Term Algebras with Length Function and Bounded Quantifier Elimination

Ting Zhang, Henny B. Sipma, Zohar Manna
Stanford University
tingz,sipma,zm@cs.stanford.edu
Motivation: Program Verification

- Term algebras can model a wide range of tree-like data structures.
- To verify programs we need to reason about these data structures.
- Programming languages often involve multiple data domains, resulting in verification conditions that span multiple theories.
- Common “mixed” constraints are combinations of data structures with integer constraints on the size of those structures.
Bounded Quantifier Elimination

In Theory:
- Term algebras have nonelementary time complexity \([FR79]\).
- The complexity lower bound remains the same for any sub-theories of term algebras \([CL89, \text{Vor96}]\).

In Practice:
- We rarely deal with formulae with a large quantifier alternation depth.
- Therefore it is worthwhile to investigate the “bounded class” of formulae.

Previous Work:
- We gave a quantifier-elimination procedure for the extended theory \([ZSM04b]\).
- But no complexity upper bound is established.
Introduction

Motivation

BQE

Outline

Term Algebras

A Quantifier Elimination Procedure for Term Algebras

Term Algebras with Length Function

A Quantifier Elimination Procedure for Term Algebras with Length Function

Complexity

Future work
Term Algebras

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

1. **TA**: The term domain.
2. **A**: A finite set of constants: $a, b, c, \ldots$
3. **C**: A finite set of constructors: $\alpha, \beta, \gamma, \ldots$
4. **S**: A finite set of selectors. For a constructor $\alpha$ with arity $k$, there are $k$ selectors $s_{1}^{\alpha}, \ldots, s_{k}^{\alpha}$ in $S$.
5. **T**: A finite set of testers. For each constructor $\alpha$ there is a corresponding tester $t_{s_{1}^{\alpha}}^{\alpha}$.

☞ Two Properties:

- The data domain is the set of data objects generated exclusively by applying constructors.
- Each data object is uniquely generated.
Definitions and Notations

- \( \alpha = (s_1^\alpha, \ldots, s_k^\alpha) \) means that \( \alpha \) is a constructor with \( \text{ar}(\alpha) = k \) and \( s_1^\alpha, \ldots, s_k^\alpha \) are the corresponding selectors of \( \alpha \).

- A term \( t \) is a **constructor term** (\( C \)-term) if the outmost function symbol of \( t \) is a constructor.

- A term \( t \) is a **selector term** (\( S \)-term) if the outmost function symbol of \( t \) is a selector.

- We assume that no constructor term appears inside selectors as simplification can always be done. For example,
  \[
  s_i^\alpha(\alpha(x_1, \ldots, x_k)) \quad \text{simplifies to} \quad x_i.
  \]

- \( L, M, N, \ldots \) denote selector sequences. For \( L = s_1, \ldots, s_n \), \( Lx \) stands for
  \[
  s_1(\ldots(s_n(x)\ldots)).
  \]

- A selector term \( s_i^\alpha(t) \) is called **proper** if \( l_s^\alpha(t) \) holds.
Axiomatization of Term Algebras

- **Construction vs. selection.**

  \[
  s_i^\alpha(x) = y \iff \exists \bar{z}_\alpha(\alpha(\bar{z}_\alpha) = x \land y = z_i) \lor (\forall \bar{z}_\alpha(\alpha(\bar{z}_\alpha) \neq x) \land x = y).
  \]

- **Unification closure.**

  \[
  \alpha(x_\alpha) = \alpha(y_\alpha) \rightarrow \bigwedge_{1 \leq i \leq \text{ar}(\alpha)} x_i = y_i.
  \]

- **Acyclicity.**

  \( t(x) \neq x \), if \( t \) is built solely by constructors and \( t \) properly contains \( x \).

- **Uniqueness.**

  \( \alpha(x_\alpha) \neq \beta(y_\beta) \), \( a \neq b \), and \( a \neq \alpha(x_\alpha) \), if \( a \) and \( b \) are distinct atoms and if \( \alpha \) and \( \beta \) are distinct constructors.

- **Domain closure.**

  \[
  \text{Is}_\alpha(x) \iff \exists \bar{z}_\alpha \alpha(\bar{z}_\alpha) = x, \quad \text{Is}_A(x) \iff \bigwedge_{\alpha \in C} \neg \text{Is}_\alpha(x).
  \]
Example: LISP lists

Signature:

\[ \langle \text{list}; \{\text{nil}\}; \{\text{cons}\}; \{\text{car}, \text{cdr}\}; \{\text{ls}_A, \text{ls}_{\text{cons}}\} \rangle \]

Axioms:

1. \( \text{ls}_A(x) \leftrightarrow \neg \text{ls}_{\text{cons}}(x) \),  
2. \( \text{car}(\text{cons}(x, y)) = x \),  
3. \( \text{cdr}(\text{cons}(x, y)) = y \),  
4. \( \text{ls}_A(x) \leftrightarrow \{\text{car}, \text{cdr}\}^+(x) = x \),  
5. \( \text{ls}_{\text{cons}}(x) \leftrightarrow \text{cons}(\text{car}(x), \text{cdr}(x)) = x \).

Formulas:

- \( \text{cons}(y, z) = \text{cons}(\text{cdr}(x), z) \rightarrow \text{cons}(\text{car}(x), y) = x \) (valid).
- \( x = \text{cons}(y, y) \rightarrow \text{cons}(\text{car}(x), y) = x \) (valid).
Quantifier Elimination Preliminary

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

\[ \exists x(A_1(x) \land \ldots \land A_n(x)), \]  

where \( A_i(x) \) (\( 1 \leq i \leq n \)) are literals [Hod93].

- We can also assume that \( A_i's \) are not of the form \( x = t \) as

\[ \exists x(x = t \land \theta(x, y)) \]

simplifies to

- \( \theta(t, y) \), if \( x \) does not occur in \( t \);
- \( \exists x\theta(x, y) \), if \( t \equiv x \);
- false, if \( t \) is a constructor term properly containing \( x \).
Definition 2 (Solved Form) \( \exists x \theta_{TA}(x, y) \) is in the solved form (with respect to \( x \)),

if \( x \) are not in equalities, not asserted to be constants and not inside selector terms.

General Idea:

☞ An existential formula in solved form has solutions under any instantiation of parameters.

Procedure Outline:

☞ A sequence of equivalence-preserving transformations will bring the input formula into a disjunction of formulae in the solved form.

☞ The whole block of existential quantifiers \( \exists x \) can be eliminated by removing all literals containing \( x \) in the matrix.
Quantifier Elimination for Term Algebras

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

- **Guess a type completion of** $\theta(x, y)$.
- **Eliminate selector terms containing** $x$.
- **Decompose relations between constructor terms**.
- **Solve equalities of the form** $Ly = t(x, y)$.
- **Eliminate variables asserted to be constants**.
- **Eliminate quantifiers and all literals containing** $x$. 
Type Completion

Definition 3  \( \Phi' \) is a type completion of \( \Phi \) if \( \Phi' \) is obtained from \( \Phi \) by adding tester predicates such that

\[
\text{for any term } s(t) \text{ either } Is_{\alpha}(t) \text{ (for some constructor } \alpha \text{) or } Is_A(t) \text{ is present in } \Phi'.
\]

Example 1  A possible type completion for \( y = \text{car} (\text{cdr}(x)) \) is

\[
y = \text{car} (\text{cdr}(x)) \land Is_{\text{cons}}(x) \land Is_A (\text{cdr}(x)).
\]

With this type information, \( y = \text{car} (\text{cdr}(x)) \) simplifies to

\[
y = \text{cdr}(x).
\]

\( \bowtie \) Guess a type completion of \( \theta(x, y) \) and simplify every selector term to a proper one.
Eliminate $S$-terms Containing $x$’s.

Replace all selector terms containing $x$ by the corresponding equivalent constructor terms.

**Example 2** Let $\alpha = (s_1^\alpha, s_2^\alpha)$.

$$\exists x (s_1^\alpha x = y \land \varphi(x, y))$$ can be rewritten as

$$\exists x_1 \exists x_2 (x_1 = y \land \varphi(\alpha(x_1, x_2), y)).$$

Similarly, $$\exists x (s_1^\alpha x \neq y \land \varphi(x, y))$$ becomes

$$\exists x_1 \exists x_2 (x_1 \neq y \land \varphi(\alpha(x_1, x_2), y)).$$

☞ It may increase the number of existential quantifiers, but leaves parameters unchanged.

☞ In the following transformations, $x$ never appear inside selector terms.
Decompose Relations between \( C \)-Terms.

- Replace

\[ \alpha(t_1, \ldots, t_i) = \alpha(t'_1, \ldots, t'_i) \]  \hspace{1cm} (2)

by

\[ \bigwedge_{1 \leq i \leq k} t_i = t'_i. \]

Repeat until no equality of the form (2) appears.

- Replace

\[ \alpha(t_1, \ldots, t_i) \neq \alpha(t'_1, \ldots, t'_i) \]  \hspace{1cm} (3)

by

\[ \bigvee_{1 \leq i \leq k} t_i \neq t'_i. \]

Repeat until no equality of the form (3) appears.
Solve equalities of the form $Ly = t(x, y)$, where

1. $L$ is a block of selectors,
2. $t(x, y)$ is a constructor term containing $x$.

The result is a set of equations in terms of $Ly$ in the selector language.

**Example 3** Suppose that $\alpha = (s_{1}^{\alpha}, s_{2}^{\alpha})$. The solution set of

$$s_{2}^{\alpha}y = \alpha(\alpha(x_{1}, y_{1}), y_{2})$$

is

$$x_{1} = s_{1}^{\alpha}s_{1}^{\alpha}s_{2}^{\alpha}y, \quad y_{1} = s_{2}^{\alpha}s_{1}^{\alpha}s_{2}^{\alpha}y, \quad y_{2} = s_{2}^{\alpha}s_{2}^{\alpha}y.$$
Eliminate Variables Asserted to Be Constant

Instantiate \( x \) to each constant to eliminate \( \exists x \) if \( x \) is asserted to be an atom. I.e.,

\[
\exists x (\text{ls}_C(x) \land \varphi(x)) \Rightarrow \bigwedge_{a \in C} \varphi(a).
\]
Eliminate Literals Containing $x$’s.

Now we can assume formulae are in the form

$$\exists x : \left[ \bigwedge_{i} x f(i) \neq t_{i}(x, y) \land \bigwedge_{i} G_{i} y g(i) \neq s_{i}(x, y) \right] \land$$

$$\bigwedge_{i} G'_{i} y g'(i) \neq s'_{i}(y) \land \bigwedge_{i} H_{i} y h(i) = H'_{i} y h'(i). \quad (4)$$

Since

$$\exists x : \left[ \bigwedge_{i} x f(i) \neq t_{i}(x, y) \land \bigwedge_{i} G_{i} y g(i) \neq s_{i}(x, y) \right] \quad (5)$$

is in solved form and hence valid, (4) is equivalent to

$$\bigwedge_{i} G'_{i} y g'(i) \neq s'_{i}(y) \land \bigwedge_{i} H_{i} y h(i) = H'_{i} y h'(i). \quad (6)$$
Language and Structure

Presburger arithmetic (PA): $\mathcal{L}_\mathbb{Z}$, $\mathcal{A}_\mathbb{Z}$.

Two-sorted language $\Sigma = \Sigma_{TA} \cup \Sigma_{\mathbb{Z}} \cup \{(.)^L\}$:

1. $\Sigma_{TA}$: signature of term algebras.
2. $\Sigma_{\mathbb{Z}}$: signature of Presburger arithmetic.
3. $(.)^L : TA \to \mathbb{N}$, the length function defined by

$$t^L = \begin{cases} 
1 & \text{if } t \text{ is an atom}, \\
\sum_{i=1}^{k} t_i^L + 1 & \text{if } t \equiv \alpha(t_1, \ldots, t_k).
\end{cases}$$

$t^L$: generalized integer terms.
### Counting Constraints

**Definition 4 (Counting Constraint)** A *counting constraint* is a predicate \( \text{CNT}^\alpha_{k,n}(x) \) \((k > 0, n \geq 0)\) that is **true** if and only if there are at least \(n+1\) different \(\alpha\)-terms of length \(x\) in \(\mathcal{A}_{TA}\) with \(k\) constants. \(\text{CNT}^\alpha_{k,n}(x)\) is similarly defined with \(\alpha\)-terms replaced by TA-terms.

**Example 4** For \(\mathcal{A}^\mathbb{Z}_{\text{list}} = (\mathcal{A}_{\text{list}}; \mathcal{A}^\mathbb{Z})\) with one constant,

\[
\text{CNT}^\text{cons}_{1,n}(x) \text{ is } x \geq 2m - 1 \land 2 \nmid x
\]

where \(m\) is the least number such that the \(m\)-th **Catalan number** \(\mathcal{C}_m = \frac{1}{m} \binom{2m-2}{m-1}\) is greater than \(n\).

Reason: \(\mathcal{C}_m\) gives the number of binary trees with \(m\) leaves (that tree has \(2m - 1\) nodes).
Equality Completion

In order to construct counting constraints, we need equality information between terms and equality information between lengths of terms.

**Definition 5 (Equality Completion)** Let $S$ be a set of TA-terms. An equality completion $\theta$ 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 $u^L = v^L$ and $u^L \neq v^L$ are in $\theta$.

**Example 5** Let $\alpha = (s_1^\alpha, s_2^\alpha)$ and $\theta$ be

$$y \neq \alpha(x, z) \wedge Is_\alpha(y).$$

A possible equality completion of $\theta$ is

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

(7)
Clusters

**Definition 6 (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 unequal terms of the same length.

- 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**.
- The **rank** of a cluster $C$, written $\text{rk}(C)$, is the length of its terms.
Clusters: Examples

Example 6  In Ex. 5, formula (7) induces two mutually independent clusters

\[ C_1 : \{[x], [z]\} \text{ and } C_2 : \{[y], [\alpha(x, z)]\} \]

with \( \text{rk}(C_1) < \text{rk}(C_2) \).

Example 7  The formula

\[ x \neq y \land x \neq z \land x^L = y^L \land x^L = z^L \land \text{ls}_\alpha(x) \land \text{ls}_\alpha(y) \]

gives two maximal clusters

\[ C'_1 : \{x, y\} \text{ and } C'_2 : \{x, z\}. \]

However, neither \( C'_1 \) nor \( C'_2 \) is closed and their ranks are incomparable.

Any equality completion induces a set of mutually independent clusters.
Length Constraint Completion

For the construction of accurate length constraints for \( x \), we need to make \( \theta_{\mathbb{Z}}(x^L, y^L) \) “complete”.

**Definition 7 (Length Constraint Completion)** Let

\[
\theta_{TA}(x, y) \equiv \theta^{(1)}_{TA}(x, y) \land \theta^{(2)}_{TA}(y) \in \mathcal{L}_{TA}, \quad \theta_{\mathbb{Z}}(x^L, y^L) \in \mathcal{L}_{\mathbb{Z}}.
\]

We say a formula \( \Theta_{\mathbb{Z}}(x^L, y^L) \) is a **completion** of \( \theta_{\mathbb{Z}}(x^L, y^L) \) in \( x \) with respect to \( \theta_{TA}(x, y) \) if the following formulae are valid:

\[
\forall y : TA \forall x : TA \left[ \theta_{TA}(x, y) \land \theta_{\mathbb{Z}}(x^L, y^L) \leftrightarrow \theta_{TA}(x, y) \land \Theta_{\mathbb{Z}}(x^L, y^L) \right]. \tag{8}
\]

\[
\forall y : TA \forall x : \mathbb{Z} \left[ \theta^{(2)}_{TA}(y) \land \Theta_{\mathbb{Z}}(x^L, y^L) \rightarrow \exists x : TA \left( \theta_{TA}(x, y) \land \Theta_{\mathbb{Z}}(x^L, y^L) \right) \right]. \tag{9}
\]
Example 8 \textit{Let}

\[
\theta_{\text{TA}}(x_1, x_2, x_3) \equiv \alpha(x_1, x_2) = x_3, \\
\theta_{\mathbb{Z}}(x_1^L, x_2^L, x_3^L) \equiv x_1^L < x_3^L \land x_2^L < x_3^L.
\]

Consider the following formulae:

\[
\Theta_{\mathbb{Z}} : \quad x_1^L + x_2^L + 1 = x_3^L \land x_1^L > 0 \land x_2^L > 0, \\
\Theta_{\mathbb{Z}}^1 : \quad x_1^L < x_3^L \land x_2^L < x_3^L \land x_1^L > 0 \land x_2^L > 0, \\
\Theta_{\mathbb{Z}}^2 : \quad x_1^L + x_2^L + 1 = x_3^L \land x_1^L > 5 \land x_2^L > 5.
\]

- \(\Theta_{\mathbb{Z}}\) is a completion of \(\theta_{\mathbb{Z}}(x_1^L, x_2^L, x_3^L)\).
- \(\Theta_{\mathbb{Z}}^1\) satisfies \(8\), it does not satisfies \(9\).
  \textit{Reason:} \(\{x_1^L = 3, x_2^L = 3, x_3^L = 4\}\).
- \(\Theta_{\mathbb{Z}}^2\) satisfies \(9\), but not \(8\).
  \textit{Reason:} \(\{x_1 = a, x_2 = a, x_3 = \alpha(a, a)\}\).
For the construction of length constraint completion, we require that $\theta_{\text{TA}}(x, y) \land \theta_{\mathbb{Z}}(x^L, y^L)$ be in “strong normal form”.

**Definition 8** We say $\theta_{\text{TA}}(x, y) \land \theta_{\mathbb{Z}}(x^L, y^L)$ is in **strong solved form** (with respect to $x$) if $\theta_{\text{TA}}(x, y)$ is in solved form and all literals of the form

$$Ly \neq t(x, y),$$

where $y \in y$ and $t(x, y)$ is a constructor term (properly) containing $x$, are redundant.

**Example 9** In Ex. 5, formula (7) is not in **strong solved form**. However, it can be made into strong solved form by adding

$$s_1^\alpha y \neq x \quad \text{or} \quad s_2^\alpha y \neq z.$$
The following predicates are needed to describe the construction algorithm:

\[
\begin{align*}
\text{Tree}(t) & : \ \exists x_1, \ldots, x_n \geq 0 \left( t^L = \left( \sum_{i=1}^{n} d_i x_i \right) + 1 \right), \\
\text{Node}^\alpha(t, t_\alpha) & : \ t^L = \sum_{i=1}^{\text{ar}(\alpha)} t_i^L + 1, \\
\text{Tree}^\alpha(t) & : \ \exists t_\alpha \left( \text{Node}^\alpha(t, t_\alpha) \land \bigwedge_{i=1}^{\text{ar}(\alpha)} \text{Tree}(t_i) \right),
\end{align*}
\]

where

- \( t_\alpha \) stands for \( t_1, \ldots, t_{\text{ar}(\alpha)} \),
- \( d_1, \ldots, d_n \) are the distinct arities of constructors.
Compute Length Constraint Completion

Algorithm 2 (Length Constraint Completion)  \textit{Input:}

\[ \theta_{TA}(x, y) \equiv \theta_{TA}^{(1)}(x, y) \land \theta_{TA}^{(2)}(y) \in \mathcal{L}_{TA}, \quad \theta_{\mathbb{Z}}(x^L, y^L) \in \mathcal{L}_{\mathbb{Z}}. \]

Initially set \( \Theta_{\mathbb{Z}}(x^L, y^L) = \theta_{\mathbb{Z}}(x^L, y^L) \). For each term \( t \) occurring in \( \theta_{TA}(x, y) \), add the following to \( \Theta_{\mathbb{Z}}(x^L, y^L) \).

- \( t^L = 1 \), if \( t \) is a constant.
- \( t^L = s^L \), if \( t = s \).
- \( \text{Tree}(t) \), if \( t \) is untyped.
- \( \text{Tree}^\alpha(t) \), if \( t \) is \( \alpha \)-typed.
- \( \text{Node}^\alpha(t, t_\alpha) \), if \( t \) is \( \alpha \)-typed with children \( t_\alpha \).
- \( \text{CNT}_{k, n}(t^L) \), if \( t \) occurs in an untyped clusters of size \( n + 1 \) and \( \mathcal{U}_{TA} \) has \( k \) constants.
- \( \text{CNT}^\alpha_{k, n}(t^L) \), if \( t \) occurs in an \( \alpha \)-cluster of size \( n + 1 \) and \( \mathcal{U}_{TA} \) has \( k \) constants.
Quantifiers Elimination on Integer Variables

**Algorithm 3 (Integer Quantifier Elimination)** We assume that formulae with quantifiers on integer variables are in the form

$$\exists z : \mathbb{Z} \ (\theta^{\mathbb{Z}}(x^L, y, z) \land \theta_{TA}(x)),$$  \hspace{1cm} (10)

where $y, z$ are integer variables and $x$ are term variables.

Since $\theta_{TA}(x)$ is in $\mathcal{L}_{TA}$, we can move $\theta_{TA}(x)$ out of the scope of $\exists z$, obtaining

$$\exists z : \mathbb{Z} \ \theta^{\mathbb{Z}}(x^L, y, z) \land \theta_{TA}(x).$$  \hspace{1cm} (11)

Now $\exists z : \mathbb{Z} \ \theta^{\mathbb{Z}}(x^L, y, z)$ is essentially a Presburger formula and we can proceed to remove the block of existential quantifiers.

In fact, we can defer the elimination of integer quantifiers until all term quantifiers have been eliminated.
Algorithm 4 We assume that formulae with quantifiers on term variables are in the form

$$\exists x : \text{TA} \ (\theta_{\text{TA}}(x, y) \land \Psi_{\mathbb{Z}}(x^L, y^L, z)),$$

where $x, y$ are term variables, $z$ are integer variables, and $\Psi_{\mathbb{Z}}(x^L, y^L, z)$ is an arbitrary Presburger formula.

Run Alg. 1 up to the last step. Apply the following subprocedures successively unless noted otherwise.

1. Equality Completion (Alg. 5).
2. Elimination of Equalities Containing $x$ (Alg. 6).
3. Propagation of Disequalities of the Form $L_y \neq t(x, y)$ (Alg. 7).
4. Reduction of Term Quantifiers to Integer Quantifiers (Alg. 8).
Compute Equality Completion

Algorithm 5 (Equality Completion) We assume the input formula is in the form (renaming the first part of (5))

\[ \exists x : \text{TA} \left[ \bigwedge_i x_f(i) \neq t_i(x, y) \land \bigwedge_i L_i y_g(i) \neq s_i(x, y) \right], \quad (13) \]

Let \( S \) be all terms including subterms which appear in (13). Guess an equality completion of \( S \) and we obtain

\[ \exists x : \text{TA} \left[ \bigwedge_i x_f(i) \neq t'_i(x, y) \land \bigwedge_i L_i y_g(i) \neq s'_i(x, y) \land \right. \]

\[ \left. \bigwedge_i x_{f'}(i) = t'_i(x, y) \land \bigwedge_i L_{i'} y_{g'}(i) = s'_i(x, y) \right]. \quad (14) \]
Eliminate Equalities Containing $x$’s

Algorithm 6 (Elimination of Equalities Containing $x$) Let $\mathcal{E}$ denote the set of equalities containing $x$. Exhaustively apply the following subprocedures until $\mathcal{E}$ is empty.

Pick an $E \in \mathcal{E}$.

- $E$ is $x = u$. Then we know $x$ does not occur in $u$ and hence we can remove $\exists x$ by substituting $u$ for all occurrences of $x$.
- $E$ is $Ly = \alpha(t_1(x, y), \ldots, t_k(x, y))$. Then replace $E$ by
  \[
  s_1^\alpha Ly = t_1(x, y), \ldots, s_k^\alpha Ly = t_k(x, y).
  \]
- $E$ is $\beta(u_1(x, y), \ldots, u_l(x, y)) = \beta(u'_1(x, y), \ldots, u'_l(x, y))$. Then replace $E$ by
  \[
  u_1(x, y) = u'_1(x, y), \ldots, u_l(x, y) = u'_l(x, y).
  \]
Propagate Disequalities

**Algorithm 7 (Propagation of Disequalites)** Let $\mathcal{D}$ denote the set of disequalities of the form

$$Ly \neq \alpha(t_1(x, y), \ldots, t_k(x, y)).$$

Exhaustively apply the following subprocedures until $\mathcal{D}$ is empty.

Pick $D \in \mathcal{D}$.

- **Disequality Splitting.** Remove $D$ from $\mathcal{D}$ and add to $\vartheta_{\text{TA}}(x, y)$

$$\neg s^\alpha(Ly) \lor \bigvee_{1 \leq i \leq k} s^\alpha_i Ly \neq t_i(x, y).$$

Return if we take $\neg s^\alpha(Ly)$; continue otherwise.
Propagate Disequivalences (2)

- **Length Splitting.** Suppose we take $s_j^\alpha Ly \neq t_j(x, y) \ (1 \leq j \leq k)$. Split on

$$ (s_j^\alpha Ly)^L = (t_j(x, y))^L \lor (s_j^\alpha Ly)^L \neq (t_j(x, y))^L. $$

Return if we take $(s_j^\alpha Ly)^L \neq (t_j(x, y))^L$; continue otherwise.

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

$$ \bigvee_{i \leq n} s_j^\alpha Ly = u_i \lor \bigwedge_{i \leq n} s_j^\alpha Ly \neq u_i $$

- If we choose any $s_j^\alpha Ly = u_i$, rerun Alg. 6 in case that $u_i$ properly contains $x$;
- If we choose $\bigwedge_{i \leq n} s_j^\alpha Ly \neq u_i$, rerun this algorithm.
Reduction of Term Quantifiers

Algorithm 8 (Reduction of Term Quantifiers to Integer Quantifiers)

Omitting the redundant disequalities of the form \( L y \neq t(x, y) \), we may assume the resulting formula be

\[
\exists x : TA \left[ \theta_{TA}^{(1)}(x, y) \land \theta_{TA}^{(2)}(y) \land \theta_{\mathbb{Z}}(x^L, y^L) \land \Psi_{\mathbb{Z}}(x^L, y^L, z) \right], \quad (15)
\]

where

- \( \theta_{TA}^{(1)}(x, y) \) is of the form \( \bigwedge_i x_{f(i)} \neq t_i(x, y) \),
- \( \theta_{TA}^{(2)}(y) \) does not contain \( x \),
- \( \theta_{\mathbb{Z}}(x^L, y^L) \) is the integer constraint obtained from Algs. 5, 7,
- and \( \Psi_{\mathbb{Z}}(x^L, y^L, z) \) is the PA formula not listed before for simplicity.
Reduction of Term Quantifiers (2)

Let $\theta_{TA}(x, y)$ denote $\theta_{TA}^{(1)}(x, y) \land \theta_{TA}^{(2)}(y)$.

Call Alg. 2 to get the completion $\Theta_{\mathbb{Z}}(x^L, y^L)$ of $\theta_{\mathbb{Z}}(x^L, y^L)$ in $x$ with respect to $\theta_{TA}(x, y)$.

Now we claim that (15) is equivalent to

$$\exists x : TA \left[ \theta_{TA}^{(1)}(x, y) \land \theta_{TA}^{(2)}(y) \land \Theta_{\mathbb{Z}}(x^L, y^L) \land \Psi_{\mathbb{Z}}(x^L, y^L, z) \right], \quad (16)$$

which in turn is equivalent to

$$\exists x^L : \mathbb{Z} \left[ \theta_{TA}^{(2)}(y) \land \Theta_{\mathbb{Z}}(x^L, y^L) \land \Psi_{\mathbb{Z}}(x^L, y^L, z) \right]. \quad (17)$$
Theorem 1  Alg. 1 eliminates a block of quantifiers in time $2^{O(n)}$.

Theorem 2  $BC_k(\mathfrak{A}_{TA})$ is decidable in $O(\exp_k(n))$.

Theorem 3  Alg. 4 eliminates a block of quantifiers in time $2^{2^{O(n)}}$.

Theorem 4  $BC_k(\mathfrak{A}_{TA}^\mathbb{Z})$ is decidable in $O(\exp_{2k}(n))$. 
Future Work

- Refine length constraint construction to reduce double-exponential blowup to one exponential.
- Apply bounded elimination to improve the decision procedure of the first-order theory of Knuth-Bendix order \([\text{ZSM04a}]\).


