On Presburger Arithmetic Extended with Modulo Counting Quantifiers*

Peter Habermehl[1] and Dietrich Kuske[2]

[1] LIAFA, University Paris Diderot, France
[2] TU Ilmenau, Germany

Abstract. We consider Presburger arithmetic (PA) extended with modulo counting quantifiers. We show that its complexity is essentially the same as that of PA, i.e., we give a doubly exponential space bound. This is done by giving and analysing a quantifier elimination procedure similar to Reddy and Loveland's procedure for PA. We also show that the complexity of the automata-based decision procedure for PA with modulo counting quantifiers has the same triple-exponential time complexity as the one for PA when using least significant bit first encoding.

1 Introduction

Presburger arithmetic is the first-order theory of the structure $\mathcal{Z}$, i.e., the integers with addition and comparision. More precisely, we also allow the binary relations $\equiv_k$ (standing for equality modulo k) for $k \geqslant 2$, and all constants $c \in \mathbb{Z}$ to appear in formulas. This theory was shown to be decidable by Presburger [17], upper bounds on the complexity of (fragments of) Presburger arithmetic can, e.g., be found in [16,18,7,2,8,20,9]. Coding integers in binary, we know since the 60's that every definable relation can be accepted by a synchronous multitape automaton. The basic idea is that a synchronous three-tape automaton can verify the equation $k + \ell = m$ (in terms of the codings of the numbers k, ℓ, and m) and synchronously rational relations are effectively closed under Boolean operations and projection. At first glance, this translation results in automata of non-elementary size since complementation of automata comes with an exponential blow-up. From Klaedtke's results [13], it follows that automata of triply-exponential size suffice and that they can be constructed in four-fold exponential time using purely automata-theoretic methods. This result was improved by Durand-Gasselin and Habermehl who showed that "small" automata can be constructed efficiently, i.e., in triply-exponential time. Their first proof [6] uses an *ad hoc* construction of automata, their second proof [5] is more uniform in the sense that it applies to the structure $\mathcal{Z}$ and to automatic structures [10,11,3] of bounded degree (improving a result from [14]). Thus, Presburger arithmetic can be decided using automata-theoretic methods in triply exponential time.

More generally, these automata-theoretic methods rely on the fact that $\mathcal{Z}$ is an automatic structure. The motivating result on automatic structures is that their

* This work was partially supported by EGIDE/DAAD-Procope TAMTV.

DOI: 10.1007/978-3-662-46678-0_24

first-order theory is decidable [10,11,3]. One line of research on automatic structures concentrated on the extension of this result to more powerful logics. One can, for instance, extend first-order logic by a modulo-counting quantifier $\exists^{(p',p)}$ saying "modulo p, there are p' elements satisfying ...". The reason is that, as in the case of $\mathcal{Z}$ and first-order logic, one can construct from a formula in this extended logic a synchronous n-tape automaton that accepts all satisfying assignments of the formula [12] (see [19] for more quantifiers with this property).[1] Since $\mathcal{Z}$ is an automatic structure, this also holds here independent of whether we code integers in base 2 or 3. Consequently, by the Cobham-Semenov theorem [4,22], any relation in $\mathcal{Z}$ definable in this extended logic is effectively semilinear and therefore definable in first-order logic not using the modulo-counting quantifier (this claim also follows from [1] that presents a quantifier elimination for Härtig's quantifier "the number of witnesses for φ equals that for ψ", see also [21]).

This paper determines the complexity of the set of all formulas in the extended logic that hold in $\mathcal{Z}$. To this aim, we first present a procedure that eliminates modulo-counting quantifiers (see the beginning of Section 3.3 for a comparision with Apelt's [1] and Schweikardt's [21] procedures). This procedure is inspired by the classical one by Reddy and Loveland [18]. As in [18], we do not analyse the complexity of this procedure, but the resulting quantifier-free formula. We obtain that every formula in the extended logic has an equivalent quantifier-free formula that uses coefficients and moduli of doubly exponential size and constants of triply exponential size. Based on this finding and classical results on solutions of linear Diophantine equations [23], we show that the theory of the structure $\mathcal{Z}$ in the extended logic can be decided in doubly exponential space. Based on the quantifier elimination, we can also show that the construction of automata from formulas using the algorithms known from the theory of automatic structures can be done in triply exponential time. Thus, the theory of the structure $\mathcal{Z}$ in the extended logic can be decided in triply exponential time using automata-theoretic methods. In summary, we obtain that adding modulo-counting quantifiers does not increase the complexity of the theory of integer addition. Proof details can be found in the full version of the paper.

2 Preliminaries

The structure. The universe of the structure $\mathcal{Z}$ is the set of integers $\mathbb{Z}$. On this set, we consider the constants $c \in \mathbb{Z}$, the binary function $+$, the binary relation $<$ and the binary relations $\equiv_k$ for $k \geqslant 2$ (with $m \equiv_k n$ iff $k \mid m - n$).

The language. We will use a sequence $\bar{x} = (x_i)_{i\in\mathbb{N}}$ of variables. A *term* is an expression $\bar{a}\,\bar{x} + c$ where $\bar{a} = (a_i)_{i\in\mathbb{N}}$ is a sequence of integers with $a_i \neq 0$ for finitely many $i \in \mathbb{N}$ and $c \in \mathbb{Z}$. Let P be an arbitrary but fixed natural number. Then *formulas* of $\mathcal{L}_P$, *Presburger's logic with modulo-counting quantifiers*, are defined by recursion:

[1] In the complete version of this extended abstract, we show that the theory of an automatic structure using *only* modulo-counting quantifiers can be non-elementary.

- If s and t are terms, then $s < t$ (also written $t > s$) and $s \equiv_k t$ are (atomic) formulas (for $k \geqslant 2$).
- If φ and ψ are formulas, then so are $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, and $\varphi \leftrightarrow \psi$.
- If φ is a formula, x is a variable, and $0 \leqslant p' < p$, $2 \leqslant p \leqslant P$ are natural numbers[2], then $\exists x\colon \varphi$ and $\exists^{(p',p)} x\colon \varphi$ are formulas.

An *evaluation* is a function f that assigns integers to variables. For x a variable and $a \in \mathbb{Z}$, we let $f[x/a]$ be the evaluation with $f[x/a](x) = a$ and $f[x/a](y) = f(y)$ for all variables $y \neq x$. We can extend in a standard way an evaluation f to a function (also denoted f) that maps terms into $\mathbb{Z}$ and formulas to the truth values $\mathtt{tt}$ and $\mathtt{ff}$. In particular, if s and t are terms, then $f(s \equiv_k t) = \mathtt{tt}$ iff $f(s) - f(t)$ is a multiple of k. Furthermore, if φ and ψ are formulas, x a variable, and $0 \leqslant p' < p$ natural numbers, then $f(\exists^{(p',p)} x\colon \varphi) = \mathtt{tt}$ iff the set $\{a \in \mathbb{Z} \mid f[x/a](\varphi) = \mathtt{tt}\}$ is finite and $|\{a \in \mathbb{Z} \mid f[x/a](\varphi) = \mathtt{tt}\}| \equiv_p p'$.

A formula φ is *valid* if $f(\varphi) = tt$ for all evaluations f. *Presburger arithmetic with modulo-counting quantifiers* is the set of all valid formulas of $\mathcal{L}_P$. For two formulas F and G, we write $F \Leftrightarrow G$ for "$f(F) = f(G)$ for all evaluations f". We define as usual addition of terms as well as multiplication of a term with an integer.

For a term $t = \bar{a}\,\bar{x} + c$ and a variable x_i, we call a_i the *coefficient* of x_i in t. If the coefficient of x_i in t is 0, then we call t an *x_i-free term.*

Let x be a variable. Then an atomic formula φ is *x-separated* if there are an x-free term t and a non-negative integer $a \in \mathbb{N}$ such that φ is of the form $ax < t$, $t < ax$, or $ax \equiv_k t$. If t is an x-free term, then, e.g., the formula $0 \equiv_k t$ is x-separated since we identified the terms $0x$ and 0.

An atomic formula is *constant separated* if it is of the form $c < s$ or $s \equiv_k c$ where s is a term and c a constant.

A formula φ with a vector of k free variables $\boldsymbol{x} = (x_1, \ldots, x_k)$ is also written as $\varphi(\boldsymbol{x})$. Then we define $[\![\varphi(\boldsymbol{x})]\!] = \{(f(x_1), \ldots, f(x_k)) \mid f$ is an evaluation such that $f(\varphi) = \mathtt{tt}\}$. We also write $\boldsymbol{a}.\boldsymbol{x} > c$ (resp. $\boldsymbol{a}.\boldsymbol{x} \equiv_k c$) for constant separated formulas with free variables $\boldsymbol{x}$.

Next, let φ be a formula. Then $\textsc{Coeff}(\varphi) \subseteq \mathbb{Z}$ is the set of integers $-1, 0, 1$ and $\pm a$ such that there is an atomic formula $s < t$ in φ such that a is a coefficient appearing in the term $s-t$. Similarly, $\textsc{Const}(\varphi) \subseteq \mathbb{Z}$ is the set of integers $-1, 0, 1$ and $\pm c$ such that there is an atomic formula $s < t$ in φ such that c is the constant term in $s-t$. The set $\textsc{Mod}(\varphi) \subseteq \mathbb{N}$ contains all integers $k \geqslant 2$ such that an atomic formula of the form $s \equiv_k t$ appears in φ. Finally, $\mathbf{P}(\varphi) = \textsc{Coeff}(\varphi) \cup \textsc{Mod}(\varphi)$.

Note that $\textsc{Coeff}(\varphi)$ and $\textsc{Const}(\varphi)$ depend on subformulas of the form $s < t$, but not on subformulas of the form $s \equiv_k t$. On the other hand, $\textsc{Mod}(\varphi)$ only depends on subformulas of the form $s \equiv_k t$.

[2] This insures that we have only finitely many quantifiers.

3 Quantifier Elimination and a Decision Procedure

3.1 Elimination of $\exists$

In this section, we will eliminate the quantifier from a formula of the form $\exists x\colon \beta$ where β is a Boolean combination of atomic formulas. Our main concern is the "size" of the resulting formula, more precisely, of the coefficients, constants, and moduli appearing in it. Neither the result (Proposition 3.3) nor the method presented here is new, but this section is meant to simplify reading and to allow the reader to grasp the new results concerning the modulo-counting quantifier.

To this aim, we define the following sets (that will turn out to overapproximate the corresponding sets of the resulting quantifier-free formula):

$$\begin{aligned}
\textsc{Coeff}'(\beta) &= \{a_1a_2 - a_3a_4 \mid a_1, a_2, a_3, a_4 \in \textsc{Coeff}(\beta)\}\\
\textsc{Const}'(\beta) &= \left\{ a_1c_1 - a_2(c_2+c) \;\middle|\; \begin{array}{l} a_1, a_2 \in \textsc{Coeff}(\beta), c_1, c_2 \in \textsc{Const}(\beta) \\ |c| \leqslant \max \textsc{Coeff}(\beta) \cdot \operatorname{lcm} \textsc{Mod}(\beta) \end{array} \right\}\\
\textsc{Mod}'(\beta) &= \{a_1a_2kp \mid a_1a_2 \in \textsc{Coeff}(\beta), k \in \textsc{Mod}(\beta), 1 \leqslant p \leqslant P\}
\end{aligned}$$

Using these sets, we formulate the following condition on the pair of formulas (β, γ):

$$\textsc{Coeff}(\gamma) \subseteq \textsc{Coeff}'(\beta)\,,\ \textsc{Const}(\gamma) \subseteq \textsc{Const}'(\beta)\,, \textsc{Mod}(\gamma) \subseteq \textsc{Mod}'(\beta) \quad (1)$$

Lemma 3.1. *Let β be a Boolean combination of x-separated atomic formulas, $ax < t$ or $t < ax$ some atomic formula from β with $a > 0$ and $-aN \leqslant c \leqslant aN$ where $N = \operatorname{lcm} \textsc{Mod}(\beta)$. There exists a Boolean combination $\beta_{a,t+c}$ of x-free atomic formulas such that $(\beta, \beta_{a,t+c})$ satisfies (1) and, for all evaluations f,*

$$f(ax) = f(t+c) \Longrightarrow f(\beta) = f(\beta_{a,t+c})\,.$$

Proof. The formula $\beta_{a,t+c}$ is obtained from β by the following replacements (where s is some x-free term and $k \geqslant 2$):

$$\begin{aligned}
a'x < s &\text{ is replaced by } a't + a'c < as\\
s < a'x &\text{ is replaced by } as < a't + a'c\\
a'x \equiv_k s &\text{ is replaced by } a't + a'c \equiv_{ak} as
\end{aligned}$$

□

Lemma 3.2. *Let x be a variable and β a Boolean combination of x-separated atomic formulas. Then there exists a Boolean combination γ of x-free atomic formulas such that (β, γ) satisfies (1) and $(\exists x\colon \beta) \Leftrightarrow \gamma$.*

Proof. Let T be the set of all pairs (a, t) such that β contains an atomic formula of the form $ax < t$ or $t < ax$ with $a > 0$ (or $T = \{(1, 0)\}$ if no such atomic formula exists). Let furthermore $N = \operatorname{lcm}(\textsc{Mod}(\beta))$ such that N is a multiple of every integer k such that an atomic formula of the form $ax \equiv_k t$ appears in β. Then $\exists x\colon \beta$ is equivalent with the formula $\gamma := \bigvee_{(a,t)\in T} \bigvee_{-aN\leqslant c\leqslant aN} (\beta_{a,t+c} \wedge 0 \equiv_a t + c)$.

□

Proposition 3.3. *Let x be a variable and α a Boolean combination of atomic formulas. Then there exists a Boolean combination γ of x-free atomic formulas such that (β, γ) satisfies* (1) *and* $(\exists x\colon \alpha) \Leftrightarrow \gamma$.

Proof. Without changing the sets Coeff etc., we can transform α into an equivalent Boolean combination β of x-separated atomic formulas. Then γ is the formula obtained from Lemma 3.2. □

3.2 Elimination of $\exists^{(p',p)}$

In this section, we want to prove a proposition analogous to Prop. 3.3, where $\exists x\colon \alpha$ is replaced by $\exists^{(p',p)} x\colon \alpha$. The crucial point is to prove the analogue of Lemma 3.2.

Lemma 3.4. *Let x be a variable, β a Boolean combination of x-separated atomic formulas, and $0 \leqslant p' < p \leqslant P$ natural numbers. Then there exists a Boolean combination of atomic formulas γ such that (β, γ) satisfies* (1) *and* $(\exists^{(p',p)} x\colon \beta) \Leftrightarrow \gamma$.

The proof of this lemma requires several claims and definitions that we demonstrate first, the actual proof of Lemma 3.4 can be found on page 381.

Let T be the set of all pairs (a, t) such that β contains an atomic formula of the form $ax < t$ or $t < ax$ with $a > 0$ (if no such formula exists, set $T = \{(1, 0)\}$).

Let S be some non-empty subset of T and let $\prec$ be a strict linear order on S. We call an evaluation f *consistent* with $\prec$ if the following hold:

- $\frac{f(s_1)}{a_1} < \frac{f(s_2)}{a_2} \iff (a_1, s_1) \prec (a_2, s_2)$ for all $(a_1, s_1), (a_2, s_2) \in S$
- for all $(a_1, t_1) \in T$, there exists $(a_2, s_2) \in S$ with $\frac{f(t_1)}{a_1} = \frac{f(s_2)}{a_2}$.

In the following, let $S = \{(a_1, s_1), (a_2, s_2), \dots, (a_n, s_n)\}$ with $(a_1, s_1) \prec (a_2, s_2) \prec \cdots \prec (a_n, s_n)$. Consider the following formulas for $0 \leqslant r < p$ and $1 \leqslant i < n$:

$$\beta_{0,r} = \exists^{(r,p)} x\colon (a_1 x < s_1 \wedge \beta) \qquad \beta_{n,r} = \exists^{(r,p)} x\colon (s_n < a_n x \wedge \beta)$$
$$\beta_{i,r} = \exists^{(r,p)} x\colon (s_i < a_i x \wedge a_{i+1} x < s_{i+1} \wedge \beta) \quad \beta'_{i,r} = \exists^{(r,p)} x\colon (x = s_i \wedge \beta)$$

If f is an evaluation, then $\beta_{0,r}$ expresses that (modulo p) there are r integers b with $f[x/b](\beta) = \mathtt{tt}$ and $b < \frac{f(s_1)}{a_1}$. Similarly, $\beta_{i,r}$ holds under f if and only if there are (modulo p) r integers b in the open interval $\left(\frac{f(s_i)}{a_i}, \frac{f(s_{i+1})}{a_{i+1}}\right)$ with $f[x/b](\beta) = \mathtt{tt}$ etc. Now consider the formula

$$\varphi^{\prec} = \bigvee \left(\bigwedge_{0 \leqslant i \leqslant n} \beta_{r_i,p} \wedge \bigwedge_{1 \leqslant i \leqslant n} \beta'_{r'_i,p} \right)$$

where the disjunction extends over all tuples $(r_0, r_1, \dots, r_n, r'_1, r'_2 \dots, r'_n)$ of integers from $\{0, 1, \dots, p-1\}$ that, modulo p, sum up to p'. For any evaluation f consistent with $\prec$, we therefore get $f(\exists^{(p',p)} x\colon \beta) = f(\varphi^{\prec})$. In order to construct γ as claimed in Lemma 3.4, it therefore suffices to eliminate the counting quantifiers from the formulas $\beta_{i,r}$ and $\beta'_{i,r}$. In this elimination procedure (detailed in the following claims), we will assume the evaluation to be consistent with $\prec$.

Claim 3.4.1. *Let $0 \leqslant r < p$. There exist Boolean combinations $\gamma_{0,r}^{\prec}$ and $\gamma_{n,r}^{\prec}$ of atomic formulas such that $(\beta, \gamma_{0,r}^{\prec})$ and $(\beta, \gamma_{n,r}^{\prec})$ satisfy (1) and $f(\beta_{0,r}) = f(\gamma_{0,r}^{\prec})$ as well as $f(\beta_{n,r}) = f(\gamma_{n,r}^{\prec})$ for all evaluations f that are consistent with $\prec$.*

We next want to eliminate the quantifier from $\beta_{i,r}$ for $1 \leqslant i < n$, i.e., we consider the integers in the open interval $\left(\frac{f(s_i)}{a_i}, \frac{f(s_{i+1})}{a_{i+1}}\right)$. It turns out to be convenient to split the set of these integers b according to $(a_i b - f(s_i)) \bmod a_i N$.

Claim 3.4.2. *For $1 \leqslant i < n$, $1 \leqslant c \leqslant a_i N$, and $0 \leqslant r < p$, set*

$$\beta_{i,r,c} = \exists^{(r,p)} c\colon (s_i < a_i x \wedge a_{i+1} x < s_{i+1} \wedge a_i x \equiv_{a_i N} s_i + c \wedge \beta)\,.$$

There exists a Boolean combination $\gamma_{i,r,c}^{\prec}$ of atomic formulas such that $(\beta, \gamma_{i,r,c}^{\prec})$ satisfies (1) and $f(\beta_{i,r,c}) = f(\gamma_{i,r,c}^{\prec})$ for all evaluations f consistent with $\prec$.

Proof. Let f be any evaluation that is consistent with $\prec$. We consider the following two sets $X \supseteq Y$:

$$X = \left\{ b \in \mathbb{Z} \;\middle|\; \frac{f(s_i)}{a_i} < b < \frac{f(s_{i+1})}{a_{i+1}}, a_i b \equiv_{a_i N} f(s_i) + c \right\} \text{ and}$$
$$Y = \{ b \in X \mid f[x/b](\beta) = \mathtt{tt} \}$$

Our aim is to construct a formula $\gamma_{i,r,c}^{\prec}$ that holds under the evaluation f if and only if $|Y| \equiv_p r$. Since the formula we construct is independent from f, this will prove the claim.

Let b be an integer from the open interval $\left(\frac{f(s_i)}{a_i}, \frac{f(s_{i+1})}{a_{i+1}}\right)$. Then $b \in X$ iff $a_i b \equiv_{a_i N} f(s_i) + c$. But this is the case iff $b \equiv_N \frac{f(s_i)+c}{a_i}$ (which, in particular, means $\frac{f(s_i)+c}{a_i} \in \mathbb{Z}$). Hence X is the set of integers of the form $\frac{f(s_i)+c}{a_i} + N \cdot k$ for some $k \in \mathbb{N}$ from the above open interval.

Next let $b_1 \in Y \subseteq X$ and $b_2 \in X$. Then $b_1 \equiv_N b_2$ and $f[x/b_1](\beta) = \mathtt{tt}$. Since N is a multiple of all moduli appearing in β, we get $f[x/b_2](\beta) = \mathtt{tt}$ and therefore $b_2 \in Y$. Hence $Y \in \{\emptyset, X\}$. Since $\frac{f(s_i)+c}{a_i} \in X$ if and only if $X \neq \emptyset$, we have $Y = X$ if $\frac{f(s_i)+c}{a_i} \in Y$ and $Y = \emptyset$ otherwise. Note that the first case occurs if and only if $f(\theta) = \mathtt{tt}$ where

$$\theta = \exists x (a_i x = s_i + c \wedge a_{i+1} x < s_{i+1} \wedge \beta)\,.$$

Now assume $\frac{f(s_i)+c}{a_i} \in Y$ which in particular implies that a_i divides $f(s_i) + c$. Then the size $|X|$ of the set X is the maximal natural number k with $\frac{f(s_i)+c}{a_i} + N \cdot k < \frac{f(s_{i+1})}{a_{i+1}}$, i.e., $|X| = k$ if and only if

$$a_{i+1}(f(s_i) + c + a_i N \cdot k) < a_i f(s_{i+1}) \leqslant a_{i+1}(f(s_i) + c + a_i N \cdot (k+1))\,.$$

Consequently, we have in this case $|Y| \equiv_p r$ if and only if $|X| \equiv_p r$ if and only if the following formula ν holds under f:

$$\nu = \exists y\colon \begin{pmatrix} a_i a_{i+1} N y < a_i s_{i+1} - a_{i+1} s_i - a_{i+1} c \\ \wedge\; a_i s_{i+1} - a_{i+1} s_i - a_{i+1} c - a_i a_{i+1} N \leqslant a_i a_{i+1} N y \\ \wedge\; y \equiv_p r \end{pmatrix}$$

So far, we showed that $f(\beta_{i,r,c}) = \mathtt{tt}$ if and only if

$$f(\theta \wedge \nu) = \mathtt{tt} \text{ or } (r = 0 \text{ and } f(\nu) = \mathtt{ff})\,. \tag{2}$$

Now, we can construct quantifier-free formulas $\bar{\theta}$ and $\bar{\nu}$ that can be shown to be equivalent to θ and ν, respectively, and to satisfy (1). $\square$

Claim 3.4.3. *Let $1 \leqslant i < n$ and $0 \leqslant r < p$. There exists a Boolean combination $\gamma^{\prec}_{i,r}$ of atomic formulas such that $(\beta, \gamma^{\prec}_{i,r})$ satisfies* (1) *and $f(\beta_{i,r}) = f(\gamma^{\prec}_{i,r})$ for all evaluations f consistent with $\prec$.*

Proof. Note that the formulas $s_i < a_i x \wedge a_{i+1}x < s_{i+1} \wedge \beta$ and

$$\bigvee_{1\leqslant c\leqslant a_iN} (s_i < a_i x \wedge a_{i+1}x < s_{i+1} \wedge a_i x \equiv_{a_iN} s_i + c \wedge \beta)$$

are equivalent and the disjunction in this formula is exclusive (i.e., every x satisfies at most one conjunct). Therefore, we can set

$$\gamma^{\prec}_{i,r} = \bigvee \bigwedge_{1\leqslant c\leqslant a_iN} \gamma^{\prec}_{i,c,r_c}$$

where the disjunction extends over all tuples $(r_1, r_2, \ldots, r_{a_iN})$ of integers from $\{0, 1, \ldots, p-1\}$ with $\sum_{1\leqslant c\leqslant a_iN} r_c \equiv_p r$. Now the claim follows from Claim 3.4.2. $\square$

Claim 3.4.4. *Let $1 \leqslant i \leqslant n$ and $0 \leqslant r < p$. There exists a Boolean combination $\delta^{\prec}_{i,r}$ of atomic formulas such that $(\beta, \delta^{\prec}_{i,r})$ satisfies* (1) *and, for all evaluations f (even those that are not consistent with $\prec$),*

$$f(\beta'_{i,r}) = f(\delta^{\prec}_{i,r})\,.$$

Proof. By Lemma 3.1, the formulas $a_i x = s_i \wedge \beta$ and $a_i x = s_i \wedge \beta_{a_i,s_i}$ are equivalent. Hence the formula

$$\delta^{\prec}_{i,r} = \begin{cases} \neg\beta_{a_i,s_i} & \text{if } r = 0 \\ \beta_{a_i,s_i} & \text{if } r = 1 \\ 0 < 0 & \text{if } r > 1 \end{cases}$$

is equivalent with $\beta'_{i,r}$. Since $\delta^{\prec}_{i,r}$ is a Boolean combination of the formulas β_{a_i,s_i} and $0 < 0$, the pair $(\beta, \delta^{\prec}_{i,r})$ satisfies (1) by Lemma 3.1. $\square$

Having shown all these claims, we now use them to finally prove Lemma 3.4.

Proof (of Lemma 3.4). Let $S \subseteq T$ be some non-empty subset of T and let $\prec$ be a strict linear order on S. As above, we let $S = \{(a_1, s_1), \ldots, (a_n, s_n)\}$ with $(a_1, s_1) \prec (a_2, s_2) \prec \cdots \prec (a_n, s_n)$. Then set

$$\gamma^{\prec} = \bigvee \left(\bigwedge_{0\leqslant i\leqslant n+1} \gamma^{\prec}_{i,r_i} \wedge \bigwedge_{1\leqslant i\leqslant n} \delta^{\prec}_{i,r'_i} \right)$$

where the disjunction extends over all tuples $(r_0, r_1, \dots, r_{n+1}, r'_1, r'_2 \dots, r'_n)$ of natural numbers from $\{0, 1, \dots, p-1\}$ with $\sum_{0 \leqslant i \leqslant n+1} r_i + \sum_{1 \leqslant i \leqslant n} r'_i \equiv_p p'$. Then $f(\varphi^{\prec}) = f(\gamma^{\prec})$ for all evaluations f that are consistent with $\prec$. Furthermore, $\gamma^{\prec}$ is a Boolean combination of atomic formulas and $(\beta, \gamma^{\prec})$ satisfies (1).

Next consider the formla

$$\alpha^{\prec} = \bigwedge_{1 \leqslant i < n} a_{i+1} s_i < a_i s_{i+1} \wedge \bigwedge_{(a,t) \in T} \bigvee_{1 \leqslant i \leqslant n} a_i t = a s_i \,.$$

Then, for any evaluation f, we have $f(\alpha^{\prec}) = \mathtt{tt}$ if and only if f is consistent with $\prec$. Since $\alpha^{\prec}$ is a Boolean combination of formulas of the form $a's < at$ with $(a, s), (a', t) \in T$, the pair $(\beta, \alpha^{\prec})$ satisfies (1).

Finally, let

$$\gamma = \bigwedge_{(*)} (\alpha^{\prec} \to \gamma^{\prec})$$

where the conjunction $(*)$ extends over all strict linear orders $\prec$ on some non-empty subset of T. $\square$

Proposition 3.5. *Let x be a variable and α a Boolean combination of atomic formulas. Let furthermore $E = \exists$ or $E = \exists^{(p',p)}$ for some $0 \leqslant p' < p$ and $2 \leqslant p \leqslant P$. Then there exists a Boolean combination γ of atomic formulas such that $(Ex\colon \alpha) \Leftrightarrow \gamma$. Furthermore, we have the following:*

$$\max \mathbf{P}(\gamma) \leqslant \max \mathbf{P}(\alpha)^3 \cdot P$$
$$\max \textsc{Const}(\gamma) \leqslant \max \textsc{Const}(\alpha) \cdot 2^{\max \mathbf{P}(\alpha)^3}$$

3.3 An Efficient Decision Procedure

Now, by induction on the quantifier depth we can obtain the following theorem.

Theorem 3.6. *Let $\varphi \in \mathcal{L}_P$ be a formula of quantifier-depth d. There exists an equivalent Boolean combination γ of atomic formulas with*

$$\max \mathbf{P}(\gamma) \leqslant (P \cdot \max \mathbf{P}(\varphi))^{4^d} \text{ and}$$
$$\max \textsc{Const}(\gamma) \leqslant 2^{(P \cdot \max \mathbf{P}(\varphi))^{4^d}} \cdot \max \textsc{Const}(\varphi)\,.$$

Comparison with Apelt's and with Schweikardt's elimination procedure. In the structure $\mathcal{Z}$, the modulo counting quantifier is a special case of Härtig's quantifier. Apelt [1] and Schweikardt [21] presented quantifier elimination procedures for Härtig's quantifier and therefore for its special case, the modulo counting quantifier. Differently from Schweikardt, we do not transform φ into disjunctive normal form, we do not normalize terms, and we do not replace a counting quantifier by many existential quantifiers. While we are not able to handle Härtig's quantifer this way, these differences allow to obtain the elementary bounds described in the theorem above. These elementary bounds are the basis for the following decision procedure.

Let $\varphi(x)$ be a Boolean combination of atomic formulas (note that x is the only free variable) and $A = \max(\mathbf{P}(\varphi) \cup \{6\})$. If φ is satisfiable, then results from [23] imply that φ has a witness of absolute value at most $A^{A^5} \cdot \max \textsc{Const}(\varphi)$. Using Theorem 3.6, we can infer a similar result for arbitrary formulas $\varphi(x)$ with one free variable. If φ has ℓ additional variables, instantiated by integers of absolute value $\leqslant N$, we can prove the following:

Corollary 3.7. *There exists $\kappa \geqslant 1$ with the following property. Consider a formula $\varphi(x, y_1, \ldots, y_\ell)$ from $\mathcal{L}_P$ of quantifier-depth d. Let $n_1, \ldots, n_\ell \in \mathbb{Z}$ with $|n_i| \leqslant N$. Then the formula $\exists x\colon \varphi(x, n_1, \ldots, n_\ell)$ is true if and only if there exists $n \in \mathbb{Z}$ such that $\varphi(n, n_1, \ldots, n_\ell)$ is true with*

$$|n| \leqslant 2^{(P \cdot \max \mathbf{P}(\varphi))^{\kappa^d}} \cdot \max \textsc{Const}(\varphi) \cdot N \cdot \max(1, \ell).$$

Next, we want to prove a similar result for the modulo-counting quantifier. Recall that $\exists^{(p',p)} x\colon \varphi(x)$ can only be true if φ has only finitely many witnesses, i.e., if the formula $\exists y \forall x\colon (\varphi(x) \to |x| \leqslant y)$ is true. Applying the above corollary, one finds a finite interval such that φ has infinitely many witnesses iff it has at least one witness in this interval. In case φ has only finitely many witnesses, then all of them are of bounded absolute value. More precisely, we get the following

Corollary 3.8. *Let κ be the constant from Corollary 3.7 and*

$$C = 2^{(P \cdot \max \mathbf{P}(\varphi))^{\kappa^{d+1}}} \cdot \max \textsc{Const}(\varphi) \cdot N \cdot \max(1, \ell)\,.$$

Let $\varphi = \varphi(x, y_1, \ldots, y_\ell) \in \mathcal{L}_P$ be a formula of quantifier-depth d, let $n_1, \ldots, n_\ell \in \mathbb{Z}$ with $|n_i| \leqslant N$. Then $\exists^{(p',p)} x\colon \varphi(x, n_1, \ldots, n_\ell)$ is true if and only if the following hold:

(1) no integer n with $C < |n| \leqslant C^2$ makes $\varphi(n, n_1, \ldots, n_\ell)$ true and
(2) $|\{n \in \mathbb{Z} \mid |n| \leqslant C$ and $\varphi(n, n_1, \ldots, n_\ell)$ is true$\}| \equiv_p p'$.

Corollaries 3.7 and 3.8 allow to evaluate the truth value of a sentence φ by, recursively, evaluating the truth value of subformulas ψ of φ with arguments of bounded size. Analysing this size carefully, one obtains

Theorem 3.9. *Presburger arithmetic with modulo-counting quantifiers is decidable in doubly exponential space.*

Note that this complexity matches the best known upper bound for Presburger arithmetic without modulo-counting quantifiers from [7].

4 Automata Based Decision Procedure

In this section we show that an automaton accepting all solutions of a formula of $\mathcal{L}_P$ can be constructed in triply exponential time. We follow the same ideas as in [6] where the same result was given for Presburger's logic.

4.1 Encoding

We represent integer vectors as finite words. We use a vectorial least significant bit first coding. For $h > 0$ we define $\Sigma_h = \{0,1\}^h$. Moreover we use the separate sign alphabet $S_h = \{+,-\}^h$ (indicating if the corresponding integer is positive or negative). Given any letter a in Σ_h or S_h we write $\pi_i(a)$ with $1 \leq i \leq h$ for its i-th component. Similarly, the i-th component of a h dimensional vector $\boldsymbol{x} \in \mathbb{Z}^h$ is denoted by $\pi_i(\boldsymbol{x})$. The symbol $+$ corresponds to 0 and $-$ corresponds to 1. In this way, to each letter $a \in \Sigma_h$ corresponds a letter $s(a) \in S_h$. Similarly to each letter $s \in S_h$ corresponds a letter $a(s) \in \Sigma_h$. Words of $\Sigma_h^* S_h$ represent h-dimensional integer vectors. A word $w_0 \dots w_n s \in \Sigma_h^* S_h$ represents the integer vector denoted by $\langle w_0 \dots w_n s\rangle$ whose i^{th} component (with $1 \leq i \leq h$) is computed as: If $s_i = +$, then $\pi_i(\langle w_0 \dots w_n s\rangle) = \sum_{j=0}^{n} 2^j . \pi_i(w_j)$ and if $s_i = -$, then $\pi_i(\langle w_0 \dots w_n s\rangle) = -2^{n+1} + \sum_{j=0}^{n} 2^j . \pi_i(w_j)$. For example, $\langle (0,1)(1,1)(1,0)(+,-)\rangle = \langle (0,1)(1,1)(1,0)\ (0,1)(+,-)\rangle = (6,-5)$. In particular, $\langle + \rangle = 0$ and $\langle - \rangle = -1$. We also define the notation $\langle . \rangle_+$ over Σ_h^* as $\langle w\rangle_+ = \langle w(+,\dots,+)\rangle$.

Remark 4.1. Let $w', w \in \Sigma_h^*, s \in S_h$. We have $\langle w'ws\rangle = \langle w'\rangle_+ + 2^{|w'|}\langle ws\rangle$.

Each vector has an infinite number of representations. Indeed for each word $w_0 \dots w_n s \in \Sigma_h^* S_h$, any word in $w_0 \dots w_n (a(s))^* s$ represents the same vector. To get a unique representation for each vector, we can take the shortest word representing it.

Given a Presburger formula $\varphi(\boldsymbol{x})$ with h free variables, we say that it defines the language $L_\varphi = \{w \in \Sigma_h^* S_h \mid \langle w\rangle \in [\![\varphi(\boldsymbol{x})]\!]\}$. Such languages are regular, called Presburger-definable and meet the following saturation property: If a representation of a vector is in the language then any other representation of that vector is also in the language. Our coding satisfies the following property [15].

Property 4.2. Any residual of a Presburger-definable language is either a Presburger-definable language, or the empty word language.

A *deterministic automaton* (DFA) is a tuple $(\Sigma, Q, q_0, Q_f, \delta)$ where Σ is the finite alphabet, Q the set of states, q_0 the initial state, $Q_f \subseteq Q$ the set of final states and δ the transition function from $Q \times \Sigma$ to Q. We suppose DFA to be complete (containing a sink state, if necessary). In a DFA accepting all solutions of a Presburger formula $\varphi(\boldsymbol{x})$ with h free variables, a word $w \in \Sigma_h^*$ leads from the initial state to a state accepting exactly all solutions of $\varphi(2^{|w|}\boldsymbol{x} + \langle w\rangle_+)$. Therefore, we can consider states (except final ones) of such automata as being Presburger formulas.

Given any Presburger-definable language L, the corresponding *uniformised Presburger-definable language* is defined by taking only one word (the shortest) representing the given vector. We obtain it by intersecting L (or the corresponding automaton) with a regular language ($\subseteq \Sigma_h^* S_h$) which forbids that words end with $a(s)s \in \Sigma_h S_h$ for some $s \in S_h$. We call this operation *uniformisation*.

4.2 Complexity of the Automata Based Decision Procedure

The well-known decision procedure for Presburger arithmetic using automata is based on recursively constructing an automaton accepting solutions of a Presburger formula by using automata constructions for handling logical connectives and quantifiers. Automata for constant separated formulas can be easily constructed. The following lemmas are from [6]. Let $\|\boldsymbol{a}\|_+ = \Sigma_{\{i \mid a_i \geq 0\}} a_i$ and $\|\boldsymbol{a}\|_- = \Sigma_{\{i \mid a_i \leq 0\}} |a_i|$. Let $\perp$ be the formula $0 < 0$.

Lemma 4.3. *The minimal DFA accepting the Presburger definable language corresponding to the formula* $\boldsymbol{a}.\boldsymbol{x} > c$ *has at most* $2 \cdot max(||\boldsymbol{a}||, |c|) + 1$ *states. Each non-final state accepts languages corresponding to formulas of the form* $\perp$ *or* $\boldsymbol{a}.\boldsymbol{x} > c'$ *with* $c' = c$ *or* $\min(c, -\|a\|_+) \leq c' < \max(c, \|a\|_-)$

Lemma 4.4. *The minimal DFA accepting the Presburger definable language corresponding to the formula* $\boldsymbol{a}.\boldsymbol{x} \equiv_{2^m(2n+1)} c$ *with* $0 \leq c < 2^m(2n+1)$ *and* $m, n \geq 0$ *has at most* $2^m(2n+1)+1$ *states. Each non-final state accepts languages corresponding to formulas of the form* $\boldsymbol{a}.\boldsymbol{x} \equiv_{2n+1} c'$ *with* $c' \in [0, 2n]$ *(this type of states is reached after* m *transitions) and* $\boldsymbol{a}.\boldsymbol{x} \equiv_{2^{m_1}(2n+1)} c'$ *where* $(m_1 = m \wedge c' = c) \vee (m_1 < m \wedge \gamma \in [0, 2^{m_1}(2n+1)-1]$ *and* $m_1 < m$.

Each logical connective ($\wedge$, $\vee$, $\leftrightarrow$, $\neg$) corresponds then naturally to operations on automata (For $\neg$ it is of course crucial to have a deterministic automaton). Furthermore to get an automaton for $\exists y \colon \varphi(y, \boldsymbol{x})$ given an automaton for $\varphi(y, \boldsymbol{x})$ one *projects away* [3] the component for y and obtains a *non-deterministic* automaton. Then, to be able to continue the recursive construction, the automaton is determinised, uniformised and minimised. Starting from an automaton of triple-exponential size, determinisation might lead to an automaton of quadruple-exponential size. However, for Presburger's logic the size of the automata during the construction is at most triple-exponential in the size of the formula [6]. We refine this analysis here to get the same upper bound for formula containing also $\exists^{(p',p)}$ quantifiers. For that we first detail the corresponding automata construction before analysing the size of the (intermediate) automata.

Automata Construction for the Modulo-Counting Quantifier. We adapt the construction of [12,19] for our particular encoding. Here it is crucial to have uniformised automata.

Lemma 4.5. *Given a DFA* $\mathcal{A}_\varphi$ *accepting the uniformised Presburger language* L_φ *defined by a formula* $\varphi(y, \boldsymbol{x})$ *of* $\mathcal{L}_P$ *one can construct a DFA* $\mathcal{A}_\psi$ *accepting the uniformised Presburger definable language* L_ψ *defined by* $\psi = \exists^{(p',p)} y \colon \varphi(y, \boldsymbol{x})$.

Proof. Without loss of generality we suppose that the value of y is given by the first component of letters of $\mathcal{A}_\varphi$. We need first some definitions. A *max-V*

[3] As the automaton should accept shortest encodings, additional transitions with a sign letter going to the final state have to be added before uniformisation.

multiset wrt. a natural number $max \geq 1$ and a set V is a multiset of elements of V such that each element appears at most max times. We denote all of these multisets by $\mathcal{M}_{max}(V)$. A max-V multiset can be seen as a multiplicity function mapping elements from V to $\{0, 1, 2, \ldots, max\}$. For positive natural numbers x and y with $y > 1$, we define $x \ mod_1 \ y = x \ mod \ y$ if $x \ mod \ y \neq 0$, $x \ mod_1 \ y = 0$ if $x = 0$ and $x \ mod_1 \ y = y$ else. Given two max-V multisets m_1, m_2 their union $m_1 \cup m_2$ is defined as $(m_1 \cup m_2)(v) = (m_1(v) + m_2(v)) \ mod_1 \ max$ for all $v \in V$.

Since $\mathcal{A}_\varphi$ is uniformised, we can suppose that $\mathcal{A}_\varphi$ has exactly one accepting state which has outgoing transitions only to the sink state. Let $\mathcal{A}_\varphi = (\Sigma_h \cup S_h, Q \cup \{F\}, q_0, \{F\}, \delta)$ with $L(\mathcal{A}) \subseteq \Sigma_h^* S_h$. We construct a DFA $\mathcal{A}_\psi = (\Sigma_{h-1} \cup S_{h-1}, Q' \cup \{F'\}, q_0', \{F'\}, \delta')$ with $L(\mathcal{A}_\psi) \subseteq \Sigma_{h-1}^* S_{h-1}$ as follows: The idea is to count modulo p how often a state can be reached (0 means unreachable) from the initial state using transitions where the first component of letters is arbitrary.

Formally, we have $Q' \subseteq \mathcal{M}_p(Q)$. Furthermore, we construct Q' starting from the multiset $q_0' = \{q_0\}$ with a modified on the fly subset construction. That means that Q' only contains *reachable* p-Q multisets of states. For each letter $a \in \Sigma_{h-1}$ and each state m (a p-Q multiset) of Q' we define a successor state $m' = \delta(m, a)$ by setting for all $q \in Q$, $m'(q) = (\sum_{q_1 \in Q} m(q_1) \cdot |\{(q_1, b) \mid \delta(q_1, (b, a)) = q$ and $b \in \{0, 1\}\}|) \ mod_1 \ p$. Now, we describe how to determine the transitions going to the final state F'. Here we have to take into account the number of times (which can be infinite) a vector corresponding to a word from Σ_{h-1}^* obtained by projection from a word w of $L(\mathcal{A}_\varphi)$ can be obtained by projection from other longer words of $L(\mathcal{A}_\varphi)$ with same prefix w. Since the automaton $\mathcal{A}_\varphi$ is uniformised each such word is only counted once. For each sign letter $s \in S_h$ with $s = (s_1, \ldots, s_h)$ we first define $s^+ = (+, s_2, \ldots, s_h)$ and $s^- = (-, s_2, \ldots, s_h)$. For each sign letter $s \in S_h$ and each state $q \in Q$, we compute then $m_{s,q}$, the (possible infinite) number of paths from q in $\mathcal{A}$ to the final state F labeled by a word from the language $(a(s^+) + a(s^-))^* s$. Then, for each sign letter $s \in S_{h-1}$ there is a transition from a state $m \in Q'$ to the final state F' iff (1) $m_{(+,s),q}$ and $m_{(-,s),q}$ are both not infinite for all $q \in Q$ with $m(q) \neq 0$ and (2) $(\sum_{q \in Q \wedge \delta(q,(+,s))=F} m(q) m_{(+,s),q} + \sum_{q \in Q \wedge \delta(q,(-,s))=F} m(q) m_{(-,s),q}) \ mod \ p = p'$. The obtained automaton is then uniformised and completed to obtain $\mathcal{A}_\psi$. $\square$

Our analysis relies on building automata for Boolean combinations of constant separated formulas. A Boolean combination of formulas $\varphi_1, \ldots, \varphi_n$ is a formula generated by $\top, \bot, \varphi_1, \ldots, \varphi_n, \neg, \vee, \wedge$ or $\not\leftrightarrow$. We denote by $\mathcal{C}(\varphi_1, \ldots, \varphi_n)$ such a Boolean combination. We build (on the fly) a product automaton whose states are Presburger formulas (not tuples of formulas).

Definition 4.6. *Given a Boolean combination of constant separated formulas $\mathcal{C}(\varphi_1(\boldsymbol{x}), \ldots, \varphi_n(\boldsymbol{x}))$ containing h free variables we define the product automaton $A_{\mathcal{C}(\varphi_1(\boldsymbol{x}), \ldots, \varphi_n(\boldsymbol{x}))} = (\Sigma_h \cup S_h, Q \cup \{F\}, q_0, \{F\}, \delta)$ by: Q is the set of Presburger formulas, F the designated final state, $q_0 = \mathcal{C}(\varphi_1(\boldsymbol{x}), \ldots, \varphi_n(\boldsymbol{x}))$ and for all $a \in \Sigma_h$, $\delta(\mathcal{C}(\psi_1(\boldsymbol{x}), \ldots, \psi_n(\boldsymbol{x})), a) = \mathcal{C}(\psi_1'(\boldsymbol{x}), \ldots, \psi_n'(\boldsymbol{x}))$ each $\psi_i(\boldsymbol{x})$ being a state, possibly $\bot$ (equivalent to $0 < 0$), of $\mathcal{A}_{\varphi_i}$ (the automaton of φ_i), and $\psi_i'(\boldsymbol{x}) = \delta_{\varphi_i}(\psi_i(\boldsymbol{x}), a)$. If $s \in S_h$, then $\delta(\mathcal{C}(\psi_1(\boldsymbol{x}), \ldots, \psi_n(\boldsymbol{x})),\ s) = F$, when $\langle s \rangle \in [\![\mathcal{C}(\psi_1(\boldsymbol{x}), \ldots, \psi_n(\boldsymbol{x}))]\!]$ and $\delta(\mathcal{C}(\psi_1(\boldsymbol{x}), \ldots, \psi_n(\boldsymbol{x})), s) = \bot$ otherwise.*

The following theorem gives a bound on the automata size for a formula in Presburger's logic with modulo-counting quantifiers. A corresponding theorem for classical Presburger's logic was given in [6] (using results from [13] where a most significant digit first encoding is used). Its proof is basically the same, as we can also eliminate all quantifiers and construct an automaton from the resulting atomic formulas. We will need the construction of the automaton later to handle the $\exists^{(p',p)}$ quantifier. We use the abbreviations $exp2(x) = 2^{2^x}$ and $exp3(x) = 2^{2^{2^x}}$. Notice that in [6] the size of the DFA was bounded by $exp3(\kappa n \log n)$.

Theorem 4.7. *The size of the minimal DFA accepting solutions of a formula $\varphi(\boldsymbol{x})$ from $\mathcal{L}_P$ with h free variables and length n is at most $exp3(\kappa n)$ for some constant κ.*

Proof. Let $d < n$ be the quantifier depth of φ. Let $\gamma(\boldsymbol{x})$ be the equivalent quantifier free formula obtained from φ using Theorem 3.6. We have $\max \mathbf{P}(\gamma) \leqslant (P \cdot \max \mathbf{P}(\varphi))^{4^d}$ and $\max \textsc{Const}(\gamma) \leqslant 2^{(P \cdot \max \mathbf{P}(\varphi))^{4^d}} \cdot \max \textsc{Const}(\varphi)$. Clearly, $\max \textsc{Const}(\gamma) \leq exp3(\kappa_1 n)$ for some constant κ_1. If we build the product automaton for γ according to Definition 4.6, a naive analysis of its size gives a quadruple-exponential, as there are possibly a quadruple exponential number of distinct inequations in γ. We give a slightly different construction of the automaton A_γ accepting solutions of γ. Let $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{t_\gamma}$ be an enumeration of all different vectors $\boldsymbol{a}$ corresponding to coefficients of variables of $\boldsymbol{x} = (x_1, \ldots, x_h)$ appearing in constant separated inequations of γ. Let $\gamma_1, \ldots, \gamma_{t'_\gamma}$ be an enumeration of all atomic formulas of the form $\boldsymbol{a}_i.\boldsymbol{x} > c_j$ with $1 \leq i \leq t_\gamma$ and c_j such that $|c_j| \in [-\|\boldsymbol{a}_i\|_+ - 1, \|\boldsymbol{a}_i\|_-]$. Due to the bound on $\max \mathbf{P}(\gamma)$ we have $t'_\gamma \leqslant exp2(\kappa_2 n)$ for some constant κ_2. Let $(\boldsymbol{b}_1, k_1), \ldots, (\boldsymbol{b}_{\boldsymbol{d}_\gamma}, k_{d_\gamma})$ be an enumeration of all different vectors $\boldsymbol{b}$ corresponding to coefficients of variables of $\boldsymbol{x} = (x_1, \ldots, x_h)$ together with its modulus appearing in constant separated modulo constraints of γ. Each k_i can be written as $k_i = k'_i \cdot k''_i$ where k'_i is the biggest possible power of 2 and k''_i odd. Let $\phi_1, \ldots, \phi_{d'_\gamma}$ be an enumeration of all modulo constraints of the form $\boldsymbol{b}_i \boldsymbol{x} \equiv_{k''_i} c_j$ with $1 \leq i \leq d_\gamma$ and $c_j < k''_i$. Again due to the bound on $\max \mathbf{P}(\gamma)$ we have $d'_\gamma \leqslant exp2(\kappa_3 n)$ for some constant κ_3.

We define $\mathcal{BC}$ to be the set of all Boolean combinations having the form $\mathcal{C}(\gamma_1, \ldots, \gamma_{t'_\gamma}, \phi_1, \ldots, \phi_{d'_\gamma})$. For each member of $\mathcal{BC}$ an automaton can be built with the product construction of Definition 4.6. All these automata are the same except for transitions leading to the final and sink states.

We describe now informally the automaton A_γ which we construct from γ. It has first the form of a complete tree starting at the initial state. Its branching factor is the size of the alphabet Σ_h and its depth is $exp2(\kappa_1 n)$. Each of the states in the tree recognises the solutions of the formula $\gamma(2^{|w|}\boldsymbol{x} + \langle w \rangle_+)$ where $w \in \Sigma_h^*$ with $|w| \leq exp2(\kappa_1 n)$ is the word leading to the state from the initial state. Then, at level $exp2(\kappa_1 n)$ there are separate automata accepting solutions of the corresponding formulas reached after reading the word leading to them. All these automata correspond to Boolean combinations of $\mathcal{BC}$. Indeed, for any constant separated formula $\zeta(\boldsymbol{x}) = \boldsymbol{a}.\boldsymbol{x} > c$ of γ and any word $w \in \Sigma_h^*$ with $|w| = exp2(\kappa_1 n)$ we have $\zeta(2^{|w|}\boldsymbol{x} + \langle w \rangle_+) \Leftrightarrow \boldsymbol{a}.\boldsymbol{x} > c'$ for some $c' \in [-\|a\|_+ - 1, \|a\|_-]$. Therefore, for any atomic inequation $\zeta(\boldsymbol{x})$ of γ, $\zeta(2^{|w|}\boldsymbol{x} + \langle w\rangle_+)$ is equivalent to some γ_i. The same is true for modulo constraints, i.e. each modulo constraint reached after w is equivalent to some ϕ_i. So, $\gamma(2^{|w|}\boldsymbol{x} + \langle w\rangle_+)$ is equivalent to a formula of $\mathcal{BC}$. Notice that in any member of $\mathcal{BC}$ *all* atomic formulas of a given form appear. That is not a restriction, since we can just expand each Boolean combination to be of this form. Let $W = \{w \in \Sigma_h^* \mid |w| = exp2(\kappa_1 n)\}$. For any $w \in W$, let $\mathcal{C}_w \in \mathcal{BC}$ be the Boolean combination equivalent to $\gamma(2^{|w|}\boldsymbol{x} + \langle w\rangle_+)$. For each $\mathcal{C}_w$ we can construct an automaton $A_{\mathcal{C}_w} = (\Sigma_h \cup S_h, Q_w \cup \{F\}, q_{w,0}, \{F\}, \delta_w)$ according to Definition 4.6. Notice that the automata $A_{\mathcal{C}_w}$ only differ in the transitions going to the final state, since the atomic formulas composing them are all the same. The final state F is the same in each automaton.

We can now give the definition of the automaton for the formula γ formally, i.e. $A_\gamma = (\Sigma_h \cup S_h, Q, q_\epsilon, \{F\}, \delta)$ where $Q = Q_1 \cup Q_2 \cup \{F\}$ with $Q_1 = \{q_w \mid w \in \Sigma_h^* \wedge |w| < exp2(\kappa_1 n)\}$ and $Q_2 = \bigcup_{w\in W} Q_w$. Furthermore, $\delta(q_w, b) = \{q_{wb}\}$ for all $b \in \Sigma_h$ and $|w| < exp2(\kappa_1 n) - 1$, $\delta(q_w, b) = \{q_{wb,0}\}$ for all $b \in \Sigma_h$ and $|w| = exp2(\kappa_1 n) - 1$ and $\delta(q, b) = \delta_w(q, b)$ for all $b \in \Sigma_h$ and $q \in Q_2$. Clearly, the number of states (and also the size) of the automaton A_γ is smaller than $exp3(\kappa n)$ for some constant κ. $\square$

When applying the construction of Lemma 4.5 to eliminate a modulo-counting quantifier, one could have a potential exponential blow-up which could lead to a quadruple exponential automaton. We can show that this is not the case by analysing the structure of the constructed automaton (similarly as in [6] for the existential quantifier) and obtain the following theorem.

Theorem 4.8. *Let $\exists y^{(p',p)} : \varphi(y, \boldsymbol{x})$ be a formula from $\mathcal{L}_P$ of size n, A the minimal DFA accepting the uniform Presburger definable language corresponding to $\varphi(y, \boldsymbol{x})$ and A' the automaton obtained for $\exists y^{(p',p)} : \varphi(y, \boldsymbol{x})$ using the construction of Lemma 4.5. Then A' is of size at most $exp3(\kappa n)$ for some constant κ.*

Corollary 4.9. *The automata based decision procedure for Presburger arithmetic with modulo-counting quantifiers takes triple-exponential time in the size of the formula.*

In [5] the complexity of the automata based construction for Presburger's logic is analysed using Ehrenfeucht-Fraïssé relations. There a most significant bit first encoding is used. An open question is to know if this approach can be also applied for modulo-counting quantifiers.

References

1. Apelt, H.: Axiomatische Untersuchungen über einige mit der Presburgerschen Arithmetik verwandten Systeme. Z. Math. Logik Grundlagen Math. 12, 131–168 (1966)
2. Berman, L.: The complexity of logical theories. Theor. Comput. Sci. 11, 71–77 (1980)

3. Blumensath, A., Grädel, E.: Finite presentations of infinite structures: Automata and interpretations. Theory of Computing Systems 37(6), 641–674 (2004)
4. Cobham, A.: On the base-dependence of sets of numbers recognizable by finite automata. Mathematical Systems Theory 3(2), 186–192 (1969)
5. Durand-Gasselin, A., Habermehl, P.: Ehrenfeucht-Fraïssé goes elementarily automatic for structures of bounded degree. In: STACS 2012, pp. 242–253. Dagstuhl Publishing (2012)
6. Durand-Gasselin, A., Habermehl, P.: On the use of non-deterministic automata for Presburger arithmetic. In: Gastin, P., Laroussinie, F. (eds.) CONCUR 2010. LNCS, vol. 6269, pp. 373–387. Springer, Heidelberg (2010)
7. Ferrante, J., Rackoff, C.W.: The computational complexity of logical theories.. Lecture Notes in Mathematics, vol. 718, 243 p. Springer, Heidelberg (1979)
8. Grädel, E.: Subclasses of Presburger arithmetic and the polynomial-time hierarchy. Theor. Comput. Sci. 56, 289–301 (1988)
9. Haase, C.: Subclasses of Presburger arithmetic and the weak EXP hierarchy. In: CSL-LICS 2014, paper no. 47, 10 pages. ACM (2014)
10. Hodgson, B.R.: On direct products of automaton decidable theories. Theoretical Computer Science 19, 331–335 (1982)
11. Khoussainov, B., Nerode, A.: Automatic presentations of structures. In: Leivant, D. (ed.) LCC 1994. LNCS, vol. 960, pp. 367–392. Springer, Heidelberg (1995)
12. Khoussainov, B., Rubin, S., Stephan, F.: Definability and regularity in automatic structures. In: Diekert, V., Habib, M. (eds.) STACS 2004. LNCS, vol. 2996, pp. 440–451. Springer, Heidelberg (2004)
13. Klaedtke, F.: Bounds on the Automata Size for Presburger Arithmetic. ACM Trans. Comput. Logic 9(2), 1–34 (2008)
14. Kuske, D., Lohrey, M.: Automatic structures of bounded degree revisited. Journal of Symbolic Logic 76(4), 1352–1380 (2011)
15. Leroux, J.: Structural Presburger Digit Vector Automata. Theoretical Computer Science 409(3), 549–556 (2008)
16. 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)
17. Presburger, M.: Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In: Sprawozdanie z 1 Kongresu Matematyków Krajow Slowiańskich, Ksiaznica Atlas, pp. 92–101. Warzaw (1930)
18. Reddy, C.R., Loveland, D.W.: Presburger Arithmetic with Bounded Quantifier Alternation. In: ACM Symposium on Theory of Computing, pp. 320–325 (1978)
19. Rubin, S.: Automata presenting structures: A survey of the finite string case. Bulletin of Symbolic Logic 14, 169–209 (2008)
20. Schöning, U.: Complexity of Presburger arithmetic with fixed quantifier dimension. Theory Comput. Syst. 30(4), 423–428 (1997)
21. Schweikardt, N.: Arithmetic, first-order logic, and counting quantifiers. ACM Trans. Comput. Log. 6(3), 634–671 (2005)
22. Semenov, A.L.: Presburgerness of predicates regular in two number systems. Sib. Math. J. 18, 289–300 (1977)
23. Von Zur Gathen, J., Sieveking, M.: A bound on solutions of linear integer equalities and inequalities. Proc. AMS 72, 155–158 (1978)