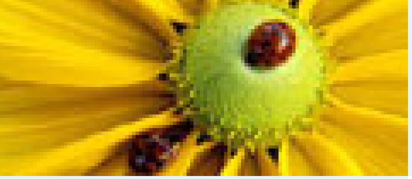
Decision Procedures for Recursive Data Structures with Integer Constraints

Ting Zhang, Henny B. Sipma, Zohar Manna

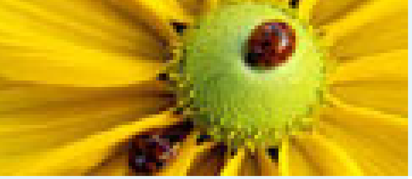
Stanford University

tingz,sipma,zm@cs.stanford.edu



Outline

- Motivation
- Recursive data structures
- Oppen's Algorithms
- Recursive data structures with integer constraints
- Decision procedure for structures with infinite atom domain
- Decision procedure for structures with finite atom domain
- Related work
- Future work



Motivation: Program Verification

- Recursive data structures are essential constructs in programming languages.
- To verify programs we need to reason about these data structures.
- Programming languages often involve multiple data domains.
- Common “mixed” constraints are combinations of data structures with integer constraints on the size of those structures.

Recursive Data Structures

Definition 1 *A data structure is **recursive** if*

> *it is partially composed of smaller or simpler instances of the same structure.*

- **No Junk**: the data domain is the set of data objects generated exclusively by applying constructors.
- **No Confusion**: each data object is uniquely generated.

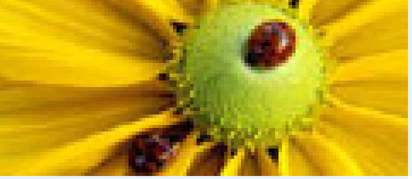
☞ Recursive Data Structures = Term Algebras.

Example 1 *A tree is composed of subtrees and leaves. Other examples include lists, stacks, counters, and records.*

Language and Structure

A **recursive data structure** $\mathfrak{A}_\lambda : \langle \lambda; \mathcal{A}, \mathcal{C}, \mathcal{S}, \mathcal{T} \rangle$ consists of

- λ: The data domain.
- $\mathcal{A}$: A set of **atoms** (constants): $a, b, c \ldots$.
- $\mathcal{C}$: A finite set of constructors: α, β, γ, $\ldots$ each of which is associated with an arity, e.g., $\alpha : \underbrace{\lambda \times \ldots \times \lambda}_{k} \rightarrow \lambda$.
- $\mathcal{S}$: A finite set of selectors: $\mathsf{s}_1^\alpha, \ldots, \mathsf{s}_k^\alpha : \lambda \rightarrow \lambda$ for each $\alpha \in \mathcal{C}$.
- $\mathcal{T}$: A finite set of testers: $\mathsf{Is}_\alpha : \lambda \rightarrow \mathcal{B}$ for each $\alpha \in \mathcal{C}$.
- A special predicate $\mathsf{Is}_\mathsf{A} : \lambda \rightarrow \mathcal{B}$.



Axiomatization

- 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:

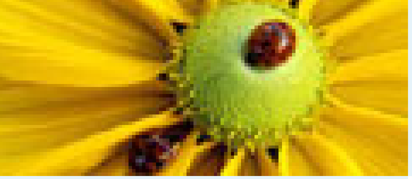
$$\langle \mathsf{list}; \{\mathsf{nil}\}; \{\mathsf{cons}\}; \{\mathsf{car}, \mathsf{cdr}\}; \{\mathsf{Is}_{\mathsf{A}}, \mathsf{Is}_{\mathsf{cons}}\}\rangle$$

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).



Directed Acyclic Graph

Definition 2 *A term t can be represented by a tree T_t such that*

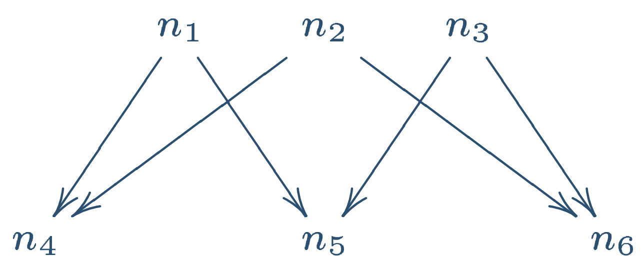
- *t is a constant or variable, then T_t is a leaf vertex labeled by t,*
- *if t is in the form $\alpha(t_1, \ldots, t_k)$, then T_t is the tree having the root labeled by t and having $T_{t_1}, \ldots, T_{t_k}$ as its subtrees.*

*A **directed acyclic graph** (DAG) G_t of t is obtained from T_t by "factoring out" the common subtrees (subterms).*

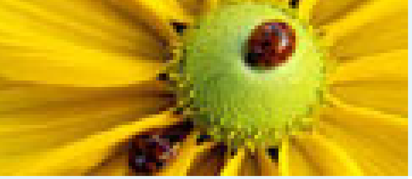
☞ The **DAG of a formula** is the DAG representing all terms in the formula.

Example: DAG Representation

$$\mathsf{cons}(y, z) = \mathsf{cons}(x, z) \ \wedge \ \mathsf{cons}(x, y) \neq x.$$



$$\begin{array}{lll} n_1 & : & \mathsf{cons}(x, y) \\ n_2 & : & \mathsf{cons}(x, z) \\ n_3 & : & \mathsf{cons}(y, z) \end{array} \qquad \begin{array}{lll} n_4 & : & x \\ n_5 & : & y \\ n_6 & : & z \end{array}$$



Bidirectional Closure

R: a binary relation.

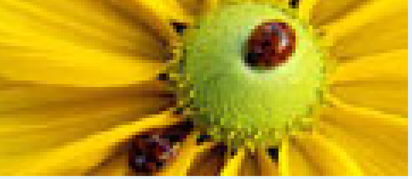
- Unification Closure $R{\downarrow}$ of R: the smallest equivalence relation extending R such that

$$\alpha(\boldsymbol{x}_\alpha) = \alpha(\boldsymbol{y}_\alpha) \to \bigwedge_{1 \le i \le \mathsf{ar}(\alpha)} x_i = y_i .$$

- Congruence Closure $R{\uparrow}$ of R: the smallest equivalence relation extending R such that

$$\bigwedge_{1 \le i \le \mathsf{ar}(\alpha)} x_i = y_i \to \alpha(\boldsymbol{x}_\alpha) = \alpha(\boldsymbol{y}_\alpha).$$

- Bidirectional Closure $R{\Updownarrow}$ = $R{\downarrow}$ + $R{\uparrow}$.



Type Completion

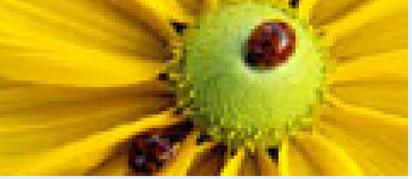
Definition 3 *Φ' is a **type completion** of Φ if Φ' is obtained from Φ by adding tester predicates such that*

> *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 2 *A possible type completion for $y = \mathsf{car}(\mathsf{cdr}(x))$ is*

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

☞ A type completion Φ' is **compatible** with Φ if the satisfiability of Φ implies that Φ' is satisfiable and if any solution of Φ' is a solution of Φ.



Oppen's Algorithm for $\mathfrak{A}_\lambda$

Algorithm 1 *Input*

$$\Phi : \underbrace{q_1 = r_1 \wedge \ldots \wedge q_k = r_k}_{\Phi_{eq}} \wedge \underbrace{s_1 \neq t_1 \wedge \ldots \wedge s_l \neq t_l}_{\Phi_{ne}} .$$

1. *Guess a type completion Φ and simplify selector terms accordingly. We still use Φ to denote the resulting formula.*
2. *Construct the DAG of Φ.*
3. *Compute the bidirectional closure $R^{\Updownarrow}$ of*

$$R = \{(q_i, r_i) \mid 1 \leq i \leq k\}.$$

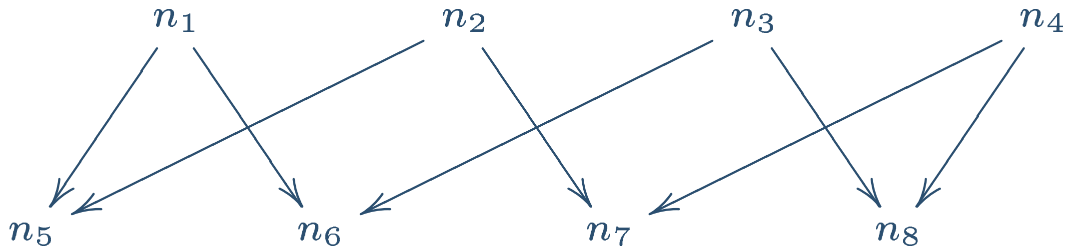
4. *Return **FAIL** if $\exists i (s_i, t_i) \in R^{\Updownarrow}$; return **SUCCESS** otherwise.*

☞ Solution = Type Completion + DAG + Bidirectional Closure.

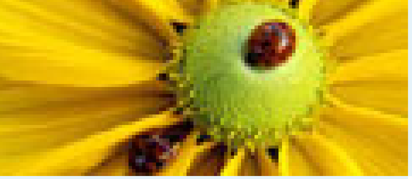
Example: Oppen's Algorithm

The following graph shows the DAG for

$$\mathsf{ls}_{\mathsf{cons}}(x) \;\wedge\; \mathsf{cons}(y, z) = \mathsf{cons}(\mathsf{cdr}(x), z) \;\wedge\; \mathsf{cons}(\mathsf{car}(x), y) \neq x.$$



$$\begin{array}{lcl@{\qquad\qquad}lcl}
n_1 & : & x & n_5 & : & \mathsf{car}(x) \\
n_2 & : & \mathsf{cons}(\mathsf{car}(x), y) & n_6 & : & \mathsf{cdr}(x) \\
n_3 & : & \mathsf{cons}(\mathsf{cdr}(x), z) & n_7 & : & y \\
n_4 & : & \mathsf{cons}(y, z) & n_8 & : & z
\end{array}$$



Example (Cont'd): Oppen's Algorithm

- Initial partition.

$$\{\{n_1\},\{n_2\},\{n_3\},\{n_4\},\{n_5\},\{n_6\},\{n_7\},\{n_8\}\}$$

- Merge n_3 and n_4 since $n_3 = n_4$.

$$\{\{n_1\},\{n_2\},\{n_3,n_4\},\{n_5\},\{n_6\},\{n_7\},\{n_8\}\}$$

- Merge n_6 and n_7 by unification closure algorithm.

$$\{\{n_1\},\{n_2\},\{n_3,n_4\},\{n_5\},\{n_6,n_7\},\{n_8\}\}$$

- Merge n_1 and n_2 by congruence closure algorithm.

$$\{\{n_1,n_2\},\{n_3,n_4\},\{n_5\},\{n_6,n_7\},\{n_8\}\}$$

☞ The conjunction is unsatisfiable since $n_1 \neq n_2$.

Language and Structure

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

Two-sorted language $\Sigma = \Sigma_\lambda \cup \Sigma_{\mathbb{Z}} \cup \{|.|\}$:

1. Σ_λ: signature of recursive data structures.
2. $\Sigma_{\mathbb{Z}}$: signature of Presburger arithmetic.
3. $|.| \ : \ \lambda \to \mathbb{N}$, the length function defined by

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

☞ $|t|$: generalized integer terms.

Two-sorted structures:

- $\mathfrak{B}^{\omega} = \langle \mathfrak{A}_\lambda^{\omega}; \mathfrak{A}_{\mathbb{Z}}; |.| \rangle$; λ contains infinitely many atoms.
- $\mathfrak{B}^{=k} = \langle \mathfrak{A}_\lambda^{=k}; \mathfrak{A}_{\mathbb{Z}}; |.| \rangle$; λ contains exactly k atoms.

Difficulty of N-O Combination

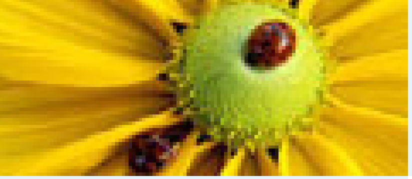
Nelson-Oppen combination methods is not directly applicable to the extended theory.

Example 3 *Consider in $\mathfrak{B}^{=1}$ with $\lambda = \{a\}$.*

$$\underbrace{|u| = 3}_{\Phi_{\mathbb{Z}}} \wedge \underbrace{u \neq \mathsf{cons}(\mathsf{cons}(a,a),a) \ \wedge \ u \neq \mathsf{cons}(a, \mathsf{cons}(a,a))}_{\Phi_{\lambda}}$$

is unsatisfiable in $\mathfrak{B}^{=1}$, while $\Phi_{\mathbb{Z}}$ is satisfiable in $\mathfrak{A}_{\mathbb{Z}}$ and Φ_{λ} is satisfiable in $\mathfrak{A}_{\lambda}$.

☞ There are "hidden" constraints on data structure length.



Length Constraint

- An arithmetic constraint Φ_Δ is a **length constraint** of Φ_λ, if

  there is one-to-one correspondence between integer variables and terms occurring in Φ_λ.
- Φ_Δ is **sound**, if

  for any satisfying assignment ν_λ of Φ_λ, $|\nu_\lambda|$ is a satisfying assignment for Φ_Δ.
- Φ_Δ is **complete**, if

  whenever Φ_λ is satisfiable, for any satisfying assignment ν_Δ of Φ_Δ there exists a satisfying assignment ν_λ of Φ_λ such that $|\nu_\lambda| = \nu_\Delta$.
- Φ_Δ is **induced** by Φ_λ, if

  Φ_Δ is both sound and complete.

Example: Length Constraint

Φ_{λ} : $\mathsf{cons}(x, y) = z$.

- Φ^1_{Δ} : $|x| < |z| \wedge |y| < |z|$ is sound but not complete.

  Reason: the integer assignment

  $$\nu_{\Delta} : \{|x| = 3, |y| = 3, |z| = 4\}$$

  can not be realized.

- Φ^2_{Δ} : $|x| + |y| = |z| \wedge |x| > 5 \wedge |y| > 0$ is complete but not sound.

  Reason: it does not satisfy the data assignment

  $$\nu_{\lambda} : \{x = a, y = a, z = \mathsf{cons}(a, a)\}.$$

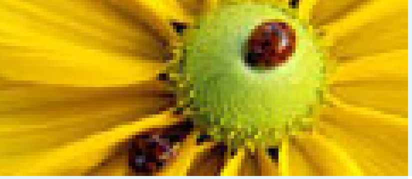
- Φ_{Δ} : $|x| + |y| = |z| \wedge |x| > 0 \wedge |y| > 0$ is both sound and complete, and hence is the induced constraint from Φ_{λ}.

Main Theorem

Main Theorem 1 *Let Φ be a mixed constraint in the form $\Phi_{\mathbb{Z}} \wedge \Phi_{\lambda}$ and Φ_{Δ} the induced length constraint with respect to Φ_{λ}. Then Φ is satisfiable in $\mathfrak{B}$ if and only if*

1. *$\Phi_{\Delta} \wedge \Phi_{\mathbb{Z}}$ is satisfiable in $\mathfrak{A}_{\mathbb{Z}}$, and*
2. *Φ_{λ} is satisfiable in $\mathfrak{A}_{\lambda}$.*

☞ The decision problem for quantifier-free theories reduces to computing the induced length constraints in Presburger arithmetic.



Notations

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

- $\boldsymbol{t}_{\alpha}$ stands for $t_1, \ldots, t_{\mathsf{ar}(\alpha)}$.
- $d_1, \ldots, d_n$ are the distinct arities of the constructors.
- $\mathsf{Tree}(t)$ is true iff $|t|$ is the length of a well-formed tree.
- $\mathsf{Node}^{\alpha}(t, \boldsymbol{t}_{\alpha})$ forces the length of an α-typed node with known children to be the sum of the lengths of its children.
- $\mathsf{Tree}^{\alpha}(t)$ states the length constraint for an α-typed tree.

Construction of $\Phi_\triangle$ in $\mathfrak{B}^\omega$

Algorithm 2 *Input:*

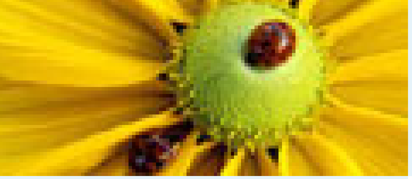
1. Φ_λ: a (type-complete) data constraint,

2. G_λ: the DAG of Φ_λ,

3. $R\Updownarrow$: the bidirectional closure obtained by Algorithm 1.

Initially set $\Phi_\triangle = \emptyset$. For each term t add the following to $\Phi_\triangle$.

- $|t| = 1$, *if t is an atom;*
- $|t| = |s|$, *if $(t, s) \in R\Updownarrow$.*
- $\mathsf{Tree}(t)$ *if t is an untyped leaf vertex.*
- $\mathsf{Node}^\alpha(t, \boldsymbol{t}_\alpha)$ *if t is an α-typed vertex with children $\boldsymbol{t}_\alpha$.*
- $\mathsf{Tree}^\alpha(t)$ *if t is an α-typed leaf vertex.*



Decision Procedure for $\mathsf{Th}^{\forall}(\mathfrak{B}^{\omega})$

Input: $\Phi_{\lambda} \wedge \Phi_{\mathbb{Z}}$.

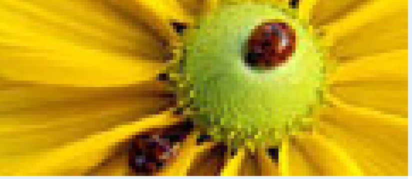
1. Guess a type completion Φ'_{λ} of Φ_{λ}.
2. Call Algorithm 1 on Φ'_{λ}.
   - Return **FAIL** if Φ'_{λ} is unsatisfiable; continue otherwise.
3. Construct $\Phi_{\triangle}$ from G'_{λ} using Algorithm 2.
   - Return **SUCCESS** if $\Phi_{\triangle} \wedge \Phi_{\mathbb{Z}}$ is satisfiable.
   - Return **FAIL** otherwise.

Example: DP for $\mathsf{Th}^{\forall}(\mathfrak{B}^{\omega})$

$$\mathsf{Is}_{\mathsf{cons}}(y) \wedge x = \mathsf{cons}(\mathsf{car}(y), y) \wedge |\mathsf{cons}(\mathsf{car}(y), y)| < 2|\mathsf{car}(x)|. \quad (1)$$

$$\begin{tikzcd}
& n_1 \arrow[ddl] \arrow[ddr] & & n_2 \arrow[dd] \arrow[dr] & & \\
& & & & n_3 \arrow[dl] \arrow[ddr] & \\
n_4 & & n_5 & n_6 & & n_7
\end{tikzcd}$$

$$\begin{array}{lcl}
n_1 & : & x \\
n_2 & : & \mathsf{cons}(\mathsf{car}(y), y) \\
n_3 & : & y \\
n_4 & : & \mathsf{car}(x) \\
n_5 & : & \mathsf{cdr}(x) \\
n_6 & : & \mathsf{car}(y) \\
n_7 & : & \mathsf{cdr}(y)
\end{array}$$



Example (Cont'd): DP for $\mathsf{Th}^{\forall}(\mathfrak{B}^{\omega})$

Introduction

Recursive Data Structures

Oppen's Algorithm

Recursive Data Structures with Integer Constraints

Decison Procedure for $\mathsf{Th}^{\forall}(\mathfrak{B}^{\omega})$

- Notations
- Construction of $\Phi_{\triangle}$
- DP for $\mathsf{Th}^{\forall}(\mathfrak{B}^{\omega})$
- Example
- Example (Cont'd)

Decison Procedure for $\mathsf{Th}^{\forall}(\mathfrak{B}^{=k})$

Related Work

Future Work

- Unification and congruence closure:

$$\{\{n_1, n_2\}, \{n_3, n_5\}, \{n_4, n_6\}, \{n_7\}\}.$$

- Induced length constraints:

$$|\mathsf{car}(x)| \geq 1 \ \wedge\ |\mathsf{cdr}(x)| \geq 1 \ \wedge\ |\mathsf{car}(y)| \geq 1 \ \wedge\ |\mathsf{cdr}(y)| \geq 1. \tag{2}$$

$$|x| = |\mathsf{cons}(\mathsf{car}(y), y)| \wedge |\mathsf{car}(x)| = |\mathsf{car}(y)| \wedge |\mathsf{cdr}(x)| = |y|. \tag{3}$$

$$|x| = |\mathsf{car}(x)| + |\mathsf{cdr}(x)| \ \wedge\ |y| = |\mathsf{car}(y)| + |\mathsf{cdr}(y)| \ \wedge \\ |\mathsf{cons}(\mathsf{car}(y), y)| = |\mathsf{car}(y)| + |y|. \tag{4}$$

(2), (3) and (4) imply $|\mathsf{cons}(\mathsf{car}(y), y)| \geq 2|\mathsf{car}(x)| + 1$.

☞ Constraint (1) is unsatisfiable.

Complication for $\mathsf{Th}^{\forall}(\mathfrak{B}^{=k})$

Suppose that the atom domain contains only one atom. Then

$$\begin{aligned}|x| = 3 \ \wedge\ \mathsf{Is}_{\mathsf{A}}(y) \ \wedge \\ x \neq \mathsf{cons}(\mathsf{cons}(y,y),y) \ \wedge\ x \neq \mathsf{cons}(y,\mathsf{cons}(y,y))\end{aligned} \tag{5}$$

is unsatisfiable while by the previous procedure

$$\begin{aligned}|y| = 1 \ \wedge\ |\mathsf{cons}(y,y)| = 2 \ \wedge \\ |\mathsf{cons}(\mathsf{cons}(y,y),y)| = 3 \ \wedge\ |\mathsf{cons}(y,\mathsf{cons}(y,y))| = 3\end{aligned} \tag{6}$$

is obviously satisfiable together with $|x| = 3$.

☞ Need to count how many distinct trees at certain length.

Counting Constraints

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

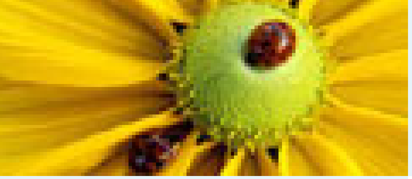
> *there are at least* $n+1$ *different* α*-terms of length* x *in the language with exactly* $k > 0$ *distinct atoms.*

Example 4 *For* $\mathfrak{B}^{=1}_{\mathsf{list}}$, $\mathsf{CNT}^{\mathsf{cons}}_{n,1}(x) \equiv x \geq m$ *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*.*

☞ $\mathsf{CNT}^{\alpha}_{k,n}(x)$ is expressible by a quantifier-free Presburger formula that can be computed in time $O(n)$.



Equality Completion

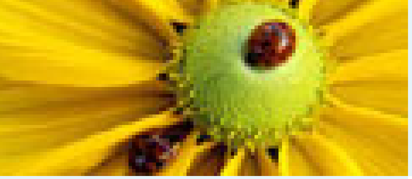
Definition 5 (Equality Completion) *Let S be a set of λ-terms. An **equality completion** θ of S is a formula consisting of the following literals:*

> *for any $u, v \in S$, exactly one of $u = v$ and $u \neq v$, and exactly one of $|u| = |v|$ and $|u| \neq |v|$ are in θ.*

Example 5 *An equality completion of $S = \{x, y, z, \alpha(x, z)\}$ is*

$$|y| = |\alpha(x, z)| \wedge |x| = |z| \wedge |y| \neq |x| \wedge \bigwedge_{t, t' \in S; t \not\equiv t'} t \neq t'. \tag{7}$$

☞ The notion of equality completion naturally generalizes to a conjunction of literals, e.g., the above is an equality completion of $\theta : y \neq \alpha(x, z)$.



Construction of Φ_Δ in $\mathfrak{B}^{=k}$

Let $\mathsf{CLT}_{n+1}(t_0, \ldots, t_n)$ denote that $t_0, \ldots, t_n$ have the same length but are pairwise unequal.

Algorithm 3 *Input:*

> Φ_λ *(type and equality complete),* G_λ *and* $R\Updownarrow$.

1. *Call Algorithm 2 to obtain* Φ_Δ.
2. *For each* t *occurring in* $\mathsf{CLT}_{n+1}(t_0, \ldots, t_n)$*, add* $\mathsf{CNT}^{\alpha}_{k,n}(|t|)$.

Example 6 *Formula* (5) *implies*

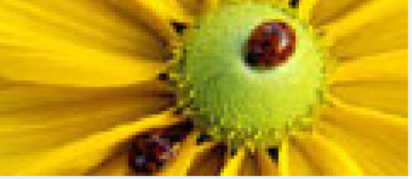
$$\mathsf{CLT}_3(x, \mathsf{cons}(\mathsf{cons}(y, y), y), \mathsf{cons}(y, \mathsf{cons}(y, y)))$$

which gives the counting constraint $|x| \geq 4$. *A contradiction.*

Decision Procedure for $\mathsf{Th}^{\forall}(\mathfrak{B}^{=k})$

Input : $\Phi_\lambda \wedge \Phi_{\mathbb{Z}}$.

1. Guess a type and equality completion Φ'_λ of Φ_λ.
2. Call Algorithm 1 on Φ'_λ.
   - Return **FAIL** if Φ'_λ is unsatisfiable; continue otherwise.
3. Construct Φ_Δ from G'_λ using Algorithm 3.
   - Return **SUCCESS** if $\Phi_\Delta \wedge \Phi_{\mathbb{Z}}$ is satisfiable.
   - Return **FAIL** otherwise.



Related Work on Arithmetic Integration

- Combining integer with sets and multisets [Zar02b, Zar02a].
- Combining integer with lists [Zar01].
- Quantifier-free theory of term algebras with Knuth-Bendix order [KV00, KV01].
- First-order theory of term algebras with Knuth-Bendix order [ZSM04a].
- First-order theory of term algebras with integer constraints [ZSM04b].

Future Work on Arithmetic Integration

- Recursive data structures with subterm relation. E.g.,

$$y \preceq \mathsf{cons}(x, \mathsf{cons}(x, x)) \rightarrow |y| \leq |x|.$$

- Queues (flat lists without concatenation). E.g.,

$$\mathsf{rcons}(\mathsf{rcons}(y, a), b) = \mathsf{cons}(b, \mathsf{cons}(a, y)) \rightarrow |y| \equiv_2 1.$$

- Word concatenation. E.g.,

$$x \circ a \circ y = y \circ b \circ x \rightarrow |x| = |y|.$$

[KV00] 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.

[KV01] 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-Verlag, 2001.

[Zar01] Calogero G. Zarba. Combining lists with integers. In Rajeev Goré, Alexander Leitsch, and Tobias Nipkow, editors, *International Joint Conference on Automated Reasoning (Short Papers)*, Technical Report DII 11/01, pages 170–179. University of Siena, Italy, 2001.

[Zar02a] Calogero G. Zarba. Combining multisets with integers. In Andrei Voronkov, editor, *Proc. of the* 18^{th} *Intl. Conference on Automated Deduction*, volume 2392 of *Lecture Notes in Artifi cial Intelligence*, pages 363–376. Springer, 2002.

[Zar02b] Calogero G. Zarba. Combining sets with integers. In Alessandro Armando, editor, *Frontiers of Combining*

Systems, volume 2309 of *Lecture Notes in Artificial Intelligence*, pages 103–116. Springer, 2002.

[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. Term algebras with length function and bounded quantifier alternation, 2004. To appear in the Proceedings of the 17^{th} International Conference on Theorem Proving in Higher Order Logics.