Linear Tree Constraints

Sabine Bauer, Martin Hofmann

PUMA Workshop, September 4, 2017
Problem Definition: Constraints on Infinite Trees

- Linear constraints with integer coefficients (as $x + y \leq 3z + w$) with variables ranging over infinite trees of nonnegative rational numbers
- Addition and comparison understood pointwise
- Lists can be shifted (\texttt{tl} operation), trees can be divided into subtrees (e.g. \texttt{left(right(x)))}
- In addition arithmetic inequalities between selected components of trees

Grammar

\[
\begin{align*}
  \text{tree} & ::= \text{variable} \mid \text{label(} \text{tree} \text{)} \\
  \text{sum} & ::= \text{tree} \mid \text{sum + sum} \\
  \text{constraint} & ::= \text{sum} \geq \text{sum}
\end{align*}
\]
Problem Definition: Constraints on Infinite Trees

- Linear constraints with integer coefficients (as \(x + y \leq 3z + w\)) with variables ranging over infinite trees of nonnegative rational numbers
- Addition and comparison understood pointwise
- Lists can be shifted (\texttt{tl} operation), trees can be divided into subtrees (e.g. \texttt{left(right(x)))}
- In addition arithmetic inequalities between selected components of trees

**Asked**

(Simultaneous) satisfiability (of all tree and arithmetic constraints) with tree entries in \(\mathbb{Q}_0^+ \cup \{\infty\}\)?
Application to resource analysis

- MH and Rodriguez: resource analysis with type systems, in particular prediction of memory usage as a function of the input size
- Object oriented language Resource Aware Java (RAJA)
- In earlier work H&R proposed tree constraints as a backend of the analysis, gave incomplete heuristic procedure to solve them
- General problem of deciding satisfiability for tree constraints remained open

N.B.: lots of related work on resource analysis, distinctive feature of RAJA: allows a very wide range of resource behaviors, not restricted to for instance polynomials.
Black Box: Translation between Programs and Constraints

- Given: Java program (certain fragment: RAJA) with main function taking list of strings as input
- Asked: Bound on memory usage as function of input size
- Technique: amortized analysis with potential method
  - gives worst case average runtime for sequences of operations
  - takes into account how the data structures change during the computation

Analysis defines potential of data structures using infinite trees which must be chosen so as to satisfy typing rules (potentials always add up)

Type inference uses unknown trees to compile program into set of constraints on those unknowns

Program can execute with an amount of memory that can be read off from the constraint solutions.
We provide the following answers:

1. The general problem as formulated by MH&Rodriguez is hard for the (notoriously difficult) Skolem-Mahler-Lech problem.

2. We analyzed the translation from resource inference $\rightarrow$ tree constraints more closely and proved that it only produces a proper subset of the general problem. We identified this fragment and called it ”unilateral”.

3. Satisfaction of unilateral list constraints are shown decidable in polynomial time.

4. We can obtain upper bounds on minimal solutions $\rightarrow$ good upper bounds on resource usage.

5. New: Satisfiability of unilateral tree constraints is also decidable.
The Tree Case
Example: tree of degree 3, labels: left, right, middle

\[ lrx \geq x, lx \geq x, mx \geq x, \]
\[ x \geq rlx, x \geq mlx. \]

Language of trees greater than \( x \), less than \( x \), and equal to \( x \):

\[ L_g = \left( lr \mid l \mid m \right)^*, \]
\[ L_l = \left( ml \mid rl \right)^*, \]
\[ L_e = m(lr)^*l = ml(rl)^*. \]

These languages are regular.
First Main Ingredient for the Decidability Proof

**Theorem**

*For each constraint system and each fixed tree \( x \), the languages \( L_g \) is regular.*
Proof of Regularity I: Proof System for Deriving Inequalities

\[
TC \vdash ux \geq ux \quad \text{(Reflexivity)}
\]

\[
TC \vdash x \geq y
\]

\[
TC \vdash lx \geq ly \quad \text{(Label)}
\]

\[
TC \vdash uy_i \geq z \\
\]

\[
x \geq y_1 + \ldots + y_n \in TC
\]

\[
TC \vdash ux \geq z \quad \text{(Transitivity)}
\]

Lemma

This proof system is sound and complete.
Proof of Regularity II: Auxiliary Set

We want: Automaton
Problem: We can both push and pop

Compute $Q = \{(x, y) \mid TC' \vdash x \geq y\}$ by the following rules

$(TC'$ is an equivalent variant of $TC$ where there are only constraints $ax \geq y, x \geq ay, x \geq y$ with $a$ only a single letter):

\[
\frac{x \geq y \in TC'}{(x, y) : Q} \quad \text{(BaseCase)}
\]

\[
\frac{x \geq ay \in TC'}{(y, z) : Q \quad az \geq w \in TC'}{(x, w) : Q} \quad \text{(DynamicProg)}
\]

This set can be computed by dynamic programming.
Proof of Regularity III: Automaton

Construct an automaton from $Q$ that accepts $L_{x,y} = \{ u \mid TC' \vdash ux \geq y \}$:

\[
\begin{align*}
(x, y) : Q & \quad (\text{Eps}) \\
\epsilon : L_{x,y}
\end{align*}
\]

\[
\begin{align*}
(x, x') : Q & \quad ax' \geq x'' : TC' \quad (x'', x''') : Q & \quad u : L_{x''', y} & \quad ua : L_{x,y} \quad (\text{ConsumeA})
\end{align*}
\]

Automaton:
Variables as states, Q-edges as $\epsilon$-moves, inequalities $ax' \geq x''$ as moves that consume $a$. Take the disjoint union of these automata for all $x$ to obtain $L_g$ for $y$. 
An Intermediate Step

TCS without arithmetic constraints trivially satisfiable (all 0, ∞).

So are TCS with all bounds for each tree in only one direction.

Lemma

The constraint system \((AC, TC)\) is satisfiable if and only if

\[
\exists v. \forall \alpha, \beta. L_{\alpha}^{\geq} \cap L_{\beta}^{\leq} \neq \emptyset \Rightarrow v(\alpha) \leq v(\beta),
\]

where \(\alpha, \beta\) are linear combinations of tree root variables \(a_i, b_j\), \(v\) is a valuation of them, and \(L_{\alpha}^{\geq}\) is the set of trees with root greater than \(\alpha\).
How we use Regularity

Using the unilateral constraint syntax, one can show that

\[ L_{\geq}^{\alpha} \cap L_{\leq}^{\beta} \subseteq \bigcup_{j} \left( \bigcap_{i} L_{\geq}^{i} \cap L_{\leq}^{j} \right) \]

right sides all finite (which is decidable)

\[ \Downarrow \]

all left sides are finite and we can build an equisatisfiable linear program for \( TC' \) (using a complete and sound proof system for inequalities with sums).

Example

\[ lx \geq x, rx \geq x, y \geq x, x \geq ly + rx + y, \]

\[ L_{\geq}^{\diamond(x)} \cap L_{\leq}^{\diamond(x)} = \{x, lx, rx, y\} \leadsto AC = \{\diamond(x) = \diamond(lx) = \diamond(rx) = \diamond(y), \]

\[ \diamond(x) \geq \diamond(lx) + \diamond(rx) + \diamond(y)\} \]
Otherwise...

- One $L = L_{a_i}^{\geq} \cap L_{b_j}^{\leq}$ is infinite.
  $\Rightarrow$ Roots of all these trees are greater than $a_i$ and less than $b_j$.

- Pumping Lemma $\Rightarrow uv^i w \in L$ for all $i$.
  $\Rightarrow$ loops in form of constraints $vz \geq z, v'z' \leq z'$.

- These loops can be brought into a "list-like" form,
  e.g. $lrz \geq lrlrz + lrlrlrz$:
  $lr$ replaced by $tl \leadsto tz \geq t^2zl + t^3z$.

- Apply the list arguments.
Lemma

Let $vz \geq z + R$ be a tree constraint, with $R$ a sum of tree variables. Let $p$ be the smallest word such that $v \in p^+$. All $w \in R$ with

$$\forall i. TC \not\models w \geq p^i z, w \geq p^i y,$$

can be eliminated. The remaining summands give a new constraint of the form

$$p^i z \geq p^{i_1} z + \ldots p^{i_n} z + z + R,$$

for which $\diamondsuit (p^k z) \_k$ form a list and with no $z$ in $R$.

If this list is strictly increasing, then it is at least linearly increasing and there is for some $i$ a contradiction to $\diamondsuit (uv^i wx) \leq b_j$. 
Third Main Ingredient for the Decidability Proof

Theorem

*Satisfiability of linear list constraints is decidable.*

The proof of it gives us a representation of all lists as a finite set of arithmetic variables.
The Introductory Example once again

Constraints:

\[ lrx \geq x, lx \geq x, mx \geq x, \]
\[ x \geq rlx, x \geq mlx. \]

We have \( L_{\Diamond(x)} = m(lr)^* l = ml(rl)^* \).

If any of the constraints had strict growth, e.g. \( lrx \geq 2x \), we would obtain a contradiction: \( x = mlrlx \geq 2x \).

Thus we may assume \( lrx = x \).
Another example

\[ rx \geq 2x, lx \geq 2x, x \geq lr, \diamond(x) = 1. \]

Then \( L^{\geq}_{\diamond(x)} \cap L^{\leq}_{\diamond(x)} \subseteq L^{\geq}_{\diamond(x)} \cap L^{\leq}_{\diamond(x)} = L^{\geq}_{x} \cap L^{\leq}_{x} = L^{=}_{x} = (lr)^{*}. \)

Schematic notation for \( x \)

\[
\begin{array}{c}
\diamond(x) \\
\diamond(lx) \& \diamond(rx) \\
\hat{c}_1 \& \hat{c}_2 \& x \& \hat{c}_3
\end{array}
\]

\[ AC = \{\diamond(lx) \geq 2\diamond(x), c_1 \geq 2\diamond(lx), c_2 \geq 2\diamond(rx), \\
\diamond(rx) \geq 2\diamond(x), \diamond(x) \geq 2\diamond(lx), c_3 \geq \diamond(rx)\}. \]

Then \( AC \) implies \( \diamond(x) = \infty, \diamond(lx) = \diamond(rx) = \infty \) or \( \diamond(x) = 0 \), contradicting \( \diamond(x) = 1. \)
Overview of the Proof

- We described how to derive all inequalities following from a constraint system.
- We characterized the set of trees greater than a fixed tree as regular language.
- We reduced easy cases directly to linear programming.
- We gave a modified elimination procedure that brings the constraints in a list-like shape and
  - we employed the list constraint solving algorithm so solve them (also via LP).

We can now decide satisfiability of linear tree constraints.
Applications and Future Work

Goal: automatic and powerful tool for object oriented analysis.

- Finish implementing and evaluate our algorithms.
- Optimizations, Complexity
- Translation between Java and RAJA or
- add Java features to RAJA (as exceptions, garbage collection, concurrent threads)

Thanks! Questions?