Term Algebras with Length Function and Bounded Quantifier Elimination

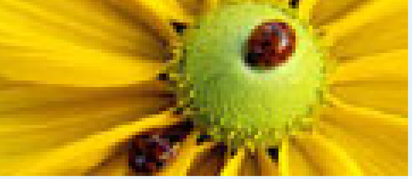
Ting Zhang, Henny B. Sipma, Zohar Manna

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, Vor96].

In Practice:

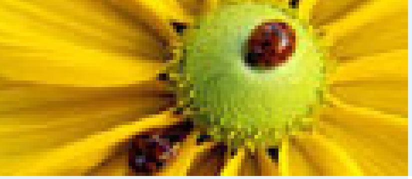
- We rarely deal with formulae with a large quantifier alternation depth.

☞ Therefore it is worthwhile to investigate the “bounded class” of formulae.

Previous Work:

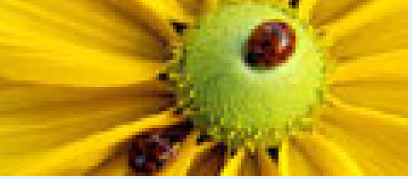
- We gave a quantifier-elimination procedure for the extended theory [ZSM04b].

☞ But no complexity upper bound is established.



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



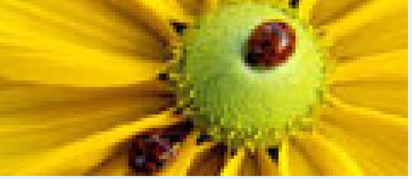
Term Algebras

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

1. TA*: The term domain.*
2. $\mathcal{A}$*: A finite set of constants:* a, b, c, $\ldots$
3. $\mathcal{C}$*: A finite set of constructors:* α, β, γ, $\ldots$.
4. $\mathcal{S}$*: A finite set of selectors. For a constructor* α *with arity* k*, there are* k *selectors* $\mathsf{s}_1^\alpha, \ldots, \mathsf{s}_k^\alpha$ *in* $\mathcal{S}$*.*
5. $\mathcal{T}$*: A finite set of testers. For each constructor* α *there is a corresponding tester* Is_α*.*

☞ 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

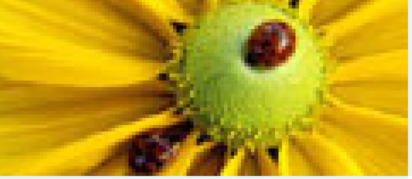
- $\alpha = (\mathsf{s}_1^\alpha, \ldots, \mathsf{s}_k^\alpha)$ means that α is a constructor with $\mathsf{ar}(\alpha) = k$ and $\mathsf{s}_1^\alpha, \ldots, \mathsf{s}_k^\alpha$ are the corresponding selectors of α.
- A term t is a **constructor term** (C-term) if the outmost function symbol of t is 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,

$$\mathsf{s}_i^\alpha(\alpha(x_1, \ldots, x_k)) \quad \textit{simplifies to} \quad x_i.$$

- $L, M, N, \ldots$ denote selector sequences. For $L = \mathsf{s}_1, \ldots, \mathsf{s}_n$, Lx stands for

$$\mathsf{s}_1(\ldots(\mathsf{s}_n(x)\ldots)).$$

- A selector term $\mathsf{s}_i^\alpha(t)$ is called **proper** if $\mathsf{Is}_\alpha(t)$ holds.



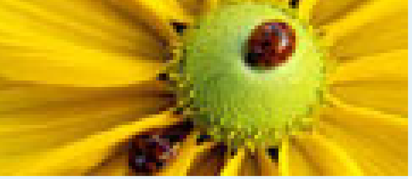
Axiomatization of Term Algebras

- Construction vs. selection.

$$\mathsf{s}_i^{\alpha}(x) = y \leftrightarrow \exists \bar{z}_\alpha \big(\alpha(\bar{z}_\alpha) = x \wedge y = z_i)\big) \vee \big(\forall \bar{z}_\alpha (\alpha(\bar{z}_\alpha) \neq x) \wedge x = y\big).$$

- Unification closure. $\alpha(\boldsymbol{x}_\alpha) = \alpha(\boldsymbol{y}_\alpha) \rightarrow \bigwedge_{1 \leq i \leq \mathsf{ar}(\alpha)} x_i = y_i.$
- Acyclicity. $t(x) \neq x$, if t is built solely by constructors and t properly contains x.
- Uniqueness. $\alpha(\boldsymbol{x}_\alpha) \neq \beta(\boldsymbol{y}_\beta)$, $a \neq b$, and $a \neq \alpha(\boldsymbol{x}_\alpha)$, if a and b are distinct atoms and if α and β are distinct constructors.
- Domain closure.

$$\mathsf{Is}_\alpha(x) \leftrightarrow \exists\, \bar{z}_\alpha \alpha(\bar{z}_\alpha) = x, \qquad \mathsf{Is}_A(x) \leftrightarrow \bigwedge_{\alpha \in \mathcal{C}} \neg \mathsf{Is}_\alpha(x).$$



Example: LISP lists

Signature:

$$\langle \mathsf{list}; \{\mathsf{nil}\}; \{\mathsf{cons}\}; \{\mathsf{car}, \mathsf{cdr}\}; \{\mathsf{Is}_{\mathsf{A}}, \mathsf{Is}_{\mathsf{cons}}\}\rangle$$

Axioms:

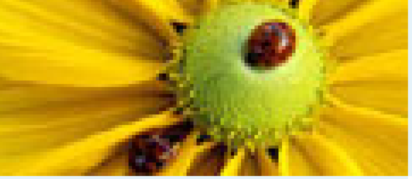
$$(1)\ \mathsf{Is}_{\mathsf{A}}(x) \leftrightarrow \neg\mathsf{Is}_{\mathsf{cons}}(x), \quad (2)\ \mathsf{car}(\mathsf{cons}(x, y)) = x,$$

$$(3)\ \mathsf{cdr}(\mathsf{cons}(x, y)) = y, \quad (4)\ \mathsf{Is}_{\mathsf{A}}(x) \leftrightarrow \{\mathsf{car}, \mathsf{cdr}\}^{+}(x) = x,$$

$$(5)\ \mathsf{Is}_{\mathsf{cons}}(x) \leftrightarrow \mathsf{cons}(\mathsf{car}(x), \mathsf{cdr}(x)) = x.$$

Formulas:

- $\mathsf{cons}(y, z) = \mathsf{cons}(\mathsf{cdr}(x), z) \rightarrow \mathsf{cons}(\mathsf{car}(x), y) = x$ (valid).
- $x = \mathsf{cons}(y, y) \rightarrow \mathsf{cons}(\mathsf{car}(x), y) = x$ (valid).



Quantifier Elimination Preliminary

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

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

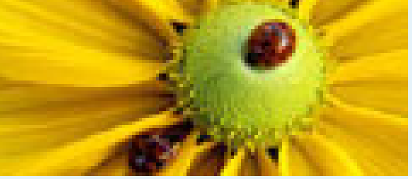
  where $A_i(\boldsymbol{x})$ $(1 \leq i \leq n)$ are literals [Hod93].

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

$$\exists x(x = t \wedge \theta(x, \boldsymbol{y}))$$

  simplifies to

  - $\theta(t, \boldsymbol{y})$, if x does not occur in t;
  - $\exists x \theta(x, \boldsymbol{y})$, if $t \equiv x$;
  - false, if t is a constructor term properly containing x.



Solved Form

Definition 2 (Solved Form) *We say* $\exists \boldsymbol{x}\, \theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y})$ *is in the* ***solved form*** *(with respect to* $\boldsymbol{x}$*),*

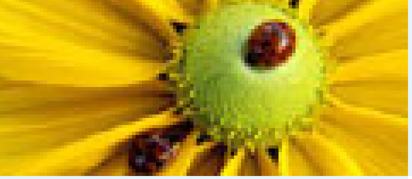
> *if* $\boldsymbol{x}$ *are not in equalities, not asserted to be constants and not inside selector terms.*

General Idea:

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

Procedure Outline:

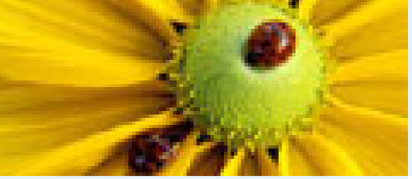
- ☞ A sequence of equivalence-preserving transformations will bring the input formula into a disjunction of formulae in the solved form.
- ☞ The whole block of existential quantifiers $\exists \boldsymbol{x}$ can be eliminated by removing all literals containing $\boldsymbol{x}$ in the matrix.



Quantifier Elimination for Term Algebras

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

- *Guess a type completion of* $\theta(\boldsymbol{x}, \boldsymbol{y})$.
- *Eliminate selector terms containing* $\boldsymbol{x}$.
- *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* $\boldsymbol{x}$.



Type Completion

Definition 3 *Φ' is a **type completion** of Φ if Φ' is obtained from Φ by adding tester predicates such that*

> *for any term $\mathsf{s}(t)$ either $\mathsf{Is}_\alpha(t)$ (for some constructor α) or $\mathsf{Is}_\mathsf{A}(t)$ is present in Φ'.*

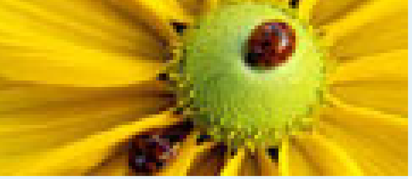
Example 1 *A possible type completion for $y = \mathsf{car}(\mathsf{cdr}(x))$ is*

$$y = \mathsf{car}(\mathsf{cdr}(x)) \wedge \mathsf{Is}_{\mathsf{cons}}(x) \wedge \mathsf{Is}_\mathsf{A}(\mathsf{cdr}(x)).$$

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

$$y = \mathsf{cdr}(x).$$

☞ Guess a type completion of $\theta(\boldsymbol{x}, \boldsymbol{y})$ 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* $\alpha = (s_1^\alpha, s_2^\alpha)$*.*

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

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

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

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

- ☞ 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

$$\alpha(t_1, \ldots, t_i) = \alpha(t'_1, \ldots, t'_i) \tag{2}$$

  by

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

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

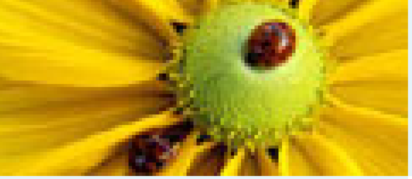
- Replace

$$\alpha(t_1, \ldots, t_i) \neq \alpha(t'_1, \ldots, t'_i) \tag{3}$$

  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 $Ly = t(x, y)$.

Solve equations of the form $Ly = t(\boldsymbol{x}, \boldsymbol{y})$, where

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

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

Example 3 *Suppose that* $\alpha = (\mathsf{s}_1^\alpha, \mathsf{s}_2^\alpha)$. *The solution set of*

$$\mathsf{s}_2^\alpha y = \alpha(\alpha(x_1, y_1), y_2)$$

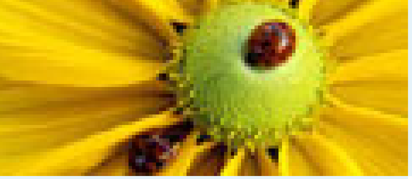
is

$$x_1 = \mathsf{s}_1^\alpha \mathsf{s}_1^\alpha \mathsf{s}_2^\alpha y, \quad y_1 = \mathsf{s}_2^\alpha \mathsf{s}_1^\alpha \mathsf{s}_2^\alpha y, \quad y_2 = \mathsf{s}_2^\alpha \mathsf{s}_2^\alpha y.$$

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.,

$$\exists x(\mathsf{Is}_{\mathsf{C}}(x) \wedge \varphi(x)) \Rightarrow \bigwedge_{a \in \mathsf{C}} \varphi(a).$$



Eliminate Literals Containing x's.

Now we can assume formulae are in the form

$$\exists \boldsymbol{x} : \left[\bigwedge_i x_{f(i)} \neq t_i(\boldsymbol{x}, \boldsymbol{y}) \wedge \bigwedge_i G_i y_{g(i)} \neq s_i(\boldsymbol{x}, \boldsymbol{y}) \right] \wedge \bigwedge_i G'_i y_{g'(i)} \neq s'_i(\boldsymbol{y}) \wedge \bigwedge_i H_i y_{h(i)} = H'_i y_{h'(i)}. \tag{4}$$

Since

$$\exists \boldsymbol{x} : \left[\bigwedge_i x_{f(i)} \neq t_i(\boldsymbol{x}, \boldsymbol{y}) \wedge \bigwedge_i G_i y_{g(i)} \neq s_i(\boldsymbol{x}, \boldsymbol{y}) \right] \tag{5}$$

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

$$\bigwedge_i G'_i y_{g'(i)} \neq s'_i(\boldsymbol{y}) \wedge \bigwedge_i H_i y_{h(i)} = H'_i y_{h'(i)}. \tag{6}$$

Language and Structure

Introduction

Term Algebras

Quantifier Elimination

QE for TA

Term Algebras with Integers

- Language and Structure

QE for “TA + Int”

Complexity

Future Work

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

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

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

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

☞ t^{L} : generalized integer terms.

Counting Constraints

Definition 4 (Counting Constraint) *A **counting constraint** is a predicate* $\mathsf{CNT}^{\alpha}_{k,n}(x)$ *(*$k > 0, n \geq 0$*) that is **true** if and only if*

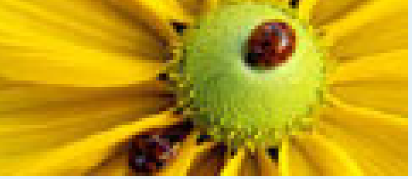
> *there are at least* $n+1$ *different* α*-terms of length* x *in* $\mathfrak{A}_{\mathsf{TA}}$ *with* k *constants.* $\mathsf{CNT}_{k,n}(x)$ *is similarly defined with* α*-terms replaced by* TA*-terms.*

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

$$\mathsf{CNT}^{\mathsf{cons}}_{1,n}(x) \quad \text{is} \quad x \geq 2m - 1 \wedge 2 \nmid x$$

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*.*

Reason: 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.

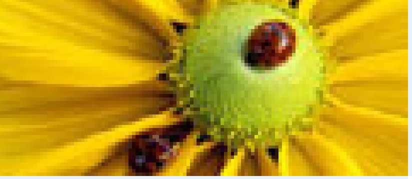
Definition 5 (Equality Completion) *Let S be a set of* TA*-terms. An* ***equality completion*** *θ of S is a formula consisting of the following literals: for any $u, v \in S$, exactly one of $u = v$ and $u \neq v$, and exactly one of $u^{\mathsf{L}} = v^{\mathsf{L}}$ and $u^{\mathsf{L}} \neq v^{\mathsf{L}}$ are in θ.*

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

$$y \neq \alpha(x, z) \wedge \mathsf{Is}_\alpha(y).$$

A possible equality completion of θ is

$$\mathsf{Is}_\alpha(y) \wedge y^{\mathsf{L}} = (\alpha(x,z))^{\mathsf{L}} \wedge x^{\mathsf{L}} = z^{\mathsf{L}} \wedge y^{\mathsf{L}} \neq x^{\mathsf{L}} \wedge \bigwedge_{t,t' \in \Sigma(\theta);\, t \not\equiv t'} t \neq t'. \tag{7}$$



Clusters

Definition 6 (Clusters) *Let* $[t]$ *denote the equivalence class containing* t *with respect to term equality. We say that*

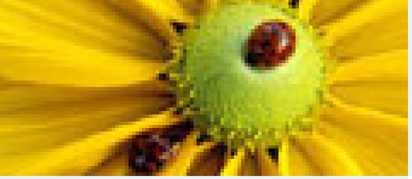
$$C = \{[t_0], \ldots, [t_n]\}$$

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 $\mathsf{rk}(C)$, 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 $\mathsf{rk}(C_1) < \mathsf{rk}(C_2)$*.*

Example 7 *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

$$C_1' : \{x, y\} \text{ and } C_2' : \{x, z\}.$$

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 $\boldsymbol{x}$, we need to make $\theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})$ “complete”.

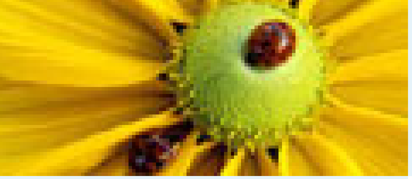
Definition 7 (Length Constraint Completion) *Let*

$$\theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y}) \equiv \theta^{(1)}_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y}) \wedge \theta^{(2)}_{\mathsf{TA}}(\boldsymbol{y}) \in \mathscr{L}_{\mathsf{TA}}, \quad \theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}) \in \mathscr{L}_{\mathbb{Z}}.$$

*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:*

$$\begin{aligned} \forall \boldsymbol{y} : \mathsf{TA}\ \forall \boldsymbol{x} : \mathsf{TA}\ \big[&\theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y}) \wedge \theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}) \\ &\leftrightarrow \theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y}) \wedge \Theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})\big]. \end{aligned} \tag{8}$$

$$\begin{aligned} \forall \boldsymbol{y} : \mathsf{TA}\ \forall \boldsymbol{x}^{\mathsf{L}} : \mathbb{Z}\ \big[&\theta^{(2)}_{\mathsf{TA}}(\boldsymbol{y}) \wedge \Theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}) \\ &\rightarrow \exists \boldsymbol{x} : \mathsf{TA}\ \big(\theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y}) \wedge \Theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})\big)\big]. \end{aligned} \tag{9}$$



Length Constraint Completion: Example

Example 8 *Let*

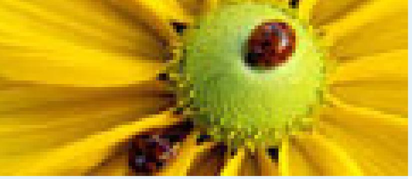
$$\theta_{\mathsf{TA}}(x_1, x_2, x_3) \equiv \alpha(x_1, x_2) = x_3,$$

$$\theta_{\mathbb{Z}}(x_1^{\mathsf{L}}, x_2^{\mathsf{L}}, x_3^{\mathsf{L}}) \equiv x_1^{\mathsf{L}} < x_3^{\mathsf{L}} \wedge x_2^{\mathsf{L}} < x_3^{\mathsf{L}}.$$

Consider the following formulae:

$$\begin{aligned}
\Theta_{\mathbb{Z}} &: \quad x_1^{\mathsf{L}} + x_2^{\mathsf{L}} + 1 = x_3^{\mathsf{L}} \wedge x_1^{\mathsf{L}} > 0 \wedge x_2^{\mathsf{L}} > 0, \\
\Theta_{\mathbb{Z}}^1 &: \quad x_1^{\mathsf{L}} < x_3^{\mathsf{L}} \wedge x_2^{\mathsf{L}} < x_3^{\mathsf{L}} \wedge x_1^{\mathsf{L}} > 0 \wedge x_2^{\mathsf{L}} > 0, \\
\Theta_{\mathbb{Z}}^2 &: \quad x_1^{\mathsf{L}} + x_2^{\mathsf{L}} + 1 = x_3^{\mathsf{L}} \wedge x_1^{\mathsf{L}} > 5 \wedge x_2^{\mathsf{L}} > 5.
\end{aligned}$$

- $\Theta_{\mathbb{Z}}$ *is a completion of* $\theta_{\mathbb{Z}}(x_1^{\mathsf{L}}, x_2^{\mathsf{L}}, x_3^{\mathsf{L}})$.
- $\Theta_{\mathbb{Z}}^1$ *satisfies* (8), *it does not satisfies* (9).
  Reason: $\{x_1^{\mathsf{L}} = 3, x_2^{\mathsf{L}} = 3, x_3^{\mathsf{L}} = 4\}$.
- $\Theta_{\mathbb{Z}}^2$ *satisfies* (9), *but not* (8).
  Reason: $\{x_1 = a, x_2 = a, x_3 = \alpha(a, a)\}$.



Strong Solved Form

For the construction of length constraint completion, we require that $\theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y}) \wedge \theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})$ be in “strong normal form”.

Definition 8 *We say $\theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y}) \wedge \theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})$ is in **strong solved form** (with respect to $\boldsymbol{x}$)*

if $\theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y})$ is in 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 (properly) containing $\boldsymbol{x}$, 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*

$$\mathsf{s}_1^{\alpha} y \neq x \quad \text{or} \quad \mathsf{s}_2^{\alpha} y \neq z.$$

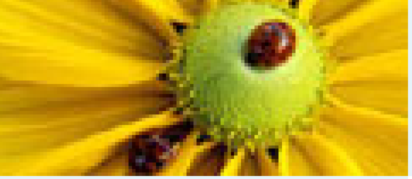
Notations

The following predicates are needed to describe the construction algorithm:

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

where

- $\boldsymbol{t}_\alpha$ stands for $t_1, \ldots, t_{\mathsf{ar}(\alpha)}$,
- $d_1, \ldots, d_n$ are the distinct arities of constructors.



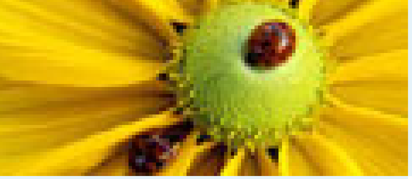
Compute Length Constraint Completion

Algorithm 2 (Length Constraint Completion) *Input:*

$$\theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y}) \equiv \theta^{(1)}_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y}) \wedge \theta^{(2)}_{\mathsf{TA}}(\boldsymbol{y}) \in \mathscr{L}_{\mathsf{TA}}, \quad \theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}) \in \mathscr{L}_{\mathbb{Z}}.$$

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 $\Theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})$.

- $t^{\mathsf{L}} = 1$, *if t is a constant.*
- $t^{\mathsf{L}} = s^{\mathsf{L}}$, *if $t = s$.*
- $\mathsf{Tree}(t)$, *if t is untyped.*
- $\mathsf{Tree}^{\alpha}(t)$, *if t is α-typed.*
- $\mathsf{Node}^{\alpha}(t, \boldsymbol{t}_{\alpha})$, *if t is α-typed with children $\boldsymbol{t}_{\alpha}$.*
- $\mathsf{CNT}_{k,n}(t^{\mathsf{L}})$, *if t occurs in an untyped clusters of size $n+1$ and $\mathfrak{A}_{\mathsf{TA}}$ has k constants.*
- $\mathsf{CNT}^{\alpha}_{k,n}(t^{\mathsf{L}})$, *if t occurs in an α-cluster of size $n+1$ and $\mathfrak{A}_{\mathsf{TA}}$ 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*

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

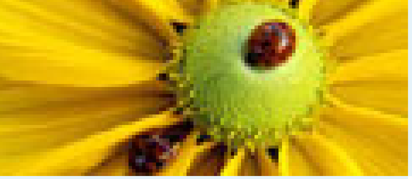
where $\boldsymbol{y}$, $\boldsymbol{z}$ *are integer variables and* $\boldsymbol{x}$ *are term variables.*

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.*

☞ In fact, we can defer the elimination of integer quantifiers until all term quantifiers have been eliminated.



Quantifiers Elimination on Term Variables

Algorithm 4 *We assume that formulae with quantifiers on term variables are in the form*

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

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.

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 $\boldsymbol{x}$ (Alg. 6).*
3. *Propagation of Disequalities of the Form $Ly \neq t(\boldsymbol{x}, \boldsymbol{y})$ (Alg. 7).*
4. *Reduction of Term Quantifi ers to Integer Quantifi ers (Alg.8).*

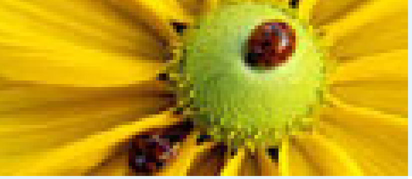
Compute Equality Completion

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

$$\exists \boldsymbol{x} : \mathsf{TA}\left[\bigwedge_i x_{f(i)} \neq t_i(\boldsymbol{x},\boldsymbol{y}) \wedge \bigwedge_i L_i y_{g(i)} \neq s_i(\boldsymbol{x},\boldsymbol{y})\right], \tag{13}$$

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

$$\exists \boldsymbol{x} : \mathsf{TA}\left[\bigwedge_i x_{f(i)} \neq t_i(\boldsymbol{x},\boldsymbol{y}) \wedge \bigwedge_i L_i y_{g(i)} \neq s_i(\boldsymbol{x},\boldsymbol{y}) \wedge \right.$$
$$\left. \bigwedge_i x_{f'(i)} = t'_i(\boldsymbol{x},\boldsymbol{y}) \wedge \bigwedge_i L'_i y_{g'(i)} = s'_i(\boldsymbol{x},\boldsymbol{y})\right]. \tag{14}$$



Eliminate Equalities Containing x's

Algorithm 6 (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.*

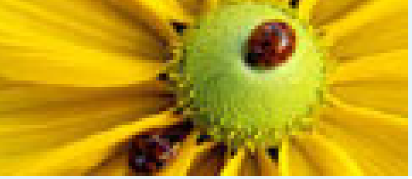
Pick an $E \in \mathcal{E}$.

- *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 $Ly = \alpha(t_1(\boldsymbol{x}, \boldsymbol{y}), \ldots, t_k(\boldsymbol{x}, \boldsymbol{y}))$. Then replace E by*

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

- *E is $\beta(u_1(\boldsymbol{x}, \boldsymbol{y}), \ldots, u_l(\boldsymbol{x}, \boldsymbol{y})) = \beta(u_1'(\boldsymbol{x}, \boldsymbol{y}), \ldots, u_l'(\boldsymbol{x}, \boldsymbol{y}))$. Then replace E by*

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



Propagate Disequalities

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

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

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 $\theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y})$*

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

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

Propagate Disequalities (2)

- Length Splitting. Suppose we take $\mathsf{s}_j^\alpha Ly \neq t_j(\boldsymbol{x}, \boldsymbol{y})$ $(1 \leq j \leq k)$. Split on

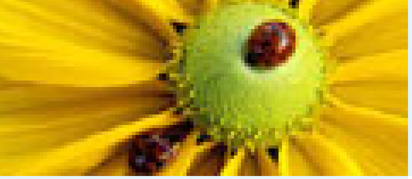
$$(\mathsf{s}_j^\alpha Ly)^{\mathsf{L}} = (t_j(\boldsymbol{x}, \boldsymbol{y}))^{\mathsf{L}} \vee (\mathsf{s}_j^\alpha Ly)^{\mathsf{L}} \neq (t_j(\boldsymbol{x}, \boldsymbol{y}))^{\mathsf{L}}.$$

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

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

$$\bigvee_{i \leq n} \mathsf{s}_j^\alpha Ly = u_i \vee \bigwedge_{i \leq n} \mathsf{s}_j^\alpha Ly \neq u_i$$

  - If we choose any $\mathsf{s}_j^\alpha Ly = u_i$, rerun Alg. 6 in case that u_i properly contains $\boldsymbol{x}$;
  - If we choose $\bigwedge_{i \leq n} \mathsf{s}_j^\alpha Ly \neq u_i$, rerun this algorithm.



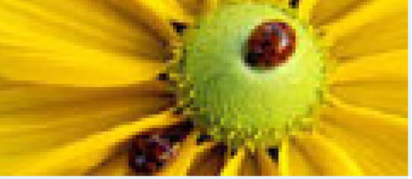
Reduction of Term Quantifiers

Algorithm 8 (Reduction of Term Quantifiers to Integer Quantifiers)
Omitting the redundant disequalities of the form $Ly \neq t(\boldsymbol{x}, \boldsymbol{y})$, we may assume the resulting formula be

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

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. 5, 7,*
- *and* $\Psi_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}}, \boldsymbol{z})$ *is the PA formula not listed before for simplicity.*



Reduction of Term Quantifiers (2)

Let $\theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y})$ denote $\theta^{(1)}_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y}) \wedge \theta^{(2)}_{\mathsf{TA}}(\boldsymbol{y})$.

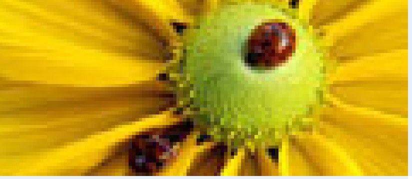
Call Alg. 2 to get the completion $\Theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})$ of $\theta_{\mathbb{Z}}(\boldsymbol{x}^{\mathsf{L}}, \boldsymbol{y}^{\mathsf{L}})$ in $\boldsymbol{x}$ with respect to $\theta_{\mathsf{TA}}(\boldsymbol{x}, \boldsymbol{y})$.

Now we claim that (15) is equivalent to

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

which in turn is equivalent to

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



Complexity

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

Theorem 2 $\mathrm{BC}_{\mathsf{k}}(\mathfrak{A}_{\mathsf{TA}})$ *is decidable in* $O(\exp_k(n))$.

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

Theorem 4 $\mathrm{BC}_{\mathsf{k}}(\mathfrak{A}^{\mathbb{Z}}_{\mathsf{TA}})$ *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 [ZSM04a].

[CL89] Hubert Comon and Pierre Lescanne. Equational problems and disunification. *Journal of Symbolic Computation*, 7:371–425, 1989.

[FR79] J. Ferrante and C. W. Rackoff. *The Computational Complexity of Logical Theories*. Springer-Verlag, 1979.

[Hod93] Wilfrid Hodges. *Model Theory*. Cambridge University Press, Cambridge, UK, 1993.

[Vor96] 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. Springer-Verlag, 1996.

[ZSM04a] Ting Zhang, Henny Sipma, and Zohar Manna. The decidability of the first-order theory of term algebras with Knuth-Bendix order, 2004. Submitted.

[ZSM04b] 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.