Decidable linear tree constraints

Sabine Bauer

LaSh Workshop 2018

Linear tree constraints arise in object-oriented resource analysis, when inferring resource types for programs in a Java-like language named RAJA (Resource Aware JAva). These types encode the heap space usage of programs, and one can calculate an upper bound on it using the constraint solutions.

We build on the work by Hofmann, Jost, and others, who carried out type based resource analysis for functional [1, 4, 5] and object-oriented languages [2, 3, 6, 7, 8, 9].

The constraints are linear inequalities between infinite trees, which have nonnegative rational or real numbers or a symbol for infinity in their nodes. These trees are added and compared pointwise. In addition to that, we have arithmetic constraints on selected nodes that have the form of a linear program.

Figures 1, 2 and 3 present the formal version of the above description.

Figure 1: Linear List Constraint Syntax

$$
\begin{aligned}
l &::= \mathbf{x} | \mathbf{tail}(\mathbf{x}) && \text{(Atomic list)} \\
t &::= l | t + t && \text{(List term)} \\
c &::= t \geq t && \text{(List constraint)}
\end{aligned}
$$

Figure 2: Linear Tree Constraint Syntax

$$
\begin{aligned}
t &::= \mathbf{x} | \mathbf{l}(\mathbf{x}), \text{ where } l \in L \text{ with } L < \infty && \text{(Atomic tree)} \\
te &::= t | te + te && \text{(Tree term)} \\
c &::= te \geq te && \text{(Tree constraint)}
\end{aligned}
$$

An example of a list constraint is $\mathbf{tail}(\mathbf{tail}(\mathbf{x})) \geq \mathbf{tail}(\mathbf{x}) + \mathbf{x}$, which states that the unknown list $\mathbf{tail}(\mathbf{tail}(\mathbf{x}))$ is growing at least as fast as the Fibonacci sequence (depending on the initial variables $\mathbf{head}(\mathbf{x})$ and $\mathbf{head}(\mathbf{tail}(\mathbf{x}))$).

Figure 3: Arithmetic Constraint Syntax

$v ::= n|\lambda|\mathbf{head}(l)$ (Arithmetic expression (number, variable or head of an atomic list))
$h ::= v|h + h$ (Head term)
$c ::= h \geq h$ (Arithmetic constraint)

An example of a tree constraint system for a tree $\mathbf{x}$ degree three with labels $\mathbf{l}, \mathbf{r}, \mathbf{m}$ is

$$\begin{aligned} \mathbf{lr}(\mathbf{x}) \geq \mathbf{x}, \mathbf{l}(\mathbf{x}) \geq \mathbf{x}, \mathbf{m}(\mathbf{x}) \geq \mathbf{x} \\ \mathbf{ml}(\mathbf{x}) \leq \mathbf{x}, \mathbf{rl}(\mathbf{x}) \leq \mathbf{x}. \end{aligned} \tag{1}$$

We study the question of simultaneous satisfiability of a system of such constraints. If we translate the tree constraints into such a linear program, we obtain an infinite number of linear inequalities, which cannot be solved directly.

We split our considerations in three variants of the problem: lists in general, a restricted sort of list constraints, and this restricted form generalized to trees.

For the first, general list constraints, there is a close connection to recurrences. We use this fact to show that in its general form this satisfiability problem is — already in the list case — hard for the famous Skolem-Mahler-Lech problem whose decidability status is still open but which is at least NP-hard.

We thus identified a subcase of the problem that still covers all instances arising from type inference in the aforementioned amortized analysis and show decidability of satisfiability for lists by a reduction to linear programming. To make this possible, we show that the list constraints are equisatisfiable over the set of periodic lists and that there are three observations:

- lists that have strict growth cannot have any upper bounds such that they can be set to infinity.
- Similarly, lists that are strictly decreasing have no lower bounds and thus can be set to zero.
- Lists with upper and lower bounds are periodic.

The algorithm that is derived from the proof has polynomial complexity.

In the case of trees replacing all trees by the analogous to periodic lists — trees with only finitely many subtrees — seems to be impossible. Still, linear tree constraints of that form are also decidable, which we prove with another approach using finite automata and word combinatorics. In contrast to the list case, the corresponding decision algorithm is no longer polynomial.

In the above example, we must calculate the set L of subtrees equal to the tree x. This can be done by intersecting the language L_1 consisting of labels of trees greater than x with the language

L_2 for trees less than x. These two languages are here obtained by iterative application of labels:

$$L_1 = (\mathbf{l} \mid \mathbf{lr} \mid \mathbf{m})^*, L_2 = (\mathbf{ml} \mid \mathbf{rl})^*, L = \mathbf{ml}(\mathbf{rl})^*$$

In other cases, it is more complicated, since we may have subtrees on both sides of the inequality sign. This corresponds to deleting label symbols when combining the constraints as in the example system (1).

We show that still this language can be described by a finite automaton and thus intersection with other regular languages is always possible and results in a regular language. By taking the union over this languages, we have now found a description of trees that are bounded only from below, only from above, or in both directions. We use this knowledge to replace all nodes bounded in only one direction by zero or infinity, and for the remaining nodes we calculate the intervals in which they must lie by an iterative procedure. We then show that the set of the inequalities stating that these intervals are nonempty is finite, by an argument that shows that all the different branchings in the tree that are strictly increasing have resemblance with the list case. Thus we can employ the arguments used there and obtain a finite number of linear inequalities that can finally be solved by an LP-solver.

We stress that the reduction to a finite number of inequalities does not entail a description of the trees by trees with only a finite number of subtrees. The reason is that the nodes that fulfill the inequalities can be split over different subtrees (as prescribed by the regular languages L_i).

The proof for the tree case yields a decision procedure that covers the entire range of constraints needed for resource analysis and we have now set the theoretical basis to analyze arbitrary RAJA programs.

References

[1] Hofmann, Martin and Jost, Steffen, Static Prediction of Heap Space Usage for First-order Functional Programs, Proceedings of the 30th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages ,POPL '03, 2003

[2] Hofmann, Martin and Jost, Steffen, Type-Based Amortised Heap-Space Analysis, Proceedings of the 15th European Symposium on Programming (ESOP), Programming Languages and Systems

[3] Hofmann, Martin and Rodriguez, Dulma , CSL: 18th EACSL Annual Conference on Computer Science Logic, Efficient Type-Checking for Amortised Heap-Space Analysis , 2009

[4] Hoffmann, Jan and Hofmann, Martin , Programming Languages and Systems: 19th European Symposium on Programming, ESOP 2010, Amortized Resource Analysis with Polynomial Potential , 2010

[5] Hoffmann, Jan and Aehlig, Klaus and Hofmann, Martin, ACM Trans. Program. Lang. Syst. , Multivariate Amortized Resource Analysis , 2012

[6] Hofmann, Martin and Rodriguez, Dulma , Proceedings of the 18th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning , LPAR'12 , Linear Constraints over Infinite Trees , 2012

[7] Hofmann, Martin and Rodriguez, Dulma, ESOP: 22nd European Symposium on Programming, Automatic Type Inference for Amortised Heap-Space Analysis, 2013

[8] Bauer, Sabine and Hofmann, Martin, LPAR-21. 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning , EPiC Series in Computing, Decidable linear list constraints, 2017

[9] Hoffmann, Jan and Das, Ankush and Weng, Shu-Chun, Towards automatic resource bound analysis for OCaml , CoRR, 2017