Term Algebras with Length Function and Bounded Quantifier Alternation

Ting Zhang, Henny B. Sipma, Zohar Manna*

Computer Science Department
Stanford University
Stanford, CA 94305-9045
{tingz,sipma,zm}@theory.stanford.edu

Abstract. Term algebras have wide applicability in computer science. Unfortunately, the decision problem for term algebras has a nonelementary lower bound, which makes the theory and any extension of it intractable in practice. However, it is often more appropriate to consider the bounded class, in which formulae can have arbitrarily long sequences of quantifiers but the quantifier alternation depth is bounded. In this paper we present new quantifier elimination procedures for the first-order theory of term algebras and for its extension with integer arithmetic. The elimination procedures deal with a block of quantifiers of the same type in one step. We show that for the bounded class of k quantifier alternations, regardless of the total number of quantifiers, the complexity of our procedures is k-fold exponential (resp. $2k$ fold exponential) for the theory of term algebras (resp. for the extended theory with integers).

1 Introduction

The theory of term algebras, also known as the theory of finite trees, axiomatizes the Herbrand universe. It has wide applicability in computer science. In programming languages many so-called recursive data structures can be modeled as term algebras [19]; in theorem proving it is essential to the unification and disunification problem [18, 3]; in logic programming, it is used to define formal semantics [14]. Other applications can be found in computational linguistics, constraint databases, pattern matching and type theory.

In this paper we consider an arithmetic extension of the theory of term algebras. Our extended language has two sorts; the integer sort $\mathbb{Z}$ and the term sort TA. Intuitively, the language is the set-theoretic union of the language of term algebras and the language of Presburger arithmetic plus the additional length function $(.)^{\mathsf{L}} : \mathsf{TA} \to \mathbb{Z}$. Formulae are formed from term literals and integer literals using logical connectives and quantifications. Term literals are exactly those literals in the language of term algebras. Integer literals are those that

* This research was supported in part by NSF grants CCR-01-21403, CCR-02-20134 and CCR-02-09237, by ARO grant DAAD19-01-1-0723, by ARPA/AF contracts F33615-00-C-1693 and F33615-99-C-3014, and by NAVY/ONR contract N00014-03-1-0939.

can be built up from integer variables (including the length function applied to TA-terms), the usual arithmetic relations and functions. This type of arithmetic extension has been used in [10, 11] to show that the quantifier-free theory of term algebras with Knuth-Bendix order is NP-complete.

Our interest originates from program verification as term algebras can model a wide range of tree-like data structures. Exampless include lists, stacks, counters, trees, records and queues. To verify programs containing these data structures we must be able to reason about these data structures. However, in program verification decision procedures for a single theory are usually not applicable as programming languages often involve multiple data domains, resulting in verification conditions that span multiple theories. A common example of such "mixed" constraints are combinations of data structures with integer constraints on the size of those structures. In [24] we gave a quantifier-elimination procedure for this extended theory.

Unfortunately the theory of term algebras has nonelementary time complexity [7, 3, 22], which makes the theory and any extension of it intractable in practice. However, as observed by many [20, 8], in consideration of the complexity of logic theories, the meaning of a formula soon becomes incomprehensible as the number of quantifier alternations increases. In practice we rarely deal with formulae with a large quantifier alternation depth. Therefore it is worthwhile to investigate the class of formulae which can have arbitrarily long sequences of quantifiers of the same kind while the total number of quantifier alternations is bounded by a constant number. We call such formulae alternation bounded.

In this paper we present new quantifier elimination procedures for the theory of term algebras as well as the extended theory with integers. Our procedures can eliminate a block of quantifiers of the same kind in one step. The complexity of both procedures is one exponential for eliminating a block of quantifiers of the same type. For the bounded class of k quantifier alternations, regardless of the total number of quantifiers, the complexity is k-fold exponential (resp. $2k$ fold exponential) for the theory of term algebras (resp. for the extended theory with integers).

Related Work and Comparison. Presburger arithmetic (PA) was first shown to be decidable in 1929 by the quantifier elimination method [6]. Efficient algorithms were later discovered by Cooper et al [5, 20]. It was shown in [20] and [8], respectively, that the upper bound and the lower bound of the bounded class in the theory of PA is one exponential lower than the whole theory.

The decidability of the first-order theory of term algebras was first shown by Mal'cev using quantifier elimination [17]. This result was reproved in different settings [16, 3, 9, 2, 1, 21, 13, 12, 24]. The lower bound of any theory of pairing functions was shown to be nonelementary in [7]; this result was strengthened in [4] to a hereditarily nonelementary lower bound. This lower bound complexity applies to the theory of term algebras as term algebras with a binary constructor can express pairing functions. Using techniques in [4, 22] showed that theories of finite trees, infinite and rational trees are all hereditarily nonelementary.

Quantifier elimination has been used to obtain decidability results for various extensions of term algebras. [16] showed the decidability of the theory of infinite and rational trees. [2] presented an elimination procedure for term algebras with membership predicate in the regular tree language. [1] presented an elimination procedure for structures of feature trees with arity constraints. [21] showed the decidability of term algebras with queues. [13] showed the decidability of term powers, which generalize products and term algebras. [24] extended the quantifier elimination procedure in [9] for term algebras with length function.

Traditionally, methods for quantifier elimination for term algebras follow one of two approaches: they either perform transformations in the constructor language [17, 3, 16, 12], or they work in the selector language [9, 21]. In the first approach formulae are reduced to a boolean combination of a specific kind of formulae called "solved forms", which only include ordinary literals. In this respect [3] is essentially a dual of [16] with the special formulae being universally quantified. In [12] selectors are used to convert solved forms to quantifier-free formulae. In the second approach, formulae are transformed into a form in which the quantified variable is not embedded in selectors and only occurs in disequalities. Methods following the first approach can deal with a block of quantifiers of the same type in one step, but it is hard to determine an upper bound on their complexity, because they all rely on the "independence lemma" ([17], page 277, also see Thm. 1 in this paper) which states that "there are enough elements to satisfy a certain set of disequalities and equalities." However, this does not hold in the language with finite signature and length function. Methods following the second approach can only handle a single quantifier at a time.

Our elimination procedures are carried out in the language with both selectors and constructors. The use of selectors is similar to [12]. The method combines the extraction of integer constraints from term constraints with a reduction of quantifiers on term variables to quantifiers on integer variables.

Paper Organization. Section 2 provides the preliminaries: it introduces the notation and terminology. Section 3 defines term algebras. Section 4 describes a new elimination procedure for the theory of term algebras. Section 5 introduces the theory of term algebras with integers arithmetic and presents the technical machinery for handling the length function. Section 6 presents the main contribution of this paper: it expands the elimination procedure in Section 4 for the extended theory and proves the correctness and complexity bounds. Section 7 concludes with some ideas for future work. Due to space limitation all proofs have been omitted from this paper. They are available for reference in the extended version of this paper at the first author's website.

2 Preliminaries

We assume the first-order syntactic notions of variables, parameters and quantifiers, and semantic notions of structures, satisfiability and validity as in [6]. We explain concepts and terminology important to this paper as follows.

A signature Σ is a set of parameters (function symbols and predicate symbols) each of which is associated with an arity. The function symbols with arity 0 are also called constants. The set of Σ-terms $\mathcal{T}(\Sigma, \mathcal{X})$ is recursively defined by: (i) every constant $c \in \Sigma$ or variable $x \in \mathcal{X}$ is a term, and (ii) if $f \in \Sigma$ is an n-place function symbol and $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term. If θ is a formula, we use $\Sigma(\theta)$ to denote the set of terms occurring in θ. The length of a term t, written $\mathsf{len}(t)$, is defined recursively by: (i) for any constant a, $\mathsf{len}(a) = 1$, and (ii) for a term $\alpha(t_1, \ldots, t_k)$, $\mathsf{len}(\alpha(t_1, \ldots, t_k)) = \sum_{i=1}^{k} \mathsf{len}(t_i) + 1$.

An atomic formula (atom) is a formula of the form $P(t_1, \ldots, t_n)$ where P is an n-place predicate symbol and $t_1, \ldots, t_n$ are terms (equality is treated as a binary predicate symbol). A literal is an atomic formula or its negation. A variable occurs free in a formula if it is not in the scope of a quantifier. A formula without quantifiers is called quantifier-free. A ground formula is a formula with no variables. A sentence is a formula in which no variable occurs free. Every quantifier-free formula can be put into disjunctive normal form, that is, a disjunction of conjunctions of literals.

We use $\boldsymbol{x}$ to denote a set of variables, say, $x_1, \ldots, x_n$, and $\exists \boldsymbol{x}$ (resp. $\forall x$) as an abbreviation of $\exists x_1, \ldots, \exists x_n$ (resp. $\forall x_1, \ldots, \forall x_n$). When we write $\theta(\boldsymbol{x})$, we mean that $\boldsymbol{x}$ occur free in θ. Any formula θ can be put into prenex form $Q_1 x_1, \ldots, Q_n x_n\ \theta(\boldsymbol{x})$, where Q_i's are either $\exists$ or $\forall$ and $\theta(\boldsymbol{x})$ is quantifier-free. We call $\theta(\boldsymbol{x})$ the matrix of θ. We say that θ has quantifier (alternation) depth m if $Q_1, \ldots, Q_n$ can be divided into m blocks such that all quantifiers in a block are of the same type and quantifiers in two consecutive blocks are different. Let $\mathsf{C_k}(\mathfrak{A})$ denote the class of true sentence in $\mathfrak{A}$ with quantifier alternation depth k.

A variable assignment σ (w.r.t. $\mathfrak{A}$) is a function that assigns each variable an element of A. We use $[\![\boldsymbol{x}]\!]\sigma$ to denote the assigned values of $\boldsymbol{x}$ under σ. We write $[\![\boldsymbol{x}]\!]$ when σ is clear from the context. A formula θ is satisfiable (resp. valid) if it evaluates to true under some (resp. all) variable assignments.

A theory T is said to admit quantifier elimination if any formula can be equivalently (modulo T) and effectively transformed into a quantifier-free formula. If a theory admits quantifier elimination, then every sentence is reducible to a ground formula. Therefore, if ground literals are decidable, then a quantifier elimination procedure becomes a decision procedure.

Presburger arithmetic (PA) is the first-order theory of addition in the arithmetic of integers. The corresponding language and structure are denoted, respectively, by $\mathscr{L}_{\mathbb{Z}}$ and $\mathfrak{A}_{\mathbb{Z}} = \langle \mathbb{Z}; 0, +, < \rangle$.

Define $\exp_0(f(n)) = f(n)$ and $\exp_{m+1}(f(n)) = 2^{\exp_m(f(n))}$.

3 Term Algebras

We present a general language and structure of term algebras. For simplicity, we do not distinguish syntactic terms in the language from semantic terms in the corresponding structure. The meaning should be clear from the context.

Definition 1. *A term algebra* $\mathfrak{A}_{\mathsf{TA}} : \langle \mathsf{TA}; \mathcal{A}, C, \mathcal{S}, \mathcal{T} \rangle$ *consists of*

1. *TA: The term domain, which consists of all terms built up from constants by applying constructors. Elements in TA are called TA-terms.*
2. *$\mathcal{A}$: A finite set of constants: $a, b, c, \ldots$*
3. *$\mathcal{C}$: A finite set of constructors: $\alpha, \beta, \gamma, \ldots$ The arity of α is denoted by $\mathsf{ar}(\alpha)$. An object is α-typed (or an α-term) if its outmost constructor is α.*
4. *$\mathcal{S}$: A finite set of selectors. For a constructor α with arity k, there are k selectors $\mathsf{s}^\alpha_1, \ldots, \mathsf{s}^\alpha_k$ in $\mathcal{S}$. For a term x, $\mathsf{s}^\alpha_i(x)$ returns the i^{th} component of x if x is an α-term and x itself otherwise.*
5. *$\mathcal{T}$: A finite set of testers. For each constructor α there is a corresponding tester Is_α. For a term x, $\mathsf{Is}_\alpha(x)$ is true if and only if x is an α-term. In addition there is a special tester Is_A such that $\mathsf{Is}_\mathsf{A}(x)$ is true if and only if x is a constant. Note that there is no need for individual constant testers as $x = a$ serves as $\mathsf{Is}_a(x)$.*

We denote by $\mathscr{L}_\mathsf{TA}$ the language for $\mathfrak{A}_\mathsf{TA}$.

Unless mentioned otherwise, in this paper we assume that $\mathscr{L}_\mathsf{TA}$ is finite. The decision problem becomes easier if we allow $\mathscr{L}_\mathsf{TA}$ to have infinitely many constants.

The theory of term algebras is axiomatizable as follows ([9]).

Proposition 1 (Axiomatization of Term Algebras [9]). *Let z_α abbreviate $z_1, \ldots, z_{\mathsf{ar}(\alpha)}$. The following formula schemes, in which variables are implicitly universally quantified over TA, axiomatize $\mathsf{Th}(\mathfrak{A}_\mathsf{TA})$.*

A. $t(x) \neq x$, *if t is built solely by constructors and t properly contains x.*
B. $a \neq b$, $a \neq \alpha(x_1 \ldots, x_{\mathsf{ar}(\alpha)})$, *and* $\alpha(x_1 \ldots, x_{\mathsf{ar}(\alpha)}) \neq \beta(y_1, \ldots, y_{\mathsf{ar}(\beta)})$, *if a and b are distinct constants and if α and β are distinct constructors.*
C. $\alpha(x_1, \ldots, x_{\mathsf{ar}(\alpha)}) = \alpha(y_1, \ldots, y_{\mathsf{ar}(\alpha)}) \rightarrow \bigwedge_{1 \leq i \leq \mathsf{ar}(\alpha)} x_i = y_i$.
D. $\mathsf{Is}_\alpha(x) \leftrightarrow \exists\, z_\alpha \alpha(z_\alpha) = x$, $\mathsf{Is}_\mathsf{A}(x) \leftrightarrow \bigwedge_{\alpha \in C} \neg \mathsf{Is}_\alpha(x)$.
E. $\mathsf{s}^\alpha_i(x) = y \leftrightarrow (\exists z_\alpha(\alpha(z_\alpha) = x \wedge y = z_i)) \vee (\forall z_\alpha(\alpha(z_\alpha) \neq x) \wedge x = y)$.

In general selectors and testers can be defined by constructors and vice versa. One direction has been shown by (D) and (E), which are pure definitional axioms.

Example 1. Consider the LISP list structure $\mathfrak{A}_\mathsf{list} = \langle \mathsf{list};\ \mathsf{cons}, \mathsf{car}, \mathsf{cdr} \rangle$ where list denotes the domain, cons is the 2-place constructor (pairing function) and car and cdr are the corresponding left and right selectors (projectors) respectively. It is not difficult to verify that $\mathfrak{A}_\mathsf{list}$ is an instance of term algebras.

We use the notation $\alpha = (\mathsf{s}^\alpha_1, \ldots, \mathsf{s}^\alpha_k)$ to mean that α is a constructor with $\mathsf{ar}(\alpha) = k$ and $\mathsf{s}^\alpha_1, \ldots, \mathsf{s}^\alpha_k$ are the corresponding selectors of α. We call a term t a constructor term (resp. selector term) if the outmost function symbol of t is a constructor (resp. a selector). We assume that no constructor term appears inside selectors as simplification can always be done. For example, $\mathsf{s}^\alpha_i(\alpha(x_1, \ldots, x_k))$ simplifies to x_i $(1 \leq i \leq k)$ and $\mathsf{s}^\beta_j(\alpha(x_1, \ldots, x_k))$ simplifies to $\alpha(x_1, \ldots, x_k)$ for $\alpha \not\equiv \beta$. We use $L, M, N, \ldots$ to denote selector sequences. If $L = \mathsf{s}_1, \ldots, \mathsf{s}_n$, Lx is an abbreviation for $\mathsf{s}_1(\ldots(\mathsf{s}_n(x)\ldots))$, and we say that the depth of x in Lx is n. The depth of x in a formula is the maximum of all depths of x in the selector terms in the formula. We say a selector term $\mathsf{s}_i^\alpha(t)$ is proper if $\mathsf{Is}_\alpha(t)$ holds. We can make selector terms proper with the type information.

Definition 2 (Type Completion). *θ' is a type completion of θ if θ' is obtained from θ by adding tester predicates such that for any term $\mathsf{s}(t)$ exactly one constant of the form $\mathsf{Is}_\alpha(t)$ (for some constructor α) or $\mathsf{Is}_\mathsf{A}(t)$ is present in θ'.*

Example 2. Let $\alpha = (\mathsf{s}_1^\alpha, \mathsf{s}_2^\alpha)$. A possible type completion for $y = \mathsf{s}_1^\alpha(\mathsf{s}_2^\alpha(x))$ is $y = \mathsf{s}_1^\alpha(\mathsf{s}_2^\alpha(x)) \wedge \mathsf{Is}_\alpha(x) \wedge \mathsf{Is}_\mathsf{A}(\mathsf{s}_2^\alpha(x))$. With this type information we can simplify $y = \mathsf{s}_1^\alpha(\mathsf{s}_2^\alpha(x))$ to $y = \mathsf{s}_2^\alpha(x)$ by Axioms (D) and (E) defined below.

4 A New Quantifier Elimination Procedure for $\mathsf{Th}(\mathfrak{A}_{\mathsf{TA}})$

In this section we present a new quantifier elimination algorithm for the theory of term algebras and show that the algorithm only needs exponential time to eliminate one block of quantifiers of the same kind. The algorithm works mainly in the constructor language while using selectors as auxiliary tools. The algorithm is also the basis for the elimination procedure for the extended theory presented in Section 6.

Normal Form It is well-known that eliminating arbitrary quantifiers reduces to eliminating existential quantifiers from formulae in the form

$$\exists \boldsymbol{x}(A_1(\boldsymbol{x}) \wedge \ldots \wedge A_n(\boldsymbol{x})), \tag{1}$$

where $A_i(\boldsymbol{x})$ $(1 \leq i \leq n)$ are literals [9]. We can also assume that $A_i's$ are not of the form $x = t$ as $\exists x(x = t \wedge \theta(x, \boldsymbol{y}))$ simplifies to $\theta(t, \boldsymbol{y})$, if x does not occur in t, to $\exists x \theta(x, \boldsymbol{y})$ if $t \equiv x$, and to false by Axiom (A) if t is a term which is built solely by constructors and properly contains x.

Nondeterminism In this paper all transformations are done on formulae of the form (1). Whenever we say "guess θ", we mean to add a valid disjunction $\bigvee_i \theta_i$ (where θ is one of the disjuncts) to the matrix of (1). When we replace θ by $\bigvee_i \theta_i$ or directly introduce $\bigvee_i \theta_i$, it should be understood that an implicit disjunctive splitting is carried out and we work on each resultant disjunct in the form (1) "simultaneously".

Simplification For simplicity, in the description of algorithms, we omit tester literals unless they are needed for correctness proof. We may also assume that the matrix of (1) is type complete and basic simplifications are carried out whenever applicable. For example, for a nonempty selector sequence L, we replace $Lx \neq x$ by true and $Lx = x$ by false. Similarly for $t(x) \neq x$ and $t(x) = x$ where $t(x)$ is a term properly containing x.

Notation In the algorithm we use the following notation: $\boldsymbol{x}$ dnote the set of existentially quantified variables; $\boldsymbol{y}$ denote the set of (implicitly) universally quantified parameters; s, t denote TA-terms; G, H denote (possibly empty) selector blocks; f, g, h denote index functions with ranges clear from the context; numerical superscripts are parenthesized. The use of index functions is to differentiate multiple occurrences of the same variables.

Note that in each step the algorithm manipulates the formula $\exists\boldsymbol{x} : \theta(\boldsymbol{x}, \boldsymbol{y})$ to produce a version of the same form (or multiple versions of the same form in case disjunctions are introduced), and thus in each step $\exists\boldsymbol{x} : \theta(\boldsymbol{x}, \boldsymbol{y})$ refers to the updated version rather than to the original input formula.

Definition 3 (Solved Form). *We say $\theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y})$ is in the* solved form *(with respect to $\boldsymbol{x}$), if $\boldsymbol{x}$ are not in equalities, not asserted to be constants (if $\mathscr{L}_{\mathsf{TA}}$ is finite) and not inside selector terms. We say $\exists\boldsymbol{x}\, \theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y})$ is in the solved form if $\theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y})$ is.*

The elimination goes as follows. A sequence of equivalent transformations will bring the input formula into a disjunction of formulae in the solved form which have solutions under any instantiation of parameters. Therefore, the whole block of existential quantifiers $\exists\boldsymbol{x}$ can be eliminated by removing all literals containing $\boldsymbol{x}$ in the matrix.

Algorithm 1. *Input: $\exists\boldsymbol{x} : \theta(\boldsymbol{x}, \boldsymbol{y})$.*

1. Type Completion. *Guess a type completion of $\theta(\boldsymbol{x}, \boldsymbol{y})$ and simplify every selector term to a proper one.*
2. Elimination of Selector Terms Containing $\boldsymbol{x}$. *Replace all selector terms containing x by the corresponding equivalent constructor terms according to Axiom* (E)*. For example $s_1^{\alpha}(x) = y$ becomes $\exists z_2, \ldots, z_k \alpha(y, z_2, \ldots, z_k) = x$ for $\mathsf{ar}(\alpha) = k$. It may increase the number of existential quantifiers, but leaves parameters unchanged.*
3. Elimination of Equalities between Constructor Terms. *Replace*
$$\alpha(t_1, \ldots, t_i) = \alpha(t'_1, \ldots, t'_i) \tag{2}$$
by $\bigwedge_{1 \le i \le k} t_i = t'_i$. Repeat until no equality of the form (2) *appears.*
4. Elimination of Disequalities between Constructor Terms. *Replace*
$$\alpha(t_1, \ldots, t_i) \neq \alpha(t'_1, \ldots, t'_i) \tag{3}$$
by $\bigvee_{1 \le i \le k} t_i \neq t'_i$. Repeat until no equality of the form (3) *appears. At this point we may assume that each disjunct (that has not been simplified to* false*) is in the form*
$$\exists\boldsymbol{x} : \Big[\bigwedge_i x_{f(i)} \neq t_i(\boldsymbol{x}, \boldsymbol{y}) \wedge \bigwedge_i G_i y_{g(i)} \neq s_i(\boldsymbol{x}, \boldsymbol{y}) \wedge \bigwedge_i H_i y_{h(i)} = u_i(\boldsymbol{x}, \boldsymbol{y})\Big]. \tag{4}$$
5. Elimination of Equalities Containing $\boldsymbol{x}$. *Solve equations of the form $H_i y_{h(i)} = u_i(\boldsymbol{x}, \boldsymbol{y})$, where $u_i(\boldsymbol{x}, \boldsymbol{y})$ is a constructor term containing $\boldsymbol{x}$, in terms of $H_i y_{h(i)}$ such that the result is a set of equations in the selector language. For example, with $\alpha = (\mathsf{s}_1^{\alpha}, \mathsf{s}_2^{\alpha})$, the solution set of $\mathsf{s}_2^{\alpha} y = \alpha(\alpha(x_1, y_1), y_2)$ is*
$$x_1 = \mathsf{s}_1^{\alpha}\mathsf{s}_1^{\alpha}\mathsf{s}_2^{\alpha} y, \quad y_1 = \mathsf{s}_2^{\alpha}\mathsf{s}_1^{\alpha}\mathsf{s}_2^{\alpha} y, \quad y_2 = \mathsf{s}_2^{\alpha}\mathsf{s}_2^{\alpha} y.$$

Solving $\bigwedge_i H_i y_{h(i)} = u_i(\boldsymbol{x}, \boldsymbol{y})$ *and eliminating all x's occurring in solved equations, we obtain*

$$\exists \boldsymbol{x} : \Big[\bigwedge_i x_{f^{(2)}(i)} \neq t_i^{(2)}(\boldsymbol{x}, \boldsymbol{y}) \wedge \bigwedge_i G_i^{(2)} y_{g^{(2)}(i)} \neq s_i^{(2)}(\boldsymbol{x}, \boldsymbol{y}) \wedge \bigwedge_i H_i^{(2)} y_{h^{(2)}(i)} = H_i^{(3)} y_{h^{(3)}(i)} \Big]. \quad (5)$$

6. Elimination of Constants. *If* $\mathscr{L}_{\mathsf{TA}}$ *has finitely many constants and for some* $x \in \boldsymbol{x}$, $\mathsf{Is}_{\mathsf{A}}(x)$ *appears in* (5), *we instantiate* x *to each constant to eliminate* $\exists x$. *We still use* (5) *to denote the resulting formula.*
7. Elimination of Quantifiers. *Rewrite* $\bigwedge_i G_i^{(2)} y_{g^{(2)}(i)} \neq s_i^{(2)}(\boldsymbol{x}, \boldsymbol{y})$ *as*

$$\bigwedge_i G_i^{(3)} y_{g^{(3)}(i)} \neq s_i^{(3)}(\boldsymbol{x}, \boldsymbol{y}) \wedge \bigwedge_i G_i^{(4)} y_{g^{(4)}(i)} \neq s_i^{(4)}(\boldsymbol{y}),$$

where $\boldsymbol{x}$ *do not appear in* $s_i^{(4)}(\boldsymbol{y})$. *Then* (5) *can be rewritten as*

$$\exists \boldsymbol{x} : \Big[\bigwedge_i x_{f^{(2)}(i)} \neq t_i^{(2)}(\boldsymbol{x}, \boldsymbol{y}) \wedge \bigwedge_i G_i^{(3)} y_{g^{(3)}(i)} \neq s_i^{(3)}(\boldsymbol{x}, \boldsymbol{y}) \Big] \wedge \bigwedge_i G_i^{(4)} y_{g^{(4)}(i)} \neq s_i^{(4)}(\boldsymbol{y}) \wedge \bigwedge_i H_i^{(2)} y_{h^{(2)}(i)} = H_i^{(3)} y_{h^{(3)}(i)}. \quad (6)$$

We claim that

$$\exists \boldsymbol{x} : \Big[\bigwedge_i x_{f^{(2)}(i)} \neq t_i^{(2)}(\boldsymbol{x}, \boldsymbol{y}) \wedge \bigwedge_i G_i^{(3)} y_{g^{(3)}(i)} \neq s_i^{(3)}(\boldsymbol{x}, \boldsymbol{y}) \Big] \quad (7)$$

is valid and hence (6) *is equivalent to*

$$\bigwedge_i G_i^{(4)} y_{g^{(4)}(i)} \neq s_i^{(4)}(\boldsymbol{y}) \wedge \bigwedge_i H_i^{(2)} y_{h^{(2)}(i)} = H_i^{(3)} y_{h^{(3)}(i)}. \quad (8)$$

Theorem 1. *All transformations in Alg. 1 preserve equivalence.*

Theorem 2. *Alg. 1 eliminates a block of quantifiers in time* $2^{O(n)}$.

Theorem 3. $\mathsf{C_k}(\mathfrak{A}_{\mathsf{TA}})$ *is decidable in* $O(\exp_k(n))$.

5 Term Algebras with Length Function

In this section we introduce the extended theory and present the technical machinery needed to handle lengths of TA-terms in the elimination procedure.

Definition 4. *The structure of the extended language is* $\mathfrak{A}^{\mathbb{Z}}_{\mathsf{TA}} = (\mathfrak{A}_{\mathsf{TA}}; \mathfrak{A}_{\mathbb{Z}}; (.)^{\mathsf{L}} : \mathsf{TA} \to \mathbb{Z})$ *where* $\mathfrak{A}_{\mathsf{TA}}$ *is a term algebra,* $\mathfrak{A}_{\mathbb{Z}}$ *is Presburger arithmetic, and* $(.)^{\mathsf{L}}$ *denotes the length function; for a term* t, $(t^{\mathsf{L}})^{\mathfrak{A}^{\mathbb{Z}}_{\mathsf{TA}}} = \mathsf{len}(t^{\mathfrak{A}^{\mathbb{Z}}_{\mathsf{TA}}})$. *We denote by* $\mathscr{L}^{\mathbb{Z}}_{\mathsf{TA}}$ *the language for* $\mathfrak{A}^{\mathbb{Z}}_{\mathsf{TA}}$.

We call terms of sort TA (resp. $\mathbb{Z}$) TA-terms (resp. integer terms), similarly for constants, variables, quantifiers and formulae. We also use "term" for "TA" when there is no confusion. A TA-term can occur inside the length function. We call this type of occurrence integer occurrence distinguishing it from the normal term occurrence.

If $\boldsymbol{t}$ is a set of TA-terms, we use $\boldsymbol{t}^{\mathsf{L}}$ to denote the set of all integer occurrences, in the context, of the form $(Lt)^{\mathsf{L}}$ where $t \in \boldsymbol{t}$ and L denotes a block of (possibly empty) selectors.

Example 3. The formula $\exists x \exists y : \mathsf{TA}\ (x \neq y \wedge x^{\mathsf{L}} = y^{\mathsf{L}})$ states that there exists at least two distinct terms $t_1, t_2 \in \mathsf{TA}$ such that $\mathsf{len}(t_1) = \mathsf{len}(t_2)$. Note that the 1^{st} occurrence of x is an ordinary term while the 2^{nd} one is integral. The same for the occurrences of y.

Instead of writing $\boldsymbol{n} = \boldsymbol{t}^{\mathsf{L}}$ to indicate the connection between term variables and the corresponding integer variables, we abuse the notation a bit by using $\boldsymbol{t}^{\mathsf{L}}$ as formal variables directly in Presburger formulae. For example, $\exists \boldsymbol{x}^{\mathsf{L}} : \mathbb{Z}\, \theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}) \rightarrow \exists \boldsymbol{x} : \mathsf{TA}\, \theta_{\mathsf{TA}}(\boldsymbol{x})$ stands for $\forall \boldsymbol{x}^{\mathsf{L}} : \mathbb{Z} \left[\theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}) \rightarrow \exists \boldsymbol{x} : \mathsf{TA}\, \theta_{\mathsf{TA}}(\boldsymbol{x})\right]$, which in turn is a shorthand for $\forall \boldsymbol{n} : \mathbb{Z} \left[\theta_{\mathbb{Z}}(\boldsymbol{n}) \rightarrow \exists \boldsymbol{x} : \mathsf{TA}\, (\theta_{\mathsf{TA}}(\boldsymbol{x}) \wedge \boldsymbol{n} = \boldsymbol{x}^{\mathsf{L}})\right]$.

5.1 Counting Constraints

As before, to eliminate $\exists \boldsymbol{x}$ from $\exists \boldsymbol{x} : \mathsf{TA}\, \theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y})$, we first puts $\exists \boldsymbol{x} : \mathsf{TA}\, \theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y})$ into the solved form. However, this alone does not suffices as the constraints on the lengths of $\boldsymbol{x}$ may restrict the solution set of $\boldsymbol{x}$.

Example 4. The truth value of $\exists x \exists y : \mathsf{TA}\, (x \neq y \wedge x^{\mathsf{L}} = y^{\mathsf{L}} = 3)$ depends on whether there exists two distinct terms of length 3.

Hence we need to know the number of distinct TA-terms at certain length.

Definition 5 (Counting Constraint). *A counting constraint is a predicate* $\mathsf{CNT}^{\alpha}_{k,n}(x)$ *(*$k > 0, n \geq 0$*) that is true if and only if there are at least* $n+1$ *different* α*-terms of length* x *in* $\mathfrak{A}_{\mathsf{TA}}$ *with* k *constants.* $\mathsf{CNT}_{k,n}(x)$ *is similarly defined with* α*-terms replaced by* TA*-terms.*

Example 5. For $\mathfrak{A}^{\mathbb{Z}}_{\mathsf{list}} = (\mathfrak{A}_{\mathsf{list}}; \mathfrak{A}_{\mathbb{Z}})$ with one constant, $\mathsf{CNT}^{\mathsf{cons}}_{1,n}(x)$ is $x \geq 2m-1 \wedge 2 \nmid m$ where m is the least number such that the m-th Catalan number $C_m = \frac{1}{m}\binom{2m-2}{m-1}$ is greater than n. This is not surprising as C_m gives the number of binary trees with m leaves (that tree has $2m-1$ nodes).

Lemma 1 ([24]). $\mathsf{CNT}^{\alpha}_{k,n}(x)$ *and* $\mathsf{CNT}_{k,n}(x)$ *are expressible in Presburger arithmetic.*

5.2 Equality Completion

Often formulae do not have all information to construct counting constraints. Consider the formula $\exists x : \mathsf{TA}\,(y_1 \neq x \wedge y_2 \neq x \wedge y_1 \neq y_2)$. Without knowing equality relations between lengths of x, y_1 and y_2, we can not find the integer constraint on the length of x. So in order to construct counting constraints, we need equality information between terms and equality information between lengths of terms.

Definition 6 (Equality Completion). *Let S be a set of TA-terms. An* equality completion *θ of S is a formula consisting of the following literals: for any $u, v \in S$, exactly one of $u = v$ and $u \neq v$, and exactly one of $u^{\mathsf{L}} = v^{\mathsf{L}}$ and $u^{\mathsf{L}} \neq v^{\mathsf{L}}$ are in θ.*

Let θ be a conjunction of literals. We say that θ' is an equality completion of θ, if θ' is a conjunction of an equality completion of $\Sigma(\theta)$ and tester literals in θ.

Example 6. Let $\mathsf{ar}(\alpha) = 2$ and θ be $y \neq \alpha(x, z) \wedge \mathsf{Is}_\alpha(y)$, then $\Sigma(\theta) = \{x, y, z, \alpha(x, z)\}$. A possible equality completion of θ is

$$y \neq \alpha(x, z) \wedge y^{\mathsf{L}} = (\alpha(x, z))^{\mathsf{L}} \wedge x^{\mathsf{L}} = z^{\mathsf{L}} \wedge y^{\mathsf{L}} \neq x^{\mathsf{L}} \wedge \bigwedge_{t, t' \in \Sigma(\theta); t \not\equiv t'} t \neq t'. \tag{9}$$

5.3 Clusters

Equality completion is an expensive operation and it is hard to maintain if the subsequent operations generate new terms (as in Alg. 6). Revisiting $\exists x : \mathsf{TA}\,(y_1 \neq x \wedge y_2 \neq x \wedge y_1 \neq y_2)$, it is easily seen that we need to know whether $y_1 = y_2$ only if we have guessed $x^{\mathsf{L}} = y_1^{\mathsf{L}} = y_2^{\mathsf{L}}$. In fact it suffices to have the equality information between terms of the same length. This leads to the notion of clusters.

Definition 7 (Clusters). *Let $[t]$ denote the equivalence class containing t with respect to term equality. We say that $C = \{[t_0], \ldots, [t_n]\}$ is a* cluster *if $t_0, \ldots, t_n$ are pairwise disjoint terms of the same length.*

For notation simplicity we may assume that a cluster is a set with each member being an (arbitrarily chosen) representative of the corresponding equivalence class. A cluster is maximal if no superset of it is a cluster. A cluster C is closed if C is maximal and for any maximal C', $C \cap C' \neq \emptyset \rightarrow C = C'$. Two distinct closed clusters are said to be mutually independent. A cluster is α-typed (called α-cluster) if all of its elements are α-typed. The notions of maximality, closedness and mutually independence naturally generalize to typed clusters. Note that an untyped maximal cluster may contain more than one typed maximal clusters. The size of a cluster is the number of equivalence class in it. The rank of a cluster C, written $\mathsf{rk}(C)$, is the length of its terms. Clusters is partially ordered by their ranks.

Example 7. In Ex. 6, (9) induces two mutually independent clusters $C_1 : \{[x], [z]\}$ and $C_2 : \{[y], [\alpha(x,z)]\}$ in which C_2 is α-typed and we have $\mathsf{rk}(C_1) < \mathsf{rk}(C_2)$. In fact any equality completion induces a set of mutually independent clusters. As another example, the formula

$$x \neq y \wedge x \neq z \wedge x^{\mathsf{L}} = y^{\mathsf{L}} \wedge x^{\mathsf{L}} = z^{\mathsf{L}} \wedge \mathsf{Is}_\alpha(x) \wedge \mathsf{Is}_\alpha(y)$$

gives two maximal clusters $C'_1 : \{x, y\}$ and $C'_2 : \{x, z\}$, in which C'_1 is α-typed but neither of them is closed and their ranks are incomparable.

5.4 Length Constraint Completion

In general, we will meet formula of the form $\exists \boldsymbol{x} : \mathsf{TA}\ \left(\theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y}) \wedge \theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})\right)$, where the lengths of $\boldsymbol{x}$ have been constrained by $\theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})$. For the construction of accurate length constraints for $\boldsymbol{x}$, we need to make $\theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})$ "complete" in the sense defined below.

Definition 8 (Length Constraint Completion). *Let $\theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y})$ be a formula of $\mathscr{L}_{\mathsf{TA}}$ and $\theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})$ be a formula of $\mathscr{L}_{\mathbb{Z}}$. Write $\theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y})$ as $\theta^{(1)}_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y}) \wedge \theta^{(2)}(\boldsymbol{y})$ such that $\theta^{(2)}(\boldsymbol{y})$ does not contain $\boldsymbol{x}$. We say a formula $\Theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})$ is a* completion *of $\theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})$ in $\boldsymbol{x}$ with respect to $\theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y})$ if the following formulae are valid:*

I. $\forall \boldsymbol{y} : \mathsf{TA}\ \forall \boldsymbol{x} : \mathsf{TA}\ \left[\theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y}) \wedge \theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}) \leftrightarrow \theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y}) \wedge \Theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})\right]$.

II. $\forall \boldsymbol{y} : \mathsf{TA}\ \forall \boldsymbol{x}^{\mathsf{L}} : \mathbb{Z}\ \left[\theta^{(2)}(\boldsymbol{y}) \wedge \Theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}) \rightarrow \exists \boldsymbol{x} : \mathsf{TA}\ \left(\theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y}) \wedge \Theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})\right)\right]$.

Example 8. Let $\mathsf{ar}(\alpha) = 2$, $\theta_{\mathsf{TA}}(x, y, z)$ be $\alpha(x, y) = z$ and $\theta_{\mathbb{Z}}(x^{\mathsf{L}}, y^{\mathsf{L}}, z^{\mathsf{L}})$ be $x^{\mathsf{L}} < z^{\mathsf{L}} \wedge y^{\mathsf{L}} < z^{\mathsf{L}}$. Consider the following formulae:

$$\begin{aligned}
\Theta_{\mathbb{Z}} &: x^{\mathsf{L}} + y^{\mathsf{L}} + 1 = z^{\mathsf{L}} \wedge x^{\mathsf{L}} > 0 \wedge y^{\mathsf{L}} > 0,\\
\Theta^1_{\mathbb{Z}} &: x^{\mathsf{L}} < z^{\mathsf{L}} \wedge y^{\mathsf{L}} < z^{\mathsf{L}} \wedge x^{\mathsf{L}} > 0 \wedge y^{\mathsf{L}} > 0,\\
\Theta^2_{\mathbb{Z}} &: x^{\mathsf{L}} + y^{\mathsf{L}} + 1 = z^{\mathsf{L}} \wedge x^{\mathsf{L}} > 5 \wedge y^{\mathsf{L}} > 5.
\end{aligned}$$

It is not hard to argue that $\Theta_{\mathbb{Z}}$ is a completion of $\theta_{\mathbb{Z}}(x^{\mathsf{L}}, y^{\mathsf{L}}, z^{\mathsf{L}})$ in $\boldsymbol{x}$ with respect to $\theta_{\mathsf{TA}}(x, y, z)$. However neither $\Theta^1_{\mathbb{Z}}$ nor $\Theta^2_{\mathbb{Z}}$ is such an completion. Though $\Theta^1_{\mathbb{Z}}$ satisfies [I], it does not satisfies [II], as the assignment $\{x^{\mathsf{L}} = 3, y^{\mathsf{L}} = 3, z^{\mathsf{L}} = 4\}$ can not be realized by any assignment for $\boldsymbol{x}$. On the other hand, $\Theta^2_{\mathbb{Z}}$ satisfies [II], but not [I], as the assignment $\{x = a, y = a, z = \alpha(a, a)\}$, where a is a constant, falsifies $\Theta^2_{\mathbb{Z}}$.

To construct a completion the following predicates are used to describe the length constraints of term tree structures:

$$\begin{aligned}
\mathsf{Tree}(t) &: \exists x_1, \ldots, x_n \geq 0\ \left(\boldsymbol{t}^{\mathsf{L}} = \left(\textstyle\sum_{i=1}^{n} d_i x_i\right) + 1\right),\\
\mathsf{Node}^\alpha(t, \boldsymbol{t}_\alpha) &: \boldsymbol{t}^{\mathsf{L}} = \textstyle\sum_{i=1}^{\mathsf{ar}(\alpha)} t_i^{\mathsf{L}} + 1,\\
\mathsf{Tree}^\alpha(t) &: \exists \boldsymbol{t}_\alpha \left(\mathsf{Node}^\alpha(t, \boldsymbol{t}_\alpha) \wedge \textstyle\bigwedge_{i=1}^{\mathsf{ar}(\alpha)} \mathsf{Tree}(t_i)\right),
\end{aligned}$$

where $\boldsymbol{t}_\alpha$ stands for $t_1, \ldots, t_{\mathsf{ar}(\alpha)}$ and and $d_1, \ldots, d_n$ are the distinct arities of constructors. The predicate $\mathsf{Tree}(t)$ is true if and only if $\boldsymbol{t}^{\mathsf{L}}$ is the length of a well-formed term. The predicate $\mathsf{Node}^\alpha(t, \boldsymbol{t}_\alpha)$ forces the length of an α-term with known children to be the sum of the lengths of its children plus 1. The predicate $\mathsf{Tree}^\alpha(t)$ states the length constraint of a well-formed α-term.

Using these predicates the following algorithm computes the completion of length constraints.

Algorithm 2 (Length Constraint Completion). *Let the input be* $\theta_{\mathsf{TA}}(\boldsymbol{x},\boldsymbol{y}) \wedge \theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}},\boldsymbol{y}^{\mathsf{L}})$, *where* $\theta_{\mathsf{TA}}(\boldsymbol{x},\boldsymbol{y})$ *is a conjunction of literals in* $\mathscr{L}_{\mathsf{TA}}$ *and* $\theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}},\boldsymbol{y}^{\mathsf{L}})$ *is a conjunction of literals in* $\mathscr{L}_{\mathbb{Z}}$. *Initially set* $\Theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}},\boldsymbol{y}^{\mathsf{L}}) = \theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}},\boldsymbol{y}^{\mathsf{L}})$. *For each term t occurring in* $\theta_{\mathsf{TA}}(\boldsymbol{x},\boldsymbol{y})$, *add the following to* $\Theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}},\boldsymbol{y}^{\mathsf{L}})$.

a. $t^{\mathsf{L}} = 1$, *if t is a constant.*
b. $t^{\mathsf{L}} = s^{\mathsf{L}}$, *if $t = s$.*
c. $\mathsf{Tree}(t)$, *if t is untyped.*
d. $\mathsf{Tree}^\alpha(t)$, *if t is α-typed.*
e. $\mathsf{Node}^\alpha(t, \boldsymbol{t}_\alpha)$, *if t is α-typed with children $\boldsymbol{t}_\alpha$.*
f. $\mathsf{CNT}_{k,n}(t^{\mathsf{L}})$, *if t occurs in an untyped clusters of size $n+1$.*
g. $\mathsf{CNT}^\alpha_{k,n}(t^{\mathsf{L}})$, *if t occurs in an α-cluster of size $n+1$.*

Definition 9. *We say* $\theta_{\mathsf{TA}}(\boldsymbol{x},\boldsymbol{y}) \wedge \theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}},\boldsymbol{y}^{\mathsf{L}})$ *is in the* strong solved form *(with respect to $\boldsymbol{x}$) if* $\theta_{\mathsf{TA}}(\boldsymbol{x},\boldsymbol{y})$ *is in the solved form and all literals of the form* $Ly \neq t(\boldsymbol{x},\boldsymbol{y})$, *where* $y \in \boldsymbol{y}$ *and* $t(\boldsymbol{x},\boldsymbol{y})$ *is a constructor term containing $\boldsymbol{x}$, are redundant.*

Example 9. In Ex. 6, (9) is not in the strong solved form. However, it can be made into the strong solved form by adding $\mathsf{s}^\alpha_1 y \neq x$ or $\mathsf{s}^\alpha_2 y \neq z$.

Lemma 2. *If* $\theta_{\mathsf{TA}}(\boldsymbol{x},\boldsymbol{y}) \wedge \theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}},\boldsymbol{y}^{\mathsf{L}})$ *is in strong solved form and induces a set of mutually independent clusters, then* $\Theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}},\boldsymbol{y}^{\mathsf{L}})$ *computed by Alg. 2 is a completion of* $\theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}},\boldsymbol{y}^{\mathsf{L}})$ *in $\boldsymbol{x}$ with respect to* $\theta_{\mathsf{TA}}(\boldsymbol{x},\boldsymbol{y})$.

6 A New Quantifier Elimination Procedure for $\mathsf{Th}(\mathfrak{A}^{\mathbb{Z}}_{\mathsf{TA}})$

In this section we expand Alg. 1 to an elimination procedure for $\mathsf{Th}(\mathfrak{A}^{\mathbb{Z}}_{\mathsf{TA}})$. Since $\mathscr{L}^{\mathbb{Z}}_{\mathsf{TA}}$ has two sorts, namely $\mathbb{Z}$ and TA, we need to show elimination of integer quantifiers as well as term quantifiers.

6.1 Eliminate Quantifiers on Integer Variables

We assume that formulae with quantifiers on integer variables are in the form

$$\exists \boldsymbol{z} : \mathbb{Z} \left(\theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}, \boldsymbol{z}) \wedge \theta_{\mathsf{TA}}(\boldsymbol{x}) \right), \tag{10}$$

where $\boldsymbol{y}, \boldsymbol{z}$ are integer variables and $\boldsymbol{x}$ are term variables. Since $\theta_{\mathsf{TA}}(\boldsymbol{x})$ is in $\mathscr{L}_{\mathsf{TA}}$, we can move $\theta_{\mathsf{TA}}(\boldsymbol{x})$ out of the scope of $\exists \boldsymbol{z}$, obtaining

$$\exists \boldsymbol{z} : \mathbb{Z}\ \theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}, \boldsymbol{z}) \wedge \theta_{\mathsf{TA}}(\boldsymbol{x}). \tag{11}$$

Now $\exists \boldsymbol{z} : \mathbb{Z}\ \theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}, \boldsymbol{z})$ is essentially a Presburger formula and we can proceed to remove the block of existential quantifiers using Cooper's method [5, 20].

6.2 Eliminate Quantifiers on Term Variables

We assume that formulae with quantifiers on term variables are in the form

$$\exists \boldsymbol{x} : \mathsf{TA}\ (\theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y}) \wedge \Psi_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}, \boldsymbol{z})), \tag{12}$$

where $\boldsymbol{x}, \boldsymbol{y}$ are term variables, $\boldsymbol{z}$ are integer variables, and $\Psi_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}, \boldsymbol{z})$ is an arbitrary Presburger formula. The following algorithm is based on Alg. 1.

Algorithm 3. *Input:* $\exists \boldsymbol{x} : \mathsf{TA}\ (\theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y}) \wedge \Psi_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}, \boldsymbol{z}))$.
To save space, we do not list $\Psi_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}, \boldsymbol{z})$ *until needed. Run Alg. 1 up to Step [7]. Apply the following subprocedures sucessively unless noted otherwise.*

1. *Equality Completion: Alg. 4.*
2. *Elimination of Equalities Containing* $\boldsymbol{x}$*: Alg. 5.*
3. *Propagation of Disequalities of the Form* $Ly \neq t(\boldsymbol{x}, \boldsymbol{y})$*: Alg. 6.*
4. *Reduction of Term Quantifiers to Integer Quantifiers: Alg. 7.*

The purpose of Steps [1]- [3] is to transform (12) *to a formula in strong normal form which induces a set of mutually independent clusters. Therefore by Alg. 2 we can construct the length constraint completion for* $\boldsymbol{x}$ *which allows us to reduce term quantifiers to integer quantifiers.*

Algorithm 4 (Equality Completion). *We assume the input formula is in the form (renaming the first part of* (12)*)*

$$\exists \boldsymbol{x} : \mathsf{TA}\ \Big[\bigwedge_i x_{f(i)} \neq t_i(\boldsymbol{x}, \boldsymbol{y}) \wedge \bigwedge_j L_j y_{g(j)} \neq s_j(\boldsymbol{x}, \boldsymbol{y}) \Big], \tag{13}$$

where t_i, s_j *are: (i) quantified variables* $\boldsymbol{x}$*, (ii) parameters* $\boldsymbol{y}$*, (iii) selector terms of parameters in the form* Ly $(y \in \boldsymbol{y})$*, (iv) constants in* C*, or (v) constructor terms built from terms in (i)-(iv). Let* S *be all terms including subterms which appear in* (13)*. Guess an equality completion of* S*. It is easily seen that an equality completion is of the form*

$$\exists \boldsymbol{x} : \mathsf{TA}\ \Big[\bigwedge_i x_{f(i)} \neq t_i(\boldsymbol{x}, \boldsymbol{y}) \wedge \bigwedge_j L_j y_{g(j)} \neq s_j(\boldsymbol{x}, \boldsymbol{y}) \wedge \\ \bigwedge_i x_{f'(i)} = t'_i(\boldsymbol{x}, \boldsymbol{y}) \wedge \bigwedge_j L'_j y_{g'(j)} = s'_j(\boldsymbol{x}, \boldsymbol{y}) \Big]. \tag{14}$$

Algorithm 5 (Elimination of Equalities Containing $\boldsymbol{x}$). *Let* $\mathcal{E}$ *denote the set of equalities containing* $\boldsymbol{x}$*. Exhaustively apply the following subprocedures until* $\mathcal{E}$ *is empty. Pick an* $E \in \mathcal{E}$*.*

A. *E is $x = u(\boldsymbol{x}, \boldsymbol{y})$. Then we know x does not occur in $u(\boldsymbol{x}, \boldsymbol{y})$ and hence we can remove $\exists x$ by substituting $u(\boldsymbol{x}, \boldsymbol{y})$ for all occurrences of x.*
B. *E is $Ly = \alpha(t_1(\boldsymbol{x}, \boldsymbol{y}), \ldots, t_k(\boldsymbol{x}, \boldsymbol{y}))$. Then replace E by*

$$\mathsf{s}_1^{\alpha} Ly = t_1(\boldsymbol{x}, \boldsymbol{y}),\ \ldots,\ \mathsf{s}_k^{\alpha} Ly = t_k(\boldsymbol{x}, \boldsymbol{y}).$$

C. E is $\beta(u_1(\boldsymbol{x},\boldsymbol{y}),\ldots,u_l(\boldsymbol{x},\boldsymbol{y})) = \gamma(u'_1(\boldsymbol{x},\boldsymbol{y}),\ldots,u'_l(\boldsymbol{x},\boldsymbol{y}))$. Then replace E by

$$u_1(\boldsymbol{x},\boldsymbol{y}) = u'_1(\boldsymbol{x},\boldsymbol{y}),\ \ldots,\ u_l(\boldsymbol{x},\boldsymbol{y}) = u'_l(\boldsymbol{x},\boldsymbol{y}).$$

Algorithm 6 (Propagation of Disequalities of the From $Ly \neq t(\boldsymbol{x},\boldsymbol{y})$). *Actually we only need to propagate those disequalities of the form $Ly \neq t(\boldsymbol{x},\boldsymbol{y})$ such that $(Ly)^{\mathsf{L}} = (t(\boldsymbol{x},\boldsymbol{y}))^{\mathsf{L}}$ and $t(\boldsymbol{x},\boldsymbol{y})$ properly contains $\boldsymbol{x}$.*

Let $\mathcal{D}$ denote the set of disequalities of the above form. Exhaustively apply the following subprocedures until $\mathcal{D}$ is empty. Pick $D : Ly \neq \alpha(t_1(\boldsymbol{x},\boldsymbol{y}),\ldots,t_k(\boldsymbol{x},\boldsymbol{y})) \in \mathcal{D}$.

A. Disequality Splitting. *Remove D from $\mathcal{D}$ and add to $\theta_{\mathsf{TA}}(\boldsymbol{x},\boldsymbol{y})$*

$$\neg \mathsf{Is}_\alpha(Ly) \vee \bigvee_{1\le i\le k} \mathsf{s}^\alpha_i Ly \neq t_i(\boldsymbol{x},\boldsymbol{y}).$$

Return if we take $\neg \mathsf{Is}_\alpha(Ly)$; continue otherwise.

B. Length Splitting. *Suppose we take $\mathsf{s}^\alpha_j Ly \neq t_j(\boldsymbol{x},\boldsymbol{y})$ ($1 \le j \le k$). Split on*

$$(\mathsf{s}^\alpha_j Ly)^{\mathsf{L}} = (t_j(\boldsymbol{x},\boldsymbol{y}))^{\mathsf{L}} \vee (\mathsf{s}^\alpha_j Ly)^{\mathsf{L}} \neq (t_j(\boldsymbol{x},\boldsymbol{y}))^{\mathsf{L}}.$$

Return if we take $(\mathsf{s}^\alpha_j Ly)^{\mathsf{L}} \neq (t_j(\boldsymbol{x},\boldsymbol{y}))^{\mathsf{L}}$; continue otherwise.

C. Equality Splitting. *Suppose the cluster of $t_j(\boldsymbol{x},\boldsymbol{y})$ contains $u_0,\ldots,u_n$. Split on*

$$\bigvee_{i\le n} \mathsf{s}^\alpha_j Ly = u_i \vee \bigwedge_{i\le n} \mathsf{s}^\alpha_j Ly \neq u_i$$

In case we take any disjunct $\mathsf{s}^\alpha_j Ly = u_i$, return if u_i does not contain $\boldsymbol{x}$; rerun Alg. 5 otherwise. The last case is that we choose $\bigwedge_{i\le n} \mathsf{s}^\alpha_j Ly \neq u_i$. This in general will increase the size of $\mathcal{D}$ if if for some of $u_i(\boldsymbol{x})$'s are also constructor terms containing $\boldsymbol{x}$, However if this happens, $u_i(\boldsymbol{x})$'s will sit in a cluster whose rank is lower than that of the cluster of $\alpha(t_1(\boldsymbol{x},\boldsymbol{y}),\ldots,t_k(\boldsymbol{x},\boldsymbol{y}))$. So eventually the size of $\mathcal{D}$ will decrease.

Algorithm 7 (Reduction of Term Quantifiers to Integer Quantifiers). *Now we can assume the resulting formula is in the form*

$$\exists \boldsymbol{x} : \mathsf{TA}\left[\theta^{(1)}_{\mathsf{TA}}(\boldsymbol{x},\boldsymbol{y}) \wedge \theta^{(2)}_{\mathsf{TA}}(\boldsymbol{y}) \wedge \theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}},\boldsymbol{y}^{\mathsf{L}}) \wedge \Psi_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}},\boldsymbol{y}^{\mathsf{L}},\boldsymbol{z})\right], \tag{15}$$

where $\theta^{(1)}_{\mathsf{TA}}(\boldsymbol{x},\boldsymbol{y})$ is of the form $\bigwedge_i x_{f(i)} \neq t_i(\boldsymbol{x},\boldsymbol{y})$, $\theta^{(2)}_{\mathsf{TA}}(\boldsymbol{y})$ does not contain $\boldsymbol{x}$, $\theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}},\boldsymbol{y}^{\mathsf{L}})$ is the integer constraint obtained from Algs. 4, 6 (Step [B]), and $\Psi_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}},\boldsymbol{y}^{\mathsf{L}},\boldsymbol{z})$ is the PA formula not listed before for simplicity. Now let $\theta_{\mathsf{TA}}(\boldsymbol{x},\boldsymbol{y})$ denote $\theta^{(1)}_{\mathsf{TA}}(\boldsymbol{x},\boldsymbol{y}) \wedge \theta^{(2)}_{\mathsf{TA}}(\boldsymbol{y})$. Call Alg. 2 to get the completion $\Theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}},\boldsymbol{y}^{\mathsf{L}})$ of $\theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}},\boldsymbol{y}^{\mathsf{L}})$ in $\boldsymbol{x}$ with respect to $\theta_{\mathsf{TA}}(\boldsymbol{x},\boldsymbol{y})$. Now we claim that (15) *is equivalent to*

$$\exists \boldsymbol{x} : \mathsf{TA}\left[\theta^{(1)}_{\mathsf{TA}}(\boldsymbol{x},\boldsymbol{y}) \wedge \theta^{(2)}_{\mathsf{TA}}(\boldsymbol{y}) \wedge \Theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}},\boldsymbol{y}^{\mathsf{L}}) \wedge \Psi_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}},\boldsymbol{y}^{\mathsf{L}},\boldsymbol{z})\right], \tag{16}$$

which in turn is equivalent to

$$\exists \boldsymbol{x}^{\mathsf{L}} : \mathbb{Z}\left[\theta^{(2)}_{\mathsf{TA}}(\boldsymbol{y}) \wedge \Theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}},\boldsymbol{y}^{\mathsf{L}}) \wedge \Psi_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}},\boldsymbol{y}^{\mathsf{L}},\boldsymbol{z})\right]. \tag{17}$$

Lemma 3. *Algs. 4,5 and 6 produces a formula in strong normal form which induces a set of mutually independent clusters.*

Theorem 4. *All transformations in Alg. 3 preserve equivalence.*

Lemma 4. *Alg. 2 computes $\Theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})$ in time $O(n)$.*

Theorem 5. *Alg. 3 eliminates a block of quantifiers in time $2^{2^{O(n)}}$.*

Theorem 6. *$\mathsf{C}_{\mathsf{k}}(\mathfrak{A}^{\mathbb{Z}}_{\mathsf{TA}})$ is decidable in $O(\exp_{2k}(n))$.*

7 Conclusion

We presented new quantifier elimination procedures for the theory of term algebras and for the extended theory with Presburger formulae. The elimination procedures deal with a block of quantifiers of the same type at one step. The complexity is k-fold exponential (resp. $2k$-fold exponential) for the bounded class with quantifier alternation depth k in the theory of term algebras (resp. in the theory of term algebras with length function).

The double exponential complexity is due to propagation of literals of the form $Ly \neq t(\boldsymbol{x}, \boldsymbol{y})$ in a cluster. We believe that more refined length constraint constuction will remove this costly operation.

We plan to apply this methods to the first-order theory of queues [21] and to the first-order theory of Knuth-Bendix order [23].

References

1. Rolf Backofen. A complete axiomatization of a theory with feature and arity constraints. *Journal of Logical Programming*, 24(1&2):37–71, 1995.
2. Hubert Comon and Catherine Delor. Equational formulae with membership constraints. *Information and Computation*, 112(2):167–216, 1994.
3. Hubert Comon and Pierre Lescanne. Equational problems and disunification. *Journal of Symbolic Computation*, 7:371–425, 1989.
4. K. J. Compton and C. W. Henson. A uniform method for proving lower bounds on the computational complexity of logical theories. *Annals of Pure and Applied Logic*, 48:1–79, 1990.
5. D. C. Cooper. Theorem proving in arithmetic without multiplication. In *Machine Intelligence*, volume 7, pages 91–99. American Elsevier, 1972.
6. H. B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, second edition, 2001.
7. J. Ferrante and C. W. Rackoff. *The Computational Complexity of Logical Theories*. Springer-Verlag, 1979.
8. Martin Fürer. The complexity of presburger arithmetic with bounded quantifer alternation depth. *Theoretical Computer Science*, 18:105–111, 1982.
9. Wilfrid Hodges. *Model Theory*. Cambridge University Press, Cambridge, UK, 1993.
10. Konstantin Korovin and Andrei Voronkov. A decision procedure for the existential theory of term algebras with the knuth-bendix ordering. In *Proc. 15th IEEE Symp. Logic in Comp. Sci.*, pages 291 – 302, 2000.

11. Konstantin Korovin and Andrei Voronkov. Knuth-Bendix constraint solving is NP-complete. In *Proceedings of 28th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 2076 of *Lecture Notes in Computer Science*, pages 979–992. Springer, 2001.
12. Viktor Kuncak and Martin Rinard. On the theory of structural subtyping. Technical Report MIT-LCS-TR-879, Massachusetts Institute of Technology, Cambridge, MA 02139, USA, January 2003.
13. Viktor Kuncak and Martin Rinard. The structural subtyping of non-recursive types is decidable. In *Proc. 18th IEEE Symp. Logic in Comp. Sci.*, pages 96–107. IEEE Computer Society Press, 2003.
14. Kenneth Kunen. Negation in logic programming. *Journal of Logic Programming*, 4(4):289–308, 1987.
15. L. Lovász. *Combinatorial Problems and Exercises*. Elsevier, Horth-Holland, 1993.
16. M. J. Maher. Complete axiomatizations of the algebras of finite, rational and infinite tree. In *Proceedings of the Third Annual Symposium on Logic in Computer Science*, pages 348–357. IEEE Computer Society Press, 1988.
17. A. I. Mal'cev. Axiomatizable classes of locally free algebras of various types. In *The Metamathematics of Algebraic Systems, Collected Papers*, chapter 23, pages 262–281. North Holland, 1971.
18. Alberto Martelli and Ugo Montanari. An efficient unification algorithm. *ACM Trans. Prog. Lang. Sys.*, 4(2):258–282, 1982.
19. Derek C. Oppen. Reasoning about recursively defined data structures. *J. ACM*, 27(3), July 1980.
20. C. R. Reddy and D. W. Loveland. Presburger arithmetic with bounded quantifier alternation. In *Proceedings of the 10th Annual Symposium on Theory of Computing*, pages 320–325. ACM Press, 1978.
21. Tatiana Rybina and Andrei Voronkov. A decision procedure for term algebras with queues. *ACM Transactions on Computational Logic*, 2(2):155–181, 2001.
22. Sergei Vorobyov. An improved lower bound for the elementary theories of trees. In *Proc. of the* 13^{th} *Intl. Conference on Automated Deduction*, volume 1104 of *LNCS*, pages 275–287, 1996.
23. Ting Zhang, Henny Sipma, and Zohar Manna. The decidability of the first-order theory of term algebras with knuth-bendix order, 2004. Submitted to CP'04. (Extended version available at theory.stanford.edu/˜tingz/papers/cp04_extended.pdf).
24. Ting Zhang, Henny Sipma, and Zohar Manna. Decision procedures for recursive data structures with integer constraints, 2004. To appear in the Proceedings of the 2^{nd} International Joint Conference on Automated Reasoning. (Extended version available at theory.stanford.edu/˜tingz/papers/ijcar04_extended.pdf).

A Proofs of Lemmas

A.1 Proof of Lemma 2.

Proof. We show that if $\theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y}) \wedge \theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})$ is in strong solved form and induces a set of mutually independent clusters, then $\Theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})$ computed by Alg. 2 is a completion of $\theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})$ in $\boldsymbol{x}$ with respect to $\theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y})$.

Condition [I] is easy to verify. Suppose $\theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y}) \wedge \theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})$ is true. It is easily seen that all conjunctive literals added by Alg. 2 are necessary constraints on the lengths of terms in $\theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y})$. Therefore we will have $\Theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})$, and hence $\theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y}) \wedge \Theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})$. The reverse direction follows directly from the construction of $\Theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})$. To verify Condition [II], we need to show

$$\forall \boldsymbol{y} : \mathsf{TA}\ \forall \boldsymbol{x}^{\mathsf{L}} : \mathbb{Z}\left[\theta^{(2)}(\boldsymbol{y}) \wedge \Theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}) \rightarrow \exists \boldsymbol{x} : \mathsf{TA}\left(\theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y}) \wedge \Theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})\right)\right]. \quad (18)$$

Recall $\theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y})$ is $\theta^{(1)}_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y}) \wedge \theta^{(2)}(\boldsymbol{y})$, and thus it suffices to establish

$$\forall \boldsymbol{y} : \mathsf{TA}\ \forall \boldsymbol{x}^{\mathsf{L}} : \mathbb{Z}\left[\Theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}) \rightarrow \exists \boldsymbol{x} : \mathsf{TA}\left(\theta^{(1)}_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y}) \wedge \Theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})\right)\right], \quad (19)$$

which is written out as

$$\forall \boldsymbol{y} : \mathsf{TA}\ \forall \boldsymbol{z} : \mathbb{Z}\left[\Theta_{\mathbb{Z}}(\boldsymbol{z}, \boldsymbol{y}^{\mathsf{L}}) \rightarrow \exists \boldsymbol{x} : \mathsf{TA}\left(\theta^{(1)}_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y}) \wedge \Theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}) \wedge \boldsymbol{z} = \boldsymbol{x}^{\mathsf{L}}\right)\right], \quad (20)$$

Let $\boldsymbol{s}$ and $\boldsymbol{p}$ be any assignments for $\boldsymbol{y}$ and $\boldsymbol{z}$, respectively, such that

$$\Theta_{\mathbb{Z}}(\boldsymbol{p}, \boldsymbol{s}^{\mathsf{L}}). \quad (21)$$

We are left to prove the satisfiability of

$$\theta^{(1)}(\boldsymbol{x}, \boldsymbol{s}) \wedge \boldsymbol{p} = \boldsymbol{x}^{\mathsf{L}} \wedge \Theta_{\mathbb{Z}}(\boldsymbol{p}, \boldsymbol{s}^{\mathsf{L}}). \quad (22)$$

Now we show how to construct an assignment $\boldsymbol{t}$ such that $\boldsymbol{t}^{\mathsf{L}} = \boldsymbol{p}$ and $\theta^{(1)}(\boldsymbol{t}, \boldsymbol{s})$ holds. Let us order all integer terms $\boldsymbol{x}^{\mathsf{L}}$ (according to $\boldsymbol{p}$) as follows.

$$\underbrace{(u^{(1)}_0)^{\mathsf{L}} = \ldots = (u^{(1)}_{n_1})^{\mathsf{L}}}_{block\ 1} < \underbrace{(u^{(2)}_0)^{\mathsf{L}} = \ldots = (u^{(2)}_{n_2})^{\mathsf{L}}}_{block\ 2} < \ldots < \underbrace{(u^{(j)}_0)^{\mathsf{L}} = \ldots = (u^{(j)}_{n_j})^{\mathsf{L}}}_{block\ j}$$

Denote by l_i the length of terms in the i^{th} block. Note that for any TA-term t occurring in $\theta^{(1)}(\boldsymbol{x}, \boldsymbol{y})$, $\boldsymbol{t}^{\mathsf{L}}$ appears in $\Theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})$ and hence in the above sequence. In general a block can contain more than one clusters. However, by Lemma 3 clusters are mutually independent, therefore, it is no harm to assume each block as a single maximal cluster (which, however, may contain more than one maximal typed clusters).

Beginning with terms in the 1^{th} block, namely $u^{(1)}_1, \ldots, u^{(1)}_{n_1}$, we incrementally construct a satisfying assignment for $\theta^{(1)}(\boldsymbol{x}, \boldsymbol{s})$. We only need to consider $u^{(j)}_{n_j}$ which contains $\boldsymbol{x}$. Obviously $u^{(1)}_i$ $(0 \le i \le n_1)$ is either a constant or a variable in $\boldsymbol{x}$

as its length is the smallest. By Alg. 2 we know that $\mathsf{CNT}_{k,n_1}(l_1)$ is in $\Theta^{\mathsf{L}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})$. As $\boldsymbol{p}$ satisfies $\Theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})$, there are at least n_1+1 different terms at length l_1. Therefore we can simply assign each $u_i^{(1)}$ (if it is a variable) a distinct TA-term. In case $u_i^{(1)}$ is asserted to be α-typed, then $\mathsf{CNT}^{\alpha}_{k,n_1}(l_1)$ should also be present in $\Theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})$, and hence similar arguemnt applies.

Now suppose that all variables in the i^{th} block have been assigned. Consider the $(i+1)^{th}$ block. At this time values of all non-variable terms in the $(i+1)^{th}$ block have been determined. This lies with the fact that $\boldsymbol{x}$ only appear in constructor terms, and hence have been assigned values by the i^{th} round. For example, suppose that $t(x)$ is a constructor term in the $(i+1)^{th}$ block. Since $x^{\mathsf{L}} < (t(x))^{\mathsf{L}}$, x was assigned by the i^{th} round and so is the value of $t(x)$. By the same argument as before, due to the presence of $\mathsf{CNT}_{k,n_{i+1}}(l_{i+1})$ in $\Theta^{\mathsf{L}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})$, we are able to assign each variable in $(i+1)^{th}$ block a different term of length l_{i+1}.

Note that the variable assignment in each round will not create any equality between terms in a block, i.e., will not violate any disequality in a cluster. The only problematic case is that a cluster may have selector terms containing $\boldsymbol{y}$ as well as constructor terms containing $\boldsymbol{x}$. For example, suppose that a cluster C in the i^{th} block contains both Ly and $\alpha(t_1(\boldsymbol{x}, \boldsymbol{y}), \ldots, t_k(\boldsymbol{x}, \boldsymbol{y}))$ for $\mathsf{ar}(\alpha) = k$. We know that $[\![Ly]\!]$ is fixed, $[\![t_1(\boldsymbol{x}, \boldsymbol{y})]\!], \ldots, [\![t_k(\boldsymbol{x}, \boldsymbol{y})]\!]$ have been determined before the i^{th} round and so is $[\![\alpha(t_1(\boldsymbol{x}, \boldsymbol{y}), \ldots, t_k(\boldsymbol{x}, \boldsymbol{y}))]\!]$. By the virtue of Step [A] we know that for some i $(1 \le i \le k)$, $\mathsf{s}^{\alpha}_i Ly \neq t_i(\boldsymbol{x}, \boldsymbol{y})$ must be in $\theta^{(1)}(\boldsymbol{x}, \boldsymbol{y})$. Since $t_i(\boldsymbol{x}, \boldsymbol{y})^{\mathsf{L}} < (\alpha(t_1(\boldsymbol{x}, \boldsymbol{y}), \ldots, t_k(\boldsymbol{x}, \boldsymbol{y})))^{\mathsf{L}}$, $t_i(\boldsymbol{x}, \boldsymbol{y})$ has been assigned before the i^{th} round such that $[\![\mathsf{s}^{\alpha}_i Ly]\!] \neq [\![t_i(\boldsymbol{x}, \boldsymbol{y})]\!]$. As a consequence we will have $[\![Ly]\!] \neq [\![\alpha(t_1(\boldsymbol{x}, \boldsymbol{y}), \ldots, t_k(\boldsymbol{x}, \boldsymbol{y}))]\!]$. That is what Alg. 6 is about, to make disequalities of the form $Ly \neq \alpha(t_1(\boldsymbol{x}, \boldsymbol{y}), \ldots, t_k(\boldsymbol{x}, \boldsymbol{y}))$ redundant.

Since at each step we can build a satisfying partial assignment for $\boldsymbol{x}$, by induction, we can eventually construct a satisfying assignment $\boldsymbol{t}$ such that $\theta^{(1)}(\boldsymbol{t}, \boldsymbol{s})$ and $\boldsymbol{t}^{\mathsf{L}} = \boldsymbol{p}$. □

A.2 Proof of Lemma 3

Proof. It suffices to show every run of Alg. 6 preserves closedness. First note that the only thing which can destroy the closedness of a cluster C is the newly generated disequalities of the from $Ly \neq x$ where $x \in C$. Suppose $C = \{t_1, \ldots, t_n\}$. If we take $(Ly)^{\mathsf{L}} \neq x^{\mathsf{L}}$ at Step [B] then $Ly \notin C$ and hence x can not be shared by C and any other clusters. Otherwise, if we take $Ly = t_i$ (for some $1 \le i \le n$) at Step [C], then Ly is "absorbed" by C. The last case is that we take $\bigwedge_{1 \le i \le n} Ly \neq t_i$. But this just makes $C \cup \{Ly\}$ a new cluster since we already have $\mathsf{rk}(C) = (Ly)^{\mathsf{L}}$. □

A.3 Proof of Lemma 4.

Proof. Obviously the computations of predicates [a]-[d] can be done in $O(n)$. Note that the sum of sizes of all clusters is of $O(n)$ and for a cluster of size r we only need to compute $\mathsf{CNT}_{k,r}(x)$ and $\mathsf{CNT}^{\alpha}_{k,r}(x)$ (for each constructor α) once. So it suffices to show that $\mathsf{CNT}_{k,n}(x)$ and $\mathsf{CNT}^{\alpha}_{k,n}(x)$ can be expressed by Presburger formulae which can be computed in time $O(n)$.

Suppose that L_{TA} has m constructors and let d_i $(1 \le i \le m)$ be the i^{th} arity. Let $f(p)$ be the number of distinct term trees of length p. For $p > 1$, we have the following recurrence relation.

$$f(p) = \sum_{h=1}^{m} \sum_{i_1+\ldots+i_{d_h}=p-1} \prod_{j=1}^{d_h} f(i_j). \tag{23}$$

The reason is as follows: there are m possible ways to label the root of a tree; for a root with d children whose lengths are $i_1, \ldots, i_d$, respectively, there are $\prod_{j=1}^{d} f(i_j)$ combinations. By dynamic programming, we can compute $f(1), f(2), \ldots$ in the bottom-up fashion. On the way we will find the first l such that $f(l) > n$ and we take $\mathsf{CNT}_{k,n}(x)$ be $x \ge l$. Similarly we can get $\mathsf{CNT}^{\alpha}_{k,n}(x)$. Let d be the maximum arity. Since there are $O(p^{d-1})$ different sequences of positive numbers whose sum is $p-1$, $f(l)$ can be obtained by $O(l^d)$ arithmetic operations. As $f(p)$ grows exponentially in p, l is at the scale of $O(\log n)$. Moreover, as all integers in the computation is less than n, any arithmetic operation costs time $O(\log n)$. Therefore the search for such l can be done in $O(n)$. $\square$

Remark 1. Note that in the proof we made a simplification by assuming that a tree can grow to have the next legitimate length. With this assumption, we have for any $x \ge l$, $f(x) > n$, and hence $\mathsf{CNT}^{\alpha}_{k,n}(x)$ is just $x \ge l$. However, it is not the case in general. To see this, let us consider an extreme example; a language with only one constant and with two constructors of arity 3 and 1000, respectively. Let $n' = f(1000)$. With no doubt, there are so many distinct trees of length 1000 (constructed using solely the constructor of arity 3), and so n' is a very huge number. But there is only one tree of length 1001 simply because 1000 can not be divided by 3 and so such a tree has to be constructed using the constructor of arity 1000. So 1000 is the least number such that $f(1000) > n' - 1$ while $f(1001) \ll n' - 1$. It should be noted that such an anomaly very unlikely happens in practice. Anyway, we can still have the $O(n)$ bound using the same argument as in [24]. Note that the length function defined in [24] is slightly different from the one in this paper.

B Proofs of Theorems

B.1 Proof of Theorem 1.

Proof. The soundness of most transformations in Alg. 1 is straightforward except the equivalence between (6) and (8). This amounts to show validity of (7). Let rewrite (7) as follows.

$$\forall \boldsymbol{y}\, \exists \boldsymbol{x} : \Big[\bigwedge_i x_{f(i)} \neq t_i(\boldsymbol{x}, \boldsymbol{y}) \wedge \bigwedge_j G_j y_{g(j)} \neq s_j(\boldsymbol{x}, \boldsymbol{y}) \Big]. \tag{24}$$

We need to show the satisfiability of

$$\bigwedge_i x_{f(i)} \neq t_i(\boldsymbol{x}, \boldsymbol{b}) \wedge \bigwedge_j G_j b_{g(j)} \neq s_j(\boldsymbol{x}, \boldsymbol{b}), \tag{25}$$

for any arbitrary sequence $\boldsymbol{b}$ of fixed TA-terms. Let us call variables asserted to be constants refrained variables. For simplicity we assume that all constructors have the same arity. The general case can be proved via minor modifications of the following argument.

First we consider the case that $\mathscr{L}_{\mathsf{TA}}$ has finitely many constants. Let n be the size of $\boldsymbol{x}$ and ξ the maximal length of terms (not containing $\boldsymbol{x}$) in (25). Let ρ be the number such that $\rho > \xi$ and there exists n distinct non-constant terms $\boldsymbol{d} = d_1, \ldots, d_n$ of length ρ. This can be done as none of x_i is refrained variable (see Step [6]). Now we claim that $\boldsymbol{d}$ satisfies (25). Let us assume that s_j contains $\boldsymbol{x}$, otherwise $G_j y_{g(j)} \neq s_j(\boldsymbol{x}, \boldsymbol{y})$ has been moved out of the scope of $\exists \boldsymbol{x}$. Since s_j is a constructor term or a variable in $\boldsymbol{x}$ and G_j is a selector sequence, $\mathsf{len}(G_j b_{g(j)}) \leq \xi$ while $\mathsf{len}(s_j(\boldsymbol{d}, \boldsymbol{b})) \geq \rho > \xi$. Therefore for all j, $G_j b_{g(j)} \neq s_j(\boldsymbol{x}, \boldsymbol{b})$. Next let us assume that t_i are constructor terms containing $\boldsymbol{x}$. Then $\mathsf{len}(d_{f(i)}) = \rho$ but $\mathsf{len}(t_i(\boldsymbol{d}, \boldsymbol{b})) > \rho$. Similarly we can show $\mathsf{len}(d_{f(i)}) \neq \mathsf{len}(t_i(\boldsymbol{d}, \boldsymbol{b}))$ for the cases where t_i is either a variable or a constant. As a consequence, for each i, $d_{f(i)} \neq t_i(\boldsymbol{d}, \boldsymbol{b})$.

If $\mathscr{L}_{\mathsf{TA}}$ has infinitely many constants, then it is not hard to argue that all disequalities containing refrained variables can be satisfied simultaneously without any restrictions on the solution set of unrefrained variables. Therefore we can ignore disequalities containing refrained variables and apply the aforementioned arguement. □

B.2 Proof of Theorem 2.

Proof. First note that we need not to be concerned with the increase of the matrix size by the substitution, since we can represent a conjunction of literals efficiently using Directed Acyclic Graph (DAG). In a DAG representation substitution can simply be done by rearranging edges in the graph. For example, consider the following sequence of formulae

$$x_1 = \alpha(x_2, x_2),\ x_2 = \alpha(x_3, x_3),\ \ldots,\ x_n = \alpha(x_{m+1}, x_{m+1})$$

Instead of generating a formula of size $O(2^m)$, the substitution only gives a linear "double-edged" path from x_1 to x_{m+1}. For details see [19].

We analyze each step of [1]-[7]. Let n be the size of the matrix. Step [1] (type completion) can be done in 2^n as for a selector block of length n there are at most 2^n combinations of tester literals. Step [2] (selector elimination) can be done in $O(n)$ as a selector term can be transformed to a formula in the constructor language in linear size. Step [3] (elimination of complex equalities) only takes time $O(n)$ and it does not increase the size of the matrix. Step [4] (elimination of complex disequalities) renders at most $2^{O(n)}$ disjuncts. Step [5] (elimination of parameters in equalities) transforms constructor terms to sets of selector terms. With introduction of new existential quantifiers, this can be done in $O(n)$. For example, for $\mathsf{ar}(\alpha) = 2$, the solved form of $y = \alpha(\alpha(x_1, x_2), x_3)$ is

$$\exists v(\mathsf{s}_1^\alpha(y) = v \wedge \mathsf{s}_1^\alpha(v) = x_1 \wedge \mathsf{s}_2^\alpha(v) = x_2 \wedge \mathsf{s}_2^\alpha(y) = x_3).$$

Step [6] (constant elimination) renders at most $2^{O(n)}$ disjuncts. Put all together, Steps [1]-[7] generate $2^{O(n)}$ disjuncts each of which has length $O(n)$. □

B.3 Proof of Theorem 3

Proof. Immediate by Thm. 2.

B.4 Proof of Theorem 5

Proof. By Thm. 2 we only need to show that Algs. 4-7 can be done in $2^{O(n^2 \log n)}$. Alg. [4] takes $2^{O(n \log n)}$. It follows from the fact that an equality completion corresponds to a valid product of a partition of terms on syntactic equality and a partition of terms on length equality. For a set of size n the number of distinct partitions is called Bell number, denoted by $B(n)$. An asymptotical expression for $B(n)$ is $\frac{1}{\sqrt{n}}\rho(n)^{n+\frac{1}{2}}e^{\rho(n)-n-1}$, where $\rho(n)$ is implicitly defined by $\rho(n)\log\rho(n) = n$ ([15]). Obviously $B(n)$ is bounded by $2^{O(n \log n)}$, and so is the number of equality completions. Alg. 5 is obviously in P. However Alg. 6 costs double exponential time. To see the reason, let us run Alg. 6 to all qualified disequalities in parallel.

At the beginning of Alg. 6, the matrix of (13) induces at most n clusters, where n is the size of (13). Therefore the height of the decision tree is bounded by n.

The run of Alg. 6 at each level generates double number of selector terms than the previous level. So at the bottom of the decision tree, there would be $O(2^n)$ newly generated selector terms. And splittings will give $O(2^{2^n})$ disjuncts each of which is of length $O(2^n)$.

By Lemma 4, $\Theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})$ obtained from Alg. 2 has size $O(2^n)$. So are the sizes of Presburger formulae obtained from Alg. 7. □

B.5 Proof of Theorem 4

Proof. Following Thm. 1, we only need to show the equivalence of (15) and (16), and the equivalence of (16) and (17). By Lemma 3 we know that $\theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y}) \wedge \theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})$ induces a set of mutually independent clusters. Then the equivalence of (15) and (16) follows directly from Lemma 2 (see [I] in Def. 8). Clearly, (16) implies (17). To show that (17) implies (16), it suffices to establish

$$\forall \boldsymbol{y} : \mathsf{TA} \left(\exists \boldsymbol{x}^{\mathsf{L}} : \mathbb{Z} \left[\theta^{(2)}_{\mathsf{TA}}(\boldsymbol{y}) \wedge \Theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}) \wedge \Psi_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}, \boldsymbol{q}) \right] \rightarrow \right.$$
$$\left. \exists \boldsymbol{x} : \mathsf{TA} \left[\theta^{(1)}_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y}) \wedge \theta^{(2)}_{\mathsf{TA}}(\boldsymbol{y}) \wedge \Theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}) \wedge \Psi_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}, \boldsymbol{q}) \right] \right), \quad (26)$$

where $\boldsymbol{q}$ is an arbitrary assignment for $\boldsymbol{z}$. Recall (26) actually stands for

$$\forall \boldsymbol{y} : \mathsf{TA}\ \forall \boldsymbol{n} : \mathbb{Z} \left[\theta^{(2)}_{\mathsf{TA}}(\boldsymbol{y}) \wedge \Theta_{\mathbb{Z}}(\boldsymbol{n}, \boldsymbol{y}^{\mathsf{L}}) \wedge \Psi_{\mathbb{Z}}(\boldsymbol{n}, \boldsymbol{y}^{\mathsf{L}}, \boldsymbol{q}) \rightarrow \right.$$
$$\left. \exists \boldsymbol{x} : \mathsf{TA} \left(\theta^{(1)}_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y}) \wedge \theta^{(2)}_{\mathsf{TA}}(\boldsymbol{y}) \wedge \Theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}) \wedge \Psi_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}, \boldsymbol{q}) \wedge \boldsymbol{x}^{\mathsf{L}} = \boldsymbol{n} \right) \right], \quad (27)$$

which can be obtained if we have

$$\forall \boldsymbol{y} : \mathsf{TA}\ \forall \boldsymbol{n} : \mathbb{Z} \left[\theta^{(2)}_{\mathsf{TA}}(\boldsymbol{y}) \wedge \Theta_{\mathbb{Z}}(\boldsymbol{n}, \boldsymbol{y}^{\mathsf{L}}) \rightarrow \right.$$
$$\left. \exists \boldsymbol{x} : \mathsf{TA} \left(\theta^{(1)}_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y}) \wedge \theta^{(2)}_{\mathsf{TA}}(\boldsymbol{y}) \wedge \Theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}) \wedge \boldsymbol{x}^{\mathsf{L}} = \boldsymbol{n} \right) \right]. \quad (28)$$

Recall $\theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y})$ is $\theta^{(1)}_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y}) \wedge \theta^{(2)}_{\mathsf{TA}}(\boldsymbol{y})$. Then (28) follows from Lemma 2 (also see [II] in Def. 8). □

B.6 Proof of Theorem 6

Proof. Consider a formula with quantifier alternation depth k. After eliminating $k-1$ blocks of quantifiers, we obtain a formula of length $O(\exp_{2k-2}(n)$. Since there is no parameters left in the last round elimination, we do not need to run the costly Alg. 6. Therefore finally we obtain $O(\exp_{2k-1}(n)$ disjunts each of which is of length $O(\exp_{2k-2}(n)$, and either is a ground TA-formula or a PA-formula with quantifier alternation depth k. The elimination of integer variables takes double exponential time [20]. Therefore the overall cost is $O(\exp_{2k}(n))$. Note that if $k = 1$, Alg. 3 only cost $O(2^n)$ as the existential theory of PA is decidable in $O(2^n)$. □