Quantifier elimination for counting extensions of Presburger arithmetic

Dmitry Chistikov[1], Christoph Haase[2], and Alessio Mansutti[2](✉)

[1] Centre for Discrete Mathematics and its Applications (DIMAP) & Department of Computer Science, University of Warwick, Coventry, UK
d.chistikov@warwick.ac.uk

[2] Department of Computer Science, University of Oxford, Oxford, UK
{christoph.haase,alessio.mansutti}@cs.ox.ac.uk

Abstract. We give a new quantifier elimination procedure for Presburger arithmetic extended with a unary counting quantifier $\exists^{=x} y\, \Phi$ that binds to the variable x the number of different y satisfying Φ. While our procedure runs in non-elementary time in general, we show that it yields nearly optimal elementary complexity results for expressive counting extensions of Presburger arithmetic, such as the *threshold counting* quantifier $\exists^{\geq c} y\, \Phi$ that requires that the number of different y satisfying Φ be at least $c \in \mathbb{N}$, where c can succinctly be defined by a Presburger formula. Our results are cast in terms of what we call the *monadically-guarded fragment* of Presburger arithmetic with unary counting quantifiers, for which we develop a 2ExpSpace decision procedure.

1 Introduction

Counting the number of solutions to an equation, or the number of elements in a set subject to constraints, is a fundamental and often computationally challenging problem studied in logic, mathematics and computer science. In discrete geometry, counting the number of integral points in a polyhedron is a canonical #P-complete problem. Barvinok's celebrated algorithm solves this problem in polynomial time when the dimension is fixed [2]. In this paper, we investigate a generalization of this problem and study algorithmic aspects of counting the number of models of formulae of *Presburger arithmetic,* the first-order theory of the integers with addition and order, and more generally, extensions of this logic with *counting quantifiers*.

Counting quantifiers such as the *Härtig quantifier*, which allows to assert equal-cardinality constraints on the sets of satisfying assignments of two given first-order formulae, have long been studied in first-order logic [6]. In first-order theories of integer arithmetic, it is compelling to consider variants of counting quantifiers that bind the number of satisfying assignments of a formula to a first-order variable. Apelt [1] and Schweikardt [10] studied the decidability of Presburger arithmetic enriched with the unary counting quantifier $\exists^{=x} y$ with the following semantics: given an assignment of integers to the first-order variables $x, z_1, \ldots, z_n$, a formula $\exists^{=x} y\, \Phi(x, y, z_1, \ldots, z_n)$ evaluates to true whenever

the number of different y satisfying $\Phi(x, y, z_1, \ldots, z_n)$ is exactly x. In both [1] and [10], decidability is shown by developing a quantifier elimination procedure for this extension of Presburger arithmetic which eliminates a counting quantifier by translating it into an equivalent quantified formula of Presburger arithmetic, i.e., one that only uses standard first-order quantifiers. This immediately gives decidability of Presburger arithmetic extended with the unary counting quantifier $\exists^{=x} y$ since Presburger arithmetic is decidable in 2ExpSpace [9,3,12]. Unfortunately, the quantifier elimination procedures in [1,10] do not yield a similar elementary upper bound for the extended theory, as the elimination of a single quantifier $\exists^{=x} y$ results in an exponential blow-up of the formula size and introduces nested first-order quantifiers. It is a widely open problem whether there is a decision procedure for Presburger arithmetic extended with the counting quantifier $\exists^{=x} y$ with elementary running time, or whether this theory admits a significantly stronger lower bound than standard Presburger arithmetic.

To shed more light on the complexity of Presburger arithmetic extended with the aforementioned unary counting quantifier, Habermehl and Kuske gave a quantifier elimination procedure for Presburger arithmetic extended with a unary *modulo counting* quantifier $\exists^{(r,q)} y$, where r and q are positive natural numbers [4]. Here, $\exists^{(r,q)} y\, \Psi(y, z_1, \ldots, z_n)$ holds whenever the number of different y satisfying $\Psi(y, z_1, \ldots, z_n)$ is congruent to r modulo q. An analysis of the growth of the constants and coefficients occurring in their procedure then enables them to derive a 2ExpSpace upper bound for the logic, matching the complexity of Presburger arithmetic on deterministic machines. This noteworthy result shows that there is still room to extend Presburger arithmetic with non-trivial counting quantifiers without increasing the computational cost of deciding the logic.

Note that in order to keep the logic decidable, the counting quantifiers considered in the literature must be *unary*. Indeed, consider a binary counting quantifier $\exists^{=x}(y_1, y_2)$ counting the number of different y_1 and y_2 satisfying a formula. Then, $\Phi(x, z) = \exists^{=x}(y_1, y_2)(0 \le y_1, y_2 < z)$ holds for $x = z^2$, which in turn allows defining multiplication, leading to undecidability of the resulting theory.

Our contribution. Following the lines of [4] while trying to avoid the limitations of the procedures in [1,10], our goal is to study decision procedures for Presburger arithmetic enriched with variants of counting quantifiers that do not increase the complexity of the Presburger arithmetic. To begin with, we develop a new quantifier elimination procedure for Presburger arithmetic with unary counting quantifiers $\exists^{=x} y$ that, in contrast to [1,10], does not require the introduction of first-order quantifiers. While the procedure still runs in non-elementary time, avoiding first-order quantification allows us not only to derive exponentially better bounds on the size of the formula obtained after eliminating a single $\exists^{=x} y$, but also to identify the sources of non-elementary growth. We exploit those observations to extend the range of counting quantifiers that can be added to Presburger arithmetic without increasing the complexity of the resulting logic.

The first type of counting quantifiers we consider is a *threshold counting quantifier* $\exists^{\ge c} y$ for some integer c. A formula $\exists^{\ge c} y\, \Psi(y, z_1, \ldots, z_n)$ evaluates to true whenever there are at least c different values of y satisfying $\Psi(y, z_1, \ldots, z_n)$. We

show that Presburger arithmetic enriched with threshold counting quantifiers can be decided in 2ExpSpace, even when the threshold c itself is succinctly given as the unique solution of a Presburger arithmetic formula. This is surprising since in Presburger arithmetic one can define numbers that are triply exponential in the size of the formula used to encode them [7, pp. 151–152]. Furthermore, we show that if we restrict c to be at most doubly exponential in the size of its encoding then Presburger arithmetic with threshold counting quantifiers is decidable in $\mathrm{STA}(*, 2^{2^{n^{\mathcal{O}(1)}}}, \mathcal{O}(n))$, matching the complexity of Presburger arithmetic [3]. Here, $\mathrm{STA}(s(n), t(n), a(n))$ is the class of all decision problems in which inputs of length n can be decided by an alternating Turing machine in space $s(n)$ and time $t(n)$ using $a(n)$ alternations, where "$*$" stands for unbounded availability of a certain resource.

Our results on the quantifier $\exists^{\geq c} x$ arise from studying a more general extension of Presburger arithmetic that relies on the notion of monadic decomposition put forward by Veanes et al. in [11] and studied by Hague et al. [5] in the context of integer linear arithmetic. Briefly, a formula $\Phi(x, y_1, \ldots, y_n)$ is said to be monadically decomposable on the variable x whenever it is equivalent to a formula of the form $\bigvee_{i \in I} \Delta_i(x) \wedge \Psi_i(y_1, \ldots, y_n)$, i.e., a formula where the satisfaction of constraints on x does not depend on the values of $y_1, \ldots, y_n$. Based on this definition, we extend Presburger arithmetic by allowing the general unary counting quantifiers $\exists^{=x} y$ to appear with guards of the form $\exists\, x(\Psi \wedge \exists^{=x} y\, \Phi)$, where Ψ is monadically decomposable on the variable x. The resulting logic is very powerful, as it not only generalizes the quantifiers $\exists^{\geq c} x$ but also the modulo counting quantifiers $\exists^{(r,q)} y$ from [4]. We establish two further results for this *monadically-guarded fragment* of Presburger arithmetic with counting quantifiers. First, we develop a 3ExpTime quantifier elimination procedure for the logic, matching the complexity of the best possible quantifier elimination procedures for Presburger arithmetic. Second, we exploit this procedure to obtain a quantifier relativization argument showing that the logic is decidable in 2ExpSpace.

2 Presburger arithmetic with counting quantifiers

General notation. The symbols $\mathbb{Z}$, $\mathbb{N}$ and $\mathbb{N}_+$ denote the set of integers, natural numbers including zero, and natural numbers without zero, respectively. We usually use $a, b, c, \ldots$ for integers, which we assume being encoded in binary. Given $n \in \mathbb{N}$, we write $[n] \stackrel{\text{def}}{=} \{0, \ldots, n-1\}$, and $\#A$ for the cardinality of a set A. If A is infinite, then $\#A = \infty$, and we postulate $n \leq \infty$ for all $n \in \mathbb{Z}$.

Structure. We consider the structure $\mathcal{Z} = \langle \mathbb{Z}, (c)_{c \in \mathbb{Z}}, +, <, (\equiv_q)_{q \in \mathbb{N}_+} \rangle$ of Presburger arithmetic, where $(c)_{c \in \mathbb{Z}}$ are constant symbols that shall be interpreted as their homographic integer numbers, the binary function symbol $+$ is interpreted as addition on $\mathbb{Z}$, the binary relation $<$ is interpreted as "less than", and $\equiv_q$ is interpreted as the modulo relation, i.e., $a \equiv_q b$ if and only if q divides $a - b$.

Basic syntax. Let $X = \{x, y, z, \dots\}$ be a countable set of first-order variables. *Linear terms*, usually denoted by t, t_1, t_2, etc., are expressions of the form $a_1x_1 + \cdots + a_dx_d + c$ where $x_1, \dots, x_d \in X$, $a_1, \dots, a_d, c \in \mathbb{Z}$. The integer a_i is the *coefficient* of the variable x_i. Variables not appearing in the linear term are tacitly assumed to have a 0 *coefficient*. A term t is said to be *x-free* if the coefficient of the variable x in t is 0. The integer c is the *constant* of the linear term. Linear terms with constant 0 are said to be *homogeneous*.

Given a term t, the lexeme $t < 0$ is understood as a *linear inequality*, and $t \equiv_q 0$ is a *modulo constraint*. Syntactically, Presburger arithmetic (PA) is the closure of linear inequalities and modulo constraints under the Boolean connectives $\wedge$ and $\neg$ (i.e., conjunction and negation, respectively) and the *first-order quantifier* $\exists y$. Presburger arithmetic with counting quantifiers (PAC) extends PA with the *(unary) counting quantifier* $\exists^{=x} y$, where x and y are two syntactically distinct variables from X. Formulae of PAC are denoted by Φ, Ψ, Γ, etc.

We write $\mathrm{vars}(\Phi)$ and $\mathrm{fv}(\Phi)$ for the set of variables and free variables of Φ, respectively, with $\mathrm{fv}(\exists^{=x} y\, \Phi) \stackrel{\text{def}}{=} \{x\} \cup (\mathrm{fv}(\Phi) \setminus \{y\})$. A *sentence* is a formula Φ with $\mathrm{fv}(\Phi) = \emptyset$. We sometimes write $\Phi(x_1, \dots, x_k)$ or $\Phi(\boldsymbol{x})$, with $\boldsymbol{x} = (x_1, \dots, x_k)$ a tuple of variables, for a formula Φ with $\mathrm{fv}(\Phi) = \{x_1, \dots, x_k\}$. We say that Φ is *z-free* if $z \in X$ does not occur in Φ. Given terms t and t', $\Phi[t'/t]$ stands for the formula obtained from Φ by syntactically replacing every occurrence of t by t'. Given $\Phi(x_1, \dots, x_k)$ and terms $t_1, \dots, t_k$, $\Phi(t_1, \dots, t_k)$ stands for $\Phi[t_1/x_1] \dots [t_k/x_k]$.

Semantics. An *assignment* is a function $\nu \colon X \to \mathbb{Z}$ assigning an integer value to every variable. As usual, we extend ν in the standard way to a function that maps every term to an element of $\mathbb{Z}$. For instance, $\nu(x+3x+2) = \nu(x)+3\nu(y)+2$. Given a variable x and an integer n, we write $\nu[n/x]$ for the assignment obtained form ν by updating the value of x to n, i.e. $\nu[n/x](x) = n$, and for all variables y distinct from x, $\nu[n/x](y) = \nu(y)$. Given a formula Φ of PAC and an assignment ν, the satisfaction relation $\nu \models \Phi$ is defined as usual for linear inequalities, modulo constraints, Boolean connectives and the existential quantifier ranging over $\mathbb{Z}$. For the counting quantifier, we define

$$\nu \models \exists^{=x} y\, \Phi \text{ if and only if } \#\{n \in \mathbb{Z} \mid \nu[n/y] \models \Phi\} = \nu(x).$$

Informally, $\exists^{=x} y\, \Phi$ is satisfied by ν if there are exactly $\nu(x)$ distinct values for the variable y that make Φ true. A formula Φ of PAC is *satisfiable* (resp. *valid*) if $\nu \models \Phi$ holds for an assignment (resp. *every assignment*) ν. A formula Φ *entails* a formula Ψ, written $\Phi \models \Psi$, whenever every assignment satisfying Φ also satisfies Ψ. We write $\Phi \Leftrightarrow \Psi$ to denote that Φ and Ψ are *equivalent*, i.e. $\Phi \models \Psi$ and $\Psi \models \Phi$.

Syntactic abbreviations. We define $\bot \stackrel{\text{def}}{=} 0 < 0$ and $\top \stackrel{\text{def}}{=} \neg\bot$. The Boolean connectives $\vee$, $\rightarrow$ and $\leftrightarrow$ and the universal first-order quantifier $\forall$ are derived as usual, and so are the (in)equalities $<$, $\leq$, $=$, $\geq$, and $>$, between terms. For instance, $t_1 < t_2$ corresponds to $t_1 - t_2 < 0$, where we tacitly manipulate $t_1 - t_2$ with standard operations of linear arithmetic to obtain an equivalent term. Similarly, $t_1 \equiv_q t_2$ is short for $t_1 - t_2 \equiv_q 0$, whereas $|t_1| + t_2 < 0$ is short for

$(t_1 < 0 \to t_2 - t_1 < 0) \wedge (t_1 \geq 0 \to t_1 + t_2 < 0)$. For a variable $x \in X$ and $r \in [q]$, we call $x \equiv_q r$ a *simple* modulo constraint. All modulo constraints introduced by our quantifier elimination procedure given in Section 3 are simple.

The counting quantifier $\exists^{\geq x} y$. Historically [1,10], the quantifier $\exists^{=x} y$ has been the unary counting quantifier of choice when it comes to PAC. However, a priori one could define PAC as the extension of PA featuring counting quantifiers $\exists^{\geq x} y$, where $\nu \models \exists^{\geq x} y\, \Phi$ holds for an assignment ν whenever there are at least $\nu(x)$ values $n \in \mathbb{Z}$ for y such that $\nu[n/y] \models \Phi$. Notice that the counting quantifier $\exists^{=y}$ can be expressed using $\exists^{\geq y}$, and vice versa:

- $\exists^{=x} y\, \Phi \;\Leftrightarrow\; \exists^{\geq x} y\, \Phi \wedge \exists x' : x' = x + 1 \wedge \neg \exists^{\geq x'} y\, \Phi$; and
- $\exists^{\geq x} y\, \Phi \;\Leftrightarrow\; (\forall z\, \exists y : |z| \leq |y| \wedge \Phi) \vee \exists x' : x' \geq x \wedge \exists^{=x'} y\, \Phi$.

Two comments are in order: first, translating a PAC formula by swapping the type of counting quantifiers using the equivalences above has the unpleasant effect of increasing the size of the formula, exponentially if the nesting depth of quantifiers is unbounded. Second, the subformula $\forall z\, \exists y : |z| \leq |y| \wedge \Phi$ used in the last equivalence states that there are infinitely many values for y that make the formula Φ true. This formula highlights the main difference between $\exists^{=x} y$ and $\exists^{\geq x} y$ quantifiers: the latter is true in the presence of infinitely many values for y, whereas the former is false. Throughout the paper, we focus on the quantifier $\exists^{=x} y$, as done in [1,10], but use this observation to argue that our results can be readily adapted to the counting quantifier $\exists^{\geq x} y$. Full details of this adaptation are given in the full version of the paper.

Parameters of formulae. To analyze quantifier-elimination procedures, following [8,12], we introduce a number of parameters for formulae of PAC:

- $|\Phi|$ denotes the *length* of the formula Φ, i.e., the number of symbols to write down φ, with numbers encoded in binary. We always assume $|\Phi| \geq 2$;
- $\mathrm{qr}(\Phi)$ (resp. $\mathrm{nr}(\Phi)$) denotes the *quantifier* (resp. *negation*) *rank* of the formula Φ, i.e., the depth of nesting of the quantifiers (resp. negations) of Φ;
- $\mathrm{fd}(\Phi)$ denotes the overall *depth* of Φ, i.e., the depth of nesting of all constructors (i.e. $\wedge$, $\neg$, $\exists x$ and $\exists^{=x} y$) in the formula Φ;
- $\mathrm{lin}(\Phi)$ is the set containing the term 0 plus all the terms t that appear in linear inequalities $t < 0$ of Φ (recall that $t_1 < t_2$ is short for $t_1 - t_2 < 0$);
- $\mathrm{hom}(\Phi)$ is the set of homogeneous linear terms obtained from all terms in $\mathrm{lin}(\Phi)$ by setting their constants to 0;
- $\mathrm{const}(\Phi)$ is the set of all constants appearing in linear terms of $\mathrm{lin}(\Phi)$; and
- $\mathrm{mod}(\Phi)$ is the set of all moduli $q \in \mathbb{N}$ appearing in modulo constraints $t_1 \equiv_q t_2$ of Φ. We postulate $1 \in \mathrm{mod}(\Phi)$, even if Φ has no modulo constraints.

Given a vector $\boldsymbol{v} = (v_1, \ldots, v_d) \in \mathbb{Z}^d$, we write $\|\boldsymbol{v}\| = \max\{|v_i| : 1 \leq i \leq d\}$ for the *infinity norm* of $\boldsymbol{v}$. Similarly, for a linear term t, we write $\|t\|$ for the maximum absolute value of a coefficient or constant appearing in t. Given a finite set of vectors or a finite set of terms A, we define $\|A\| = \max\{\|a\| : a \in A\}$. Given a matrix $A \in \mathbb{Z}^{n \times d}$, its infinity norm is the maximal infinity norm of its column vectors. Notice that $\|\mathrm{lin}(\Phi)\| = \|\mathrm{hom}(\Phi) \cup \mathrm{const}(\Phi)\|$. For a formula Φ, we define $\|\Phi\| \stackrel{\text{def}}{=} \|\mathrm{lin}(\Phi) \cup \mathrm{mod}(\Phi)\|$.

Complexity remarks. The proposition below characterizes the complexity of PA.

Proposition 1 ([3]). *Presburger arithmetic is* $\mathrm{STA}(*, 2^{2^{n^{\mathcal{O}(1)}}}, \mathcal{O}(n))$*-complete.*

To be more precise, the number of alternations required to decide the validity or satisfiability of a formula Φ from Presburger arithmetic is linear in $\mathrm{nr}(\Phi)$. Notice that $2\textsc{NExpTime} \subseteq \mathrm{STA}(*, 2^{2^{n^{\mathcal{O}(1)}}}, \mathcal{O}(n)) \subseteq 2\textsc{ExpSpace}$.

3 A quantifier elimination procedure for PAC

In this section, we develop a new quantifier elimination procedure (QE procedure) for the counting quantifier $\exists^{=x} y$:

Proposition 2. *Let Φ be quantifier-free. Then $\exists^{=x} y\, \Phi$ is equivalent to a Boolean combination of linear inequalities and simple modulo constraints.*

We quantify the growth of parameters in the formula in Section 4. Upper bounds on this growth are at the core of our results. Without any bounds (as stated), Proposition 2 is known and can be obtained by chaining the quantifier elimination procedure developed by Schweikardt [10] together with the standard quantifier elimination procedure for Presburger arithmetic. An advantage of our QE procedure for the quantifier $\exists^{=x} y$ is that it avoids the introduction of additional $\exists$- and $\forall$-quantifiers when eliminating a counting quantifier on which Schweikardt's procedure relies. More precisely, given a formula $\exists^{=x} y\, \Phi$ where Φ is quantifier-free (q.f. in short), the QE procedure in [10] requires a full transformation of Φ into disjunctive normal form, and eliminates the quantifier $\exists^{=x} y$ by introducing first-order quantifiers, producing an equivalent formula Ψ of Presburger arithmetic. This strategy comes at a cost: the size of the q.f. formula obtained after removing the quantifiers from Ψ is doubly exponential in the size of $\exists^{=x} y\, \Phi$. By avoiding the introduction of first-order quantifiers, our QE procedure already exponentially improves upon Schweikardt's procedure.

Our QE procedure performs a series of formula manipulations, divided into five steps. At the end of the i-th step, the procedure produces a formula Φ_i equivalent to the original formula $\exists^{=x} y\, \Phi$. Ultimately, Φ_5 is a Boolean combination of inequalities and simple modulo constraints allowing us to establish Proposition 2. In this section, we present the procedure and briefly discuss its correctness, leaving the computational analysis of parameters $\mathrm{lin}(\Phi_5)$, $\mathrm{hom}(\Phi_5)$, $\mathrm{const}(\Phi_5)$ and $\mathrm{mod}(\Phi_5)$ to subsequent sections.

Step I: Normalise the coefficients of y. Given the input formula $\Phi_0 = \exists^{=x} y\, \Phi$, with Φ q.f., the first step of the procedure is a standard step for QE procedures for Presburger arithmetic. It produces an equivalent formula Φ_1 in which all non-zero coefficients of y appearing in a linear term are normalized to 1 or -1. For simplicity, we first translate every modulo constraint in Φ into simple modulo constraints, by relying on the lemma below.

Lemma 1. *Every constraint* $t \equiv_q 0$ *is equivalent to a Boolean combination* Ψ *of simple modulo constraints such that* $\mathrm{vars}(\Psi) \subseteq \mathrm{vars}(t \equiv_q 0)$ *and* $\mathrm{mod}(\Psi) = \{q\}$.

The first step of our QE procedure is as follows:

1 Translate every modulo constraint in Φ into simple modulo constraints (Lemma 1).
2 Let k be the lcm of the absolute values of all coefficients of y appearing in $\hom(\Phi)$.
3 Let Φ' be the formula obtained from Φ by applying the following three rewrite rules to each linear inequality and simple modulo constraint in which y appears:

- $ay + t < 0 \longrightarrow ky + (k/a) \cdot t < 0$, if $a > 0$,
- $ay + t < 0 \longrightarrow -ky - (k/a) \cdot t < 0$, if $a < 0$, and
- $y \equiv_q r \longrightarrow ky \equiv_{kq} kr$,

where t is a term, $q \geq 1$ and $r \in [q]$:

4 Define $\Phi_1 \stackrel{\text{def}}{=} \exists^{=x} y\, (y \equiv_k 0 \wedge \Phi'[y/ky])$.

Claim 1. $\Phi_0 \Leftrightarrow \Phi_1$, and in Φ_1, all non-zero coefficients of y are either 1 or -1.

Step II: Subdivide the formula according to term orderings and residue classes. We define an *ordering for a set of linear terms* T to be a formula of the form

$$(t_1 \lhd_1 t_2) \wedge (t_2 \lhd_2 t_3) \wedge \cdots \wedge (t_{n-1} \lhd_{n-1} t_n), \tag{1}$$

where $\{t_1, \ldots, t_n\} = T$ and $\{\lhd_1, \ldots, \lhd_{n-1}\} \subseteq \{<, =\}$.

Lemma 2. *There is an algorithm that, given a set* T *of* n *linear terms over* d *variables, computes in time* $n^{\mathcal{O}(d)} \log \|T\|^{\mathcal{O}(1)}$ *a set* $\{O_1, \ldots, O_o\}$ *of orderings for* T *s.t. (1)* $o = \mathcal{O}(n^{2d})$*, (2)* $\top \Leftrightarrow \bigvee_{i=1}^{o} O_i$*, (3)* $\bot \Leftrightarrow O_i \wedge O_j$ *whenever* $i \neq j$.

Lemma 2 is proven analogously to [13, Proposition 5.1].

The second step of our QE procedure is as follows:

5 Let T be the set of all y-free terms t such that t, $y - t$ or $-y + t$ belongs to $\mathrm{lin}(\Phi_1)$.
6 Using Lemma 2, build a set $\{O_1, \ldots, O_o\}$ of orderings for the terms T.
7 Let $Z \stackrel{\text{def}}{=} \mathrm{vars}(\Phi)$ and $m \stackrel{\text{def}}{=} \mathrm{lcm}(\mathrm{mod}(\Phi_1))$.
8 For every $i \in [1, o]$ and every $r\colon Z \to [m]$, let $\Gamma_{i,r} \stackrel{\text{def}}{=} O_i \wedge (\bigwedge_{z \in Z} z \equiv_m r(z))$.
9 Define $\Phi_2 \stackrel{\text{def}}{=} \bigvee_{i=1}^{o} \bigvee_{r\colon Z \to [m]} (\Gamma_{i,r} \wedge \Phi_1)$.

Claim 2. $\Phi_1 \Leftrightarrow \Phi_2$.

In Steps III to V of the procedure, we focus on each disjunct of Φ_2 separately, iterating over all $i \in [1, o]$, hence over all orderings, and all $r\colon Z \to [m]$, i.e., functions assigning residue classes modulo m to the variables in Z.

Step III: Split the range of y into segments. Recall that $\Phi_1 = \exists^{=x} y\, \Psi$, where Ψ is some Boolean combination of inequalities and modulo constraints with variables from $\mathrm{vars}(\Phi)$ in which the non-zero coefficients of y are either 1 or -1. Let $T|_{O_i} \stackrel{\text{def}}{=} (t'_1, \cdots, t'_\ell)$ be the tuple of all the terms in T that the formula O_i asserts pairwise non-equal, taken in the ascending order. In other words, we obtain $t'_1, \ldots, t'_\ell$ by removing from the sequence $t_1, \ldots, t_n$ in Equation (1) all terms t_{j+1} for which $\lhd_j$ is $=$. Let $seg(y, O_i)$ be the set of formulae

$$\big\{\, y < t'_1,\ \ y = t'_1,\ \ (t'_{i-1} < y \wedge y < t'_i),\ \ y = t'_i,\ \ t'_\ell < y \ :\ i \in [2,\ell] \big\}.$$

We have $\#\mathit{seg}(y, O_i) = 2\ell + 1$. Given $\kappa \in \mathit{seg}(y, O_i)$, the formula $O_i \wedge \kappa$ imparts a linear ordering on the terms $T \cup \{y\}$. This enables us to "almost evaluate" Ψ:

Lemma 3. *For every $\kappa \in \mathit{seg}(y, O_i)$, there is a Boolean combination $\Psi^{i,r}_\kappa$ of simple modulo constraints such that $\mathrm{vars}(\Psi^{i,r}_\kappa) = \{y\}$, $\mathrm{mod}(\Psi^{i,r}_\kappa) \subseteq \mathrm{mod}(\Psi)$ and*

$$\Gamma_{i,r} \wedge \kappa \wedge \Psi \ \ \Leftrightarrow\ \ \Gamma_{i,r} \wedge \kappa \wedge \Psi^{i,r}_\kappa.$$

Our QE procedure manipulates Φ_2 as follows:

For every $i \in [1, o]$ and every $r\colon Z \to [m]$:
  Let $\mathit{seg}(y, O_i) = \{\kappa_0, \ldots, \kappa_{2\ell}\}$.
  For every $j \in [0, 2\ell]$, consider the formula $\Psi^{i,r}_{\kappa_j}$ from Lemma 3.
  Let $\Phi^{i,r}_3 = \exists x_0 \ldots \exists x_{2\ell} \big(x = \sum_{j=0}^{2\ell} x_j \ \wedge\ \bigwedge_{j=0}^{2\ell} \exists^{=x_j} y (\kappa_j \wedge \Psi^{i,r}_{\kappa_j})\big)$.
Define $\Phi_3 \stackrel{\text{def}}{=} \bigvee_{i=1}^{o} \bigvee_{r\colon Z \to [m]} (\Gamma_{i,r} \wedge \Phi^{i,r}_3)$.

Claim 3. $\Phi_2 \Leftrightarrow \Phi_3$.

Step IV: Compute the number of solutions for each segment. We next aim at eliminating the counting quantifiers introduced in Step III in the sub-formulae $\exists^{=x_j} y (\kappa_j \wedge \Psi^{i,r}_{\kappa_j})$. We go over each $\kappa \in \mathit{seg}(y, O_i)$, and consider three cases depending on whether it specifies (syntactically) an infinite interval, a finite segment, or a single value for y.

Notice that r is in fact an assignment to variables, so $r(t) \in \mathbb{Z}$ is well-defined for every term t with free variables Z. For all $i \in [1, o]$ and $r\colon Z \to [m]$, given $T|_{O_i} = (t'_1, \ldots, t'_\ell)$ the procedure computes the following numbers $c_1, \ldots, c_\ell$, $p_2, \ldots, p_\ell$ and $r_2, \ldots, r_\ell$.

For every $j \in [1, \ell]$:
  If $\Psi^{i,r}_\kappa[r(t'_j)/y]$ is true, where $\kappa = (y = t'_j)$, then let $c_j \stackrel{\text{def}}{=} 1$, else let $c_j \stackrel{\text{def}}{=} 0$.
For every $j \in [2, \ell]$:
  Let $p_j \in [0, m]$ be the number of $y \in [m]$ satisfying $\Psi^{i,r}_\kappa(y)$.
  Let $\underline{u}_j = (r(t'_{j-1}) \bmod m)$.
  Let $\overline{u}_j$ be the smallest integer congruent to $r(t'_j)$ modulo m and greater than $\underline{u}_j$.
  Let $r'_j \in [0, m]$ be the number of $y \in [\underline{u}_j + 1, \overline{u}_j - 1]$ satisfying $\Psi^{i,r}_\kappa(y)$.
  Let $r_j \in [-m^2, m^2]$ be such that $r_j = -p_j \cdot (\overline{u}_j - \underline{u}_j) + m \cdot r'_j$.

Lemma 4. *Given a formula $\Psi^{i,r}_\kappa$ and $m, \underline{u}_j, \overline{u}_j$, the numbers p_j and r'_j can be computed in #P, or by a deterministic algorithm with running time $\mathcal{O}(m \cdot |\Psi^{i,r}_\kappa|)$.*

The numbers c_j, p_j, r_j determine, for each $\kappa \in \mathit{seg}(y, O_i)$, how many assignments to the variable y satisfy the formula $\Psi^{i,r}_\kappa$ in the conjunction $\Gamma_{i,r} \wedge \kappa \wedge \Psi^{i,r}_\kappa$. Intuitively, this is c_j for κ of the form $y = t'_j$, and $(p_j(t'_j - t'_{j-1}) + r_j)/m$ for κ of the form $t'_{j-1} < y \wedge y < t'_j$. We say "intuitively" here, because in the latter case the expression above depends on other variables so is not, strictly speaking, a number. The following claims formalize this intuition:

Claim 4. Let $\kappa \in \{y < t'_1,\ t'_\ell < y\}$. If $\Psi^{i,r}_\kappa(y)$ is satisfiable, then $\Phi^{i,r}_3 \Leftrightarrow \bot$.

Claim 5. Let $j \in [1,\ell]$, $\kappa = (y = t'_j)$, $z \in X$. Then, $\exists^{=z} y\,(\kappa \wedge \Psi^{i,r}_\kappa) \ \Leftrightarrow\ z = c_j$.

Claim 6. Let $\kappa = (t'_{j-1} < y \wedge y < t'_j)$ for some $j \in [2,\ell]$ and let z be a fresh variable. Then, $\Gamma_{i,r} \wedge \exists^{=z} y\,(\kappa \wedge \Psi^{i,r}_\kappa) \ \Leftrightarrow\ \Gamma_{i,r} \wedge mz = p_j(t'_j - t'_{j-1}) + r_j$.

The procedure manipulates the formula Φ_3 as follows:

23 For every $i \in [1,o]$ and every $r\colon Z \to [m]$:
24 If $\Psi^{i,r}_\kappa(y)$ is satisfiable for some $\kappa \in \{y < t'_1,\ t'_\ell < y\}$, then let $\Phi^{i,r}_4 \stackrel{\text{def}}{=} \bot$,
25 else $\Phi^{i,r}_4 \stackrel{\text{def}}{=} \exists x_2 \ldots \exists x_\ell \big(x = \sum_{j=2}^{\ell} x_j + \sum_{j=1}^{\ell} c_j \wedge \bigwedge_{j=2}^{\ell} mx_j = p_j(t'_j - t'_{j-1}) + r_j\big)$.
26 Define $\Phi_4 \stackrel{\text{def}}{=} \bigvee_{i=1}^{o} \bigvee_{r\colon Z\to[m]} (\Gamma_{i,r} \wedge \Phi^{i,r}_4)$.

Claim 7. $\Phi_3 \Leftrightarrow \Phi_4$.

Step V: Sum up the solutions. It remains to get rid of the variables x_i introduced earlier. For each disjunct $\Gamma_{i,r} \wedge \Phi^{i,r}_4$ of Φ_4, we use the notation from Step IV.

27 For every $i \in [1,o]$ and every $r\colon Z \to [m]$:
28 If $\Phi^{i,r}_4 = \bot$, then let $\Phi^{i,r}_5 \stackrel{\text{def}}{=} \bot$,
29 else let $\Phi^{i,r}_5 \stackrel{\text{def}}{=} mx = \sum_{j=2}^{\ell}(p_j(t'_j - t'_{j-1}) + r_j) + m \cdot \sum_{j=1}^{\ell} c_j$.
30 Let $\Phi_5 \stackrel{\text{def}}{=} \bigvee_{i=1}^{o} \bigvee_{r\colon Z\to[m]} (\Gamma_{i,r} \wedge \Phi^{i,r}_5)$.

The procedure outputs Φ_5. The following claim implies Proposition 2.

Claim 8. $\Phi_4 \Leftrightarrow \Phi_5$. The formula Φ_5 is quantifier-free.

4 Discussion, summary of results and roadmap

The QE procedure for a single counting quantifier $\exists^{=x} y$ from Section 3 forms the basis of our results. In this section we discuss its use and lay out its applications.

Analysis of the procedure. The next lemma establishes the growth of the formulae and their parameters in our quantifier elimination procedure.

Lemma 5. *Let Φ_5 be obtained from applying the QE procedure of Section 3 to a formula $\exists^{=y} x\,\Phi$, where Φ is quantifier-free and $\#\mathrm{vars}(\Phi) = d$. Then:*

$$\mathrm{mod}(\Phi_5) = \{m\} \quad \textit{with } m = k \cdot \mathrm{lcm}(\mathrm{mod}(\Phi)) \textit{ and } k \le \|\mathrm{hom}(\Phi)\|^{\#\mathrm{hom}(\Phi)},$$
$$\#\mathrm{lin}(\Phi_5) \le N^{O(d)}, \qquad \|\mathrm{lin}(\Phi_5)\| \le \mathcal{O}(N) \cdot \|\mathrm{lin}(\Phi)\|,$$
$$\#\mathrm{hom}(\Phi_5) \le N^{O(d)}, \quad \|\mathrm{hom}(\Phi_5)\| \le \mathcal{O}(N) \cdot \|\mathrm{hom}(\Phi)\|, \quad \textit{with } N = m^2 \cdot \#\mathrm{lin}(\Phi).$$

Remark 1. With minor changes to our procedure, one can obtain a QE procedure for the quantifier $\exists^{\ge x} y$. In particular, since $\exists^{\ge x} y\,\Phi$ is true if there are infinitely many values for y that satisfy Φ, Claim 4 needs to be updated so that $\Phi^{i,r}_3 \Leftrightarrow \top$ is deduced, instead of $\Phi^{i,r}_3 \Leftrightarrow \bot$. Other minor adaptations are required, e.g. equalities "$x = \ldots$" and counting quantifiers $\exists^{=x_j} y$ appearing in Line 13 must be updated to "$x \le \ldots$" and $\exists^{\ge x_j} y$. The resulting QE procedure for $\exists^{\ge x} y$ still adheres to the bounds in Lemma 5.

A consequence of Lemma 5 is that our QE procedure gives an algorithm for deciding a formula Φ from PAC featuring multiple counting quantifiers $\exists^{=x} y$ in time $2^{\cdot^{\cdot^{\cdot^{2}}}}$, where the height of the tower is linear in the quantifier rank of Φ. Indeed, in view of the upper bounds and equations given by Lemma 5 for $\#\hom(\Phi_5)$, N, m, and k, we observe that the upper bound for $\#\hom(\Phi_5)$ is exponential in $\#\hom(\Phi)$. This means that more fine-grained bounds are necessary for decision procedures with elementary complexity, i.e., with a running time bounded from above by a k-fold exponential in the size of the input formula.

Elementary decision procedures. In view of this growth of the parameters, it is natural to ask ourselves whether our QE procedure is perhaps naïvely disregarding important properties of the underlying arithmetic theory that could lead to better bounds. A good test in this direction is to check whether improved bounds can be achieved when the procedure runs on restricted forms of counting quantifiers. In the remainder of the paper we show that this is the case, and explain how the growth of parameters can be countered for restricted quantifiers, obtaining 3ExpTime quantifier elimination procedures as well as 2ExpSpace decision procedures for extensions of PA with a variety of counting quantifiers.

As an example, let us consider Presburger arithmetic enriched with threshold quantifiers $\exists^{\geq c} y\, \Phi$, where $c \in \mathbb{N}$ is written in binary. These are satisfied whenever there are at least c distinct values for the variable y that make the formula Φ true. Notice that the threshold counting quantifiers $\exists^{\geq c} y$ are a syntactic generalization of the first-order quantifiers, as $\exists^{\geq 1} y\, \Phi \Leftrightarrow \exists y\, \Phi$. Interestingly enough, one can translate threshold quantifiers into standard Presburger arithmetic with just a polynomial increase in the size of the formula. For simplicity, assume that the threshold c is a power of 2. Then, the quantifier $\exists^{\geq c} y$ can be internalized in PA by relying on the equivalence

$$\exists^{\geq 2g} y\, \Phi(y, \boldsymbol{z}) \Leftrightarrow \exists u\, \forall v\, \exists^{\geq g} y : (v = 0 \leftrightarrow y < u) \wedge \Phi(y, \boldsymbol{z})$$

as well as $\exists^{\geq 1} y\, \Phi \Leftrightarrow \exists y\, \Phi$. However, in terms of decision procedures, this is an inadequate solution, as it comes at the cost of introducing $2 \log_2 c$ many quantifier alternations. Building upon the QE procedure from Section 3, we show how to directly eliminate threshold quantifiers. This proves that the increase in alternation depth that depends on the threshold c is unnecessary.

Theorem 1. *The validity of a formula Φ from Presburger arithmetic with threshold counting quantifiers can be decided in* $\mathrm{STA}(*, 2^{2^{|\Phi|^{\mathcal{O}(1)}}}, \mathcal{O}(\mathrm{fd}(\Phi)))$.

This result matches the complexity of deciding standard PA in the case of unbounded alternation depth. Thus, PA can be enriched with threshold quantifiers with almost no computational overhead. Note that a slight increase in number of alternations is still required, and goes from $\mathcal{O}(\mathrm{nr}(\Phi))$ for PA to $\mathcal{O}(\mathrm{fd}(\Phi))$ for PA with threshold counting quantifiers.

We further strengthen Theorem 1, extending it to the case where the threshold c is encoded even more succinctly, as the unique solution of a PA formula

$\Phi(x)$ as long as this solution is bounded doubly-exponentially in $|\Phi|$. An example of such a formula is $\Phi(x) = \exists z : z = 1 \wedge \Psi_n(x, z)$, where

$$\Psi_0(x, z) \stackrel{\text{def}}{=} x = 2z,$$
$$\Psi_{n+1}(x, z) \stackrel{\text{def}}{=} \exists y \forall a \forall b : (a = x \wedge b = y) \vee (a = y \wedge b = z) \rightarrow \Psi_n(a, b),$$

and the only solution is given by $x = 2^{2^n}$ [7, Lecture 23], whilst $|\Phi| = \mathcal{O}(n)$. The crux of our results lies in the identification of a fragment of PAC that we call *monadically-guarded*, for which the following theorem can be established.

Theorem 2. *Monadically-guarded PAC is decidable in* 2ExpSpace.

In the next section, we introduce the monadically-guarded fragment of PAC and discuss extensions of PA that can be captured by this fragment. In Section 6, by adding post-processing to the procedure from Section 3, we show how to deal with any monadically-guarded counting quantifiers in 3ExpTime. In Section 7 we establish Theorem 2 by designing a quantifier relativization argument, continuing the direction of research due to [12]. In Section 8 we prove Theorem 1.

5 The monadically-guarded fragment of PAC

Fix a logic $\mathcal{L}$. A formula $\Phi(x, \boldsymbol{z})$ from $\mathcal{L}$, where $\boldsymbol{z}$ is a tuple of variables not including x, is said to be *monadically decomposable* on the variable x whenever

$$\Phi \Leftrightarrow \Psi, \text{ for some } \Psi \stackrel{\text{def}}{=} \bigvee_{i \in I} (\Delta_i(x) \wedge \Gamma_i(\boldsymbol{z})),$$

where Δ_i and Γ_i are formulae from $\mathcal{L}$. In this case, Ψ is said to be a *monadic decomposition of* Φ on the variable x.

The notion of monadic decomposition has been put forward by Veanes et al. in [11], as a general simplification technique that improves the performance of solvers. Here, our interest lies in studying whether the notion of monadic decomposability can bring complexity advantages for Presburger arithmetic with counting quantifiers. With this in mind, we consider formulae of PAC that we call *monadically-guarded*: those in which the quantifiers $\exists^{=x} y$ only appear in subformulae of the form $\exists x \, (\Psi \wedge \exists^{=x} y \, \Phi)$, where Φ and Ψ are themselves from the monadically-guarded fragment of PAC, x does not occur in Φ, and Ψ is monadically decomposable on the variable x. The *monadically-guarded fragment* of PAC is understood as the set of all formulae from PAC that are monadically-guarded. This fragment captures several interesting extensions of PA:

– It can express that the number of different y satisfying $\Phi(y, \boldsymbol{z})$ lies in an arithmetic progression b, $b + p$, $b + 2 \cdot p$, $b + i \cdot p$, ..., with $b, p \in \mathbb{N}$. That is,

$$\exists x (x \geq b \wedge x \equiv_p b \wedge \exists^{=x} y \, \Phi(y, \boldsymbol{z})).$$

This type of monadically-guarded formulae extends the *modulo counting* quantifiers studied by Habermehl and Kuske [4]. Modulo counting quantifiers are written as $\exists^{(r,q)} y \, \Phi$ and hold whenever the number of different y satisfying Φ is congruent to r modulo q. Hence, $\exists^{(r,q)} y \, \Phi \Leftrightarrow \exists x \, (x \equiv_p r \wedge \exists^{=x} y \Phi)$.

Moreover, in the monadically-guarded fragment, we can replace the integer r with an arbitrary linear term t with variables from $\boldsymbol{z}$, since the modulo constraint $x \equiv_p t$ can be monadically decomposed into $\bigvee_{r\in[p]}(x \equiv_p r \wedge t \equiv_p r)$.

– As we recalled in the previous section with the formula $\Psi_n(x, z)$, it is known that PA allows one to succinctly encode numbers that are doubly or triply exponentially large with respect to the size of the formula. For instance, one can define a formula $L_n(x)$, again of size polynomial in n, that is true whenever x is the product of all primes in the interval $[2, 2^{2^n}]$ (see [7, Lecture 24]). In this case, $x \geq 2^{c2^{2^n}}$ for some fixed $c > 0$. The monadically-guarded fragment of PAC allows one to use these succinct representations as guards of counting quantifiers. For instance, $\exists x(L_n(x) \wedge \exists^{=x} y\, \Psi(y, \boldsymbol{z}))$ is true whenever the number of y satisfying $\Psi(y, \boldsymbol{z})$ is the product of all primes in $[2, 2^{2^n}]$.

Hague et al. [5] proved that constructing the monadic decomposition of a quantifier-free formula can be done in exponential time. More precisely, given a q.f. formula $\Phi(x, \boldsymbol{y})$ from PA that is monadically decomposable on x, in [5] it is shown that there is a natural number B of magnitude exponential in $|\Phi|$ that makes the following formula $\Psi_B(x, \boldsymbol{y})$ a monadic decomposition of Φ on x:

$$\Psi_B \stackrel{\text{def}}{=} \bigvee_{c=0}^{m-1} \big((x \geq B \wedge x \equiv_m c \wedge \Phi(B + c, \boldsymbol{y})) \vee (x \leq -B \wedge x \equiv_m c \wedge \Phi(-B - c, \boldsymbol{y}))\big) \\ \vee \bigvee_{c=-B+1}^{B-1} (x = c \wedge \Phi(c, \boldsymbol{y})),$$

where $m = \mathrm{lcm}(\mathrm{mod}(\Phi))$. We study the arguments presented in [5] and refine the bound B, tracking dependencies on several formula parameters separately. We find that B is polynomial in $\|\Phi\|$; it is only exponential in $\#\mathrm{mod}(\Phi)$ and in the number of variables of the tuple $\boldsymbol{y}$.

Proposition 3. *Let $\Phi(x, \boldsymbol{y})$ be a q.f. formula from PA, where $\boldsymbol{y} = (y_1, \ldots, y_d)$. Let $m = \mathrm{lcm}(\mathrm{mod}(\Phi))$ and $B = 2^{48d^2}(m \cdot \|\mathrm{lin}(\Phi)\|)^{6d} + 1$. If Φ is monadically decomposable on x, then the formula Ψ_B is such a decomposition.*

Together with our QE procedure, Proposition 3 shows that it is decidable to check whether a formula of PAC is monadically decomposable (on a certain variable). Due to Theorem 2, this problem is in 2ExpSpace for formulae of the monadically-guarded fragment of PAC. Besides, notice that all formulae having one free variable are monadic decompositions of themselves.

Our QE procedure for the monadically-guarded fragment of PAC, outlined below, makes use of the sharper bound obtained in Proposition 3.

6 Eliminating monadically-guarded counting quantifiers

Consider a formula $\Phi_0 = \exists x(\Psi \wedge \exists^{=x} y\, \Phi)$, where Φ and Ψ are quantifier-free formulae, x does not occur in Φ, and Ψ is monadically decomposable on x. By relying on the QE procedure introduced in Section 3, we show how to obtain a quantifier-free formula equivalent to Φ_0. W.l.o.g., we assume that all free variables distinct from x and y and occurring in Φ and Ψ come from the tuple of variables $\boldsymbol{z}$.

Below, let $\Psi' = \bigvee_{k\in K} \Delta_k(x) \wedge \Psi_k(\boldsymbol{z})$ be the monadic decomposition of Ψ on the variable x computed according to Proposition 3. Recall that this means that each Δ_k is a formula having one among the following three forms:

$$x \geq B \wedge x \equiv_q c; \qquad x \leq -B \wedge x \equiv_q c; \text{ or } \qquad x = r,$$

where $q \stackrel{\text{def}}{=} \text{lcm}(\text{mod}(\Psi))$, $c \in [q]$, $r \in [-B+1, B-1]$ and B is a fixed natural number. Let us also consider the formula Φ_5 obtained from performing the QE procedure for the $\exists^{=x} y$ counting quantifier on $\exists^{=x} y\, \Phi$, so that $\Phi_0 \Leftrightarrow \exists x(\Psi' \wedge \Phi_5)$. In particular, recall that $\Phi_5 \stackrel{\text{def}}{=} \bigvee_{i=1}^{o} \bigvee_{r\colon Z\to[m]} (\Gamma_{i,r} \wedge \Phi_5^{i,r})$, where Z is the set of variables appearing in $\boldsymbol{z}$, $m = \text{lcm}(\text{mod}(\Phi))$ and $\Gamma_{i,r} = O_i \wedge (\bigwedge_{w\in Z} w \equiv_m r(w))$ is a conjunction of an ordering O_i and simple modulo constraints with variables from Z. Hence, $\Gamma_{i,r}$ is x-free. Moreover, $\Phi_5^{i,r}$ is either $\bot$ or a formula of the form

$$mx = \textstyle\sum_{j=2}^{\ell} (p_j(t'_j - t'_{j-1}) + r_j) + m \cdot \sum_{j=1}^{\ell} c_j. \tag{2}$$

where the terms $t'_1, \ldots, t'_\ell$ are from T (where T is defined as in Step II of Section 3), and hence x-free. Therefore, the following property holds.

Claim 9. In Φ_5, x only appears on the left-hand side of equalities of the form (2).

This inconspicuous claim, together with the shape of Δ_k, is at the heart of our QE procedure eliminating x from the formula $\exists x(\Psi' \wedge \Phi_5)$. Indeed, after distributing the existential quantifier $\exists x$ and all conjunctions over disjunctions of $\Psi' \wedge \Phi_5$, we end up with a disjunction of formulae of the form $\exists x : \Delta_k(x) \wedge \Psi_k(\boldsymbol{z}) \wedge \Gamma_{i,r} \wedge \Phi_5^{i,r}$, and let us consider one such disjunct with $\Delta_k(x) = (x \geq B \wedge x \equiv_q c)$ and $\Phi_5^{i,r}$ as in Equation (2). The variable x can be eliminated with a simple substitution, rewriting $\Delta_k(x) \wedge \Phi_5^{i,r}$ as the new formula $\widetilde{t} \geq m \cdot B \wedge \widetilde{t} \equiv_{m\cdot q} m \cdot c$, where $\widetilde{t}$ is the right-hand side of Equation (2). The correctness of this rewrite step follows simply from the equivalences $x \geq B \Leftrightarrow m \cdot x \geq m \cdot B$ and $x \equiv_q c \Leftrightarrow m \cdot x \equiv_{m\cdot q} m \cdot c$, with $m \geq 1$. In a similar way, we can treat all possible cases for the different forms of $\Delta_k(x)$ and $\Phi_5^{i,r}$. We obtain a formula

$$\Psi_k(\boldsymbol{z}) \wedge \Gamma_{i,r} \wedge \widetilde{t} \geq m \cdot B \wedge \widetilde{t} \equiv_{m\cdot q} m \cdot c. \tag{3}$$

The number of homogeneous terms across all such disjuncts is still prohibitive as it was in Φ_5. Now comes the key simplification step; we deal with the inequality $\widetilde{t} \geq m \cdot B$ and with the modulo constraint $\widetilde{t} \equiv_{m\cdot q} m \cdot c$.

Consider the former first. By definition, all the coefficients p_j of Equation (2) are non-negative, and thanks to the ordering O_i appearing in $\Gamma_{i,r}$, in every valuation ν satisfying the formula in Equation (3) we have $\nu(t'_j - t'_{j-1}) \geq 0$. Therefore, the inequality $\widetilde{t} \geq m \cdot B$ can be translated into a formula of the form $\bigvee_{g\in G} \bigwedge_{j=2}^{\ell} t'_j - t'_{j-1} \geq d_{g,j}$, where each $d_{g,j}$ is non-negative and, for every $g \in G$, the sum $\sum_{j=2}^{\ell} p_j d_{g,j}$ is at least $e \stackrel{\text{def}}{=} m(B - \sum_{j=1}^{\ell} c_j) - \sum_{j=2}^{\ell} r_j$. To compute this formula efficiently, we appeal to Lemma 2, with respect to the set of terms $\{t'_j - t'_{j-1} \mid j \in [2, \ell]\} \cup [0, e]$.

Lemma 6. *Let* $d = |\mathrm{fv}(O_i \wedge \widetilde{t} \geq m \cdot B)|$. *In time* $(e+\ell)^{\mathcal{O}(d)} \log(B \cdot \|O_i\|)^{\mathcal{O}(1)}$ *one can compute a formula* $\Theta = \bigvee_{g \in G} \bigwedge_{j=2}^{\ell} t'_j - t'_{j-1} \geq d_{g,j}$ *s.t. (1)* $d_{g,j} \in [0, e+1]$, *(2)* $\#G \leq \mathcal{O}((e+\ell)^{2d})$, *and (3)* $O_i \wedge \widetilde{t} \geq m \cdot B \;\Leftrightarrow\; O_i \wedge \Theta$.

A similar simplification can be done for the modulo constraint $\widetilde{t} \equiv_{m \cdot q} m \cdot c$: we guess residue classes of variables in $\widetilde{t}$ modulo $m \cdot q$, rewriting $\widetilde{t} \equiv_{m \cdot q} m \cdot c$ into $\bigvee_{s\colon Z \to [m \cdot q]} (\widetilde{t} \equiv_{m \cdot q} m \cdot c \wedge \bigwedge_{z \in Z} z \equiv_{m \cdot q} s(z))$ and then replace, in each disjunct, $\widetilde{t} \equiv_{m \cdot q} m \cdot c$ by $\top$ or $\bot$, according to the satisfaction of $s(\widetilde{t}) \equiv_{m \cdot q} m \cdot c$.

The steps just discussed forms the post-processing phase of our QE procedure for the monadically-guarded fragment of PAC. Thanks to Lemma 6, we can show that the set of homogeneous terms of the resulting quantifier free formula Φ', equivalent to Φ_0, is the set of homogeneous terms in the monadic decomposition Ψ', together with terms of the form $t - t'$ with t and t' belong to the set T defined in Line 5. But $\#\mathrm{hom}(\Psi') = \mathcal{O}(\#\mathrm{hom}(\Phi_0))$, and thus:

Lemma 7. $\#\mathrm{hom}(\Phi') \leq \mathcal{O}(\#\mathrm{hom}(\Phi_0)^2)$.

Running time. Lemma 7 is the key to obtaining an elementary QE procedure. In particular, this improvement over the exponential dependence of $\#\mathrm{hom}(\Phi_5)$ on $\#\mathrm{hom}(\Phi)$ from our "baseline" Lemma 5 leads to the following bounds on the elimination of an arbitrary number of monadically-guarded quantifiers.

Lemma 8. *Let* Ω *be a formula from the monadically-guarded fragment of PAC, with quantifier rank* d. *There is an equivalent quantifier-free formula* Υ *such that*

- $\#\mathrm{hom}(\Upsilon) \leq |\Omega|^{2^{\mathcal{O}(d)}}$ *and* $\#\mathrm{mod}(\Upsilon) \leq \mathcal{O}(|\Omega|)$;
- $\#\mathrm{lin}(\Upsilon), \|\mathrm{const}(\Upsilon)\|, \|\mathrm{hom}(\Upsilon)\|$ *and* $\|\mathrm{mod}(\Upsilon)\|$ *are at most* $2^{|\Omega|^{2^{\mathcal{O}(d)}}}$.

Proof idea. In a nutshell, the bounds of Lemma 8 are obtained by first iterating Lemma 7 across all quantifier elimination rounds. This results in the doubly exponential bound $|\Omega|^{2^{\mathcal{O}(d)}}$ on the cardinality of the set of homogeneous terms throughout the entire procedure. With this bound in hand, exponentiation on the right-hand side of the inequalities of Section 3 does not blow the parameters above triple exponential. □

Subsequent analysis leads to the following result.

Theorem 3. *There is a* 3ExpTime *quantifier elimination procedure for the monadically-guarded fragment of PAC.*

Theorem 3 follows by combining Lemma 8 with upper bounds on the running time of a single quantifier elimination round. These upper bounds are all subsumed by the size of the obtained formulae, except possibly for the subdivision procedure of Step II (Lemma 2), the model counting procedure of Step IV (Lemma 4), and the further subdivision performed by Lemma 6. For Lemmas 2 and 6, the running time is only exponential in the size of the original formula, and thus polynomial time in the size of the obtained formula, as long as the latter

has at least exponential size. For Lemma 4, observe that $m \leq \|\mathrm{mod}(\Upsilon)\|$, where Υ is the quantifier-free formula of Lemma 8. Therefore, the bounds of Lemma 8 suffice for a triply exponential time overall.

Remark 2. Only small updates are necessary to treat monadically-guarded formulae of the form $\exists x(\Psi(x, \boldsymbol{z}) \wedge \exists^{\geq x} y \Phi(y, \boldsymbol{z}))$. Again, these updates deal with the fact that, contrary to $\exists^{=x} y \Phi$, the formula $\exists^{\geq x} y \Phi$ is true whenever there are infinitely many y satisfying Φ, or alternatively when x corresponds to a non-positive number. Then, Lemma 8 can be established for formulae of PAC containing both monadically-guarded quantifiers $\exists^{=x}$ and $\exists^{\geq x}$.

7 The monadically-guarded fragment is in doubly exponential space

In this section, we prove Theorem 2. Theorem 3 shows that our QE procedure has the same asymptotic running time as the standard QE procedures for PA. Historically, bounds obtained from the latter lead to computationally optimal decision procedures based on quantifier relativisation [12,4]. More precisely, given a formula Φ from PA, the QE procedures allow us to conclude that there is a bound C, of bitsize at most doubly exponential in $|\Phi|$, such that $\exists x\, \Phi \Leftrightarrow \exists x : -C \leq x \leq C \wedge \Phi$ holds (a *small-model property*). Then, a *quantifier relativisation procedure* follows the semantics of the formula and naïvely tries all the possible assignments to x in $[-C, C]$ whenever a quantifier $\exists x$ is encountered. With some bookkeeping, this procedure runs in 2ExpSpace. In this section, we show that this is also the case for our QE procedure, leading to a 2ExpSpace relativisation procedure for the monadically-guarded fragment of PAC, proving Theorem 2.

First of all, we need to recall a folklore result regarding the existence of infinitely many solutions of a quantifier-free Presburger formula.

Lemma 9. *Let ν be an assignment and $\Phi(y, \boldsymbol{z})$ be a q.f. formula of PA, where $\boldsymbol{z}$ has d variables. Let $C \stackrel{\text{def}}{=} \|\Phi\| \cdot d \cdot \max\{1, |\nu(z)| : z \text{ is in } \boldsymbol{z}\} + \|\Phi\|^{\#\mathrm{mod}(\Phi)} + 1$.*

1. *If there are finitely many $n \in \mathbb{Z}$ s.t. $\nu[n/y] \models \Phi$, then they all satisfy $|n| \leq C$.*
2. *If there are infinitely many $n \in \mathbb{Z}$ such that $\nu[n/y] \models \Phi$, then for every $j \in \mathbb{N}_+$ there is such an n satisfying $j \cdot C < |n| \leq (j+1) \cdot C$.*

Together with Lemma 8, this result leads to the relativisation of first-order quantifiers in the context of PAC.

Lemma 10. *There is a constant c with the following property. Let ν be an assignment, $\Phi(y, \boldsymbol{z})$ be a monadically-guarded formula of PAC, where $\boldsymbol{z}$ has d variables, and let $C \stackrel{\text{def}}{=} 2^{|\Psi|^{2^{c \cdot d}}} \cdot \max\{1, |\nu(z)| : z \text{ is in } \boldsymbol{z}\}$. Then, $\nu \models \exists y\, \Phi$ if and only if $\nu[n/y] \models \Phi$ holds for some $n \in \mathbb{Z}$ with $|n| \leq 3 \cdot C$.*

We want to derive a similar lemma for monadically guarded counting quantifiers. First of all, we consider a formula $\Phi = \exists^{=x} y\, \Psi(y, \boldsymbol{z})$ where Ψ is a monadically guarded formula. Recall that Φ is satisfied by an assignment ν whenever the number of distinct values $n \in \mathbb{Z}$ such that $\nu[n/y] \models \Psi$ is finite and equal to $\nu(x)$. By relying on Lemmas 8 and 9, we show the following lemma.

Lemma 11. *There is a constant c with the following property. Let ν be an assignment, and consider a formula $\Phi = \exists^{=x} y\, \Psi(y, \boldsymbol{z})$ such that Ψ is a monadically guarded formula of quantifier rank d. Let $C \stackrel{\text{def}}{=} 2^{|\Psi|^{2^{c \cdot d}}} \cdot \max\{1, |\nu(z)| : z \text{ is in } \boldsymbol{z}\}$. Then, $\nu \models \Phi$ iff (i) $\nu[n/y] \not\models \Psi$, for every $n \in \mathbb{Z}$ with $C < |n| \leq 3 \cdot C$; and (ii) $\#\{n \in \mathbb{Z} : |n| \leq C \text{ and } \nu[n/y] \models \Psi\} = \nu(x)$.*

We now consider the outermost quantifier x of a monadically-guarded formula $\Theta = \exists x\, (\Psi(x, \boldsymbol{z}) \wedge \exists^{=x} y\, \Phi(y, \boldsymbol{z}))$, and aim at finding relativisation bounds for the variable x. Notice that the subformula $\Psi(x, \boldsymbol{z}) \wedge \exists^{=x} y\, \Phi(y, \boldsymbol{z})$ is not, strictly speaking, in the monadically-guarded fragment of PAC. However, we can first apply Lemma 8 and obtain quantifier-free formulae $\widehat{\Psi}$ and Φ' equivalent to Ψ and Φ, respectively. Then, we apply the QE procedure of Section 3 on input $\exists^{=x} y\, \Phi'$, producing an equivalent quantifier-free formula $\widehat{\Phi}$. We have $\Theta \Leftrightarrow \exists x\, (\widehat{\Psi} \wedge \widehat{\Phi})$, where $\widehat{\Psi} \wedge \widehat{\Phi}$ is quantifier-free. Similarly to Lemma 10, we can now obtain relativisation bounds from $\exists x\, (\widehat{\Psi} \wedge \widehat{\Phi})$ by relying on Lemma 9:

Lemma 12. *There is a constant c with the following property. Let ν be an assignment, and let $\Theta = \exists x\, (\Psi(x, \boldsymbol{z}) \wedge \exists^{=x} y\, \Phi(y, \boldsymbol{z}))$ be a monadically-guarded formula of quantifier rank d. Define $C \stackrel{\text{def}}{=} 2^{|\Theta|^{2^{c \cdot d}}} \cdot \max\{1, |\nu(z)| : z \text{ is in } \boldsymbol{z}\}$. Then, $\nu \models \Theta$ if and only if there is $n \in \mathbb{N}$ s.t. $n \leq C$ and $\nu[n/x] \models \Psi \wedge \exists^{=x} y\, \Phi$.*

Lemmas 10 to 12 allow to evaluate the truth of a sentence of the monadically-guarded fragment of PAC by recursively evaluating the truth of its subformulae, and iterating over a finite set of values when considering first-order and counting quantifiers. As all the considered values admit a binary encoding that is doubly exponential in the size of the input formula, this proves Theorem 2.

8 A complexity characterisation

By Theorem 2, for deterministic machines, the monadically-guarded fragment of PAC is no harder than standard Presburger arithmetic, and the same is true when considering monadically-guarded quantifiers $\exists^{\geq x} y$ (Remark 2). However, by Proposition 1, PA is not complete for 2ExpSpace, but rather for the complexity class $\mathrm{STA}(*, 2^{n^{\mathcal{O}(1)}}, \mathcal{O}(n))$. This leads to the natural question on whether the monadically-guarded fragment of PAC is also complete for the same STA class. While we leave this question open, in this section we show a completeness result in the restricted case where all monadically-guarded quantifiers appear in the form $\exists x(\Psi(x) \wedge \exists^{\geq x} y\, \Phi)$, where $\Psi(x)$ is any formula from PAC having all models bounded by $2^{2^{|\Psi|}}$, in absolute value. For brevity, let us denote this fragment by F. As F extends PA, proving the following upper bound suffices.

Theorem 4. *The validity of a sentence Φ in F can be decided by an alternating Turing machine with runtime $2^{2^{|\Phi|^{\mathcal{O}(1)}}}$ and performing $\mathcal{O}(\mathrm{fd}(\Phi))$ alternations.*

Since the equivalence $\exists^{\geq c} y\, \Phi \Leftrightarrow \exists x : x = c \wedge \exists^{\geq x} y\, \Phi$, where $c \in \mathbb{Z}$ is written in binary, shows that F contains PA enriched with threshold counting quantifiers,

Function check(V : non-empty set of assignments, Φ : formula from F) $\to \{\top, \bot\}$

1 check(V, $t < 0$) = if $\nu \models t < 0$ holds for all $\nu \in V$ then return $\top$ else return $\bot$.
2 check(V, $\Phi_1 \vee \Phi_2$) = if $\exists V_1, V_2 : V = V_1 \cup V_2$ and check(V_1,Φ_1) = check(V_2,Φ_2) = $\top$
3 then return $\top$ else return $\bot$.
4 check(V, $\neg\Psi$) = if $\exists \nu \in V$: check($\{\nu\}$, Ψ) = $\top$ then return $\bot$ else return $\top$.
5 check(V, $\exists x(\Psi(x) \wedge \exists^{\geq x} y\, \Theta(y, \boldsymbol{z}))$) = if there is a family $(W_\nu)_{\nu \in V}$ such that
6 • check($\bigcup_{\nu \in V} W_\nu$, $\Psi \wedge \Theta$) = $\top$, and
7 • each W_ν is either $\{\nu[k^{(\nu)}/x][n^{(\nu)}/y]\}$ or $\{\nu[k^{(\nu)}/x][n_i^{(\nu)}/y] : i \in [1, k^{(\nu)}]\}$
8 where $|k^{(\nu)}| \leq 2^{2^{|\Psi|}}$
9 let C be defined as in Lemma 13, w.r.t. ν and Ψ;
10 $n^{(\nu)} \in \mathbb{Z}$ such that $C < |n^{(\nu)}| \leq 3 \cdot C$;
11 $|n_i^{(\nu)}| \leq C$ for every $i \in [1, k^{(\nu)}]$; and
12 $n_i^{(\nu)} \neq n_j^{(\nu)}$ for every two distinct $i, j \in [1, k^{(\nu)}]$
13 then return $\top$ else return $\bot$.

Fig. 1. Deciding whether a formula Φ from F is satisfied by all assignments in V.

this result implies Theorem 1. To establish Theorem 4, the first step is to rely on Lemmas 8 and 9 and adapt the proof of Lemma 11 to obtain a quantifier relativisation argument for the counting quantifier $\exists^{\geq x} y$.

Lemma 13. *There is a constant c with the following property. Let ν be an assignment, and consider a formula $\Phi = \exists^{\geq x} y\, \Psi(y, \boldsymbol{z})$ such that Ψ is a monadically guarded formula of quantifier rank d. Let $C \stackrel{\text{def}}{=} 2^{|\Psi|^{2^{c \cdot d}}} \cdot \max\{1, |\nu(z)| : z \text{ is in } \boldsymbol{z}\}$. Then, $\nu \models \Phi$ iff (i) there is $n \in \mathbb{Z}$ s.t. $\nu[n/y] \models \Psi$ and $C < |n| \leq 3 \cdot C$, or (ii) $\#\{n \in \mathbb{Z} : |n| \leq C \text{ and } \nu[n/y] \models \Psi\} \geq \nu(x)$.*

With Lemma 13 at hand, designing an algorithm that can be implemented as an alternating Turing machine with resources bounded as in Theorem 4 is simple. The function check$(\cdot, \cdot)$ given in Figure 1 provides such an algorithm.

Lemma 14. check(V, Φ) *returns $\top$ if and only if for all $\nu \in V$, $\nu \models \Phi$.*

When Φ is a sentence, i.e. $\text{fv}(\Phi) = \emptyset$, this lemma implies that Φ is valid if and only if check$(\{\nu\}, \Phi) = \top$, where ν is an arbitrary assignment. Then, Theorem 4 follows by establishing that check$(\cdot, \cdot)$ can be implemented with an alternating Turing machine that, on input $(\{\nu\}, \Phi)$ where Φ is a sentence in F, runs in time $2^{2^{|\Phi|^{\mathcal{O}(1)}}}$ and performs $\mathcal{O}(\text{fd}(\Phi))$ many alternations. We see the existential quantifications on V_1, V_2, ν and $(W_\nu)_{\nu \in V}$ in Lines 2, 4 and 5 as guesses done by the alternating Turing machine. The computation in Line 1 is done deterministically in time polynomial in the encoding of V and $t < 0$. In Line 2, the alternating Turing machine decides which branch among check(V_1,Φ_1) and check(V_2,Φ_2) must be evaluated, at the cost of one alternation. In this way, alternations occur only in the case of check(V,$\Phi_1 \vee \Phi_2$) and check(V,$\neg\Psi$), as the latter returns the negation of the assertion "$\exists \nu \in V$: check($\{\nu\}$, Ψ) = $\top$". This leads to $\mathcal{O}(\text{fd}(\Phi))$

many alternations overall. Let us now discuss the runtime of check$(\cdot,\cdot)$, again on alternating Turing machines. Assume that, after a certain number of recursive calls including at most $r \leq \mathrm{qr}(\Phi)$ calls to Line 5, the algorithm evaluates the input (V', Ψ). Then, the number of assignments in V' is bounded by $2^{r \cdot 2^{|\Phi|}}$ (this correspond to the case where each W_ν in Line 7 contains the maximum amount of assignments, according to $k^{(\nu)}$), and following the bounds on the numbers $n^{(\nu)}$ and $n_i^{(\nu)}$ in Lines 10 and 11 and by Lemma 13, all these assignments map each variable to an integer that is, in absolute value, bounded by $2^{r \cdot |\Phi|^{2^{c \cdot d}}}$, where c is the constant of Lemma 13 and d is the number of variables in Φ. So, as the number of recursive calls to check$(\cdot,\cdot)$ is bounded by $|\Phi|$, no more than $|\Phi| \cdot 2^{\mathrm{qr}(\Phi) \cdot 2^{|\Phi|}} \cdot \log_2(2^{\mathrm{qr}(\Phi) \cdot |\Phi|^{2^{c \cdot d}}}) \leq 2^{2^{(c+3)|\Phi|}}$ space is required to represent all possible sets of assignments that are generated throughout the evaluation of check$(\cdot,\cdot)$. All the assignments are guessed by the alternating Turing machine and thus, when also accounting for the computation done in Line 1, we conclude that check$(\{\nu\}, \Phi)$ runs in time $2^{2^{|\Phi|^{\mathcal{O}(1)}}}$.

9 Conclusion

We developed a new quantifier elimination procedure for Presburger arithmetic extended with the unary counting quantifiers (PAC), and adapted it for its monadically-guarded fragment. While the existence of an algorithm for PAC running in elementary time is wide open, our procedure runs in 3ExpTime on the monadically-guarded fragment and leads to the small-model property and relativisation argument, which show that this logic is decidable in 2ExpSpace. When it comes to deterministic algorithms, this matches the complexity of deciding standard Presburger arithmetic. However, fully settling the complexity of the monadically-guarded fragment of Presburger arithmetic seems to require a generalisation of the STA complexity framework to capture counting mechanisms, which we leave as an avenue for further investigation. In this direction, we have shown that Presburger arithmetic is still STA$(*, 2^{2^{n^{\mathcal{O}(1)}}}, O(n))$-complete when enriched with threshold quantifiers $\exists^{\geq c} y$, for the case of c written in binary but also even for the case of c represented succinctly as a solution of a Presburger formula Φ, characterising a number that may be doubly exponential in $|\Phi|$.

With respect to our QE procedure for (general) unary counting quantifiers $\exists^{=x}$, we have pinpointed precisely where the non-elementary growth occurs. It remains to be seen whether our procedure can be further improved, or if, possibly based on insights obtained from it, a non-elementary lower bound for Presburger arithmetic extended with the $\exists^{=x} y$ quantifier can be established.

Acknowledgments. We are grateful to our reviewers for drawing our attention to the translation of threshold counting into standard PA. This work is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 852769, ARiAT).

References

1. Apelt, H.: Axiomatische Untersuchungen über einige mit der Presburgerschen Arithmetik verwandte Systeme. Math. Log. Q. **12**(1), 131–168 (1966)
2. Barvinok, A.I.: A polynomial time algorithm for counting integral points in polyhedra when the dimension is fixed. Math. Oper. Res. **19**(4), 769–779 (1994)
3. Berman, L.: The complexity of logical theories. Theor. Comput. Sci. **11**(1), 71–77 (1980)
4. Habermehl, P., Kuske, D.: On Presburger arithmetic extended with modulo counting quantifiers. In: Proc. Foundations of Software Science and Computation Structures, FoSSaCS. Lect. Notes Comput. Sc., vol. 9034, pp. 375–389. Springer (2015)
5. Hague, M., Lin, A.W., Rümmer, P., Wu, Z.: Monadic decomposition in integer linear arithmetic. In: Proc. International Joint Conference on Automated Reasoning, IJCAR. Lect. Notes Comput. Sc., vol. 12166, pp. 122–140. Springer (2020)
6. Herre, H., Krynicki, M., Pinus, A., Väänänen, J.: The Härtig quantifier: A survey. J. Symb. Comput. **56**(4), 1153–1183 (1991)
7. Kozen, D.: Theory of Computation. Texts in Computer Science, Springer (2006)
8. Oppen, D.C.: A $2^{2^{2^{pn}}}$ upper bound on the complexity of Presburger arithmetic. J. Comput. Syst. Sci. **16**(3), 323–332 (1978)
9. Presburger, M.: Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In: Comptes Rendus du I congres de Mathematiciens des Pays Slaves, pp. 92–101. American Mathematical Society (1929)
10. Schweikardt, N.: Arithmetic, first-order logic, and counting quantifiers. ACM Trans. Comput. Log. **6**(3), 634–671 (2005)
11. Veanes, M., Bjørner, N., Nachmanson, L., Bereg, S.: Monadic decomposition. J. ACM **64**(2), 14:1–14:28 (2017)
12. Weispfenning, V.: The complexity of almost linear diophantine problems. J. Symb. Comput. **10**(5), 395–404 (1990)
13. Woods, K.: Presburger arithmetic, rational generating functions, and quasi-polynomials. J. Symb. Log. **80**(2), 433–449 (2015)