




# Monadic Decomposition in Integer Linear Arithmetic

Matthew Hague<sup>1</sup>  , Anthony W. Lin<sup>2</sup> , Philipp Rümmer<sup>3</sup> ,  
and Zhilin Wu<sup>4</sup>

<sup>1</sup> Royal Holloway, University of London, Egham, UK  
`matthew.hague@rhul.ac.uk`

<sup>2</sup> TU Kaiserslautern, Kaiserslautern, Germany

<sup>3</sup> Uppsala University, Uppsala, Sweden

<sup>4</sup> State Key Laboratory of Computer Science, Institute of Software,  
Chinese Academy of Sciences, Beijing, China

**Abstract.** Monadic decomposability is a notion of variable independence, which asks whether a given formula in a first-order theory is expressible as a Boolean combination of monadic predicates in the theory. Recently, Veanes et al. showed the usefulness of monadic decomposability in the context of SMT (i.e. the input formula is quantifier-free), and found various interesting applications including string analysis. However, checking monadic decomposability is undecidable in general. Decidability for certain theories is known (e.g. Presburger Arithmetic, Tarski's Real-Closed Field), but there are very few results regarding their computational complexity. In this paper, we study monadic decomposability of integer linear arithmetic in the setting of SMT. We show that this decision problem is coNP-complete and, when monadically decomposable, a formula admits a decomposition of exponential size in the worst case. We provide a new application of our results to string constraint solving with length constraints. We then extend our results to variadic decomposability, where predicates could admit multiple free variables (in contrast to monadic decomposability). Finally, we give an application to quantifier elimination in integer linear arithmetic where the variables in a block of quantifiers, if independent, could be eliminated with an exponential (instead of the standard doubly exponential) blow-up.

## 1 Introduction

A formula  $\phi(\bar{x})$  in some theory  $\mathcal{L}$  is *monadically decomposable* if it is  $\mathcal{L}$ -equivalent to a Boolean combination of monadic predicates in  $\mathcal{L}$ , i.e., to a *monadic decomposition* of  $\phi$ . Monadic decomposability measures how tightly the free variables in  $\phi$  are coupled. For example,  $x = y$  is not monadically decomposable in any (finitary) logic over an infinite domain, but  $x + y \geq 2$  can be decomposed, in Presburger arithmetic over natural numbers, since it can be written as  $x \geq 2 \vee (x \geq 1 \wedge y \geq 1) \vee y \geq 2$ .

Veanes *et al.* [24] initiated the study of monadic decomposability in the setting of Satisfiability Modulo Theories, wherein formulas are required to be

quantifier-free. Monadic decomposability has many applications, including symbolic transducers [11] and string analysis [24]. Although the problem was shown to be in general undecidable, a generic semi-algorithm for outputting monadic decompositions (if decomposable) was provided. A termination check could in fact be added if the input formula belongs to a theory for which monadic decomposability is decidable, e.g., linear arithmetic, Tarski's Real-Closed Field, and the theory of uninterpreted functions. Hitherto, not much is known about the computational complexity of monadic decomposability problems for many first-order theories (in particular, quantifier-free theories), and about practical algorithms. This was an open problem raised by Veanes *et al.* in [24].

Monadic decomposability is intimately connected to the variable partition problem, first studied by Libkin [19] nearly 20 years ago. In particular, a monadic decomposition gives rise to a partition of the free variables  $\bar{x}$  of a formula  $\phi(\bar{x})$ , wherein each part consists of a single variable. More precisely, take a partition  $\Pi = \{Y_1, \dots, Y_m\}$  of  $\bar{x}$  into sets  $Y_i$  of variables, with linearizations  $\bar{y}_i$ . The formula  $\phi(\bar{x})$  is  $\Pi$ -decomposable (in some theory  $\mathcal{L}$ ) if it is  $\mathcal{L}$ -equivalent to a boolean combination of formulas of the form  $\Delta(\bar{y}_i)$ . As suggested in [19], such *variadic decompositions* of  $\phi(\bar{x})$  have potential applications in optimization of database query processing and quantifier elimination. The author gave a general condition for the decidability of variable independence in first-order theories. This result is unfortunately not easily applicable in the SMT setting for at least two reasons: (i) the full first-order theory might be undecidable (e.g. theory of uninterpreted functions), and (ii) even for a first-order theory that admits decidable monadic decompositions, the complexity of the algorithm obtained from [19] could be too prohibitive for the quantifier-free fragment. One example that epitomizes (ii) is the problem of determining whether a given relation  $R \subseteq (\Sigma^*)^k$  over strings represented by a regular transducer could be expressed as a boolean combination of monadic predicates. The result of [19] would give a double exponential-time algorithm for monadic decomposability, whereas it was recently shown in [5] to be solvable in polynomial-time (resp. polynomial-space) when the transducer is given as a deterministic (resp. nondeterministic) machine.

*Contributions.* First, we determine the complexity of deciding monadic decomposability and outputting monadic decompositions (if they exist) for the theory of integer linear arithmetic in the setting of SMT. Our result is summarized in Theorem 1.

**Theorem 1 (Monadic Decomposability).** *Given a quantifier-free formula  $\phi$  of Presburger Arithmetic, it is coNP-complete to decide if  $\phi$  is monadically decomposable. This is efficiently reducible to unsatisfiability of quantifier-free Presburger formulas. Moreover, if a decomposition exists, it can be constructed in exponential time.*

We show a new application of monadic decomposability in integer linear arithmetic for SMT over strings, which is currently a very active research area, e.g., see [1–3, 6, 9, 12, 16, 18, 20, 22, 23]. One problem that makes string constraint solving

difficult is the presence of additional *length constraints*, which forces the lengths of the strings in the solutions to satisfy certain linear arithmetic constraints. Whereas satisfiability of string equations with regular constraints is PSPACE-complete (e.g. see [13,17]), it is a long-standing open problem [7,14] whether word equations with length constraints are decidable. Length constraints are omnipresent in Kaluza [22], arguably the first serious string constraint benchmarks obtained from real-world JavaScript applications. Using our monadic decomposability solver, we show that 90% of the Kaluza benchmarks are in fact in a decidable fragment of string constraints, since occurring length constraints can be completely removed by means of decomposition.

Next we extend our result to variadic decomposability (cf. [19]).

**Theorem 2 (Variadic Decomposability).** *It is coNP-complete to decide if  $\phi$  is  $\Pi$ -decomposable, given a quantifier-free formula  $\phi(\bar{x})$  of Presburger Arithmetic and a partition  $\Pi = \{Y_1, \dots, Y_n\}$  of  $\bar{x}$ . This is efficiently reducible to unsatisfiability of quantifier-free Presburger formulas. Moreover, if a decomposition exists, it can be constructed in exponential time.*

We show how this could be applied to quantifier elimination. In particular, we show that if a formula  $\phi(\bar{y}) = \exists \bar{x}. \psi(\bar{x}, \bar{y})$ , where  $\psi$  is quantifier-free, is  $\{X, Y\}$ -decomposable—where  $\bar{x}$  and  $\bar{y}$  are linearizations of the variables in  $X$  and  $Y$ —then we can compute in exponential time a formula  $\theta(\bar{y})$  such that  $\langle \mathbb{N}, + \rangle \models \theta \leftrightarrow \phi$ , i.e., avoiding the standard double-exponential blow-up (cf. [25]).

*Organization.* Preliminaries are in Sect. 2. Results on monadic (resp. variadic) decomposition are in Sect. 3 (resp. Sect. 4) and applications appear in Sect. 5.

## 2 Preliminaries

### 2.1 Presburger Syntax

In this paper we study the problem of monadic decomposition for formulas in linear integer arithmetic. All of our results are presented for Presburger arithmetic over natural numbers, but they can be adapted easily to all integers.

**Definition 1 (Fragments of Presburger Arithmetic).** *A formula  $\phi$  of Presburger arithmetic is a formula of the form  $\mathcal{Q}_1 x_1 \dots \mathcal{Q}_n x_n. \psi$  where  $\mathcal{Q}_i \in \{\forall, \exists\}$  and  $\psi$  is a quantifier-free Presburger formula:*

$$\psi := \sum_i a_i x_i \sim b \mid ax \equiv_k by \mid x \equiv_k c \mid \phi_1 \wedge \phi_2 \mid \neg \phi$$

where  $a_i, a, b \in \mathbb{Z}$ ,  $k, c \in \mathbb{N}$  with  $0 \leq c < k$ , variables  $x_i, x, y$  range over  $\mathbb{N}$ , and  $\sim \in \{\leq, \geq\}$ . The operator  $\equiv_k$  denotes equality modulo  $k$ , i.e.,  $s \equiv_k t$  whenever  $s - t$  is a multiple of  $k$ . Formulas of the shape  $\sum_i a_i x_i \sim b$ ,  $ax \equiv_k by$ , or  $x \equiv_k c$  are called atoms.

Existential Presburger formulas are formulas of the form  $\exists x_1, \dots, x_n. \psi$  for some quantifier-free Presburger formula  $\psi$ . We let  $QF(\mathbb{N})$  (resp.  $\exists^*(\mathbb{N})$ ) denote the set of all quantifier-free (resp., existential) Presburger formulas.

Let  $\bar{x} = (x_1, \dots, x_n)$  be a tuple of integer variables. We write  $f(\bar{x}) = \sum_i a_i x_i$  for a linear sum over  $\bar{x}$ . Let  $\bar{y} = (y_1, \dots, y_m)$ . By slight abuse of notation, we may also write  $\phi(\bar{x}, \bar{y})$  to denote a  $\text{QF}(\mathbb{N})$  formula over the variables  $x_1, \dots, x_n, y_1, \dots, y_m$ .

## 2.2 Monadic Decomposability

A quantifier-free formula  $\phi$  is called *monadic* if every atom in  $\phi$  contains at most one variable, and it is called *monadically decomposable* if  $\phi$  is equivalent to a monadic formula  $\phi'$ . In this case,  $\phi'$  is also called a *decomposition* of  $\phi$ . For our main results we use a slightly refined notion of a formula being decomposable:

**Definition 2 (Monadically Decomposable on  $x$ ).** Fix a logic  $\mathcal{L}$  (e.g.  $\text{QF}(\mathbb{N})$  or  $\exists^*(\mathbb{N})$ ). We say a formula  $\phi(x_1, \dots, x_n)$  in  $\mathcal{L}$  is monadically decomposable on  $x_i$  whenever

$$\phi(x_1, \dots, x_n) \equiv \bigvee_j \Delta_j(x_i) \wedge \psi_j(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$$

for some formulas  $\Delta_j$  and  $\psi_j$  in  $\mathcal{L}$ .

It can be observed that a formula is monadically decomposable if and only if it is monadically decomposable on all variables occurring in the formula (cf. Lemma 1). We expand on this for the variadic case below.

We recall the following characterization of monadic decomposability for formulas  $\phi(x, y)$  with two free variables (cf. [5, 8, 19, 24]), which holds regardless of the theory under consideration. This can be extended easily to formulas with  $k$  variables, but is not needed in this paper. Given a formula  $\phi(x, y)$ , define the formula  $\sim$  as follows:

$$x \sim x' := \forall y, y'. (\phi(x, y) \wedge \phi(x', y') \rightarrow (\phi(x', y) \wedge \phi(x, y')))$$

**Proposition 1.** *The relation  $\sim$  is an equivalence relation. Furthermore,  $\phi(x, y)$  is monadically decomposable iff  $\sim$  has a finite index (i.e. the number of  $\sim$ -equivalence classes is finite).*

Using this proposition, it is easy to show that over a structure with an infinite domain (e.g. integer linear arithmetic) the formula  $x = y$  is not monadically decomposable. As was noted already in [19], to check monadic decomposability of a formula  $\phi$  in Presburger Arithmetic in general, we may simply check if there is an upper bound  $B$  on the smallest representation of every  $\sim$ -equivalence class, i.e.,

$$\exists B. \forall x. \exists x_s. (x_s \leq B \wedge x_s \sim x).$$

However, to derive tight complexity bounds for checking monadic decomposability, this approach is problematic, since the above characterisation has multiple quantifier alternations. Using known results (e.g. [15]), one would only obtain an upper bound in the weak exponential hierarchy [15], which only admits double-exponential time algorithms.

### 2.3 Variadic Decomposability

The notion of a *variadic decomposition* generalises monadic decomposition by considering partitions of the occurring variables.

**Definition 3 ( *$\Pi$ -Decomposable*).** Fix a logic  $\mathcal{L}$  (e.g.  $QF(\mathbb{N})$  or  $\exists^*(\mathbb{N})$ ). Take a formula  $\phi(x_1, \dots, x_n)$  in  $\mathcal{L}$  and a partition  $\Pi = \{Y_1, \dots, Y_m\}$  of  $x_1, \dots, x_n$ . We say  $\phi$  is  $\Pi$ -decomposable whenever

$$\phi(x_1, \dots, x_n) \equiv \bigvee_i \Delta_i^1(\bar{y}_1) \wedge \dots \wedge \Delta_i^m(\bar{y}_m)$$

for some formulas  $\Delta_i^j$  in  $\mathcal{L}$  and linearizations  $\bar{y}_j$  of  $Y_j$ .

Observe that a formula  $\phi(x_1, \dots, x_n)$  is monadically decomposable on  $x_i$  iff it is  $\Pi$ -decomposable with  $\Pi = \{\{x_i\}, \{x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n\}\}$ . Moreover, we say a formula  $\phi$  over the set of variables  $X$  is *variadic decomposable on  $Y$*  whenever it is  $\Pi$ -decomposable with  $\Pi = \{Y, X \setminus Y\}$ .

General  $\Pi$ -decompositions can be computed by decomposing on binary partitions  $\{Y, X \setminus Y\}$ , which is why we focus on this binary case in the rest of the paper. We argue why this is the case below.

Let a formula  $\phi$  and  $\Pi = \{Y_1, \dots, Y_m\}$  be given. We can first decompose separately on each  $\{Y_i, Y\}$  where  $Y = Y_1 \cup \dots \cup Y_{i-1} \cup Y_{i+1} \cup \dots \cup Y_m$ . Using the algorithm in Sect. 4 we obtain for each  $i$  a decomposition of a specific form:

$$\bigvee_j \Delta_j^i(\bar{y}_i) \wedge \phi(\bar{y}_1, \dots, \bar{y}_{i-1}, \bar{c}_j^i, \bar{y}_{i+1}, \dots, \bar{y}_m).$$

Note, these decompositions can be performed independently using the algorithm in Sect. 4 and the second conjunct of each disjunct is  $\phi$  with  $y_i$  replaced by fixed constants  $\bar{c}_j$ . Additionally, each  $\Delta_j^i$  is polynomial in size and each  $\bar{c}_j^i$  can be represented with polynomially many bits. We note also that our algorithm ensures that each  $\Delta_j^i$  is satisfiable.

Given such decompositions, we can recursively decompose  $\phi$  on  $\Pi$ . We first use the above decomposition for  $i = 1$  and obtain

$$\bigvee_j \Delta_j^1(\bar{y}_1) \wedge \phi(\bar{c}_j^1, \bar{y}_2, \dots, \bar{y}_m).$$

Next, we use the decomposition for  $i = 2$  to decompose the copies of  $\phi$  in the decomposition above. We obtain

$$\bigvee_{j_1} \left( \Delta_{j_1}^1(\bar{y}_1) \wedge \bigvee_{j_2} \Delta_{j_2}^2(\bar{y}_2) \wedge \phi(\bar{c}_{j_1}^1, \bar{c}_{j_2}^2, \bar{y}_3, \dots, \bar{y}_m) \right).$$

This process repeats until all  $Y_i$  have been considered. If  $\phi$  is  $\Pi$ -decomposable, we find a decomposition. If  $\phi$  is not  $\Pi$ -decomposable, then it

would not be possible to do the independent decompositions for each  $i$ . Thus, for  $\Pi = \{Y_1, \dots, Y_m\}$ , we can use variadic decompositions on  $Y_i$  to compute  $\Pi$ -decompositions.

The above algorithm runs in exponential time due both to the exponential size of the decompositions and the branching caused by the disjuncts. If we are only interested in whether a formula is  $\Pi$ -decomposable, it is enough to ask whether it is decomposable on  $Y_i$  for *each*  $i$ . In particular, a formula  $\phi(\bar{x})$  is monadically decomposable iff  $\phi$  is decomposable for each variable  $y \in \bar{x}$ . Since the complexity class coNP is closed under intersection, we obtain the following:

**Lemma 1.** *A coNP upper bound for monadic decomposability on a given variable  $y$  implies a coNP upper bound for monadic decomposability. Likewise, a coNP upper bound for variadic decomposability on a given subset  $Y$  of variables implies a coNP upper bound for  $\Pi$ -decomposability.*

### 2.4 Example

Consider the formula  $\phi(x, y, z)$  given by  $z = x + 2y \wedge z < 5$ . This formula is monadically decomposable, which means, it is  $\Pi$ -decomposable for  $\Pi = \{\{x\}, \{y\}, \{z\}\}$ .

Our algorithm will first take a decomposition on  $x$  and might obtain  $\bigvee_{i=0}^4 \Delta_i^1(x) \wedge \phi(i, y, z)$  where  $\Delta_i^1(x) = (x = i)$  and  $\phi(i, y, z) = (z = i + 2y) \wedge z < 5$ . Next, we use a decomposition on  $y$ . For each  $\phi(i, y, z)$  we substitute  $\bigvee_{j=0}^{2-\lceil \frac{i}{2} \rceil} y = j \wedge \phi(i, j, z)$ , and as the final decomposition we get

$$\bigvee_{i=0}^4 \bigvee_{j=0}^{2-\lceil \frac{i}{2} \rceil} x = i \wedge y = j \wedge z = i + 2j.$$

## 3 Monadic Decomposability

### 3.1 Lower Bounds

We first show that unsatisfiability of Boolean formulas can be reduced to monadic decomposability of formulas with only two variables, directly implying coNP-hardness:

**Lemma 2 (coNP-Hardness).** *Deciding whether a formula  $\phi(x, y)$  in  $QF(\mathbb{N})$  is monadic decomposable is coNP-hard.*

*Proof.* We reduce from unsatisfiability of propositional formulas to monadic decomposability of  $\phi(x, y)$ . Take a propositional formula  $S(x_1, \dots, x_n)$ . Let  $p_1, \dots, p_n$  be the first  $n$  primes. Let  $\psi(x)$  be the formula obtained from  $S$  by replacing each occurrence of  $x_i$  by  $x \equiv_{p_i} 0$ . Given an assignment  $\nu : \{x_1, \dots, x_n\} \rightarrow \{0, 1\}$ , we let

$$H_\nu = \{m \in \mathbb{N} \mid \forall 1 \leq i \leq n. (m \equiv_{p_i} 0 \leftrightarrow \nu(x_i) = 1)\}.$$

Thanks to the Chinese Remainder Theorem,  $H_\nu$  is non-empty and periodic with period  $p = \prod_{i=1}^n p_i$ , which implies that  $H_\nu$  is infinite for every  $\nu$ . We also have that  $\nu \models S$  iff, for each  $n \in H_\nu$ ,  $\psi(n)$  is true.

Now define  $\phi(x, y) = (\psi(x) \wedge x = y)$ . If  $S$  is unsatisfiable, then  $\psi$  is unsatisfiable and so it is decomposable. Conversely, if  $S$  can be satisfied by some assignment  $\nu$ , then  $\phi(m, m)$  is true for all (infinitely many)  $m \in H_\nu$ . Since all solutions to  $\phi(x, y)$  imply that  $x = y$ , by Proposition 1 we have that  $\phi$  is not monadically decomposable.  $\square$

We next provide exponential lower bounds for decompositions in either *disjunctive normal form* (DNF) or *conjunctive normal form* (CNF). DNF has been frequently used to represent monadic decompositions by previous papers (e.g. [5, 8, 19]), and it is most suitable for applications in quantifier elimination.

**Lemma 3 (Size of Decomposition).** *There exists a family  $\{\phi_n(x, y)\}_{n \in \mathbb{N}}$  of formulas in  $QF(\mathbb{N})$  such that  $\phi_n$  grows linearly in  $n$ , while the smallest decomposition on  $x$  in DNF/CNF is exponential in  $n$ .*

*Proof.* Consider the formulas  $\phi_n(x, y) = (x + y \leq 2^n)$ . Using a binary encoding of constants, the size of the formulas is linear in  $n$ . We show that decompositions in DNF/CNF must be exponential in size.

*Disjunctive:* Suppose  $\psi_n(x, y) = \bigvee_i \psi_i^x(x) \wedge \psi_i^y(y)$  is a monadic decomposition in DNF. Each disjunct  $\psi_i^x(x) \wedge \psi_i^y(y)$ , if it is satisfiable at all, has an upper right corner  $(x_i, y_i)$  such that  $\psi_i^x(x_i) \wedge \psi_i^y(y_i)$  holds, but  $\psi_i^x(x) \wedge \psi_i^y(y) \Rightarrow x \leq x_i \wedge y \leq y_i$ . This immediately implies that exponentially many disjuncts are needed to cover the exponentially many points on the line  $x + y = 2^n$ .

*Conjunctive:* Suppose  $\psi_n(x, y)$  is a succinct monadic decomposition of  $\phi_n$  in CNF. Since  $\neg\psi_n(x, y) \equiv 2^n + 1 \leq x + y \equiv (2^n - x + 1) + (2^n - y) \leq 2^n$ , it follows that  $\neg\psi_n(2^n - x + 1, 2^n - y) \equiv (2^n - (2^n - x + 1) + 1) + (2^n - (2^n - y)) \leq 2^n \equiv x + y \leq 2^n$ . Therefore,  $\neg\psi_n(2^n - x + 1, 2^n - y)$  is a succinct decomposition of  $\phi_n$  in DNF, contradicting the lower bound for DNFs.  $\square$

### 3.2 Upper Bound

We prove Theorem 1. Following Lemma 1, it suffices to show that testing decomposability on a variable  $x$  is in coNP and that a decomposition can be computed in exponential time. Assume without loss of generality that we have  $\phi(x, \bar{y})$  where  $\bar{y} = (y_1, \dots, y_n)$ , and that we are decomposing on the first variable  $x$ .

We claim that  $\phi$  is monadically decomposable on  $x$  iff

$$\forall x_1, x_2 \geq B. \forall \bar{y}. \text{SameDiv}(x_1, x_2, \bar{y}) \Rightarrow (\phi(x_1, \bar{y}) \iff \phi(x_2, \bar{y}))$$

where  $B$  is a bound exponential in the size of  $\phi$  and SameDiv is a formula asserting that  $x_1$  and  $x_2$  satisfy the same divisibility constraints. This bound is computable in polynomial time and is described in Sect. 3.4. To define SameDiv, let Divs be the set of all divisibility constraints  $az_1 \equiv_k bz_2$  or  $z_1 \equiv_k c$  appearing (syntactically) in  $\phi$ . Assume without loss of generality that  $x$  always appears on

the left-hand side of a divisibility constraint (i.e., in the  $z_1$  position of  $az_1 \equiv_k bz_2$ ). We then define

$$\text{SameDiv}(x_1, x_2, \bar{y}) = \left( \bigwedge_{ax \equiv_k bz \in \text{Divs}} (ax_1 \equiv_k bz) \iff (ax_2 \equiv_k bz) \right) \wedge \left( \bigwedge_{x \equiv_k c \in \text{Divs}} (x_1 \equiv_k c) \iff (x_2 \equiv_k c) \right).$$

We prove the claim in the following sections and simultaneously show how to construct the decomposition. Once we have established the above, we can test non-decomposability on  $x$  by checking

$$\exists x_1, x_2 \geq B. \exists \bar{y}. \text{SameDiv}(x_1, x_2, \bar{y}) \wedge \phi(x_1, \bar{y}) \wedge \neg \phi(x_2, \bar{y})$$

which is decidable in NP. Thus we obtain a coNP decision procedure because the above formula is polynomial in the size of  $\phi$ .

**Example.** We consider some examples. First consider the formula  $x = y$  that cannot be decomposed on  $x$ . Since there are no divisibility constraints, SameDiv is simply *true*. It is straightforward to see that,  $\forall B. \exists x_1, x_2 \geq B. \exists y. \text{true} \wedge x_1 = y \wedge x_2 \neq y$ , for example by setting  $x_1 = B$ ,  $x_2 = B + 1$ , and  $y = B$ .

Now consider the monadically decomposable formula

$$\phi(x, y, z) = x + 2y \geq 5 \wedge z < 5 \wedge x \equiv_2 y.$$

In this case  $\text{SameDiv}(x_1, x_2, y, z) = (x_1 \equiv_2 y \iff x_2 \equiv_2 y)$ . We can verify

$$\forall x_1, x_2 \geq B. \forall y, z. \text{SameDiv}(x_1, x_2, y, z) \Rightarrow (\phi(x_1, y, z) \iff \phi(x_2, y, z))$$

holds, as it will be the case that  $5 < B$  and for all  $x > 5$  the formula  $\phi$  will hold whenever  $x \equiv_2 y$  holds and  $z < 5$ . The precondition SameDiv ensures that the if and only if holds. We will construct the decomposition in the next section.

**Expanded Divisibility Constraints.** Observe that divisibility constraints are always decomposable. In particular,  $az_1 \equiv_k bz_2$  is equivalent to a finite disjunction of clauses  $z_1 \equiv_{k'} c \wedge z_2 \equiv_{k'} c$  where  $k'$  and  $c$  are bounded by a multiple of  $a, b$  and  $k$ . The expansion is exponential in size, since the values up to  $k'$  have to be enumerated explicitly.

We define XDivs be the set of all constraints of the form  $x \equiv_k c'$  where  $0 \leq c' < k$  and  $x \equiv_k c$  appears directly in  $\phi$  or in the expansion of the divisibility constraints of  $\phi$ . This set will be used in the next sections.

### 3.3 Soundness

We show that if

$$\forall x_1, x_2 \geq B. \forall \bar{y}. \text{SameDiv}(x_1, x_2, \bar{y}) \Rightarrow (\phi(x_1, \bar{y}) \iff \phi(x_2, \bar{y}))$$



then  $\phi$  is decomposable on  $x$ . We do this by constructing the decomposition.

Although there are doubly exponentially many subsets  $D \subseteq \text{XDivs}$ , there are only exponentially many *maximal consistent* subsets. We implicitly restrict  $D$  to such subsets. This is because, for any  $k$ , there is no value of  $x$  such that  $x \equiv_k c$  and  $x \equiv_k c'$  both hold with  $c \neq c'$  but  $c, c' \in \{0, \dots, k-1\}$ . For any maximal consistent set  $D \subseteq \text{XDivs}$ , let  $c_D$  be the smallest integer greater than or equal to  $B$  satisfying all constraints in  $D$ . Note, since  $D$  is maximal, a value that satisfies all constraints in  $D$  also does not satisfy an constraints not in  $D$ . The number  $c_D$  can be represented using polynomially many bits.

We can now decompose  $\phi$  into

$$\left( \begin{array}{c} (x = 0 \wedge \phi(0, \bar{y})) \\ \vee \dots \vee \\ (x = B - 1 \wedge \phi(B - 1, \bar{y})) \end{array} \right) \vee \bigvee_{D \subseteq \text{XDivs}} \left( x \geq B \wedge \bigwedge_{x \equiv_k c \in D} x \equiv_k c \wedge \phi(c_D, \bar{y}) \right).$$

This formula is exponential in the size of  $\phi$  if  $D$  only ranges over the maximal consistent subsets of  $\text{XDivs}$ . For values of  $x$  less than  $B$ , equivalence with the original formula is immediate. For larger values, we use the fact that, from our original assumption, for any values  $x_1$  and  $x_2$  that satisfy the same divisibility constraints, we have  $\phi(x_1, \bar{y})$  iff  $\phi(x_2, \bar{y})$ . Hence, we can substitute the values  $c_D$  in these cases.

**Example.** We return to  $\phi(x, y, z) = x + 2y \geq 5 \wedge z < 5 \wedge x \equiv_2 y$  and compute the decomposition on  $x$ . Assuming  $B$  is odd, the decomposition will be as follows. In our presentation we slightly simplify the formula. Strictly speaking  $x \equiv_2 y$  should be expanded to  $(x \equiv_2 0 \wedge y \equiv_2 0) \vee (x \equiv_2 1 \wedge y \equiv_2 1)$ . We simplify these to  $y \equiv_2 0$  and  $y \equiv_2 1$ , respectively, when instantiated with concrete values of  $x$ .

$$\begin{aligned} & (x = 0 \wedge (0 + 2y \geq 5 \wedge z < 5 \wedge y \equiv_2 0)) \vee \\ & (x = 1 \wedge (1 + 2y \geq 5 \wedge z < 5 \wedge y \equiv_2 1)) \\ & \quad \vee \dots \vee \\ & (x = B - 1 \wedge (B - 1 + 2y \geq 5 \wedge z < 5 \wedge y \equiv_2 0)) \vee \\ & ((x \equiv_2 0 \wedge x \geq B) \wedge (B + 1 + 2y \geq 5 \wedge z < 5 \wedge y \equiv_2 0)) \vee \\ & ((x \equiv_2 1 \wedge x \geq B) \wedge (B + 2y \geq 5 \wedge z < 5 \wedge y \equiv_2 1)) \end{aligned}$$

### 3.4 Completeness

We now show that every formula  $\phi$  decomposable on  $x$  satisfies

$$\forall x_1, x_2 \geq B. \forall \bar{y}. \text{SameDiv}(x_1, x_2, \bar{y}) \Rightarrow (\phi(x_1, \bar{y}) \iff \phi(x_2, \bar{y})).$$

We first show that some  $B$  must exist. Once the existence has been established, we can argue that it must be at most exponential in  $\phi$ .

**Existence of the Bound.** If  $\phi(x, \bar{y})$  is decomposable on  $x$ , then there is an equivalent formula  $\bigvee_i \Delta_i(x) \wedge \psi_i(\bar{y})$ . It is known that every formula  $\Delta(x)$  is

satisfied by a finite union of arithmetic progressions  $a + jb$ . Let  $B$  be larger than the largest value of  $a$  in the arithmetic progressions satisfying the  $\Delta_i(x)$ .

We show when  $\text{SameDiv}(x_1, x_2, \bar{y})$  then  $\phi(x_1, \bar{y})$  iff  $\phi(x_2, \bar{y})$  for all values  $x_1, x_2 \geq B$  and  $\bar{y}$ . Assume towards a contradiction that we have values  $x_1, x_2$  and a tuple of values  $\bar{y}$  such that  $\text{SameDiv}(x_1, x_2, \bar{y})$  and  $\phi(x_1, \bar{y})$ , but not  $\phi(x_2, \bar{y})$ .

Let  $k$  be the product of all  $k'$  appearing in some divisibility constraint  $x \equiv_{k'} c$  in  $\text{XDivs}$ . We know that there is some disjunct of the monadic decomposition such that  $\Delta(x_1) \wedge \psi(\bar{y})$  holds. Moreover, let  $x_1$  belong to the arithmetic progression  $a + jb$ . Since  $x_1 \geq B > a$  we know that  $\Delta(x'_1) \wedge \psi(\bar{y})$  also holds for any  $x'_1 = x_1 + j'bk$ . That is, we can pump  $x_1$  by adding a multiple of  $bk$ , while staying in the same arithmetic progression and satisfying the same divisibility constraints.

Similarly, let  $d$  be the product of all  $b$  appearing in the (finite number of) arithmetic progressions that define the monadic decomposition of  $\phi$ , limited to disjuncts such that  $\psi_i(\bar{y})$  holds. Since  $\phi(x_2, \bar{y})$  does not hold, then  $\phi(x'_2, \bar{y})$  also does not hold for any  $x'_2 = x_2 + jdk$ . This means that we can pump  $x_2$  staying outside of the arithmetic progressions defining permissible values of  $x$  for the given values  $\bar{y}$ , whilst additionally satisfying the same divisibility constraints.

Now, for each value of  $x'_1$  satisfying  $\phi(x'_1, \bar{y})$  we can consider the disjunctive normal form of  $\phi$ . By expanding the divisibility constraints, a disjunct becomes a conjunction of terms of the form, where  $f$  represents some linear function on  $\bar{y}$ ,

1.  $ax + f(\bar{y}) \leq c$  or  $ax + f(\bar{y}) \geq c$ , or
2.  $y_i \equiv_{k'} c$  or  $x \equiv_{k'} c$ .

Since there are infinitely many  $x'_1$ , we can choose one disjunct satisfied by infinitely many  $x'_1$ . This means that for constraints of the form  $ax + f(\bar{y}) \leq c$  or  $ax + f(\bar{y}) \geq c$  with a non-zero  $a$ , then  $a$  must be negative or positive respectively (or zero). Otherwise, only a finite number of values of  $x$  would be permitted.

We know that  $x'_2$  and  $\bar{y}$  do not satisfy the disjunct. We argue that this is a contradiction by considering each term in turn. Since there are infinitely many  $x'_2$  we can assume without loss of generality that  $x'_2 > x'_1$ .

1. If  $ax + f(\bar{y}) \leq c$  (resp.  $ax + f(\bar{y}) \geq c$ ) appears and is satisfied by  $x'_1$ , then  $a$  must be negative or zero (resp. positive or zero) and  $x'_2$  will also satisfy the atom.
2. Atoms of the form  $y_i \equiv_{k'} c$  do not distinguish values of  $x$  and thus are satisfied for both  $x'_1$  and  $x'_2$ . We cannot have  $x'_1 \equiv_{k'} c$  but not  $x'_2 \equiv_{k'} c$  since  $x'_1$  and  $x'_2$  satisfy the same divisibility constraints.

Thus, it cannot be the case that  $x'_1$  satisfies the disjunct, while  $x'_2$  does not. This is our required contradiction. Hence, for all  $x_1, x_2 \geq B$  and  $\bar{y}$  such that  $\text{SameDiv}(x_1, x_2, \bar{y})$  it must be the case that  $\phi(x_1, \bar{y})$  iff  $\phi(x_2, \bar{y})$ . We have thus established the existence of a bound  $B$ .

**Size of the Bound.** We now argue that this bound is exponential in the size of  $\phi$ , and can thus be encoded in a polynomial number of bits.

Consider the formula that is essentially the negation of our property.

$$\chi(x_1, x_2, \bar{y}) = \text{SameDiv}(x_1, x_2) \wedge \phi(x_1, \bar{y}) \wedge \neg\phi(x_2, \bar{y}).$$

There is some computable bound  $B'$  exponential in the size of  $\chi$  (and thus  $\phi$ ) such that, if there exists  $x_1, x_2 \geq B'$  and some  $\bar{y}$  such that  $\chi(x_1, x_2, \bar{y})$  holds, then there are infinitely many  $x'_1$  and  $x'_2$  such that for some  $\bar{y}'$  we have that  $\chi(x'_1, x'_2, \bar{y}')$  holds. An argument for the existence of this bound is given in the full version. In short, we first convert the formula above into a disjunction of conjunctions of linear equalities, using a linear number of slack variables to encode inequalities and divisibility constraints. Then, using a result of Chistikov and Haase [10], we set  $B' = 2^{dnm+3}$  where  $d$  is the number of bits needed to encode the largest constant in the converted formula (polynomially related to the size of the formula above),  $n$  is the maximum number of linear equalities in any disjunct, and  $m$  is the number of variables (including slack variables).

Now, assume that the smallest  $B$  is larger than  $B'$ . That is

$$\forall x_1, x_2 \geq B. \forall \bar{y}. \text{SameDiv}(x_1, x_2, \bar{y}) \Rightarrow (\phi(x_1, \bar{y}) \iff \phi(x_2, \bar{y}))$$

holds, but it does not hold that

$$\forall x_1, x_2 \geq B'. \forall \bar{y}. \text{SameDiv}(x_1, x_2, \bar{y}) \Rightarrow (\phi(x_1, \bar{y}) \iff \phi(x_2, \bar{y}))$$

This implies there exists some  $x_1, x_2 \geq B'$  and  $\bar{y}$  such that  $\chi(x_1, x_2, \bar{y})$  holds. Thus, there are infinitely many such  $x'_1$  and  $x'_2$ , contradicting the fact that all  $x'_1, x'_2 \geq B$  do not satisfy the property. Thus, we take  $B'$  as the value of  $B$ . It is computable in polynomial time, exponential in size, and representable in a polynomial number of bits.

## 4 Variadic Decomposability

We consider decomposition along several variables instead of just one. In this section, we assume without loss of generality that  $\phi$  is given in positive normal form and all (in)equalities rearranged into the form  $\sum_i a_i x_i \geq b$ . We may use negation  $\neg\phi$  as a shorthand. We require this form because later we use the set of all linear equations in the DNF of a formula. Since negation alters the linear equations, it is more convenient to assume that negation has already been eliminated.

### 4.1 $\Pi$ -Decomposability

As described in Sect. 2.3, we refine the notion of  $\Pi$ -decomposability to separate only a single set  $Y_i$  in  $\Pi = \{Y_1, \dots, Y_n\}$ . Without loss of generality, we assume we are given a formula  $\phi(\bar{x}, \bar{y})$  and we separate the variables in  $\bar{x}$  from  $\bar{y}$ .

In particular, given a formula  $\phi(\bar{x}, \bar{y})$  we aim to decompose the formula into  $\phi(\bar{x}, \bar{y}) \equiv \bigvee_j \Delta_j(\bar{x}) \wedge \psi_j(\bar{y})$  for some QF( $\mathbb{N}$ ) formulas  $\Delta_i$  and  $\psi_i$ .

### 4.2 Decomposition

We show that testing whether a given formula  $\phi$  is variadic decomposable on  $\bar{x}$  is in coNP. This proves Theorem 2 as the coNP lower bound follows from the monadic case.

**Lemma 4 (Decomposing on  $\bar{x}$ ).** *Given a  $QF(\mathbb{N})$  formula  $\phi(\bar{x}, \bar{y})$  there is a coNP algorithm to decide if  $\phi$  is variadic decomposable on  $\bar{x}$ . Moreover, if a decomposition exists, it can be constructed in exponential-time and is exponential in size.*

Let  $F$  be the set of all  $f$  such that  $f(\bar{x}) + g(\bar{y}) \geq b$  is a linear inequality appearing in  $\phi$ . Our approach will divide the points of  $\bar{x}$  into regions where all points within a region can be paired with the same values of  $\bar{y}$  to satisfy the formula. These regions are given by a bound  $B$ . If  $f(\bar{x})$  is within the bound, then two points  $\bar{x}_1$  and  $\bar{x}_2$  are in the same region if  $f(\bar{x}_1) = f(\bar{x}_2)$ . If two points are outside the bound, then by a pumping argument we can show that we have  $\phi(\bar{x}_1, \bar{y})$  iff  $\phi(\bar{x}_2, \bar{y})$ .

Let  $\hat{r} = (\text{UB}, \text{EQ})$  be a partition of  $F$  into unbounded and bounded functions (where EQ refers to equality being asserted over bounded functions as shown below). Define for each  $\hat{r} = (\text{UB}, \text{EQ})$

$$\text{Region}_{\hat{r}}(\bar{x}_1, \bar{x}_2) \triangleq \left( \bigwedge_{f \in \text{EQ}} f(\bar{x}_1) = f(\bar{x}_2) \right).$$

Note, this formula intentionally does not say anything about the unbounded functions. This is important when we need to derive a bound—such a derivation cannot use a pre-existing bound.

We also need to extend SameDiv to account for  $\bar{x}_1$  and  $\bar{x}_2$  being vectors. This is a straightforward extension asserting that each variable in  $\bar{x}_1$  satisfies the same divisibility constraints as its counterpart in  $\bar{x}_2$ . Again, let Divs be the set of all divisibility constraints  $az_1 \equiv_k bz_2$  appearing (syntactically) in  $\phi$ . Let  $x_i$ ,  $x_i^1$  and  $x_i^2$  denote the  $i$ th variable of  $\bar{x}$ ,  $\bar{x}_1$ , and  $\bar{x}_2$  respectively. Assume without loss of generality that variables in  $\bar{x}$  always either appear on the left-hand side of a divisibility constraint (i.e. in the  $z_1$  position) or on both sides. Define

$$\text{SameDiv}(x_1, x_2, \bar{y}) = \bigwedge_{\substack{ax_i \equiv_k bz \in \text{Divs}, \\ z \neq x_j}} \left( \begin{array}{c} (ax_i^1 \equiv_k bz) \\ \iff \\ (ax_i^2 \equiv_k bz) \end{array} \right) \wedge \bigwedge_{\substack{ax_i \equiv_k bx_j \\ \in \text{Divs}}} \left( \begin{array}{c} (ax_i^1 \equiv_k bx_j^1) \\ \iff \\ (ax_i^2 \equiv_k bx_j^2) \end{array} \right) \wedge \bigwedge_{\substack{x_i \equiv_k c \\ \in \text{Divs}}} \left( \begin{array}{c} (x_i^1 \equiv_k c) \\ \iff \\ (x_i^2 \equiv_k c) \end{array} \right).$$

Next, we introduce an operator for comparing a vector of variables with a bound. For  $a \in \mathbb{Z}$  let  $\text{abs}(a)$  denote the absolute value of  $a$ . Given a bound  $B$  and some  $\hat{r} = (\text{UB}, \text{EQ})$  let

$$(\bar{x} \geq_{\hat{r}} B) \triangleq \bigwedge_{f \in \text{UB}} \text{abs}(f(\bar{x})) \geq B \wedge \bigwedge_{f \in \text{EQ}} \text{abs}(f(\bar{x})) < B.$$

We claim there is an exponential bound  $B$  such that  $\phi$  is variadic decomposable iff for all  $\hat{r}$  we have

$$\forall \bar{x}_1, \bar{x}_2 \geq_{\hat{r}} B. \forall \bar{y}. \left( \begin{array}{c} \text{Region}_{\hat{r}}(\bar{x}_1, \bar{x}_2) \\ \wedge \\ \text{SameDiv}(\bar{x}_1, \bar{x}_2, \bar{y}) \end{array} \right) \Rightarrow \left( \begin{array}{c} \phi(\bar{x}_1, \bar{y}) \\ \iff \\ \phi(\bar{x}_2, \bar{y}) \end{array} \right) \quad (\text{DC-}\hat{r})$$

Note, unsatisfiability can be tested in NP. First guess  $\hat{r}$ , then guess  $\bar{x}_1, \bar{x}_2, \bar{y}$ .

We prove soundness of the claim in the next section. Completeness is an extension of the argument for the monadic case and is given in the full version. In the monadic case, we were able to take some values of  $x_1, x_2 > B$  such that both satisfied the same divisibility constraints, but one value satisfied the formula while the other did not. Since these values were large, we derived an infinite number of such value pairs with increasing values. We then used these growing solutions to show that it was impossible for the value of  $x_1$  to satisfy the formula, while the value of  $x_2$  does not, as they were both beyond the distinguishing power of the linear inequalities. The argument for the variadic case is similar, with the values of  $x_1$  and  $x_2$  being replaced by the values of  $f(\bar{x}_1)$  and  $f(\bar{x}_2)$ .

### 4.3 Soundness

Assume there is an exponential bound  $B$  such that for each  $\hat{r}$ , Equation DC- $\hat{r}$  holds. We show how to produce a decomposition.

As in the monadic case (Sect. 3.3), let XDivs be the set of all constraints of the form  $\bar{x}_i \equiv_k c$  in the expansion of the divisibility constraints of  $\phi$ . Observe again that there are only exponentially many maximal consistent subsets  $D \subseteq \text{XDivs}$ . For each  $D$  fix a vector of values  $\bar{c}_D$  that satisfies all constraints in  $D$  and is encodable in a polynomial number of bits. Furthermore, we define

$$\text{Div}_D(\bar{z}) \triangleq \bigwedge_{x_i \equiv_k c \in D} z_i \equiv_k c.$$

For each  $\hat{r}$  and  $D$  we can define an equivalence relation over values of  $\bar{x}$  such that  $\bar{x} \geq_{\hat{r}} B$  and  $\text{Div}_D(\bar{x})$ .

$$(\bar{x}_1 \stackrel{D}{=} \bar{x}_2) \triangleq (\bar{x}_1 \geq_{\hat{r}} B \wedge \bar{x}_2 \geq_{\hat{r}} B \wedge \text{Region}_{\hat{r}}(\bar{x}_1, \bar{x}_2) \wedge \text{Div}_D(\bar{x}_1) \wedge \text{Div}_D(\bar{x}_2)).$$

Observe each equivalence relation has an exponential number of equivalence classes depending on the values of the bounded  $f$ . Let  $C_{\hat{r}}^D$  be a set of minimal representatives from each equivalence class such that each representative is representable in a polynomial number of bits. These can be computed by solving an existential Presburger constraint for each set of values of the bounded  $f$ . In particular, for each  $\hat{r} = (\text{UB}, \text{EQ})$  and assignments  $\text{abs}(c_f) < B$  for each  $f \in \text{EQ}$ , we select a solution to the equation

$$\bar{x} \geq_{\hat{r}} B \wedge \bigwedge_{f \in \text{EQ}} f(\bar{x}) = c_f \wedge \text{Div}_D(\bar{x})$$

if such a solution exists. If no such solution exists, the assignment can be ignored.

The decomposition is

$$\bigvee_{\hat{r}} \bigvee_D \bigvee_{\bar{c} \in C_{\hat{r}}^D} (\bar{x} \geq_{\hat{r}} B \wedge \text{Region}_{\hat{r}}(\bar{x}, \bar{c}) \wedge \text{Div}_D(\bar{x}) \wedge \phi(\bar{c}, \bar{y})).$$

The correctness of this decomposition follows from the Equations DC- $\hat{r}$ . For any values  $\bar{c}_{\bar{x}}$  and  $\bar{c}_{\bar{y}}$  of  $\bar{x}$  and  $\bar{y}$ , first assume  $\phi(\bar{c}_{\bar{x}}, \bar{c}_{\bar{y}})$  holds. Since there is some disjunct in the decomposition for which it holds that  $\bar{c}_{\bar{x}} \geq_{\hat{r}} B \wedge \text{Region}_{\hat{r}}(\bar{c}_{\bar{x}}, \bar{c}) \wedge \text{Div}_D(\bar{x})$  then, by applying Equation DC- $\hat{r}$  we get  $\phi(\bar{c}, \bar{c}_{\bar{y}})$  as required. Conversely, if some disjunct of the decomposition holds, we can apply Equation DC- $\hat{r}$  and obtain  $\phi(\bar{c}_{\bar{x}}, \bar{c}_{\bar{y}})$ .

## 5 Applications of Decomposition

### 5.1 Monadic Decomposition in String Solving

The development of effective techniques for solving string constraints has received a lot of attention over the last years, motivated by applications ranging from program verification [2, 16] and security analysis [22, 23] to the analysis of access policies of cloud services [4]. Strings give rise to a rich theory that may combine, depending on the studied fragment, (i) word equations, i.e., equations over the free monoid generated by some finite (but often large) alphabet, (ii) regular expression constraints, (iii) transduction, i.e., constraints described by finite-state automata with multiple tracks, (iv) conversion functions, e.g. between integer variables and strings encoding numbers in binary or decimal notation, (v) length constraints, i.e., arithmetic constraints on the length of strings.

**Table 1.** Statistics about the Kaluza benchmarks [22]. It should be noted (and is well-known [18]) that the categories “sat” and “unsat” do not (always) imply the status of the benchmarks, they only represent the way the benchmarks were organised by the Kaluza authors.

Folder	#Benchmarks	Benchmarks with <code>str.len</code>	Decomposition checks	Decomposition checks succeeded
sat/small	19804	2185	2183	2155
sat/big	1741	1318	1317	56
unsat/small	11365	3910	2919	2919
unsat/big	14374	13813	6786	3362
Total	47284	21226	13205	8492

The handling of length constraints has turned out to be particularly challenging in this context, both practically and theoretically. Even for the combination of word equations (or even just quadratic word equations) with length

constraints, decidability of the (quantifier-free) theory is a long-standing open problem [21]. At the same time, length constraints are quite frequently used in applications; they are needed, for instance, when encoding operations like `indexOf` or `substring`, or also when splitting a string into the parts separated by some delimiter. In standard benchmark libraries for string constraints, like the Kaluza set [22], benchmarks with length constraints occur in large numbers.

The notion of monadic decomposition is in this setting important, since any *monadic* length constraint (in Presburger arithmetic) can be reduced to a Boolean combination of regular expression constraints, and is therefore easier to handle than the general case.

**Proposition 2.** *Satisfiability of a quantifier-free formula  $\phi = \phi_{eq} \wedge \phi_{regex} \wedge \phi_{len}$  consisting of word equations, regular expression constraints, and monadically decomposable length constraints is decidable.*

*Proof.* Suppose  $w_1, \dots, w_n$  are the string variables occurring in  $\phi$ , and  $|w_1|, \dots, |w_n|$  the terms representing their length. A decision procedure can first compute a monadic representation  $\phi'_{len}$  of  $\phi_{len}$  over lengths  $|w_1|, \dots, |w_n|$ , and then turn each atom  $\Delta(|w_i|)$  in  $\phi'_{len}$  into an equivalent regular membership constraint  $w_i \in \mathcal{L}_\Delta$ . This is possible because the Presburger formula  $\Delta$  can be represented as a semi-linear set, which can directly be translated to a regular expression. Decidability follows from the decidability of word equations combined with regular expression constraints [13].  $\square$

This motivates the use of monadic decomposition as a standard pre-processing step in string solvers, transforming away those length constraints that can be turned into monadic form. To evaluate the effectiveness of such an optimisation, we implemented the decomposition check defined in Sect. 3.2, and used it within the string SMT solver OSTRICH [9] to determine the number of Kaluza benchmarks with monadic decomposable length constraints.<sup>1</sup> The results are summarised in Table 1:

- Of altogether 47 284 benchmarks, 21 226 contain the `str.len` function, and therefore length constraints. This number was determined by a simple textual analysis of the benchmarks.
- Running our decomposition check in OSTRICH, in 13 205 of the 21 226 cases length constraints were found that could be analysed. The remaining 8 021 problems were proven unsatisfiable without ever reaching the string theory solver in OSTRICH, i.e., as a result of pre-processing the input formula, or because Boolean reasoning discovered obvious inconsistencies in the problems.
- In 8 492 of the 13 205 cases, all analysed length constraints were found to be monadically decomposable; 4 713 of the benchmarks contained length constraints that could not be decomposed.

<sup>1</sup> Branch “modec” of <https://github.com/uuverifiers/ostrich>, which also contains detailed logs of the experiments.

This means that 42 571 of the Kaluza benchmarks (slightly more than 90%) do in principle not require support for length constraints in a string solver, either because there are no length constraints, or because length constraints can be decomposed and then turned into regular expression constraints.

Even with a largely unoptimised implementation, the time required to check whether length constraints can be decomposed was negligible in case of the Kaluza benchmarks, with the longest check requiring 2.1 s (on an AMD Opteron 2220 SE machine). The maximum number of variables in a length constraint was 140.

## 5.2 Variadic Decomposition in Quantifier Elimination

A second natural application of decomposition is *quantifier elimination*, i.e., the problem of deriving an equivalent quantifier-free formula  $\phi'$  for a given formula  $\phi$  with quantifiers. In Presburger arithmetic, for a formula  $\phi = \exists x_1, \dots, x_n. \psi$  with  $n$  quantifiers but no quantifier alternations, quantifier elimination in the worst case causes a doubly-exponential increase in formula size [25].

Variadic decomposition can be used to eliminate quantifiers with a smaller worst-case increase in size, provided that the matrix of a quantifier formula can be decomposed. Suppose  $\phi = \exists \bar{x}. \psi(\bar{x}, \bar{y})$  is given and  $\psi$  is variadic decomposable on  $\bar{x}$ , i.e.,

$$\psi(\bar{x}, \bar{y}) \equiv \bigvee_j \Delta_j(\bar{x}) \wedge \psi_j(\bar{y})$$

This means that the existential quantifiers can be distributed over the disjunction, and their elimination turns into a simpler satisfiability check:

$$\exists \bar{x}. \psi(\bar{x}, \bar{y}) \equiv \bigvee_j \exists \bar{x}. \Delta_j(\bar{x}) \wedge \psi_j(\bar{y}) \equiv \bigvee_{j: \Delta_j(\bar{x}) \text{ is sat}} \psi_j(\bar{y})$$

Universal quantifiers can be handled in a similar way by negating the matrix first.

**Proposition 3.** *Take a formula  $\phi(\bar{y}) = \exists \bar{x}. \psi(\bar{x}, \bar{y})$  in Presburger arithmetic in which  $\psi$  is quantifier-free and variadic decomposable on  $\bar{x}$ . Then there is a quantifier-free formula  $\phi'(\bar{y})$  that is equivalent to  $\phi$  and at most singly-exponentially bigger than  $\phi$ .*

Checking whether a formula can be decomposed is therefore a simple optimisation that can be added to any quantifier elimination procedure for Presburger arithmetic.

## 6 Conclusion and Future Work

We have shown that the monadic and variadic decomposability problem for  $\text{QF}(\mathbb{N})$  is coNP-complete. Moreover, when a decomposition exists, it is at most exponential in size and can be computed in exponential time. This formula size



is tight for decompositions presented in either disjunctive or conjunctive normal form.

We gave two applications of our results. The first was in string constraint solving. In program analysis, string constraints are often mixed with numerical constraints on the lengths of the strings (for example, via the `indexOf` function). Length constraints significantly complicate the analysis of strings. However, if the string constraints permit a monadic decomposition, they may be reduced to regular constraints and thus eliminated. We analysed the well-known Kaluza benchmarks and showed that less than 10% of the benchmarks contained length constraints that could not be decomposed.

For the second application, we showed that the doubly exponential blow-up caused by quantifier elimination can be limited to a singly exponential blow up whenever the formula is decomposable on the quantified variables. Thus, variadic decomposition can form an optimisation step in a quantifier elimination algorithm.

Interesting problems are opened up by our results. It would be interesting to study lower bounds for general boolean formulas. If smaller decompositions are possible, they would be useful for applications in string solving.

Second, we may consider variadic decomposition where a partition  $\Pi$  is not given as part of the input. Instead, one must check whether a  $\Pi$ -decomposition exists for some non-trivial  $\Pi$ . This variant of the problem has a simple  $\Sigma_2^P$  algorithm that first guesses some  $\Pi$  and then verifies  $\Pi$ -decomposability. However, the only known lower bound is coNP, which follows the same argument as monadic decomposability. A better algorithm would not improve the worst-case complexity for our quantifier elimination application, but it might provide a way to quickly identify a subset of a block of quantifiers that can be eliminated quickly with  $\Pi$ -decompositions.

**Acknowledgments.** We thank Christoph Haase, Leonid Libkin, and Pascal Bergsträßer for their help during the preparation of this work. Matthew Hague is supported by EPSRC [EP/T00021X/1]. Anthony Lin is supported by the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement no 759969), and by Max-Planck Fellowship. Philipp Rümmer is supported by the Swedish Research Council (VR) under grant 2018-04727, and by the Swedish Foundation for Strategic Research (SSF) under the project WebSec (Ref. RIT17-0011). Zhilin Wu is partially supported by the NSFC grant No. 61872340, Guangdong Science and Technology Department grant (No. 2018B010107004), and the INRIA-CAS joint research project VIP.

## References

1. Abdulla, P.A., et al.: TRAU: SMT solver for string constraints. In: Formal Methods in Computer Aided Design, FMCAD 2018 (2018)
2. Abdulla, P.A., et al.: String constraints for verification. In: Biere, A., Bloem, R. (eds.) CAV 2014. LNCS, vol. 8559, pp. 150–166. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-08867-9\\_10](https://doi.org/10.1007/978-3-319-08867-9_10)

3. Amadini, R., Gange, G., Stuckey, P.J.: Sweep-based propagation for string constraint solving. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI 2018), the 30th Innovative Applications of Artificial Intelligence (IAAI 2018), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI 2018), New Orleans, Louisiana, USA, 2–7 February 2018, pp. 6557–6564 (2018). <https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16223>
4. Backes, J., et al.: Semantic-based automated reasoning for AWS access policies using SMT. In: Bjørner, N., Gurfinkel, A. (eds.) 2018 Formal Methods in Computer Aided Design, FMCAD 2018, Austin, TX, USA, 30 October–2 November 2018, pp. 1–9. IEEE (2018). <https://doi.org/10.23919/FMCAD.2018.8602994>
5. Barceló, P., Hong, C., Le, X.B., Lin, A.W., Niskanen, R.: Monadic decomposability of regular relations. In: 46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, Patras, Greece, pp. 103:1–103:14 (2019). <https://doi.org/10.4230/LIPIcs.ICALP.2019.103>
6. Berzish, M., Ganesh, V., Zheng, Y.: Z3str3: a string solver with theory-aware heuristics. In: 2017 Formal Methods in Computer Aided Design, FMCAD 2017, Vienna, Austria, 2–6 October 2017, pp. 55–59. IEEE (2017). <https://doi.org/10.23919/FMCAD.2017.8102241>
7. Büchi, J.R., Senger, S.: Definability in the existential theory of concatenation and undecidable extensions of this theory. In: Mac, L.S., Siefkes, D. (eds.) The Collected Works of J. Richard Büchi, pp. 671–683. Springer, Heidelberg (1990). [https://doi.org/10.1007/978-1-4613-8928-6\\_37](https://doi.org/10.1007/978-1-4613-8928-6_37)
8. Carton, O., Choffrut, C., Grigorieff, S.: Decision problems among the main sub-families of rational relations. *ITA* **40**(2), 255–275 (2006). <https://doi.org/10.1051/ita:2006005>
9. Chen, T., Hague, M., Lin, A.W., Rümmer, P., Wu, Z.: Decision procedures for path feasibility of string-manipulating programs with complex operations. *CoRR* abs/1811.03167 (2018). <https://arxiv.org/abs/1811.03167>
10. Chistikov, D., Haase, C.: The taming of the semi-linear set. In: Chatzigiannakis, I., Mitzenmacher, M., Rabani, Y., Sangiorgi, D. (eds.) 43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016). Leibniz International Proceedings in Informatics (LIPIcs), vol. 55, pp. 128:1–128:13. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2016). <https://doi.org/10.4230/LIPIcs.ICALP.2016.128>. <http://drops.dagstuhl.de/opus/volltexte/2016/6263>
11. D’Antoni, L., Veanes, M.: The power of symbolic automata and transducers. In: Majumdar, R., Kunčák, V. (eds.) CAV 2017, Part 1. LNCS, vol. 10426, pp. 47–67. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-63387-9\\_3](https://doi.org/10.1007/978-3-319-63387-9_3)
12. Day, J.D., Ehlers, T., Kulczynski, M., Manea, F., Nowotka, D., Poulsen, D.B.: On solving word equations using SAT. In: Filiot, E., Jungers, R., Potapov, I. (eds.) RP 2019. LNCS, vol. 11674, pp. 93–106. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-30806-3\\_8](https://doi.org/10.1007/978-3-030-30806-3_8)
13. Diekert, V.: Makanin’s algorithm. In: Lothaire, M. (ed.) Algebraic Combinatorics on Words, Encyclopedia of Mathematics and its Applications, vol. 90, chap. 12, pp. 387–442. Cambridge University Press (2002). <https://doi.org/10.1017/CBO9781107326019.013>
14. Ganesh, V., Minnes, M., Solar-Lezama, A., Rinard, M.: Word equations with length constraints: what’s decidable? In: Biere, A., Nahir, A., Vos, T. (eds.) HVC 2012. LNCS, vol. 7857, pp. 209–226. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-39611-3\\_21](https://doi.org/10.1007/978-3-642-39611-3_21)

15. Haase, C.: Subclasses of presburger arithmetic and the weak EXP hierarchy. In: Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS 2014, Vienna, Austria, 14–18 July 2014, pp. 47:1–47:10 (2014). <https://doi.org/10.1145/2603088.2603092>
16. Hojjat, H., Rümmer, P., Shamakhi, A.: On strings in software model checking. In: Lin, A.W. (ed.) APLAS 2019. LNCS, vol. 11893, pp. 19–30. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-34175-6\\_2](https://doi.org/10.1007/978-3-030-34175-6_2)
17. Jez, A.: Word equations in linear space. CoRR abs/1702.00736 (2017). <http://arxiv.org/abs/1702.00736>
18. Liang, T., Reynolds, A., Tinelli, C., Barrett, C., Deters, M.: A DPLL( $T$ ) theory solver for a theory of strings and regular expressions. In: Biere, A., Bloem, R. (eds.) CAV 2014. LNCS, vol. 8559, pp. 646–662. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-08867-9\\_43](https://doi.org/10.1007/978-3-319-08867-9_43)
19. Libkin, L.: Variable independence for first-order definable constraints. ACM Trans. Comput. Log. 4(4), 431–451 (2003). <https://doi.org/10.1145/937555.937557>
20. Lin, A.W., Barceló, P.: String solving with word equations and transducers: towards a logic for analysing mutation XSS. In: Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, pp. 123–136. Springer (2016). <https://doi.org/10.1145/2837614.2837641>
21. Lin, A.W., Majumdar, R.: Quadratic word equations with length constraints, counter systems, and presburger arithmetic with divisibility. In: Lahiri, S.K., Wang, C. (eds.) ATVA 2018. LNCS, vol. 11138, pp. 352–369. Springer, Cham (2018). [https://doi.org/10.1007/978-3-030-01090-4\\_21](https://doi.org/10.1007/978-3-030-01090-4_21)
22. Saxena, P., Akhawe, D., Hanna, S., Mao, F., McCamant, S., Song, D.: A symbolic execution framework for JavaScript. In: 31st IEEE Symposium on Security and Privacy, S&P 2010, Berkeley/Oakland, California, USA, 16–19 May 2010, pp. 513–528. IEEE (2010). <https://doi.org/10.1109/SP.2010.38>
23. Trinh, M., Chu, D., Jaffar, J.: S3: a symbolic string solver for vulnerability detection in web applications. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, CCS 2014, pp. 1232–1243. ACM (2014). <https://doi.org/10.1145/2660267.2660372>
24. Veanes, M., Bjørner, N., Nachmanson, L., Bereg, S.: Monadic decomposition. J. ACM 64(2), 14:1–14:28 (2017). <https://doi.org/10.1145/3040488>
25. Weispfenning, V.: Complexity and uniformity of elimination in presburger arithmetic. In: Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation, ISSAC 1997, Maui, Hawaii, USA, 21–23 July 1997, pp. 48–53 (1997)