III.** *For every row $j \in [i+1, e+d]$ and column $k \in [i+1, g+1+r+d]$, the entry in position (j,k) of M_i is equal to the determinant of the matrix $M_0[1, \dots, i, j;\ 1, \dots, i, k]$.*

Moreover:

- **IV.** *The coefficient $f_i(t)$ of x_{i+1} in the equality guessed in line 14 during the $(i+1)$th iteration of the **foreach** loop of line 10 is $\det(M_0[1, \dots, i+1; 1, \dots, i+1])$.*
- **V.** *The factor used for the division of line 22 during the $(i+1)$th iteration of the **foreach** loop of line 10 is $f_{i-1}(t)$ (postulating $f_{-1}(t) \coloneqq 1$). These divisions are without remainder.*

Where to find the proof. Item (a) and Item (b) in Lemma 9 of the full version of [11] imply Item III, Item IV, and the fact that factor used for the division of line 22 during the $(i+1)$th iteration of the **foreach** loop of line 10 is $f_{i-1}(t)$. The proofs of Item (a) and Item (b)

follow [1], which iteratively uses the Desnanot–Jacobi identity. Lemma 9 also shows that the divisions are all without remainder for all constraints in the formulae $\varphi_0, \ldots, \varphi_q$, except for the divisibility constraints added in line 23 (see Item (c) of Lemma 9). Under the assumption that the divisibilities performed in line 22 are also without remainder for the added divisibility constraints, Item (a) from Lemma 14 of [11] establishes Item I. The assumption is ultimately discharged in Lemma 15 of [11]: all divisions performed in line 23 are without remainder (which completes the proof of Item V). Lastly, Item II follows directly from the definition of G_i, which is a $(e + d) \times i$ diagonal matrix.

A note: for the most part, it is straightforward to see that the aforementioned lemmas in [11, Appendix B] work on any integral domain (a non-zero commutative ring in which the product of non-zero elements is non-zero). The only non-trivial step is given in the proof of [11, Lemma 14]. There, the proof requires taking an inverse N^{-1} of some integer matrix N with non-zero determinant. Recall that a matrix is invertible over the integers only if the determinant is ± 1. However, because we only know $\det N \neq 0$, this step of the proof considers matrices over the rationals $\mathbb{Q}$ instead, so that N^{-1} is guaranteed to exist. (This is not a problem, because in any case the proof of [11, Lemma 14] then continues by showing that the entries of M_i are indeed the ring elements given in the statement of Proposition 20.) For an arbitrary integral domain (such as $\mathbb{Z}[t]$), instead of $\mathbb{Q}$ we can use its *field of fractions*, that is the smallest field in which the integral domain can be embedded. Such a field exists for any integral domain. ◀

Proposition 20 will be very useful for both the proofs of correctness and complexity of the algorithm. We will also need the following property on the coefficients of the slack variables.

▶ **Lemma 21.** *For all $i \in [0, q]$, all slack variables in φ_i that are not assigned a value by B_i occur in φ_i with identical coefficients, namely $f_{i-1}(t)$ (setting $f_{-1}(t) \coloneqq 1$). Moreover, each of these slack variables occur in a single constraint of φ_i, it is an equality.*

Proof. See Lemmas 11 and 12 from [11] a fully detailed proof. Briefly, the second statement of the lemma is simple to show: it is clearly true for φ_0, and in all substitutions $[\![\frac{-\tau}{f(t)} \,/\, x]\!]$ performed by the algorithm, the term τ only feature slack variables (if any) for which the map B already assigns a value. Since substitutions are local to each atomic formula, it is thus not possible for an "unassigned" slack variable y to end up in multiple constraints. Then, in the column k of M_i corresponding to the variable y, there is at most one non-zero coefficient, and it is located in a row j among the first e (y occurs in an equality). More precisely, because of how the matrix S_i is defined, $j > i$ (y is unassigned). Then, we are in the case of Proposition 20.III, and the coefficient of y in φ_i is $\det(M_0[1, \ldots, i, j;\ 1, \ldots, i, k])$. The last column of $M_0[1, \ldots, i, j;\ 1, \ldots, i, k]$ is $(0, \ldots, 0, 1)$, and so this determinant is equal to $\det(M_0[1, \ldots, i;\ 1, \ldots, i])$. By Proposition 20.IV, this is equal to $f_{i-1}(t)$. ◀

▶ **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.*

Proof. We look at the polynomial $m(t)$ initially defined in line 6 and used to define the range of all maps B_β. In any run of the algorithm, this polynomial is built from polynomials appearing as divisors of divisibility constraints in φ (which are here assumed to be integers), together with polynomials $f_i(t)$ considered in Item IV of Proposition 20. Since in φ all coefficients of variables in $\boldsymbol{x}$ are integers, from Proposition 20 and by definition of the matrix M_q we conclude that all $f_i(t)$ are integers. Therefore, $m(t)$ is an integer throughout the procedure, and the statement follows. ◀

A.2 Correctness of BoundedQE (proof of Lemma 9)

Recall that the input of BoundedQE is a $\exists \boldsymbol{x} : \varphi(\boldsymbol{x}, \boldsymbol{z})$, where φ is a positive Boolean combination of linear PrA[t] constraints, $\boldsymbol{x} = (x_1, \dots, x_n)$ and $\boldsymbol{z} = (x_{n+1}, \dots, x_g)$.

For the proof of correctness, we would like to proceed as in Appendix A.1 and fix a non-deterministic branch β of the algorithm, which produces a path of the form:

$$\varphi \xrightarrow{(Z,\pm)} (\varphi_0, m_0, \ell_0, \pm_0, B_0, \chi_0) \xrightarrow{\delta_0} (\varphi_1, m_1, \ell_1, \pm_1, B_1, \chi_1) \xrightarrow{\delta_1} \dots \\ \dots \xrightarrow{\delta_{n-1}} (\varphi_n, m_n, \ell_n, \pm_n, B_n, \chi_n) \to \psi_\beta$$

where the various objects in it are as defined in Appendix A.1. Correctness is however a global property that ranges all non-deterministic executions, and we will thus need to consider the whole computation tree of the procedure. We will still use the path above as a reference (it will soon be clear in what sense). An important detail: in this appendix, we are not working under the assumption **(∗)** used to derive the bounds on φ.

We divide the proof of correctness into three steps:

- Step 1 involves the first edge of paths as the one above, concerning the guesses of $(Z, \pm)$.
- Step 2 concerns the iterations of the **foreach** loop of line 10, involving the guesses δ_i.
- Step 3 concerns all that happens after the **foreach** loop of line 10 completes. (In this step the algorithm is deterministic.)

Step 1 (lines 1–9). Let us focus on the first edge of the path, that is,

$$\varphi \xrightarrow{(Z,\pm)} (\varphi_0, m_0, \ell_0, \pm_0, B_0, \chi_0).$$

We refer the reader to Appendix A.1 for the description of $(\varphi_0, m_0, \ell_0, \pm_0, B_0, \chi_0)$ (Lemma 22 below recalls properties of this tuple that suffice for this appendix). In the guess $(Z, \pm)$, recall that Z is a subset of $E \coloneqq \{f(t) : \text{the relation } (f(t) \mid \cdot) \text{ occurs in } \varphi\}$, and $\pm$ is the sign used to define $m_0(t) \coloneqq \pm \prod_{g \in E \setminus Z} g$. The formula φ_0 already contains all slack variables $\boldsymbol{y} = (y_1, \dots, y_r)$ added in line 9 (note that the number of slack variables does not depend on the guessed Z and $\pm$, because the number of inequalities in φ is the same across non-deterministic branches).

Let us write S_0 for the set of all tuples $(\varphi_0, m_0(t), \ell_0(t), \pm_0, B_0)$ obtained by considering all the different choices for the pair $(Z, \pm)$. We start with some simple observations:

▶ **Lemma 22.** *for every $(\varphi_0, m_0, \ell_0, \pm_0, B_0, \chi_0) \in S_0$:*

1. *φ_0 is a positive Boolean combination linear of equalities and divisibility constraints.*
2. *The formula χ_0 is the inequality $m_0(t) > 0$.*
3. *For every divisibility $(f(t) \mid \cdot)$ in φ_0, the divisor $f(t)$ is a factor of $m_0(t)$.*
4. *The polynomial $\ell_0(t)$ is the constant 1, and $\pm_0$ is the symbol $+$.*
5. *The map B_0 is empty.*

Proof. The proof is straightforward (see lines 6–9, or the discussion in Appendix A.1). ◀

The following lemma express the key "equivalence" between φ and the elements in S_0:

▶ **Lemma 23.** *The formula φ is equivalent to $\bigvee_{(\varphi_0, m_0, \ell_0, \pm_0, B_0, \chi_0) \in S_0} \boldsymbol{y} \in \mathbb{N} : \varphi_0 \wedge \chi_0$.*

Proof. Below, given $Z \subseteq E$, let us write φ_Z for the formula obtained from φ by replacing every divisibility $f(t) \mid \tau$, where $f \in Z$, with $\tau = 0$. We also write φ'_Z for the formula obtained from φ_Z by replacing each inequality $\tau \leq 0$ with $\tau + y = 0$, where y is a slack variable from $\boldsymbol{y}$

(distinct inequalities receive distinct slack variables). Following lines 1–9, we see that for every $(\varphi_0, m_0, \ell_0, \pm_0, B_0, \chi_0) \in S_0$, the formula φ_0 is of the form $(\varphi'_Z \wedge \bigwedge_{f\in Z} f(t) = 0)$, for some $Z \subseteq E$, and moreover $m_0(t)$ is equal to $\pm(\prod_{g\in E\setminus Z} g(t)) > 0$, for some sign $\pm$ among $+$ and $-$. The following chain of equivalences then proves the lemma:

$$
\begin{aligned}
& \varphi \\
\iff & \bigvee_{Z\subseteq E} \Big(\varphi_Z \wedge \bigwedge_{f\in Z} f(t) = 0 \wedge \bigwedge_{g\in E\setminus Z} g(t) \neq 0\Big) \\
& \qquad\qquad \text{(by definition of divisibility constraint)} \\
\iff & \bigvee_{Z\subseteq E} \bigvee_{\pm\in\{+,-\}} \Big(\varphi_Z \wedge \bigwedge_{f\in Z} f(t) = 0 \wedge (\prod_{g\in E\setminus Z} g(t)) \neq 0\Big) \\
& \qquad\qquad \text{(since } a \neq 0 \wedge b \neq 0 \iff a \cdot b \neq 0) \\
\iff & \bigvee_{Z\subseteq E} \bigvee_{\pm\in\{+,-\}} \Big(\varphi_Z \wedge \bigwedge_{f\in Z} f(t) = 0 \wedge m(t) > 0\Big) \\
& \qquad\qquad \text{(guessing a sign for } \textstyle\prod_{g\in E\setminus Z} g(t)\text{, and by definition of } m(t)) \\
\iff & \bigvee_{Z\subseteq E} \bigvee_{\pm\in\{+,-\}} \Big((\exists \boldsymbol{y} \in \mathbb{N} : \varphi'_Z) \wedge \bigwedge_{f\in Z} f(t) = 0 \wedge m(t) > 0\Big) \\
& \qquad\qquad \text{(because } \tau \leq 0 \iff \exists y \in \mathbb{N} : \tau + y = 0\text{, and moreover } \varphi_Z \text{ is a} \\
& \qquad\qquad \text{positive Boolean combination of linear PrA}[t] \text{ constraints, hence} \\
& \qquad\qquad \text{we can push } \exists y \text{ outside the scope of all its Boolean connectives)} \\
\iff & \bigvee_{(\varphi_0,m_0,\ell_0,\pm_0,B_0,\chi_0)\in S_0} \boldsymbol{y} \in \mathbb{N} : \varphi_0 \wedge \chi_0 \qquad \text{(by def. of } S_0\text{, } \chi_0\text{, and } \varphi_0) \quad \blacktriangleleft
\end{aligned}
$$

Step 2 (lines 10–23). Let us write S_i for the set of all tuples $(\varphi_i, m_i, \ell_i, \pm_i, B_i, \chi_i)$ obtained across all non-deterministic branches of Algorithm 3, after the **foreach** loop of line 10 has been executed i times. Note that S_0 coincides with the homonymous set defined before Lemma 22. Recall that this loop executes n times, considering the variables $x_1, \ldots, x_n$ (in this order).

Let $i \in [0, n-1]$. Back to our "path view", we will now look at edges of the form

$$(\varphi_i, m_i, \ell_i, \pm_i, B_i, \chi_i) \xrightarrow{\delta} (\varphi_{i+1}, m_{i+1}, \ell_{i+1}, \pm_{i+1}, B_{i+1}, \chi_{i+1}),$$

where:

- the tuple $(\varphi_i, m_i, \ell_i, \pm_i, B_i, \chi_i)$ belongs to S_i,
- δ is either $\top$ (this case occurs when the $(i+1)$th iteration of the **foreach** loop executes the **continue** statement of line 13) or a pair $(f_i(t) \cdot x_{i+1} + \tau_i = 0,\ \pm_{i+1})$ (corresponding to the two guesses of lines 14 and 16), and
- $(\varphi_{i+1}, m_{i+1}, \ell_{i+1}, \pm_{i+1}, B_{i+1}, \chi_{i+1}) \in S_{i+1}$.

Let $i \in [0, n]$, and consider $\Phi := (\varphi_i, m_i, \ell_i, \pm_i, B_i, \chi_i) \in S_i$. Let us denote by $\boldsymbol{y'}$ the slack variables from $\boldsymbol{y}$ that are *unassigned* in B_i (that is, not in its domain), and $\boldsymbol{v}$ for all the variables that are assigned in B_i. We write $F(\Phi)$ for the formula:

$$F(\Phi) := \exists \boldsymbol{y'} \in \mathbb{N}\, \exists \boldsymbol{v} \leq B\, (\varphi_i \wedge \chi_i).$$

Observe that then the equivalence in Lemma 23 can be states as $\varphi \iff \bigvee_{\Phi\in S_0} F(\Phi)$, since in this case B_0 is the empty map (Lemma 22).

The goal for this step is to prove the following result:

▶ **Lemma 24.** *Let $i \in [0, n-1]$ and $\Phi \in S_i$. There is a subset $G \subseteq S_{i+1}$ satisfying*

$$\exists x_{i+1} F(\Phi) \iff \bigvee\nolimits_{\Psi\in G} F(\Psi).$$

As done in Step 1, let us start with some simple facts:

▶ **Lemma 25.** *Let $i \in [0, n]$ and consider a tuple $(\varphi_i, m_i, \ell_i, \pm_i, B_i, \chi_i) \in S_i$. We have:*

1. *φ_i is a positive Boolean combination of linear equalities and divisibility constraints.*
2. *For every divisibility $(g(t) \mid \cdot)$ in φ_i, the divisor $g(t)$ is a factor of $m_i(t)$.*
3. *The formula χ_i is a conjunction of inequalities of the form $h(t) > 0$, with $h \in \mathbb{Z}[t]$.*
4. *The formula χ_i entails both $m_i(t) > 0$ and $\pm_i \ell_i(t) > 0$.*

Proof. The proof is by induction on i.

base case: $i = 0$. Directly from Lemma 22.

induction step: $i \geq 1$. Consider $(\varphi_i, m_i, \ell_i, \pm_i, B_i, \chi_i) \in S_i$, as well as the label δ of an edge such that there is $(\varphi_{i-1}, m_{i-1}, \ell_{i-1}, \pm_{i-1}, B_{i-1}, \chi_{i-1}) \in S_{i-1}$ satisfying

$$(\varphi_{i-1}, m_{i-1}, \ell_{i-1}, \pm_{i-1}, B_{i-1}, \chi_{i-1}) \xrightarrow{\delta} (\varphi_i, m_i, \ell_i, \pm_i, B_i, \chi_i).$$

By induction hypothesis, the tuple $(\varphi_{i-1}, m_{i-1}, \ell_{i-1}, \pm_{i-1}, B_{i-1}, \chi_{i-1})$ satisfies Items 1–4. We proceed by cases on δ:

case: $\delta = \top$. We have $(\varphi_i, m_i, \ell_i, \pm_i, \chi_i) = (\varphi_{i-1}, m_{i-1}, \ell_{i-1}, \pm_{i-1}, \chi_{i-1})$ (because the **foreach** loop executes the **continue** statement of line 13). Items 1–4 are thus satisfied.

case: $(f(t) \cdot x_i + \tau = 0, \pm)$. Following lines 15, 17 and 18, we have:

$$\ell_i \coloneqq f(t), \qquad m_i \coloneqq \pm f(t) \cdot m_{i-1}(t),$$
$$\pm_i \coloneqq \pm, \qquad \chi_i \coloneqq (\chi_{i-1} \wedge \pm f(t) > 0).$$

proof of Item 1: By induction hypothesis, φ_{i-1} is a positive Boolean combination of linear equalities and divisibility constraints. The formula φ_i is obtained from φ_{i-1} by applying lines 21–23, which preserve this property.

proof of Item 2: By induction hypothesis, for every divisibility $(g(t) \mid \cdot)$ in φ_{i-1}, the divisor $g(t)$ is a factor of $m_{i-1}(t)$. Line 21 multiplies each divisor $g(t)$ of φ_{i-1} by $f(t)$. The resulting polynomial $g(t) \cdot f(t)$ is a factor of $m_i(t) = \pm f(t) \cdot m_{i-1}(t)$. Line 22 divides $g(t) \cdot f(t)$ by the common factor $\ell_{i-1}(t)$. From Proposition 20.V these polynomial divisions are without remainder, and therefore the result is still a factor of $m_i(t)$. Lastly, line 23 adds the divisibility $(f(t) \mid \tau)$; whose divisor is again a factor of $m_i(t) = \pm f(t) \cdot m_{i-1}(t)$.

proof of Item 3: Follows by definition of χ_i and the fact that χ_{i-1} satisfies Item 3.

proof of Item 4: Directly from the its definition, χ_i implies both $\pm_i \ell_i > 0$ and $m_i > 0$ (the latter because χ_{i-1} implies $m_{i-1} > 0$ by induction hypothesis). ◀

Let us consider a tuple $(\varphi_i, m_i, \ell_i, \pm_i, B_i, \chi_i) \in S_i$. For the time being, it is more convenient to reason purely in terms of formulae equivalences, neglecting their connections with the operations performed by Algorithm 3. We will come back to Algorithm 3 later.

We rely on the following notion:

▶ **Definition 26** (Partial formulae of φ_i)**.** *Let I, A_0 and A_1 be a partition of the set $T \coloneqq \{(f(t), \tau) : \text{the equation } f(t) \cdot x_{i+1} + \tau = 0 \text{ occurs in } \varphi_i\}$. Let us write $(\varphi_i|_{I,T})$ for the formula obtained from φ_i by replacing with $\bot$ every equation $f(t) \cdot x_{i+1} + \tau = 0$ such that $(f(t), \tau) \in I$, and with $\top$ every equation $f(t) \cdot x_{i+1} + \tau = 0$ such that $(f(t), \tau) \in T \setminus I$.*

The following formula is said to be a partial formula *of φ_i:*

$$(\varphi_i|_{I,T}) \wedge \bigwedge_{(f(t),\tau) \in A_0} (f(t) = 0 \wedge \tau = 0) \wedge \bigwedge_{(f(t),\tau) \in A_1} (f(t) \neq 0 \wedge f(t) \cdot x_{i+1} + \tau = 0).$$

We denote by $\mathcal{P}(\varphi_i)$ the set of all partial formulae *of φ_i. Given $\gamma \in \mathcal{P}(\varphi_i)$, we write $I(\gamma)$, $A_0(\gamma)$ and $A_1(\gamma)$ for the sets I, A_0 and A_1 used to define γ, respectively.*

The idea behind the notion of partial formulae is the following. In any solution ν to φ_i, some atomic formulae are satisfied (below, "active") and some are not (below, "inactive"). (We recall in passing that ν is a map assigning an integer to each variable in $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$, and to the parameter t, such that the integers assigned to the slack variables $\boldsymbol{y}$ are non-negative.) The formula $(\varphi_i|_{I,T})$ in Definition 26 guesses the "activity" of each equality featuring x_{i+1}; replacing inactive equalities with $\bot$ and active equalities with $\top$. Among the active equalities $f(t) \cdot x_{i+1} + \tau = 0$ (with respect to ν), we distinguish two types, depending on whether $f(\nu(t)) = 0$. In Definition 26, the first type corresponds to the set A_0: here $f(t)$ is guessed to be 0, and so $f(t) \cdot x_{i+1} + \tau = 0$ becomes equivalent to $f(t) = 0 \wedge \tau = 0$. The second type corresponds to the set A_1: here $f(t)$ is guessed non-zero, and therefore $f(t) \cdot x_{i+1} + \tau = 0$ becomes (trivially) equivalent to $f(t) \neq 0 \wedge f(t) \cdot x_{i+1} + \tau = 0$.

The next two lemmas show that no information is lost when moving from φ_i to $\mathcal{P}(\varphi_i)$.

▶ **Lemma 27.** *Every formula $\gamma \in \mathcal{P}(\varphi_i)$ implies φ_i.*

Proof. Let $\gamma \in \mathcal{P}(\varphi_i)$, and $I \coloneqq I(\gamma)$, $A_0 \coloneqq A_0(\gamma)$, $A_1 \coloneqq A_1(\gamma)$ and $T \coloneqq I \cup A_0 \cup A_1$. Following Definition 26, the formula $(\varphi_i|_{I,T})$ has the same Boolean structure as φ_i, the only difference is that all equalities from T are replaced with $\bot$ or $\top$. Moreover, since φ_i is a positive Boolean combination, both φ_i and $(\varphi_i|_{I,T})$ only contain the connectives $\vee$ and $\wedge$. Let us then refer to subformulae of φ_i and $(\varphi_i|_{I,T})$ by using finite words from the alphabet $\{L, R\}$, where L selects the left disjunct/conjunct of a Boolean connective, and R selects the right one. Given a word $w \in \{L, R\}^*$, we write $[\varphi_i]_w$ for the subformula of φ_i corresponding to w (and proceed similarly for $(\varphi_i|_{I,T})$). For example, $\big[(a \vee b) \wedge ((c \vee d) \wedge e)\big]_{RL} = c \vee d$. Therefore, $[\varphi_i]_\epsilon = \varphi_i$ and $[(\varphi_i|_{I,T})]_\epsilon = (\varphi_i|_{I,T})$, where ϵ stands for the empty word. We leave $[\varphi_i]_w$ undefined whenever w does not lead to a subformula of φ_i (e.g., $\big[(a \vee b) \wedge ((c \vee d) \wedge e)\big]_{LLL}$ is undefined), and write $\mathcal{W}$ for the set of words $w \in \{L, R\}^*$ for which $[\varphi_i]_w$ is defined. This also corresponds to the set of words $w \in \{L, R\}^*$ for which $[(\varphi_i|_{I,T})]_w$ is defined.

Let ν be a solution to γ. We prove that for every $w \in \mathcal{W}$, if ν satisfies $[(\varphi_i|_{I,T})]_w$ then it also satisfies $[\varphi_i]_w$. Since ν satisfies $(\varphi_i|_{I,T})$ (as it satisfies γ), this suffices to conclude that ν satisfies φ_i, as required. The proof is by induction on the words of $\mathcal{W}$, with induction hypothesis asserting that the statement holds for all strictly longer words in this set.

base case: $w \in \mathcal{W}$ is not a strict prefix of some word in $\mathcal{W}$. In this case, $\alpha \coloneqq [\varphi_i]_w$ and $\beta \coloneqq [(\varphi_i|_{I,T})]_w$ are atomic formulae. We split the proof into four cases:

- **case: α is not an equation $f(t) \cdot x_{i+1} + \tau = 0$.** We have $\alpha = \beta$ (by def. of $(\varphi_i|_{I,T})$), and therefore the statement trivially follows.
- **case: α is an equation $f(t) \cdot x_{i+1} + \tau = 0$ and $(f(t), \tau) \in I$.** We have $\beta = \bot$ (by definition of $(\varphi_i|_{I,T})$). Hence, ν does not satisfy β, and the statement follows.
- **case: α is an equation $f(t) \cdot x_{i+1} + \tau = 0$ and $(f(t), \tau) \in A_0$.** By def. of $(\varphi_i|_{I,T})$ we have $\beta = \top$. Since ν satisfies γ, it also satisfies $f(t) = 0 \wedge \tau = 0$ (by definition of γ). This conjunction entails $f(t) \cdot x_{i+1} + \tau = 0$; and so ν satisfies α.
- **case: α is an equation $f(t) \cdot x_{i+1} + \tau = 0$ and $(f(t), \tau) \in A_1$.** Since ν satisfies γ, it also satisfies $f(t) \neq 0 \wedge f(t) \cdot x_{i+1} + \tau = 0$ (by definition of γ). Hence, ν satisfies α.

induction step: $w \in \mathcal{W}$ is a strict prefix of some word in $\mathcal{W}$. From the definition of $\mathcal{W}$, we have $[\varphi_i]_w = [\varphi_i]_{wL} \oplus [\varphi_i]_{wR}$ and $[(\varphi_i|_{I,T})]_w = [(\varphi_i|_{I,T})]_{wL} \oplus [(\varphi_i|_{I,T})]_{wR}$, where $\oplus \in \{\wedge, \vee\}$. We split the proof into two cases, depending on $\oplus$.

- **case: $\oplus$ is $\wedge$.** Suppose that ν satisfies $[(\varphi_i|_{I,T})]_{wL} \wedge [(\varphi_i|_{I,T})]_{wR}$. By induction hypothesis, ν satisfies $[\varphi_i]_{wL}$ and $[\varphi_i]_{wR}$. That is, ν satisfies $[\varphi_i]_{wL} \wedge [\varphi_i]_{wR}$.

case: $\oplus$ is $\vee$. Suppose that ν satisfies $[(\varphi_i|_{I,T})]_{wL} \vee [(\varphi_i|_{I,T})]_{wR}$. There is $D \in \{L, R\}$ such that ν satisfies $[(\varphi_i|_{I,T})]_{wD}$. By induction hypothesis, ν satisfies $[\varphi_i]_{wD}$. Therefore, ν satisfies $[\varphi_i]_{wL} \vee [\varphi_i]_{wR}$. ◀

▶ **Lemma 28.** *The formula φ_i implies $\bigvee_{\gamma \in \mathcal{P}(\varphi_i)} \gamma$.*

Proof. Consider a solution ν to φ_i. Let T be defined as in Definition 26 as the set $\{(f(t), \tau) : \text{the equation } f(t) \cdot x_{i+1} + \tau = 0 \text{ occurs in } \varphi_i\}$. Let I and A_0 and A_1 be the following partition of T, induced by ν:

$$I \coloneqq \{(f(t), \tau) \in T : \nu \text{ does not satisfy } f(t) \cdot x_{i+1} + \tau = 0\}$$
$$A_0 \coloneqq \{(f(t), \tau) \in T : \nu \text{ satisfies } f(t) \cdot x_{i+1} + \tau = 0 \wedge f(t) = 0\}$$
$$A_1 \coloneqq \{(f(t), \tau) \in T : \nu \text{ satisfies } f(t) \cdot x_{i+1} + \tau = 0 \wedge f(t) \neq 0\}.$$

From Definition 26, it is simple to see that ν satisfies the partial formula γ of φ_i such that $I(\gamma) = I$, $A_0(\gamma) = A_0$ and $A_1(\gamma) = A_1$. Indeed, the fact that ν satisfies $(\varphi_i|_{I,T})$ follows from the fact that $(\varphi_i|_{I,T})$ is obtained from φ_i by replacing each equality featuring x_{i+1} with $\top$ (when ν satisfies the equality) or $\bot$ (when ν does not satisfy the equality). Moreover, for every $(f(t), \tau) \in A_0$, ν satisfies $f(t) \cdot x_{i+1} + \tau = 0 \wedge f(t) = 0$, which in turn means that ν satisfies $\tau = 0 \wedge f(t) = 0$. For every $(f(t), \tau) \in A_1$, ν trivially satisfies $f(t) \cdot x_{i+1} + \tau = 0 \wedge f(t) \neq 0$. ◀

We now discuss three cases, depending on the set $A_1(\gamma)$. These cases will later corresponds to different branches in Algorithm 3, from which we will prove Lemma 24.

Here is the first case:

▶ **Lemma 29.** *Let $\gamma \in \mathcal{P}(\varphi_i)$ such that $A_1(\gamma) = \emptyset$. Then,*

$$\exists x_{i+1}(\gamma \wedge \chi_i) \quad \textit{implies} \quad \exists x_{i+1}(0 \leq x_{i+1} \leq m_i(t) - 1 \wedge \gamma \wedge \chi_i).$$

Proof. Since $A_1(\gamma) = \emptyset$, in γ the variable x_{i+1} only occurs in divisibility constraints. Consider then a solution ν to $\gamma \wedge \chi_i$. Let $g_1(t), \dots, g_k(t)$ be the list of divisors occurring in divisibility constraints of φ_i. These are also all the divisors occurring in γ (by definition of γ). Each divisor g_j ($j \in [1, k]$) is evaluated to an integer $g_j(\nu(t))$. From Lemma 25:

- χ_i implies $m(t) > 0$, and the variable x_{i+1} does not occur in this formula.
- $g_j(t)$ is a factor of $m_i(t)$.

Therefore, $m(\nu(t))$ is positive, and it is divided by $g_j(\nu(t))$ (in particular, $g_j(\nu(t))$ is non-zero). Let $L \coloneqq \operatorname{lcm}\{g_j(\nu(t)) : j \in [1, k]\}$ (recall: lcm returns a non-negative integer). All numbers of the form $\nu(x_{i+1}) + j \cdot L$, for $j \in \mathbb{Z}$, have the same remainder modulo every $g_j(\nu(t))$. Since x_{i+1} only occurs in divisibility constraints, updating $\nu(t)$ to any of these values still yields a solution to $\gamma \wedge \chi_i$. In particular, one such value must lie in $[0, L - 1]$. Then, to conclude the proof, it suffices to observe that $L \leq m(\nu(t))$, as L divides $m(\nu(t)) > 0$. ◀

Let us denote by $\boldsymbol{y}'$ the slack variables from $\boldsymbol{y}$ that are unassigned in B_i. Given $\gamma \in \mathcal{P}(\varphi_i)$, we "divide" the set $A_1(\gamma)$ into two sets $A_1^{\text{eq}}(\gamma)$ and $A_1^{\text{slk}}(\gamma)$:

$$A_1^{\text{eq}}(\gamma) \coloneqq \{(f(t), \tau) \in A_1(\gamma) : \tau \text{ does not contain a slack variable from } \boldsymbol{y}'\},$$
$$A_1^{\text{slk}}(\gamma) \coloneqq \{(\sigma, y) : \sigma = (f(t), \tau) \in A_1(\gamma) \setminus A_1^{\text{eq}}(\gamma), \text{ and } y \text{ occurs in } \boldsymbol{y}' \text{ and } \tau\}.$$

For the second of the three cases, let us consider when $A_1^{\text{eq}}(\gamma) \neq \emptyset$. Let $(f(t), \tau) \in A_1^{\text{eq}}(\gamma)$. Any solution to γ satisfies $f(t) \cdot x_{i+1} + \tau = 0 \wedge f(t) \neq 0$, and this equality does not feature any unassigned slack variable. There is then little to do, we can simply substitute x_{i+1} for $\frac{-\tau}{f(t)}$, eliminating this variable without producing any new bounded quantifier (more on that later). We are left with the third case (the most interesting one):

▶ **Lemma 30.** *Let $\gamma \in \mathcal{P}(\varphi_i)$ with $A_1^{slk}(\gamma) \neq \emptyset$ and $A_1^{eq}(\gamma) = \emptyset$. Then,*

$$\exists \boldsymbol{y}' \in \mathbb{N}\, \exists x_{i+1}(\gamma \wedge \chi_i) \;\; \textit{implies} \bigvee_{\substack{((f(t),\tau),y)\in A_1^{slk}(\gamma)\\ \pm\in\{+,-\}}} \exists \boldsymbol{y}' \in \mathbb{N}\, \exists x_{i+1}(0 \leq y \leq \pm f(t)\cdot m_i(t) - 1 \wedge \gamma \wedge \chi_i).$$

Proof. We follow arguments in [11, Lemma 17]. Let $A_1^{\text{slk}}(\gamma) = \{((g_1,\tau_1),y_1'),\ldots,((g_q,\tau_1),y_q')\}$. Let ν be a solution to $\gamma \wedge \chi_i$, which in particular assigns an integer to x_{i+1} and non-negative integers to each of the slack variables $y_1',\ldots,y_q'$. Observe that each integer $g_j(\nu(t))$ is non-zero, because (g_j,τ_j) is an element of $A_1(\gamma)$.

Consider the auxiliary rational vector $\boldsymbol{v}^\nu = (v_1,\ldots,v_q)$ where $v_j \coloneqq \frac{\nu(y_j')}{|g_j(\nu(t))|}$. Each v_j is positive. Suppose now that the values given by ν to the variables $x_{i+1}, y_1', \ldots, y_q'$ are those that *minimize* the smallest possible component of $\boldsymbol{v}^\nu$. Without loss of generality, assume v_1 to be such component. In other words, no matter how we change the values given to $x_{i+1}, y_1', \ldots, y_q'$ in the solution ν, if the resulting map ξ is still a solution, then the minimal component in its auxiliary rational vector $\boldsymbol{v}^\xi$ is greater or equal than v_1.

We show that $v_1 < m_i(\nu(t))$ or, in other words, $\nu(y_1') \leq |g_1(\nu(t))| \cdot m_i(\nu(t)) - 1$, proving the lemma. (Recall that χ_i implies $m(t) > 0$ by Lemma 25, hence $m_i(\nu(t))$ is positive).

For the sake of contradiction, suppose $v_1 \geq m_i(\nu(t))$; and note that this implies $v_j \geq m_i(\nu(t))$ for all $j \in [1,q]$. Let $\mu(t)$ be the coefficient of the slack variable y_1' in τ_1. From Lemma 21, for every $j \in [1,q]$, τ_j is the only term in which y_j' occurs, and moreover its coefficient is $\mu(t)$. We also know that $\mu(\nu(t))$ cannot be zero. Indeed, from Lemma 21, $\mu(t)$ is the polynomial $f(t)$ from line 14, with respect to the last time this line was executed by the **foreach** loop of line 10 (with $f(t) = 1$ if no such previous execution occurred). Let us say that this happened during the kth iteration of the loop, with $k \leq i$. Then, χ_k contains the constraint $\pm_k \mu(t) > 0$. Iterations between the kth and before the (current) $(i+1)$th one do not change the formula χ (the **continue** statement of line 13 is executed instead), hence χ_i still contains the constraint $\pm_k \mu(t) > 0$, which forces $\mu(t)$ to be non-zero.

Let us denote by ξ the assignment that agrees with ν on all possible variables except x_{i+1} and $y_1', \ldots, y_q'$, and such that:

$$\begin{aligned} \xi(x_{i+1}) &= \nu(x_{i+1}) \pm \mu(\nu(t)) \cdot m_i(\nu(t)), \\ \xi(y_j') &= \nu(y_j') \mp g_j(\nu(t)) \cdot m_j(\nu(t)) \qquad\qquad \text{for every } j \in [1,q], \end{aligned}$$

where $\mp$ stands for $-$ whenever $g_1(\nu(t)) > 0$, and for $+$ otherwise, whereas $\pm \in \{+,-\}$ is the sign opposite to $\mp$. Clearly, $\frac{\xi(y_1')}{|g_1(\nu(t))|} < v_1$, since $\xi(y_1')$ is obtained by adding a negative number to $\nu(y_1')$. Moreover, for every $j \in [1,q]$, we have $\xi(y_j') \geq 0$; because $v_j \geq m_i(\nu(t))$ and we have (in the worse case) only removed $|g_j(\nu(t))| \cdot m_j(\nu(t))$ from $\nu(y_j')$.

To conclude the proof it suffices to show that ξ still satisfies $\gamma \wedge \chi_i$. Indeed, in this way $\frac{\xi(y_1')}{|g_1(\nu(t))|} < v_1$ contradicts the fact that v_1 is the minimal possible value that entries in the auxiliary vector $\boldsymbol{v}^\nu$ can take as we change the values of $x_{i+1}, y_1', \ldots, y_q'$ (while preserving the satisfaction of $\gamma \wedge \chi_i$).

Clearly, the map ξ satisfies all equalities and divisibility constraints not featuring x_{i+1} (hence, in particular, it satisfies χ_i). Similarly, it satisfies also all divisibility constraints featuring x_{i+1}, since by Lemma 25 we have that every divisor in these constraints is a factor of $m_i(t)$, and $\xi(x_{i+1})$ has been defined by shifting $\nu(x_{i+1})$ by a multiple of $m_i(\nu(t))$. Lastly, the equalities in γ involving x_{i+1} are still all satisfies. Indeed, given $j \in [1,q]$, let us write

$\tau_j = \rho_j + \mu(t) \cdot y'_j$. We have:

$$\begin{aligned}
&g_j(\xi(t)) \cdot \xi(x_{i+1}) + \rho_j + \mu(\xi(t)) \cdot \xi(y'_j) \\
&= g_j(\nu(t)) \cdot \big(\nu(x_{i+1}) \pm \mu(\nu(t)) \cdot m_i(\nu(t))\big) + \rho_j + \mu(\nu(t)) \cdot \big(\nu(y'_j) \mp g_j(\nu(t)) \cdot m_i(\nu(t))\big) \\
&= g_j(\nu(t)) \cdot \nu(x_{i+1}) + \rho_j + \mu(\nu(t)) \cdot \xi(y_j) \\
&= 0
\end{aligned}$$

◀

We now combine Lemmas 27–30:

▶ **Lemma 31.** *The formula* $\exists \boldsymbol{y}' \in \mathbb{N}\, \exists x_{i+1}(\varphi_i \wedge \chi_i)$ *is equivalent to*

$$\begin{aligned}
&\Big(\exists \boldsymbol{y}' \in \mathbb{N}\, \exists x_{i+1}\big(0 \le x \le m_i(t) - 1 \wedge \varphi_i \wedge \chi_i\big)\Big) \vee \\
&\bigvee_{\substack{\gamma \in \mathcal{P}(\varphi_i) \\ \pm \in \{+,-\}}} \Big(\bigvee_{(f(t),\tau) \in A_1^{eq}(\gamma)} \Big(\exists \boldsymbol{y}' \in \mathbb{N}\, \exists x_{i+1}\big(f(t) \cdot x_{i+1} + \tau = 0 \wedge \pm f(t) > 0 \wedge \varphi_i \wedge \chi_i\big)\Big) \vee \\
&\bigvee_{((f(t),\tau),y) \in A_1^{slk}(\gamma)} \Big(\exists \boldsymbol{y}' \in \mathbb{N}\, \exists x_{i+1}\big(f(t) \cdot x_{i+1} + \tau = 0 \wedge 0 \le y \le \pm f(t) \cdot m_i(t) - 1 \wedge \varphi_i \wedge \chi_i\big)\Big)\Big).
\end{aligned}$$

Proof. The right to left direction is trivial, since every disjunct "specializes" the formula $\exists \boldsymbol{y}' \in \mathbb{N}\, \exists x_{i+1}(\varphi_i \wedge \chi_i)$ with further constraints. For the left to right direction, by Lemma 28 we have that $\exists \boldsymbol{y}' \in \mathbb{N}\, \exists x_{i+1}(\varphi_i \wedge \chi_i)$ implies $\bigvee_{\gamma \in \mathcal{P}(\varphi_i)} \exists \boldsymbol{y}' \in \mathbb{N}\, \exists x_{i+1}(\gamma \wedge \chi_i)$. Let us then fix $\gamma \in \mathcal{P}(\varphi_i)$, and show that $\exists \boldsymbol{y}' \in \mathbb{N}\, \exists x_{i+1}(\gamma \wedge \chi_i)$ implies one of the disjunct of the right-hand side of the equivalence. We again consider three cases separately, depending on the set $A_1(\gamma)$.

case: $A_1(\gamma) = \emptyset$. From Lemma 29, the formula $\exists \boldsymbol{y}' \in \mathbb{N}\, \exists x_{i+1}(\gamma \wedge \chi_i)$ implies the formula $\exists \boldsymbol{y}' \in \mathbb{N}\, \exists x_{i+1}(0 \le x \le m_i(t) - 1 \wedge \gamma \wedge \chi_i)$. From Lemma 27, the latter formula implies

$$\exists \boldsymbol{y}' \in \mathbb{N}\, \exists x_{i+1}(0 \le x \le m_i(t) - 1 \wedge \varphi_i \wedge \chi_i).$$

This is one of the disjuncts on the right-hand side of the equivalence (see first line).

case: $A_1^{\mathsf{eq}}(\gamma) \neq \emptyset$. In this case, one of the conjuncts occurring in the formula γ is a formula $f(t) \cdot x_{i+1} + \tau = 0 \wedge f(t) \neq 0$ such that $(f(t), \tau) \in A_1^{\mathrm{eq}}(\gamma)$. Since γ implies φ_i (Lemma 27), $\exists \boldsymbol{y}' \in \mathbb{N}\, \exists x_{i+1}(\gamma \wedge \chi_i)$ implies $\exists \boldsymbol{y}' \in \mathbb{N}\, \exists x_{i+1}(f(t) \cdot x_{i+1} + \tau = 0 \wedge f(t) \neq 0 \wedge \varphi_i \wedge \chi_i)$. By "guessing" the sign of $f(t)$, we can rewrite the latter formula as

$$\bigvee_{\pm \in \{+,-\}} \exists \boldsymbol{y}' \in \mathbb{N}\, \exists x_{i+1}(f(t) \cdot x_{i+1} + \tau = 0 \wedge \pm f(t) > 0 \wedge \varphi_i \wedge \chi_i).$$

This is one of the disjunct on the right-hand side of the equivalence (see second line).

case: $A_1^{\mathsf{eq}}(\gamma) = \emptyset$ and $A_1^{\mathsf{slk}}(\gamma) \neq \emptyset$. From Lemma 30, $\exists \boldsymbol{y}' \in \mathbb{N}\, \exists x_{i+1}(\gamma \wedge \chi_i)$ implies

$$\bigvee_{\substack{((f(t),\tau),y) \in A_1^{\mathrm{slk}}(\gamma) \\ \pm \in \{+,-\}}} \exists \boldsymbol{y}' \in \mathbb{N}\, \exists x_{i+1}(0 \le y \le \pm f(t) \cdot m_i(t) - 1 \wedge \gamma \wedge \chi_i).$$

In the above formula, we can replace γ by φ_i (as the former implies the latter by Lemma 27), to obtain a formula that is still implied by $\exists \boldsymbol{y}' \in \mathbb{N}\, \exists x_{i+1}(\gamma \wedge \chi_i)$. Each disjunct in the resulting formula appears on the right-hand side of the equivalence (see third line). ◀

The next lemma completes the elimination of x_{i+1} from the second and third case of the formula in Lemma 31.

▶ **Lemma 32.** *Consider a formula* $\psi \coloneqq \exists x_{i+1}(f(t)\cdot x_{i+1}+\tau = 0 \wedge \pm f(t) > 0 \wedge \varphi_i \wedge \chi_i)$*, where* $f(t)\cdot x_{i+1}+\tau = 0$ *is an equation in* φ_i *and* $\pm \in \{+,-\}$*. Let* φ_{i+1} *be the formula obtained from* φ_i *by executing lines 21–23 of Algorithm 3, where the division is done with respect to* ℓ_i*. Then,* ψ *is equivalent to* $\pm f(t) > 0 \wedge \varphi_{i+1} \wedge \chi_i$*.*

Proof. By definition of the vigorous substitution, it is easy to see that ψ is equivalent to $\exists x_{i+1}(f(t)\cdot x_{i+1}+\tau = 0 \wedge \pm f(t) > 0 \wedge \varphi_i[\![\frac{-\tau}{f(t)} \,/\, x_{i+1}]\!] \wedge \chi_i)$. As explained in Appendix A.1, all polynomials from $\mathbb{Z}[t]$ occurring in inequalities and divisibility constraints (divisors included) of $\varphi_i[\![\frac{-\tau}{f(t)} \,/\, x_{i+1}]\!]$ are divisible by the polynomial ℓ_i (without remainder). Line 22 performs this division. The resulting formula φ_i' is equivalent to $\varphi_i[\![\frac{-\tau}{f(t)} \,/\, x_{i+1}]\!]$. Lastly, observe that x_{i+1} occurs now only in the equality $f(t)\cdot x_{i+1}+\tau = 0$, which we can rewrite as $f(t) \mid \tau$, since x_{i+1} ranges over the integers. The formula φ_{i+1} is defined as $\varphi_i' \wedge f(t) \mid \tau$ (see line 23). Therefore, ψ is equivalent to $\pm f(t) > 0 \wedge \varphi_{i+1} \wedge \chi_i$. ◀

We are now ready to prove Lemma 24:

▶ **Lemma 24.** *Let* $i \in [0, n-1]$ *and* $\Phi \in S_i$*. There is a subset* $G \subseteq S_{i+1}$ *satisfying*

$$\exists x_{i+1} F(\Phi) \iff \bigvee\nolimits_{\Psi \in G} F(\Psi).$$

Proof. Let $\Phi = (\varphi_i, m_i, \ell_i, \pm_i, B_i, \chi_i) \in S_i$, $\boldsymbol{v}$ be the variables assigned in B_i, and $\boldsymbol{y'}$ be the slack variables that are unassigned in B_i. By Lemma 31, $\exists x_{i+1} F(\Phi)$ is equivalent to

$$\begin{aligned}
&\Big(\exists \boldsymbol{v} \le B_i\, \exists \boldsymbol{y'} \in \mathbb{N}\, \exists x_{i+1}\big(0 \le x \le m_i(t) - 1 \wedge \varphi_i \wedge \chi_i\big)\Big) \vee \\
\bigvee_{\substack{\gamma \in \mathcal{P}(\varphi_i)\\ \pm \in \{+,-\}}} &\Big(\bigvee_{(f(t),\tau) \in A_1^{\mathrm{eq}}(\gamma)} \Big(\exists \boldsymbol{v} \le B_i\, \exists \boldsymbol{y'} \in \mathbb{N}\, \exists x_{i+1}\big(f(t)\cdot x_{i+1}+\tau = 0 \wedge \pm f(t) > 0 \wedge \varphi_i \wedge \chi_i\big)\Big) \vee \\
&\bigvee_{((f(t),\tau),y) \in A_1^{\mathrm{slk}}(\gamma)} \Big(\exists \boldsymbol{v} \le B_i\, \exists \boldsymbol{y'} \in \mathbb{N}\, \exists x_{i+1}\big(f(t)\cdot x_{i+1}+\tau = 0 \wedge 0 \le y \le \pm f(t)\cdot m_i(t) - 1 \wedge \varphi_i \wedge \chi_i\big)\Big)\Big).
\end{aligned}$$

We now consider each disjunction ψ of this formula, and show that S_{i+1} contains an element Ψ such that $F(\Psi)$ is equivalent to ψ. Naturally, the element Ψ will be such that $\Phi \xrightarrow{\delta} \Psi$, for a suitable choice of the guess δ. Once more, we divide the proof in three cases.

case 1: Consider the formula $\psi \coloneqq \exists \boldsymbol{v} \le B_i\, \exists \boldsymbol{y'} \in \mathbb{N}\, \exists x_{i+1}\big(0 \le x \le m_i(t) - 1 \wedge \varphi_i \wedge \chi_i\big)$. Let $\delta = \top$, that is, consider the case where the **foreach** loop executes the **continue** statement of line 13. Starting from Ψ, in this case the algorithm simply adds x_{i+1} to the keys of B_i, with value $m(t) - 1$. Let B_{i+1} be the resulting map. This completes the iteration of the **foreach** loop, and therefore S_{i+1} contains the tuple $\Psi \coloneqq (\varphi_i, m_i, \ell_i, \pm_i, B_{i+1}, \chi_i)$. Clearly, $F(\Psi)$ is equivalent to ψ.

case 2: Consider the formula

$$\psi \coloneqq \exists \boldsymbol{v} \le B_i\, \exists \boldsymbol{y'} \in \mathbb{N}\, \exists x_{i+1}\big(f(t)\cdot x_{i+1}+\tau = 0 \wedge \pm f(t) > 0 \wedge \varphi_i \wedge \chi_i\big),$$

where $(f(t),\tau) \in A_1^{\mathrm{eq}}(\gamma)$, $\gamma \in \mathcal{P}(\varphi_i)$ and $\pm \in \{+,-\}$. Since $(f(t),\tau) \in A_1^{\mathrm{eq}}(\gamma)$, the formula φ_i contains the equality $f(t)\cdot x_{i+1}+\tau$, and τ does not feature any unassigned slack variable. Let $\delta = (f(t)\cdot x_{i+1}+\tau = 0, \pm)$, that is, the equation and sign $\pm$ are guessed in lines 14 and 16. Note that line 17 updates χ, so that $\chi_{i+1} = \chi_i \wedge (\pm f(t) > 0)$. By Lemma 32, lines 21–23 updates φ_i into a formula φ_{i+1} such that $\exists x_{i+1}\big(f(t)\cdot x_{i+1}+\tau = 0 \wedge \varphi_i \wedge \chi_{i+1}\big)$ is equivalent to $(\varphi_{i+1} \wedge \chi_{i+1})$. Furthermore, the algorithm gives $m_{i+1} \coloneqq \pm f(t)\cdot m_i$, $\pm_{i+1} \coloneqq \pm$ and $\ell_{i+1} = f(t)$. The tuple $\Psi \coloneqq (\varphi_{i+1}, m_{i+1}, \ell_{i+1}, \pm_{i+1}, B_i, \chi_{i+1})$ belongs to S_{i+1}, and $F(\Psi) = \exists \boldsymbol{v} \le B_i\, \exists \boldsymbol{y'} \in \mathbb{N}\, (\varphi_{i+1} \wedge \chi_{i+1})$ is equivalent to ψ.

case 3: This case considers formulae

$$\exists \boldsymbol{v} \leq B_i\, \exists \boldsymbol{y'} \in \mathbb{N}\, \exists x_{i+1} \big(f(t) \cdot x_{i+1} + \tau = 0 \wedge 0 \leq y \leq \pm f(t) \cdot m_i(t) - 1 \wedge \varphi_i \wedge \chi_i \big).$$

Observe that the subformula $0 \leq y \leq \pm f(t) \cdot m_i(t) - 1 \wedge \chi_i$ implies $\pm f(t) > 0$ (recall that χ_i implies $m_i(t) > 0$). After adding this implies inequality, this case is analogous to the previous one, the only difference being that the map B_i is now updated with a bound $\pm f(t) \cdot m_i(t) - 1$ for the slack variable y. ◀

Step 3 (lines 24–26). The last step of the procedure is deterministic and simply removes the remaining slack variables. Here is its "specification":

▶ **Lemma 33.** *Let $(\varphi_n, m_n, \ell_n, \pm_n, B_n, \chi_n) \in S_n$, and let $\boldsymbol{y'}$ be the slack variables that are unassigned in B_n. Let ψ be the formula obtained by applying the **foreach** loop of line 24 to the formula φ_n, with respect to B_n and $\pm_n$. Then, $\exists \boldsymbol{y'} \in \mathbb{N}(\varphi_n \wedge \chi_n) \iff \psi \wedge \chi_n$.*

Proof idea. The proof follows essentially verbatim the proof of [11, Lemma 18] and it is thus not duplicated here. We give instead the intuition on how [11, Lemma 18] is proven, by considering the case of $n = 1$. The full proof given in [11] iterates the same reasoning across all formulae $\varphi_2, \ldots, \varphi_n$.

In the case $n = 1$, the **foreach** loop of line 10 iterates only once. Let us suppose that, during this iteration, the procedure guesses an equation $f(t) \cdot x_1 + \tau = 0$ and $\pm \in \{+, -\}$ (whether τ has a slack variable or not is irrelevant here, so let us assume it does not for simplicity). The formula χ contains the constraint $\pm f(t) > 0$ asserting that $\pm$ is the sign of $f(t)$. Let $g(t) \cdot x_1 + \rho + y = 0$ be another equation from φ, featuring the unassigned slack variable y. (y has coefficient 1 because we are performing the first iteration of the loop.)

The substitution $[\![\frac{-\tau}{f(t)} \,/\, x]\!]$ updates $g(t) \cdot x_1 + \rho + y = 0$ to $-g(t) \cdot \tau + f(t) \cdot \rho + f(t) \cdot y = 0$. Line 25 modifies this term to $-g(t) \cdot \tau + f(t) \cdot \rho \sim 0$, where $\sim$ stands for $\leq$ whenever $\pm$ is the symbol $+$, and otherwise $\sim$ stands for $\geq$.

First, notice that $\exists y \in \mathbb{N}\, (-g(t) \cdot \tau + f(t) \cdot \rho + f(t) \cdot y = 0 \wedge \pm f(t) > 0)$ implies the inequality $-g(t) \cdot \tau + f(t) \cdot \rho \sim 0$. Indeed, since y is non-negative, the equality is forcing $-g(t) \cdot \tau + f(t) \cdot \rho$ to either be zero (when y is zero) or to have a sign opposite to the one of $f(t)$. The left to right direction of the lemma follows from this reason alone.

The right-to-left direction is more subtle, because $-g(t) \cdot \tau + f(t) \cdot \rho \sim 0 \wedge \pm f(t) > 0$ alone does not imply $\exists y \in \mathbb{N} : -g(t) \cdot \tau + f(t) \cdot \rho + f(t) \cdot y = 0$. For this direction to hold, it is sufficient (and necessary) to further assume that $-g(t) \cdot \tau + f(t) \cdot \rho$ is a multiple of $f(t)$; which in turns implies that $-g(t) \cdot \tau$ has to be a multiple of $f(t)$. But indeed, the procedure adds in line 23 the division $f(t) \mid \tau$, which is preserved by the updates performed in Line 25. This is the key observation used to prove the right to left direction of the lemma. ◀

Putting all together. We can now complete the proof of correctness:

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

Proof. Let $\exists x_1, \ldots, x_n \varphi(x_1, \ldots, x_n, \boldsymbol{z})$ be the input formula. By Lemma 23, across non-deterministic branches, lines 1–9 build the formulae $F(\Phi)$, with $\Phi \in S_0$, such that

$$\exists x_1, \ldots, x_n \varphi \iff \bigvee_{\Phi \in S_0} \exists x_1, \ldots, x_n F(\Phi).$$

By iterated application of Lemma 24, after the **foreach** loop of line 10 completes, (again across non-deterministic branches), the algorithm considers the formulae $F(\Psi)$ with $\Psi \in S_n$,

and we have

$$\exists x_1, \ldots, x_n \varphi \iff \bigvee_{\Psi \in S_n} F(\Psi).$$

Given $\Psi = (\varphi_n, m_n, \ell_n, \pm_n, B_n, \chi_n) \in S_n$, let $F(\Psi) = \exists \boldsymbol{y'} \in \mathbb{N} : \exists \boldsymbol{v} \leq B_n(\varphi_n \wedge \chi_n)$, where $\boldsymbol{y'}$ and $\boldsymbol{v}$ are, respectively, the unassigned and assigned variables in B_n. By Lemma 33, lines 24–26 produce a formula $F'(\Psi) = \exists \boldsymbol{v} \leq B_n(\varphi'_n \wedge \chi_n)$ equivalent to $F(\Psi)$. The output of the algorithm is $\bigvee_{\Psi \in S_n} F'(\Psi)$, which is thus equivalent to $\exists x_1, \ldots, x_n \varphi$. ◀

A.3 Complexity of BoundedQE

We now move to the complexity of the procedure. Recall the paremeters of PrA[t] formulae, which were introduced in Section 6.

- $\mathit{atom}(\varphi) \coloneqq \big($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 $\varphi\big)$,
- $\langle \mathit{const} \rangle(\varphi) \coloneqq \max\{\langle f \rangle : f \in \mathbb{Z}[t] \text{ occurs in } \varphi\}$,
- $\langle B \rangle \coloneqq \max\{0, \langle B(w) \rangle : w \text{ is in the domain of } B\}$.

With this notation, let us prove the following lemma, which gives us essentially the proof that Algorithm 3 runs in non-deterministic polynomial time (see Lemma 35 below).

▶ **Lemma 34.** *Let $\exists \boldsymbol{x} : \varphi(\boldsymbol{x}, \boldsymbol{z})$ be a formula in input of* BoundedQE*, where φ is a positive Boolean combination of linear* PrA[t] *constraints, $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{z} = (x_{n+1}, \ldots, x_g)$, where $n \geq 1$. Then, for every branch output $\exists \boldsymbol{w} \leq B : \psi(\boldsymbol{w}, \boldsymbol{z})$ we have:*

$$\textit{if } \begin{cases} \mathit{atom}(\varphi) & \leq a \\ \mathit{vars}(\varphi) & = g \\ \mathit{func}(\varphi) & = 0 \\ \langle \mathit{const} \rangle(\varphi) & \leq c \end{cases} \quad \textit{then } \begin{cases} \mathit{atom}(\psi) & \leq 2 \cdot (n + a) + 1 \\ \mathit{vars}(\psi) & = g \\ \mathit{func}(\psi) & = 0 \\ \langle \mathit{const} \rangle(\psi) & \leq (n + a)^3 \cdot c^3 \end{cases}$$

Moreover, $\boldsymbol{w}$ contains n variables and $\langle B \rangle \leq (n + a)^{11} \cdot c^9$.

Proof. First observe that the same as the input formula φ, the formula ψ does not feature remainder functions $(\cdot \bmod f(t))$ nor integer divisions $\left\lfloor \frac{\cdot}{f(t)} \right\rfloor$, and we have $\mathit{func}(\varphi) = \mathit{func}(\psi) = 0$. Let us now consider other parameters.

Bound on $\mathit{atom}(\psi)$**:** The relevant lines are: line 4, line 7, line 17, line 23, and line 26. Line 4 adds one conjunct for each element in the set Z guessed in line 1. The cardinality of this set is bounded by a. In parallel, the algorithm constructs a formula χ, which has initially one atomic formula (line 7). Lines 17 and 23 add, respectively, one conjunct to χ and one conjunct to φ for each variable in $\boldsymbol{x}$ for which the non-deterministic branching performed in line 11 does not lead to executing the **continue** instruction in line 13. Let us assume that line 23 is executed $q \leq n$ times. Therefore, when the algorithm reaches line 24, the formula φ has thus at most $a + a + q$ atomic formulae and the formula χ has at most $q + 1$ atomic formulae. The last line 26 returns the conjunction of φ and χ, and we thus conclude that $\mathit{atom}(\psi) \leq a + a + q + q + 1 \leq 2 \cdot (n + a) + 1$.

Bound on $\mathit{vars}(\psi)$**:** The procedure simply "replaces" the n variables from $\boldsymbol{x}$ with variables $\boldsymbol{w}$ under the scope of bounded quantifiers. Following lines 20 and 12, each variable in $\boldsymbol{x}$ corresponds to a variable in $\boldsymbol{w}$ ($\boldsymbol{w}$ thus features n variables). Therefore, $\mathit{vars}(\psi) \leq v$.

Bound on $\langle const\rangle(\psi)$: First observe that $\langle const\rangle(\psi) = max\{\langle const\rangle(\varphi'), \langle const\rangle(\chi')\}$, where φ' and χ' are the systems stored in the variables φ and χ in the last line 26. From line 7 and line 17 we see that $\langle const\rangle(\chi') = \max\{\langle \prod N\rangle, \langle f_0\rangle, \ldots, \langle f_{q-1}\rangle\}$ for the set N such that $N \subseteq \{f(t) : \text{the relation } (f(t) \mid \cdot) \text{ occurs in } \varphi\}$ and equations $f_0(t) \cdot x_1 + \tau_0 = 0$, $\ldots$, $f_{q-1}(t) \cdot x_q + \tau_{q-1} = 0$ are guessed in line 14 within the main **foreach** loop from line 10. Let us now upper-bound the bit size of $f_0, \ldots, f_{q-1}$ together with the bit size of the coefficients and constants of the formula φ'.

This bound uses in a crucial way Proposition 20. From the definitions of the matrices $M_0, \ldots, M_q$, given at the beginning of the appendix, we see that all coefficients of the variables and constants of the terms in φ' are entries of the matrix M_q located in columns among $1, \ldots, g+r+1$, where g is the overall number of free variables in the input formula φ (including both $\boldsymbol{x}$ and $\boldsymbol{z}$), and r is the number of slack variables introduced by the algorithm in line 9.

Following Proposition 20, we see that each f_i is the determinant of $B[1, \ldots, i{+}1; 1, \ldots, i{+}1]$ and that the entries of M_q are determinants of $q \times q$ or $(q+1) \times (q+1)$ sub-matrices of M_0. All entries in the columns $1, \ldots, g$ of M_0 are coefficients of the variables $\boldsymbol{x}$ and $\boldsymbol{z}$ in the formula stored in φ at the beginning of the loop in line 10, whereas the $(g+1)$-th column contains constants of terms in this formula, and the bit size of all these entries is bounded by $\langle const\rangle(\varphi)$. (In particular, the equalities $f(t) = 0$ added in line 4 do not feature any variable, and $f(t)$ is an entry located in the $(g+1)$-th column of M_0.) All entries in the columns $g+2, \ldots, g+r+1$ of M_0 are either 0 or 1 (slack variables have coefficient 1 initially). Therefore, to establish the bound on $\langle const\rangle(\varphi')$ and $\langle f_0\rangle, \ldots, \langle f_{q-1}\rangle$, it suffices to bound the bit size of these determinants. Below, we provide the computation for a $(q+1) \times (q+1)$ sub-matrix, which subsumes the one for $q \times q$ matrices.

Let $B \in \mathbb{Z}[t]^{(q+1)\times(q+1)}$ be a matrix with entries of bit size at most $c \geq \max(1, \langle const\rangle(\varphi))$:

$$B \coloneqq \begin{pmatrix} b_{1,1} & \cdots & b_{1,q+1} \\ \vdots & \ddots & \vdots \\ b_{q+1,1} & \cdots & b_{q+1,q+1} \end{pmatrix}.$$

Let us write S_{q+1} for the set of all permutations of $(1, \ldots, q+1)$, and $\operatorname{sgn}(\sigma)$ for the signature of a permutation σ, i.e., $\operatorname{sgn}(\sigma) = +1$ if σ can be obtained from $(1, \ldots, q+1)$ by exchanging entries an even number of times, and $\operatorname{sgn}(\sigma) = -1$ otherwise. Let us also write $\sigma(i)$ for the ith entry of σ. The determinant of B is the element of $\mathbb{Z}[t]$ defined as

$$\det B = \sum_{\sigma \in S_{q+1}} \operatorname{sgn}(\sigma) \cdot b_{1,\sigma(1)} \cdot \ldots \cdot b_{q+1,\sigma(q+1)}.$$

Let $D \coloneqq \max_{i,j} \deg(b_{i,j})$ and $H \coloneqq \max_{i,j} h(b_{i,j})$. We first bound the degree and height of $\det B$. For the degree, we have:

$$\deg(\det B) \leq \max_{\sigma \in S_{q+1}} \left(\deg(b_{1,\sigma(1)}) + \ldots + \deg(b_{q+1,\sigma(q+1)})\right) \leq (q+1) \cdot D.$$

To bound the height, consider first a polynomial $b_{1,\sigma(1)} \cdot \ldots \cdot b_{q+1,\sigma(q+1)}$, for some $\sigma \in S_{q+1}$. Writing $b_{j,\sigma(j)} = \sum_{i=0}^{D} g_j \cdot t^j$, we have

$$b_{1,\sigma(1)} \cdot \ldots \cdot b_{q+1,\sigma(q+1)} = \sum\nolimits_{j_1,\ldots,j_{q+1} \in [0,D]} \prod\nolimits_{i=1}^{(q+1)} g_{j_i} \cdot t^{j_i}.$$

When performing all multiplications and additions, the height of $b_{1,\sigma(1)} \cdot \ldots \cdot b_{q+1,\sigma(q+1)}$ is thus bounded by $(D+1)^{(q+1)} \cdot H^{q+1}$. Therefore,

$$h(\det B) \leq (q+1)^{q+1} \cdot (D+1)^{(q+1)} \cdot H^{q+1}. \tag{5}$$

We can now bound the bit size of $\det B$:

$$\begin{aligned}\langle \det B\rangle &= (\deg(\det B)+1)\cdot(\lceil \log_2(h(\det B)+1)\rceil+1)\\ &\le ((q+1)\cdot D+1)\cdot(\lceil \log_2(1+(q+1)^{q+1}\cdot(D+1)^{(q+1)}\cdot H^{q+1})\rceil+1)\\ &\le (q+1)\cdot(D+1)\cdot\big((q+1)\cdot\lceil \log_2((q+1)\cdot(D+1)\cdot(H+1))\rceil+1\big)\\ &\le (q+1)^2\cdot(D+1)\cdot\big(\lceil \log_2((q+1)\cdot(D+1))+\log_2(H+1)\rceil+1\big)\\ &\le (q+1)^3\cdot(D+1)^2\cdot\big(\lceil \log_2(H+1)\rceil+1\big)\\ &\le (n+1)^3\cdot c^3.\end{aligned}$$

It remains to compare the bounds for $\langle const\rangle(\varphi')$ and $\langle const\rangle(\chi')$. Let us define the integers $D \coloneqq \max\{\deg(f) : f \in N\}$ and $H \coloneqq \max\{h(f) : f \in N\}$, then we have:

$$\deg(\langle \textstyle\prod N\rangle) \le a\cdot D \qquad\qquad h(\langle \textstyle\prod N\rangle) \le (D+1)^a\cdot H^a.$$

Therefore,

$$\begin{aligned}\langle \textstyle\prod N\rangle &\le (a\cdot D+1)\cdot(\lceil \log_2((D+1)^a\cdot H^a+1)\rceil+1)\\ &\le a\cdot(D+1)\cdot\big(a\cdot\lceil \log_2((D+1)\cdot(H+1))\rceil+1\big)\\ &\le a^2\cdot(D+1)^2\cdot\big(\lceil \log_2(H+1)\rceil+1\big)\\ &\le a^2\cdot c^2,\end{aligned}$$

and we can conclude that

$$\begin{aligned}\langle const\rangle(\psi) &= max\{\langle const\rangle(\varphi'),\langle const\rangle(\chi')\}\\ &\le \max\{(n+1)^3\cdot c^3, a^2\cdot c^2\}\\ &\le (n+a)^3\cdot c^3.\end{aligned}$$

Bound on $\langle B(w)\rangle$: The relevant lines are line 6, line 12, line 18, and line 20. Line 6 defines a product of polynomials $m(t) = \pm\prod N$ for $N \subseteq \{f(t) : \text{the relation } (f(t) \mid \cdot) \text{ occurs in } \varphi\}$. Let $f_0(t)\cdot x_1+\tau_0 = 0, \ldots, f_{q-1}(t)\cdot x_q+\tau_{q-1}=0$ be the equations guessed in line 14 during the run of the branch β. Because of the updates performed in line 18, when line 20 is executed the $(i+1)$-th time, the bound given to the slack variable y is $\pm f_0(t)\cdot\cdots\cdot f_i(t)\cdot\prod N-1$. Therefore, the bound given to variables in line 12 is $\pm f_0(t)\cdot\cdots\cdot f_{q-1}(t)\cdot\prod N-1$. Let us now define

$$\begin{aligned}D &\coloneqq \max(\{\deg(f) : f\in N\}\cup\{\deg(f_i) : i\in[1,q]\})\\ H &\coloneqq \max(\{h(f) : f\in N\}\cup\{h(f_i) : i\in[1,q]\}).\end{aligned}$$

If we consider for every $i\in[0,q-1]$ the polynomial $f(t) \coloneqq f_0(t)\cdot\cdots\cdot f_i(t)\cdot\prod N-1$, then (since $\#N\le a$ and $q\le n$) we have:

$$\deg(f)\le(n+a)\cdot D \qquad\qquad h(f)\le(D+1)^{n+a}\cdot H^{n+a}+1,$$

where the bound on $h(f)$ is found using similar arguments as the ones used to derive the bound in Equation (5). Therefore, for every variable w in $\boldsymbol{w}$, we obtain

$$\begin{aligned}\langle B(w)\rangle &\le ((n+a)\cdot D+1)\cdot(\lceil \log_2((D+1)^{n+a}\cdot H^{n+a}+2)\rceil+1)\\ &\le (n+a)\cdot(D+1)\cdot\big((n+a)\cdot\lceil \log_2((D+1)\cdot(H+1))\rceil+1\big)\\ &\le (n+a)^2\cdot(D+1)^2\cdot\big(\lceil \log_2(H+1)\rceil+1\big)\\ &\le (n+a)^2\cdot(n+1)^9\cdot c^9\\ &\le (n+a)^{11}\cdot c^9.\end{aligned}$$

In the second-to-last line, we have used the fact that $\langle f_i \rangle \leq (n+1)^3 \cdot c^3$ (recall that f_i is the determinant of $B[1, \ldots, i+1; 1, \ldots, i+1]$). ◀

▶ **Lemma 35.** *The algorithm* BoundedQE *runs in non-deterministic polynomial time.*

Proof. Following the proof of Lemma 34, we now know that all objects BoundedQE constructs during its execution are of polynomial bit size. With this information, the proof that the algorithm only requires non-deterministic polynomial time is simple: it suffices to show that each line of the algorithm can be implemented in non-deterministic polynomial time, and that all loops only iterate a polynomial number of times.

Regarding the operations performed, a quick glance at the pseudocode of BoundedQE reveals that, in fact, except for the lines that perform guesses, all other operations can be implemented in polynomial time. In particular, note that the polynomial divisions performed in line 22 can be performed in polynomial time with the standard Euclidean algorithm.

For the number of loops iterations, the **foreach** loop of line 2 iterates at most $a \coloneqq atom(\varphi)$ times; the **foreach** loop of line 10 iterates n times, where n is the number of eliminated variables; and the **foreach** loop of line 24 iterates at most $2(n+a)+1$ times (following the bound on $atom(\psi)$ from Lemma 34). ◀

B Extended material for Section 5

The main result of Section 5 is the correctness proof for Algorithm 4 (ElimBounded), which was only sketched in that section. In this appendix, we first formally prove Lemma 14 (with the help of two auxiliary lemmas). Together with the correctness of BoundedQE (Lemma 9, proven in Appendix A.2), this will give us the necessary equivalences to complete the proof of Lemma 15. The complexity of Algorithm 4 is analysed at the end of the appendix.

In this appendix, we write $\#\boldsymbol{x}$ for the number of variables in the vector $\boldsymbol{x}$.

B.1 Proof of Lemma 12

▶ **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 $\varphi_\varnothing$.*

Proof. It is simple to see that this lemma follows as soon as we prove that, in any solution to the input formula, the base t expansion of every bounded variable w has at most $\langle B(w) \rangle$ t-digits. For this, it suffices to establish $|g(k)| \leq k^{\langle g \rangle}$, for every $g \in \mathbb{Z}[t]$ and $k \geq 2$. Consider a non-zero polynomial $g(t) \coloneqq \sum_{j=0}^{d} a_j \cdot t^j$, where $a_0, \ldots, a_i \in \mathbb{Z}$. (Trivially, $0 \leq k^{\langle 0 \rangle}$.) Let $a \coloneqq \max\{|a_j| : j \in [0, d]\}$. Then,

$$|g(k)| = \left| \sum_{j=0}^{d} a_j \cdot k^j \right| \leq (d+1) \cdot a \cdot k^d \leq k^{d + \log_k((d+1)\cdot a)} \leq k^{d + \log_2((d+1)\cdot a)},$$

and analysing the exponent we see

$$d + \log_2((d+1)\cdot a) \leq d + (d+1) \cdot \log_2(a+1) \leq (d+1) \cdot (\log_2(a+1) + 1) \leq \langle g \rangle. \quad ◀$$

B.2 Proof of Lemma 14

Before delving into the proof of this lemma, we need two auxiliary results (Lemma 36 and Lemma 37). The first lemma describes the PrA$[t]$ constraints produced during the execution of the body of the **while** loop.

▶ **Lemma 36.** *At the beginning of every iteration of the* ***while*** *loop in line 7, every constraint which has a non-integer coefficient of a variable from* $\boldsymbol{y}$ *has the form*

$$\sigma(\boldsymbol{y}) \cdot t + \rho(\boldsymbol{y}) + \left\lfloor \frac{\tau(\boldsymbol{z})}{t^k} \right\rfloor \sim 0 \qquad \text{with } \sim \text{ in } \{=, \leq\}, \tag{6}$$

where $\sigma(\boldsymbol{y})$ *is a linear* PrA$[t]$ *term,* $\rho(\boldsymbol{y})$ *is a linear term with coefficients in* $\mathbb{Z}$*,* $\tau(\boldsymbol{z})$ *is either a linear non-shifted* PrA$[t]$ *term or* $(\tau'(\boldsymbol{z}) \bmod f(t))$*, where* τ' *is linear non-shifted,* k *is a non-negative integer, and* $\left\lfloor \frac{\tau}{t^0} \right\rfloor = \tau$*.*

Proof. By induction on the number of iterations $k \geq 0$. Initially, when the constraint $(\eta \sim 0)$ is taken from the formula $\varphi_\varnothing$, the term η has one of the two forms:

- $\sigma(\boldsymbol{y}) \cdot t + \rho(\boldsymbol{y}) + \tau(\boldsymbol{z})$, where we have split the linear PrA$[t]$ term over the variables $\boldsymbol{y}$ into $\sigma(\boldsymbol{y}) \cdot t$ and a linear term with integer coefficients $\rho(\boldsymbol{y})$; or
- $\sigma(\boldsymbol{y}) \cdot t + \rho(\boldsymbol{y}) + (\tau(\boldsymbol{z}) \bmod f(t))$, where, again, $\rho(\boldsymbol{y})$ is a linear term with integer coefficients.

Here, in both cases, $\tau(\boldsymbol{z})$ is a linear non-shifted term. It is clear that Equation (6) unifies both items.

Let us assume that at the beginning of the k-th iteration every term with a non-integer coefficient of a variable from $\boldsymbol{y}$ has the form Equation (6). Consider the updates in φ performed during one iteration of the loop. This formula is only modified in line 17, where the constraint $(\eta \leq 0)$ is replaced with a conjunction

$$\gamma \wedge \left(\sigma(\boldsymbol{y}) + r + \left\lfloor \frac{\left\lfloor \frac{\tau(\boldsymbol{z})}{t^k} \right\rfloor}{t} \right\rfloor \sim 0\right), \tag{7}$$

where r is an integer (see lines 11 and 16) and γ is either an equality $(t \cdot r = \rho(\boldsymbol{y}))$ (line 13) or a conjunction of two inequalities $(t \cdot r \leq \rho(\boldsymbol{y})) \wedge (\rho(\boldsymbol{y}) \leq t \cdot (r+1) - 1)$ (line 15).

By induction hypothesis, $\sigma(\boldsymbol{y})$ is a linear term with coefficients in $\mathbb{Z}[t]$. Let us represent this term as $\sigma'(\boldsymbol{y}) \cdot t + \rho'(\boldsymbol{y})$, where ρ' is a linear term with integer coefficients. The term $\sigma'(\boldsymbol{y})$ can be equal to zero; this happens when $\sigma(\boldsymbol{y})$ is a linear term with integer coefficients. We can now rewrite Equation (7) as

$$\gamma \wedge \left(\sigma'(\boldsymbol{y}) \cdot t + (\rho'(\boldsymbol{y}) + r) + \left\lfloor \frac{\tau(\boldsymbol{z})}{t^{k+1}} \right\rfloor \sim 0\right). \tag{8}$$

It remains to notice that, by induction hypothesis, $\rho(\boldsymbol{y})$ is a linear term over $\mathbb{Z}$, and thus there are no occurrences of a variable from $\boldsymbol{y}$ with a non-integer coefficient in γ. ◀

The "divisions by the parameter t" are formalised in the second auxiliary lemma, which gives us the two necessary equivalences already introduced in the sketch of the proof of Lemma 14 given in the body of the paper.

▶ **Lemma 37.** *Let* $C, D \in \mathbb{Z}$*, with* $C \leq D$*. Then, for the parameter* $t \in \mathbb{N}$ *such that* $t \geq 2$*,* $z \in \mathbb{Z}$*, and* $w \in [t \cdot C, t \cdot D]$*, the following equivalences hold:*

(a) $t \cdot z + w = 0 \iff \bigvee_{r \in [C,D]} (z + r = 0 \wedge t \cdot r = w)$*,*

(b) $t \cdot z + w < 0 \iff \bigvee_{r \in [C,D]} (z + r < 0 \wedge t \cdot r \leq w \wedge w < t \cdot (r+1))$*.*

Proof. Since we know that $w \in [t \cdot C, t \cdot D]$, there is an integer $r^* \in [C, D]$ such that $\left\lfloor \frac{w}{t} \right\rfloor = r^*$.

Item (a): For the left-to-right direction, the equality $t \cdot z + w = 0$ implies that w is divisible by t. Hence, $w = t \cdot r^*$, and we have $t \cdot z + t \cdot r^* = 0$ which can be rewritten as $z + r^* = 0$ (since $t \geq 2$). The right-to-left direction is trivial.

Item (b): Observe that the fact that $t \cdot r \leq w \wedge w < t \cdot (r+1)$ for some $r \in [C, D]$ means that $r = r^*$, and it is sufficient to prove the equivalence $t \cdot z + w < 0 \iff z + r^* < 0$. Below, we write $\{x\}$ for the fractional part of a real number $x \in \mathbb{R}$, that is, $\{x\}$ is the only real number such that $0 \leq \{x\} < 1$ and $x = \lfloor x \rfloor + \{x\}$. We have:

$$\begin{aligned}
t \cdot z + w < 0 &\iff t \cdot z + t \cdot \frac{w}{t} < 0 && \text{recall: } t \geq 2 \\
&\iff t \cdot z + t \cdot \left(\left\lfloor \frac{w}{t} \right\rfloor + \left\{ \frac{w}{t} \right\} \right) < 0 \\
&\iff z + \left(\left\lfloor \frac{w}{t} \right\rfloor + \left\{ \frac{w}{t} \right\} \right) < 0 \\
&\iff z + \left\lfloor \frac{w}{t} \right\rfloor < 0 && \text{since } 0 \leq \left\{ \frac{w}{t} \right\} < 1 \\
&\iff z + r^* < 0. && \blacktriangleleft
\end{aligned}$$

Thanks to the auxiliary Lemmas 36 and 37, we are now ready to prove Lemma 14:

▶ **Lemma 14.** *Consider $\boldsymbol{s} \in R_i$ with $\deg(\boldsymbol{y}, \varphi_{\boldsymbol{s}}) > 0$, and the set $G \coloneqq \{\boldsymbol{s}r \in R_{i+1} : r \in \mathbb{Z}\}$. 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. The $(i+1)$-th iteration of the **while** loop in lines 7–17 picks a constraint $(\eta \sim 0)$ from $\varphi_{\boldsymbol{s}}$ and divides it using Lemma 37. Since this lemma works with strict inequalities, if the symbol $\sim$ is the non-strict inequality sign $\leq$, in line 8 we transform $(\eta \leq 0)$ into an equivalent formula $(\eta - 1 < 0)$. We further assume that, in this case, η has been replaced with $\eta - 1$, and we work with the strict inequality $(\eta < 0)$.

By Lemma 36, every constraint $(\eta \sim 0)$, which has an occurrence of a variable from $\boldsymbol{y}$ with a non-integer coefficient, has the form (6), where now the symbol $\sim$ is from $\{=, <\}$. The term η is represented as

$$\eta = \eta' \cdot t + \rho', \text{ where } \quad \eta' = \sigma(\boldsymbol{y}) + \left\lfloor \frac{\tau(\boldsymbol{z})}{t^{k+1}} \right\rfloor \quad \text{and} \quad \rho' = \rho(\boldsymbol{y}) + \left(\left\lfloor \frac{\tau(\boldsymbol{z})}{t^k} \right\rfloor \bmod t \right)$$

Notice that the term ρ' is exactly the term assigned to the variable ρ in line 10 of Algorithm 4. This term is a sum of a linear polynomial with integer coefficients $\rho(\boldsymbol{y})$ and a linear occurrence of the integer remainder function with coefficient 1. Therefore, we can compute $\|\rho'\|_1$ and obtain that $\rho' \in \left[t \cdot (-\|\rho'\|_1), t \cdot \|\rho\|_1\right]$. Indeed, every variable from $\boldsymbol{y}$ and the summand with linear occurrence of $(\cdot \bmod t)$ belong to the segment $[0, t-1]$. With these bounds on ρ', we can apply Lemma 37 to the constraints $(\eta' \cdot t + \rho' = 0)$ and $(\eta' \cdot t + \rho' < 0)$ for the integers $C = (-\|\rho'\|_1)$ and $D = \|\rho'\|_1$. This gives us two equivalences

$$(\eta' \cdot t + \rho' = 0) \iff \bigvee_{r \in [-\|\rho'\|_1,\, \|\rho'\|_1]} (\eta' + r = 0) \quad \wedge\, (t \cdot r = \rho') \tag{9}$$

$$(\eta' \cdot t + \rho' < 0) \iff \bigvee_{r \in [-\|\rho'\|_1,\, \|\rho'\|_1]} (\eta' + r + 1 \leq 0) \wedge \underbrace{(t \cdot r \leq \rho') \wedge (\rho' < t \cdot (r+1))}_{\gamma}. \tag{10}$$

In the second equivalence, the strict inequality $(\eta' + r < 0)$ originated from (b) is replaced with the equivalent non-strict one $(\eta' + r + 1 \leq 0)$. This replacement is done in Algorithm 4 by line 16 and does not affect the formula γ, which is defined above, in line 15.

Since $\varphi_{\boldsymbol{s}}$ is a positive Boolean combination of atomic formulae, the logical equivalences $(\psi_1 \vee \psi_2) \wedge \psi_3 \iff (\psi_1 \wedge \psi_3) \vee (\psi_2 \wedge \psi_3)$ and $(\psi_1 \vee \psi_2) \vee \psi_3 \iff (\psi_1 \vee \psi_3) \vee (\psi_2 \vee \psi_3)$ together with Equations (9) and (10) give us the equivalence

$$\varphi_{\boldsymbol{s}} \iff \bigvee_{r \,\in\, [-\|\rho'\|_1,\, \|\rho'\|_1]} \varphi_{\boldsymbol{s}r}. \tag{11}$$

The formula $\varphi_{\boldsymbol{s}r}$ is the result of replacement of the constraint $(\eta \sim 0)$ with the conjunction on the right-hand side of Equation (9) if the symbol $\sim$ is $=$, and with the conjunction on the right-hand side of Equation (10) if $\sim$ is the strict inequality sign $<$. The formula $\varphi_{\boldsymbol{s}r}$ is assigned to φ in line 17, and this finishes the $(i+1)$-th iteration of the **while** loop.

Line 11 guesses an integer r in $[-\|\rho'\|_1, \|\rho'\|_1]$, and thus $G = \{\boldsymbol{s}r : r \in [-\|\rho'\|_1, -\|\rho'\|_1]\}$, and Equation (11) imply Item **(i)**. To prove Item **(ii)**, observe that

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

Indeed, the constraints from γ have degree 0 and $\deg(\theta + \tau \sim 0) = \deg(\theta \sim 0)$ for every PrA$[t]$ term τ not featuring variables from $\boldsymbol{y}$. Thus, Lemma 14 is proved. ◀

Let us now inductively apply Lemma 14 to show that the **while** loop performs at most $\deg(\boldsymbol{y}, \varphi_\varnothing)$ iterations, and that the disjunction, across non-deterministic branches, over all formulae $\varphi_{\boldsymbol{r}}$ obtained at the end of this loop is equivalent to $\varphi_\varnothing$.

▶ **Lemma 38.** *The **while** loop of line 7 performs at most* $\deg(\boldsymbol{y}, \varphi_\varnothing)$ *iterations.*

Proof. Directly from Item **(ii)** of Lemma 14. ◀

▶ **Lemma 39.** *Let* $R_* \coloneqq \{\boldsymbol{s} \in R_i : i \in \mathbb{N}, \deg(\boldsymbol{y}, \varphi_{\boldsymbol{s}}) = 0\}$. *We have* $\varphi_\varnothing \iff \bigvee_{\boldsymbol{s} \in R_*} \varphi_{\boldsymbol{s}}$.

Proof. Let us define a set $R_*^{(i)} \coloneqq \{\boldsymbol{s} \in R_j : j \in \mathbb{N}, \text{ and } \deg(\boldsymbol{y}, \varphi_{\boldsymbol{s}}) = 0 \text{ or } i = j\}$, and an non-negative integer $m \coloneqq \deg(\boldsymbol{y}, \varphi_\varnothing)$. By induction on i from m to 0, we prove the following equivalence: $\bigvee_{\boldsymbol{s} \in R_*^{(i)}} \varphi_{\boldsymbol{s}} \iff \bigvee_{\boldsymbol{s} \in R_*} \varphi_{\boldsymbol{s}}$. The statement then follows from $R_*^{(0)} = \{\varnothing\}$.

For the base case $i = m$, directly from Lemma 38 we have $R_*^{(m)} = R_*$. For the induction step, assume that $\bigvee_{\boldsymbol{s} \in R_*^{(i+1)}} \varphi_{\boldsymbol{s}} \iff \bigvee_{\boldsymbol{s} \in R_*} \varphi_{\boldsymbol{s}}$ (induction hypothesis), and let $\boldsymbol{r} \in R_i$. If we have $\deg(\boldsymbol{y}, \varphi_{\boldsymbol{r}}) = 0$, then the sequence $\boldsymbol{r}$ is also in the set $R_*^{(i+1)}$. Else $\deg(\boldsymbol{y}, \varphi_{\boldsymbol{r}}) > 0$, and by Item **(i)** of Lemma 14 we have $\varphi_{\boldsymbol{r}} \iff \bigvee_{\boldsymbol{g} \in G} \varphi_{\boldsymbol{g}}$, where $G \coloneqq \{\boldsymbol{r}g \in R_{i+1} : g \in \mathbb{Z}\}$. Since $G \subseteq R_*^{(i+1)}$, we conclude that $\bigvee_{\boldsymbol{s} \in R_*^{(i)}} \varphi_{\boldsymbol{s}} \iff \bigvee_{\boldsymbol{s} \in R_*^{(i+1)}} \varphi_{\boldsymbol{s}}$, which together with the induction hypothesis concludes the proof. ◀

B.3 Correctness of ElimBounded (proof of Lemma 15)

We can now complete the proof of Lemma 15. By Lemma 38, the **while** loop does at most $\deg(\boldsymbol{y}, \varphi_\varnothing)$ iterations, and Lemma 39 implies that $\varphi_\varnothing$ is equivalent to $\bigvee_{\boldsymbol{r} \in R_*} \varphi_{\boldsymbol{r}}$, where $R_* \coloneqq \{\boldsymbol{s} \in R_i : i \in \mathbb{N}, \deg(\boldsymbol{y}, \varphi_{\boldsymbol{s}}) = 0\}$. Every constraint in this disjunction as the form

$$f(t) + \tau(\boldsymbol{z}) + \sum_{i=1}^{N} a_i \cdot y_i \sim 0,$$

where $\boldsymbol{y} = (y_1, \ldots, y_N)$, the coefficients $a_1, \ldots, a_N$ are in $\mathbb{Z}$, f is in $\mathbb{Z}[t]$, $\tau(\boldsymbol{z})$ is a non-shifted PrA$[t]$ term, and the symbol $\sim$ is, as usual, from the set $\{=, \leq\}$. Recall that $N = \#\boldsymbol{w} \cdot M$, because every variable y_i is a t-digit of a variable $w \in \boldsymbol{w}$ from the input formula. We now see that the variables $\boldsymbol{y}$ can be eliminated from $\exists \boldsymbol{y} : \varphi_{\boldsymbol{r}}$ via any quantifier elimination procedure for PrA. In order to perform this step efficiently, we call BoundedQE in line 22.

Formally, BoundedQE works with positive Boolean combinations of linear constraints with coefficients in $\mathbb{Z}[t]$, and it is necessary to replace each term $\tau(\boldsymbol{z})$ with a fresh variable. These replacements are performed by lines 18–21; let us call $\varphi'_{\boldsymbol{r}}(\boldsymbol{y}, \boldsymbol{z}')$ the resulting formula.

The map S collects the key-value pairs $(z', \tau(\boldsymbol{z}))$ in order to restore the terms with the variables $\boldsymbol{z}$ after elimination of $\boldsymbol{y}$ from $\varphi'_{\boldsymbol{r}}$. This, in particular, means that $\varphi_{\boldsymbol{r}} = \varphi'_{\boldsymbol{r}}[[S(z') \,/\, z'] : z' \in \boldsymbol{z}']$.

By Lemmas 9 and 10, the result of application of the non-deterministic procedure BoundedQE to $\exists \boldsymbol{y} : \varphi'_{\boldsymbol{r}}(\boldsymbol{y}, \boldsymbol{z}')$ is a bounded formula $\exists \boldsymbol{w}'_\delta \le B'_\delta : \psi_\delta(\boldsymbol{w}'_\delta, \boldsymbol{z}')$, where for every variable $w \in \boldsymbol{w}'_\delta$, the bound $B'_\delta(w)$ is a non-negative integer, and we have the equivalence

$$\exists \boldsymbol{y} : \varphi'_{\boldsymbol{r}}(\boldsymbol{y}, \boldsymbol{z}') \iff \bigvee_{\delta \in \Delta_{\boldsymbol{r}}} \exists \boldsymbol{w}'_\delta \le B'_\delta : \psi_\delta(\boldsymbol{w}'_\delta, \boldsymbol{z}'), \tag{12}$$

where the set $\Delta_{\boldsymbol{r}}$ is the union over all non-deterministic branches of BoundedQE. Since this algorithm only produces a bounded existential formula, to eliminate every bounded variable $w \in \boldsymbol{w}'_\delta$ we guess an integer g from the segment $[0, B'_\delta(w)]$ and replace w with g in ψ_δ. This is done by lines 23–25. Let us gather all the substitutions $[g \,/\, w]$ from line 25 together and denote by $\boldsymbol{g}$ the corresponding sequence of substitutions. The set of all possible values of $\boldsymbol{g}$ for the formula ψ_δ will be denoted by G_δ.

Proof of Lemma 15. Let us define the set $\mathcal{B}$ as the union of all sequences $(\boldsymbol{r}, \delta, \boldsymbol{g})$, where the sequence $\boldsymbol{r}$ is from R_*, δ is in $\Delta_{\boldsymbol{r}}$, and $\boldsymbol{g} \in G_\delta$. For every sequence $\beta = (\boldsymbol{r}, \delta, \boldsymbol{g}) \in \mathcal{B}$, denote by $\psi'_\beta(\boldsymbol{z}')$ the result of application of the substitutions $\boldsymbol{g}$ to the formula $\psi_\delta(\boldsymbol{w}'_\delta, \boldsymbol{z}')$. We obtain the following chain of equivalences:

$$\begin{aligned}
&\exists \boldsymbol{w} \le B : \varphi(\boldsymbol{w}, \boldsymbol{z}) \\
\iff &\exists \boldsymbol{y} : \varphi_\varnothing(\boldsymbol{y}, \boldsymbol{z}) && \text{Lemma 12} \\
\iff &\exists \boldsymbol{y} : \bigvee_{\boldsymbol{r} \in R_*} \varphi_{\boldsymbol{r}}(\boldsymbol{y}, \boldsymbol{z}) && \text{Lemma 39} \\
\iff &\bigvee_{\boldsymbol{r} \in R_*} \big(\exists \boldsymbol{y} : \varphi'_{\boldsymbol{r}}(\boldsymbol{y}, \boldsymbol{z}')\big)[\boldsymbol{S}] && \varphi_{\boldsymbol{r}} = \varphi'_{\boldsymbol{r}}[\boldsymbol{S}] \text{ for } \boldsymbol{S} = \{[S(z') \,/\, z'] : z' \in \boldsymbol{z}'\} \\
\iff &\bigvee_{\boldsymbol{r} \in R_*} \bigvee_{\delta \in \Delta_{\boldsymbol{r}}} \big(\exists \boldsymbol{w}'_\delta \le B'_\delta : \psi_\delta(\boldsymbol{w}'_\delta, \boldsymbol{z}')\big)[\boldsymbol{S}] && \text{Equation (12)} \\
\iff &\bigvee_{\beta \in \mathcal{B}} \psi'_\beta(\boldsymbol{z}')[\boldsymbol{S}] && \text{by definition of } \mathcal{B} \\
\iff &\bigvee_{\beta \in \mathcal{B}} \psi_\beta(\boldsymbol{z}). && \text{where } \psi_\beta(\boldsymbol{z}) \coloneqq \psi'_\beta(\boldsymbol{z}')[\boldsymbol{S}]
\end{aligned}$$

Hence, we have obtained the desired equivalence from the specification of Algorithm 4. ◀

Before moving to the complexity of ElimBounded, let us establish Lemma 16.

▶ **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.*

Proof. The first statement follows from the fact that, in line 9, ElimBounded splits the term η as $(\sigma(\boldsymbol{y}) \cdot t + \rho(\boldsymbol{y}) + \tau(\boldsymbol{z}))$, where ρ does not contain t, and τ *is non-shifted.* All functions introduced by the algorithm are applied to these terms $\tau(\boldsymbol{z})$. Following the specification of ElimBounded, occurrences of functions that comes instead from the input formula are of the form $(\tau'(\boldsymbol{z}) \bmod f(t))$, where τ' is again linear and non-shifted.

For the second statement, observe that the input of ElimBounded does not contain any divisibility constraint, and only BoundedQE adds these constraints. From Lemma 10, the divisibility constraints added are of the form $(d \mid \cdot)$, where d is an integer. ◀

B.4 Complexity of ElimBounded

We now proceed to the complexity analysis, which uses the parameters of formulae defined in Appendix A.3.

▶ **Lemma 40.** *Let $\exists \boldsymbol{w} \leq B : \varphi(\boldsymbol{w}, \boldsymbol{z})$ be a formula in input of* ElimBounded*, where φ is a positive Boolean combination of linear (in)equalities with coefficients in $\mathbb{Z}[t]$ and constraints $\sigma(\boldsymbol{w}) + (\tau(\boldsymbol{z}) \bmod f(t)) = 0$, with σ linear, and τ linear and non-shifted. Let $\boldsymbol{w} = (x_1, \ldots, x_n)$ and $\boldsymbol{z} = (x_{n+1}, \ldots, x_g)$, where $n \geq 1$. Then, for every branch output $\psi(\boldsymbol{z})$ of the algorithm, the following holds:*

$$\text{if } \begin{cases} \mathit{atom}(\varphi) & \leq a \\ \mathit{func}(\varphi) & \leq 1 \\ \langle \mathit{const} \rangle(\varphi) & \leq c \end{cases} \quad \text{then } \begin{cases} \mathit{atom}(\psi) & \leq 2^7 \cdot a \cdot n \cdot c \cdot \langle B \rangle \\ \mathit{func}(\psi) & \leq 2^7 \cdot a \cdot n \cdot c \cdot \langle B \rangle \\ \langle \mathit{const} \rangle(\psi) & \leq (2^{11} \cdot a \cdot n^3 \cdot c^3 \cdot \langle B \rangle^4)^{16} \end{cases}$$

Proof. The same as in the proof of Lemma 34, we consider each parameter separately:

Bound on $\mathit{atom}(\psi)$: In the first line of ElimBounded, the bounds on the variables $\boldsymbol{w}$, i.e., 2 inequalities for each of the n variables, are explicitly added to the formula. Then, in line 6, we add $2 \cdot (\langle B \rangle + 1)$ inequalities for every $w \in \boldsymbol{w}$ specifying that the variables introduced by line 5 are t-digits. The same as in Section 5, let us denote by $\varphi_\varnothing$ the formula obtainded from φ after the execution of the **foreach** loop in line 4. We see that

$$\mathit{atom}(\varphi_\varnothing) \leq a + 2 \cdot n + 2 \cdot n \cdot (\langle B \rangle + 1) = a + 2 \cdot n \cdot (\langle B \rangle + 2).$$

By Lemma 38, the **while** loop in line 7 performs at most $\deg(\boldsymbol{y}, \varphi_\varnothing)$ iterations. Observe that for every variable from $\boldsymbol{y}$, its coefficent $f(t)$ is such that $\deg(f) \leq \langle B \rangle + \langle \mathit{const} \rangle(\varphi)$ (see line 6, where the coefficients of the newly introduced variables are equal to a coefficient of x multiplied by t^k for $k \in [0, \langle B \rangle]$.) Since $\varphi_\varnothing$ have at most $(a + 2 \cdot n)$ constraints where the variables $\boldsymbol{y}$ can have non-integer coefficients, i.e., the constraints of φ and those added by line 1, we conclude that $\deg(\boldsymbol{y}, \varphi_\varnothing) \leq (a + 2 \cdot n) \cdot (\langle B \rangle + c)$.

Every iteration of the **while** loop replaces one constraint with either two (lines 13 and 17) or three constraints (lines 15 and 17). Here and in the following, φ' will be the formula obtained by the end of execution of this loop. We see that

$$\begin{aligned} \mathit{atom}(\varphi') &\leq a + 2 \cdot n \cdot (\langle B \rangle + 2) + 3 \cdot (a + 2 \cdot n) \cdot (\langle B \rangle + c) \\ &\leq a + 6 \cdot n \cdot \langle B \rangle + 18 \cdot a \cdot n \cdot \langle B \rangle \cdot c \\ &\leq 2^5 \cdot a \cdot n \cdot c \cdot \langle B \rangle \end{aligned}$$

Lines 19–21 and 23–25 does not change the number of atomic formulae, and in order to obtain the bounds on $\mathit{atom}(\psi)$ it remains to apply Lemma 34:

$$\begin{aligned} \mathit{atom}(\psi) &\leq 2 \cdot (\#\boldsymbol{y} + \mathit{atom}(\varphi')) + 1 \\ &\leq 2 \cdot (n \cdot (\langle B \rangle + 1) + 2^5 \cdot a \cdot n \cdot c \cdot \langle B \rangle + 1) \\ &\leq 2^7 \cdot a \cdot n \cdot c \cdot \langle B \rangle. \end{aligned}$$

Bound on $\mathit{func}(\psi)$: To estimate the number of occurrences of $\left\lfloor \frac{\cdot}{t^d} \right\rfloor$ and $(\cdot \bmod f(t))$ in each atomic formula of ψ, consider lines 10 and 17. By Lemma 36, the term η in line 9 contains at most one occurrence of $\left\lfloor \frac{\cdot}{t^d} \right\rfloor$ and at most one occurrence of $(\cdot \bmod f(t))$. Hence, every constraint of the formula γ (see lines 10, 13, and 15) has at most at most 2 occurrences of $(\cdot \bmod f(t))$ and at most 1 occurrence of $\left\lfloor \frac{\cdot}{t^d} \right\rfloor$. These constraints have only

integer coefficients of the variables in $\boldsymbol{y}$ and will not be taken into account by the next iterations of the **while** loop. Line 17 only changes the power of the parameter t in the denominator (rewriting $\left\lfloor \frac{\left\lfloor \frac{\tau}{t} \right\rfloor}{t^d} \right\rfloor$ as $\left\lfloor \frac{\tau}{t^{d+1}} \right\rfloor$) and does not change the number of occurrences of $(\cdot \bmod f(t))$ and $\left\lfloor \frac{\cdot}{t^d} \right\rfloor$ in the constraint $(\eta \sim 0)$, which was replaced. For the formula φ' (obtained by the end of execution of the **while** loop), we see that $\mathit{func}(\varphi') \leq 3$.

Lines 19–21 introduce at most $\mathit{atom}(\varphi')$ new variables $\boldsymbol{z'}$, and for every $z' \in \boldsymbol{z'}$, the term $S(z')$ has at most 3 occurrences of $(\cdot \bmod f(t))$ and $\left\lfloor \frac{\cdot}{t^d} \right\rfloor$. The variables $\boldsymbol{z'}$ will occur in linear constraints with integer coefficients $\psi'(\boldsymbol{w'}, \boldsymbol{z'})$ produced by the non-deterministic procedure BoundedQE in line 22. Therefore, after the replacement in ψ' all the variables $z' \in \boldsymbol{z'}$ with $S(z')$ in line 26, we obtain the bounds

$$\mathit{func}(\psi) \leq \mathit{func}(\varphi') \cdot \mathit{atom}(\varphi') \leq 2^7 \cdot a \cdot n \cdot c \cdot \langle B \rangle.$$

Bound on $\langle const \rangle(\psi)$: The value of $\langle const \rangle$ for the formulae added to φ in line 1 is clearly bounded by $\langle B \rangle$. After the replacement of w in line 6, the coefficients of the variables $\boldsymbol{y}$ are at most $\max\{\langle f(t) \cdot t^{\langle B \rangle} \rangle : f(t) \text{ is a coefficient of } w \in \boldsymbol{w}\}$. Before we continue, let us observe that

▶ Remark 41. For any two polynomials $p_1, p_2 \in \mathbb{Z}[t]$ we have $\langle p_1 \cdot p_2 \rangle \leq \langle p_1 \rangle \cdot \langle p_2 \rangle$.

This follows from the following sequence of inequalities:

$$\begin{aligned}
\langle p_1 \cdot p_2 \rangle &\leq (\deg(p_1) + \deg(p_2) + 1) \cdot (\lceil \log_2(h(p_1) \cdot h(p_2) + 1) \rceil + 1) \\
&\leq (\deg(p_1) + \deg(p_2) + 1) \cdot (\lceil \log_2(h(p_1) + 1) \rceil + \lceil \log_2(h(p_2) + 1) \rceil + 1) \\
&\leq (\deg(p_1) + 1) \cdot (\deg(p_2) + 1) \cdot (\lceil \log_2(h(p_1) + 1) \rceil + 1) \cdot (\lceil \log_2(h(p_2) + 1) \rceil + 1) \\
&\leq \langle p_1 \rangle \cdot \langle p_2 \rangle.
\end{aligned}$$

Therefore, we have

$$\langle const \rangle(\varphi_\varnothing) \leq \max\{c, \langle B \rangle, c \cdot \langle t^{\langle B \rangle} \rangle, \langle t - 1 \rangle\} \leq c \cdot (\langle B \rangle + 1) \cdot 2 \leq 4 \cdot c \cdot \langle B \rangle,$$

where, as before, $\varphi_\varnothing$ is the formula obtainded from φ after the execution of the **foreach** loop in line 4. Let us now consider the constants of the formula φ'.

It is sufficient to estimate the size of the polynomials in the constraints introduced for an inequality $(\eta \leq 0)$, which is affected by line 17 at most $K \coloneqq \langle B \rangle + c$ times. Since the term $\tau(\boldsymbol{z})$ is non-shifted, at every $(k+1)$-th iteration of the **while** loop the constant term of η goes to the linear term with integer coefficients $\rho_{k+1}(\boldsymbol{y})$, which is the sum of the coefficient $\rho_k(\boldsymbol{y})$ of t^k in $\sigma(\boldsymbol{y})$, and of the integer r_k guessed in line 11 during the k-th iteration (see line 17). Here, we assume that $r_0 = 0$. It is clear that $\|\rho_{k+1}\|_1 \leq \|\rho'_k\|_1 + |r_k|$ and, moreover, that *for every* $k \in [0, K-1]$ we have

$$\langle \|\rho'_k\|_1 \rangle \leq \langle (\#\boldsymbol{y} + 1) \rangle \cdot \langle const \rangle(\varphi_\varnothing) \leq n \cdot (\langle B \rangle + 1) \cdot 4 \cdot c \cdot \langle B \rangle \leq 2^3 \cdot n \cdot c \cdot \langle B \rangle^2.$$

The integer r_k is guessed in line 11 from the segment $[-\|\rho_k\|_1 - 1, \|\rho_k\|_1 + 1]$ (see the previous line 10 for the '+1'). Therefore, by the end of the execution of the loop, we obtain

$$\begin{aligned}
\langle \|\rho_K\|_1 \rangle &\leq \langle \|\rho'_{K-1}\|_1 + |r_{K-1}| \rangle \\
&\leq \langle \|\rho'_{K-1}\|_1 + \|\rho_{K-1}\|_1 + 1 \rangle \\
&\leq \langle \|\rho'_{K-1}\|_1 + \|\rho'_{K-2}\|_1 + |r_{K-2}| + 1 \rangle \leq \dots \\
&\leq \langle K \cdot \max\{\|\rho'_k\|_1 : k \in [0, K-1]\} + |r_0| + (K-1) \rangle \\
&\leq (\langle B \rangle + c) \cdot (2^3 \cdot n \cdot c \cdot \langle B \rangle^2 + 1) \leq 2^5 \cdot n \cdot c^2 \cdot \langle B \rangle^3.
\end{aligned}$$

We can now upper-bound $\langle const\rangle(\varphi')$, where, again, φ' is the formula obtained by the end of execution of the loop. The linear constraints introduced by lines 13 and 15 have coefficients bounded by $\|\rho_K\|_1 + 2$; the coefficients of the term σ are now integers bounded the same as $\|\rho'_k\|_1$ for any $k \in [0, K]$, and thus are bounded by $\|\rho_K\|_1$. The coefficients of the variables $\boldsymbol{z}$ in $\tau(\boldsymbol{z})$ does not change, i.e., their bit size is at most $\langle const\rangle(\varphi_\varnothing)$; the denominator of the term $\left\lfloor \frac{\tau(\boldsymbol{z})}{t^K} \right\rfloor$ is the polynomial t^K and, by definition, $\langle t^K\rangle = (K+1)\cdot(1+1) \leq 4\cdot K < \langle \|\rho_K\|_1\rangle$. Summarising,

$$\langle const\rangle(\varphi') \;\leq\; \langle \|\rho_K\|_1 + 2\rangle \leq 2^7 \cdot n \cdot c^2 \cdot \langle B\rangle^3.$$

Lines 19–21 replace the PrA$[t]$ terms with $\boldsymbol{z}$ with new variables $\boldsymbol{z'}$, which will be restored in the formula in line 26. For this reason, we first consider the bounds on the bit size of integer coefficients of $\boldsymbol{z'}$ in the output $\exists \boldsymbol{w'} \leq B' : \psi'(\boldsymbol{w'}, \boldsymbol{z'})$ of the non-deterministc procedure BoundedQE. Applying the bounds from Lemma 34, we obtain

$$\begin{aligned}
\langle const\rangle(\psi') &\leq (\#\boldsymbol{y} + atom(\varphi'))^3 \cdot \langle const\rangle(\varphi')^3 \\
&\leq (n\cdot(\langle B\rangle + 1) + 2^5 \cdot a \cdot n \cdot c \cdot \langle B\rangle)^3 \cdot \langle const\rangle(\varphi')^3 \\
&\leq (2^6 \cdot a \cdot n \cdot c \cdot \langle B\rangle \cdot 2^7 \cdot n \cdot c^2 \cdot \langle B\rangle^3)^3 \\
&\leq (2^{13} \cdot a \cdot n^2 \cdot c^3 \cdot \langle B\rangle^4)^3.
\end{aligned}$$

The size of the vector of variables $\boldsymbol{w'}$ is equal to $\#\boldsymbol{y}$, and every constraint of ψ' is a linear (in)equality or divisibility with integer coefficients. Denote by $\psi''(\boldsymbol{z'})$ the result of the replacements performed by lines 23–25. Using the bounds from Lemma 34 on the bit size of $B'(w')$ for the variables $w' \in \boldsymbol{w'}$, the constants of ψ'' are such that

$$\begin{aligned}
\langle const\rangle(\psi'') &\leq (\#\boldsymbol{y}+1)\cdot\langle const\rangle(\psi')\cdot\left((\#\boldsymbol{y} + atom(\varphi'))^{11}\cdot\langle const\rangle(\varphi')^9\right) \\
&\leq 2\cdot\langle const\rangle(\psi')\cdot\#\boldsymbol{y}\cdot(\#\boldsymbol{y} + atom(\varphi'))^{11}\cdot\langle const\rangle(\varphi')^9 \\
&\leq 2\cdot\langle const\rangle(\psi')\cdot(2\cdot n\cdot\langle B\rangle)^{12}\cdot(2^5\cdot a\cdot n\cdot c\cdot\langle B\rangle)^{11}\cdot(2^7\cdot n\cdot c^2\cdot\langle B\rangle^3)^9 \\
&= \langle const\rangle(\psi')\cdot 2^{131}\cdot a^{11}\cdot n^{32}\cdot c^{29}\cdot\langle B\rangle^{50} \\
&\leq 2^{39}\cdot(a\cdot n^2\cdot c^3\cdot\langle B\rangle^4)^3\cdot 2^{131}\cdot a^{11}\cdot n^{32}\cdot c^{29}\cdot\langle B\rangle^{50} \\
&= 2^{170}\cdot a^{14}\cdot n^{38}\cdot c^{38}\cdot\langle B\rangle^{62} \\
&< (2^{11}\cdot a\cdot n^3\cdot c^3\cdot\langle B\rangle^4)^{16}.
\end{aligned}$$

It remains to observe that the replacements of line 26 transforms $\psi''(\boldsymbol{z'})$ into $\psi(\boldsymbol{z})$, where the constants have the desired bounds:

$$\begin{aligned}
\langle const\rangle(\psi) &\leq \max\{\langle const\rangle(\psi''), \langle const\rangle(\psi')\cdot\langle const\rangle(\varphi')\} \\
&\leq \max\{\langle const\rangle(\psi''), (2^{13}\cdot a\cdot n^2\cdot c^3\cdot\langle B\rangle^4)^3\cdot 2^7\cdot n\cdot c^2\cdot\langle B\rangle^3\} \\
&= \langle const\rangle(\psi'') \;<\; (2^{11}\cdot a\cdot n^3\cdot c^3\cdot\langle B\rangle^4)^{16}.
\end{aligned}$$

◀

C Complexity of ElimDiv and PrA$[t]$-QE

To prove Lemma 19, which describes the complexity of Algorithm 1 (PrA$[t]$-QE), we first formally prove correctness of Algorithm 2 (ElimDiv) together with an analog of Lemmas 34 and 40 for this algorithm (see Lemma 42 below). Combining bounds from Lemmas 34, 40, and 42, we obtain the desired result.

C.1 Correctness and complexity of ElimDiv

An informal description of the procedure ElimDiv is given in Section 3: every linear divisibility in the input formula is transformed into an equality by means of a bounded existential quantifier. Below we formalize the correctness of the transformation:

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

Proof. It is sufficient to prove that the divisibility $f(t) \mid \sigma(\boldsymbol{w}) + \tau(\boldsymbol{z})$, which is picked from the input formula ψ, is equivalent to either $\exists y : f(t) \cdot y + \sigma(\boldsymbol{w}) + (\tau \bmod f(t)) = 0$ or $\exists y : (-f(t) \cdot y + \sigma(\boldsymbol{w}) + (\tau \bmod f(t)) = 0)$ for a variable $y \in [0, t^d]$, where d is a positive integer defined in line 3.

The direction from right to left is obvious. For the converse direction, notice that if there is no upper bound on the value of the non-negative integer variable y, then the replacement directly follows from the definition of the integer divisibility predicate. The sign of the coefficent of y corresponds to the sign of the integer $\frac{\sigma+(\tau \bmod f(t))}{f(t)}$: when it is positive, line 5 should take '$-$', and vice versa. Let us show that the bitlength of this integer is less than $\langle t^d \rangle = 2 \cdot (d+1)$, and this will justify the bound $y \in [0, t^d]$ introduced in line 4. Recall that we represent the linear PrA$[t]$ term σ as $f_0(t) + \sum_{i=1}^{n} f_i(t) \cdot w_i$, where $\boldsymbol{w} = (w_1, \ldots, w_n)$ is a vector of variables bounded by the map B, and d is defined as

$$d = (n+3) \cdot \max\{\langle f \rangle, \langle f_0 \rangle, \langle f_i \rangle \cdot \langle B(w_i) \rangle \ : i \in [1..n]\}.$$

Using Remark 41, we obtain

$$\begin{aligned}
\left\langle \frac{\sigma + (\tau \bmod f(t))}{f(t)} \right\rangle \ &\leq \ \langle \sigma + (\tau \bmod f(t)) \rangle \ \leq \ \langle f_0(t) + \sum_{i=1}^{n} f_i(t) \cdot B(w_i) + f(t) \rangle \\
&\leq \ \langle n+2 \rangle \cdot \max\{\langle f \rangle, \langle f_0 \rangle, \langle f_i \cdot B(w_i) \rangle \ : i \in [1..n]\} \\
&\leq \ (\lceil \log_2(n+3) \rceil + 1) \cdot \max\{\langle f \rangle, \langle f_0 \rangle, \langle f_i \rangle \cdot \langle B(w_i) \rangle \ : i \in [1..n]\} \\
&\leq \ d < \langle t^d \rangle.
\end{aligned}$$

◀

We now move to the complexity of the procedure.

▶ **Lemma 42.** *Let $\exists \boldsymbol{w} \leq B : \psi(\boldsymbol{w}, \boldsymbol{z})$ be a formula in input of ElimDiv, where $\boldsymbol{w} = (x_1, \ldots, x_n)$, $\boldsymbol{z} = (x_{n+1}, \ldots, x_g)$ with $n \geq 1$ and ψ is a positive Boolean combination of linear constraints. Then, for every branch output $\exists \boldsymbol{w}' \leq B' : \psi'(\boldsymbol{w}', \boldsymbol{z})$ of the algorithm, the following holds:*

$$\textit{if} \ \begin{cases} \mathit{atom}(\psi) & \leq a \\ \mathit{vars}(\psi) & = g \\ \mathit{func}(\psi) & = 0 \\ \langle \mathit{const} \rangle(\psi) & \leq c \end{cases} \quad \textit{then} \ \begin{cases} \mathit{atom}(\psi') & \leq a \\ \mathit{vars}(\psi') & \leq g + a \\ \mathit{func}(\psi') & \leq 1 \\ \langle \mathit{const} \rangle(\psi') & \leq c \end{cases}$$

Moreover, the vector $\boldsymbol{w}'$ contains $\mathit{vars}(\psi') + n - g$ variables, and $\langle B' \rangle \leq 2 \cdot (n+4) \cdot c \cdot \langle B \rangle$.

Proof. The bounds on the parameters *atom*, *vars*, *func*, and $\langle \mathit{const} \rangle$ are trivial. They are obtained by inspecting the lines of the pseudocode of Algorithm 2. The number of constraints does not change: line 6 replaces a linear divisibility with an equality with a single occurrence of the function $(\cdot \bmod f(t))$. Also notice that the polynomial coefficients occurring in this new equality are already present in the input formula ψ, and thus $\langle \mathit{const} \rangle(\psi) = \langle \mathit{const} \rangle(\psi')$. Finally, since the **foreach** loop has at most $\mathit{atom}(\psi)$ iterations, line 4 introduces at most a

new variables, each of which is added to the list of bounded variables $\boldsymbol{w}'$. It is clear that $\#\boldsymbol{w}' = (\mathit{vars}(\psi') - \#\boldsymbol{z}) = \mathit{vars}(\psi') + n - g$.

To upper-bound the bitlength of $B'(w')$ for the newly introduced variables $w' \in \boldsymbol{w}'$, consider lines 3 and 4. Let $\langle B' \rangle \coloneqq \max\{\langle B'(w') \rangle : w' \in \boldsymbol{w}'\}$, then we see that

$$\langle B' \rangle \leq \max\{\langle B \rangle, \langle t^{(n+3)\cdot c \cdot \langle B \rangle} \rangle\} \leq \max\{\langle B \rangle, 2 \cdot (n+4) \cdot c \cdot \langle B \rangle\} = 2 \cdot (n+4) \cdot c \cdot \langle B \rangle. \quad \blacktriangleleft$$

C.2 Complexity of PrA$[t]$-QE

The main Algorithm 1 (PrA$[t]$-QE) calls Algorithms 2–4, and the bounds from Lemmas 34, 40, and 42 will give us the upper bounds on the values of the four parameters from these lemmas. The next lemma essentially proves Lemma 19 from Section 6.

▶ **Lemma 43.** *Let $\exists \boldsymbol{x} : \varphi(\boldsymbol{x}, \boldsymbol{z})$ be a formula in input of* PrA$[t]$*-QE, where φ is a positive Boolean combination of* PrA$[t]$ *constraints. Let $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{z} = (x_{n+1}, \ldots, x_g)$, where $n \geq 1$, then for every branch output $\psi(\boldsymbol{z})$ of the algorithm, the following holds:*

$$\textit{if} \begin{cases} \mathit{atom}(\psi) & \leq a \\ \mathit{func}(\psi) & \leq d \\ \langle \mathit{const} \rangle(\psi) & \leq c \end{cases} \textit{then} \begin{cases} \mathit{atom}(\psi) & \leq 2^5 \cdot (n+d+4) \cdot (a+n+3\cdot d)^{19} \cdot c^{15} \\ \mathit{func}(\psi) & \leq 2^5 \cdot (n+d+4) \cdot (a+n+3\cdot d)^{19} \cdot c^{15} \\ \langle \mathit{const} \rangle(\psi) & \leq \left(2^6 \cdot (n+d+4) \cdot (a+n+3\cdot d)^{18} \cdot c^{15}\right)^{64}. \end{cases}$$

Proof. The two **while** loops in the pre-processing part remove all functions from the input formula. Denote by $\exists \boldsymbol{w}_0 : \varphi_0(\boldsymbol{w}_0, \boldsymbol{z})$ the result of this step. These loops together perform d iterations, and add at most 2 new constraints to the formula (see line 3, line 7 and line 11). Therefore, $\mathit{atom}(\varphi_0) \leq a + 2 \cdot d$. Next, these loops also introduce at most one variable during each iteration: $\mathit{vars}(\varphi_0) \leq g + d$. It is easy to see that $\mathit{func}(\varphi_0) = 0$ and $\langle \mathit{const} \rangle(\varphi_0) = \langle \mathit{const} \rangle(\varphi)$. Finally, the bounds introduced by line 10 (denote the corresponding map by B_0) can be bounded as follows: $\langle B_0 \rangle \leq 2 \cdot c$.

We are now going to apply the bounds constructed by Lemmas 34, 40, and 42 sequentially to the quintuple $(\mathit{atom}(\varphi_0), \mathit{vars}(\varphi_0), \mathit{func}(\varphi_0), \langle \mathit{const} \rangle(\varphi_0), \langle B_0 \rangle) = (a+2\cdot d,\ g+d,\ 0,\ c,\ 2\cdot c)$. Moreover, we know that $\#\boldsymbol{w}_0 \leq n + d$.

BoundedQE: Applying Lemma 40, we obtain that every branch output of BoundedQE on input φ_0 will have at most $2 \cdot (n+d+a+2\cdot d)+1 \leq 2 \cdot (a+n+3\cdot d)+1$ atomic formulae. It is now easy to obtain the bounds on parameters for every formula $\exists \boldsymbol{w}_1 \leq B_1 : \varphi_1(\boldsymbol{w}_1, \boldsymbol{z})$ given on input of algorithm ElimDiv:

$$\begin{cases} \mathit{atom}(\varphi_1) & \leq 2 \cdot (a+n+3\cdot d)+1 \\ \mathit{vars}(\varphi_1) & \leq g + d = (n+d) + \#\boldsymbol{z} \\ \mathit{func}(\varphi_1) & \leq 0 \\ \langle \mathit{const} \rangle(\varphi_1) & \leq (a+n+3\cdot d)^3 \cdot c^3 \\ \langle B_1 \rangle & \leq (a+n+3\cdot d)^{11} \cdot c^9, \end{cases}$$

because the bound on $\langle B_1 \rangle$, which is the union of B_0 and the map B from the output of BoundedQE is such that $\langle B_1 \rangle \leq \max\{(a+n+3\cdot d)^{11} \cdot c^9, \langle B_0 \rangle\} = (a+n+3\cdot d)^{11} \cdot c^9$. Lemma 40 also says that $\#\boldsymbol{w}_1 = \#\boldsymbol{w}_0 \leq n + d$.

ElimDiv: According to Lemma 42, algorithm ElimBounded will get a bounded formula $\exists \boldsymbol{w}_2 \leq B_2 : \varphi_2(\boldsymbol{w}_2, \boldsymbol{z})$ with the same values of the parameters *atom* and $\langle \mathit{const} \rangle$ as in formula φ_1. It is easy to derive the bounds for $\mathit{vars}(\varphi_2) \leq \mathit{vars}(\varphi_1) + \mathit{atom}(\varphi_1) \leq$

$g + d + 2 \cdot (a + n + 3 \cdot d) + 1 \leq 3 \cdot (a + n + 3 \cdot d) + \#\boldsymbol{z}$ and $\mathit{func}(\varphi_2) \leq 1$. The bit size of the bounds is such that

$$\begin{aligned}\langle B_2 \rangle &\leq 2 \cdot (n + d + 4) \cdot (a + n + 3 \cdot d)^3 \cdot c^3 \cdot (a + n + 3 \cdot d)^{11} \cdot c^9 \\ &\leq 2 \cdot (n + d + 4) \cdot (a + n + 3 \cdot d)^{14} \cdot c^{12}.\end{aligned}$$

To sum up, $\#\boldsymbol{w}_2 \leq 3 \cdot (a + n + 3 \cdot d)$ and we have

$$\begin{cases} \mathit{atom}(\varphi_2) & \leq 2 \cdot (a + n + 3 \cdot d) + 1 \\ \mathit{vars}(\varphi_2) & \leq 3 \cdot (a + n + 3 \cdot d) + \#\boldsymbol{z} \\ \mathit{func}(\varphi_2) & \leq 1 \\ \langle \mathit{const} \rangle(\varphi_2) & \leq (a + n + 3 \cdot d)^3 \cdot c^3 \\ \langle B_2 \rangle & \leq 2 \cdot (n + d + 4) \cdot (a + n + 3 \cdot d)^{14} \cdot c^{12}. \end{cases}$$

ElimBounded: We are now going to apply Lemma 40 to the formula $\exists \boldsymbol{w}_2 \leq B_2 : \varphi_2(\boldsymbol{w}_2, \boldsymbol{z})$.

Bound on $\mathit{atom}(\psi)$:

$$\begin{aligned}\mathit{atom}(\psi) &\leq \mathit{atom}(\varphi_2) \cdot 3 \cdot (a + n + 3 \cdot d) \cdot (a + n + 3 \cdot d)^3 \cdot c^3 \cdot \langle B_2 \rangle \\ &\leq (2 \cdot (a + n + 3 \cdot d) + 1) \cdot 3 \cdot (a + n + 3 \cdot d)^4 \cdot c^3 \cdot \langle B_2 \rangle \\ &\leq 9 \cdot (a + n + 3 \cdot d)^5 \cdot c^3 \cdot \langle B_2 \rangle \\ &\leq 9 \cdot (a + n + 3 \cdot d)^5 \cdot c^3 \cdot 2 \cdot (n + d + 4) \cdot (a + n + 3 \cdot d)^{14} \cdot c^{12} \\ &\leq 2^5 \cdot (n + d + 4) \cdot (a + n + 3 \cdot d)^{19} \cdot c^{15}.\end{aligned}$$

Bound on $\mathit{func}(\psi)$: the same as $\mathit{atom}(\psi)$ above.

Bound on $\langle \mathit{const} \rangle(\psi)$:

$$\begin{aligned}\mathit{func}(\psi) &\leq \left(2^{11} \cdot \mathit{atom}(\varphi_2) \cdot (3 \cdot (a + n + 3 \cdot d)^4 \cdot c^3)^3 \cdot \langle B_2 \rangle^4\right)^{16} \\ &\leq \left(2^{11} \cdot (2 \cdot (a + n + 3 \cdot d) + 1) \cdot (3 \cdot (a + n + 3 \cdot d)^4 \cdot c^3)^3 \cdot \langle B_2 \rangle^4\right)^{16} \\ &\leq \left(2^{18} \cdot (a + n + 3 \cdot d)^{13} \cdot c^9 \cdot \langle B_2 \rangle^4\right)^{16} \\ &\leq \left(2^{18} \cdot (a + n + 3 \cdot d)^{13} \cdot c^9 \cdot 2^4 \cdot (n + d + 4)^4 \cdot (a + n + 3 \cdot d)^{56} \cdot c^{48}\right)^{16} \\ &= \left(2^{22} \cdot (n + d + 4)^4 \cdot (a + n + 3 \cdot d)^{69} \cdot c^{57}\right)^{16} \\ &\leq \left(2^6 \cdot (n + d + 4) \cdot (a + n + 3 \cdot d)^{18} \cdot c^{15}\right)^{64}.\end{aligned}$$

◀

Proof of Lemma 19. As for Lemma 35, following the proof of Lemma 43, we now know that all object constructed by PrA[t]-QE during its execution are of polynomial bit size. It is then simple to deduce that the algorithm runs in non-deterministic polynomial time; it suffices to check the number of iterations of the various loops, and the sets used for the various guesses performed by the algorithm.

In Lines 1–11, the number of iterations in both **while** loops is bounded by the size of the input formula φ, and guesses (lines 6 and 9) are all on a 2-element domain (for the non-deterministic branches).

We already know from Lemma 35 that the computation performed by BoundedQE takes non-deterministic polynomial time. Since its output has size polynomial in the input formula φ, the **foreach** loop in algorithm ElimDiv also performs only polynomially many iterations; all guesses are again on a 2-element domain (line 5).

Going into ElimBounded, the loop in line 4 only executes $\#\boldsymbol{w}$ times, adding at most $\#\boldsymbol{w} \cdot \langle B \rangle$ variables $\boldsymbol{y}$ overall (a polynomial amount). Let $\varphi_\varnothing$ be the formula obtained after

these lines. The $\boldsymbol{y}$-degree $\deg(\boldsymbol{y}, \varphi_\varnothing)$ is polynomial in the input formula φ. By Lemma 14, the **while** loop of line 7 only iterates a polynomial amount of times. Following Lemma 40, the guesses performed in line 11 are over an interval of numbers having all polynomial bit size. After the **foreach** loop of line 19 (whose number of iterations is, again, polynomial), the algorithm calls again BoundedQE, which returns an output in (non-deterministic) polynomial time. In this output, all elements in the range of the map B' are integers of polynomial bit size. Then, in the **foreach** loop of line 23 (executing polynomially many times) all guesses are over a range of polynomial-size integers. ◀