EPiC Series in Computing

Volume 46, 2017, Pages 181–199

LPAR-21. 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning

Decidable Linear List Constraints

Sabine Bauer[1]* and Martin Hofmann[2]

[1] University of Munich, Germany
Sabine.Bauer@ifi.lmu.de
[2] University of Munich, Germany
Hofmann@ifi.lmu.de

Abstract

We present new results on a constraint satisfaction problem arising from the inference of resource types in automatic amortized analysis for object-oriented programs by Rodriguez and Hofmann.These constraints are essentially linear inequalities between infinite lists of nonnegative rational numbers which are added and compared pointwise. We study the question of satisfiability of a system of such constraints in two variants with significantly different complexity. We show that in its general form (which is the original formulation presented by Hofmann and Rodriguez at LPAR 2012) this satisfiability problem is hard for the famous Skolem-Mahler-Lech problem whose decidability status is still open but which is at least NP-hard. We then identify a subcase of the problem that still covers all instances arising from type inference in the aforementioned amortized analysis and show decidability of satisfiability in polynomial time by a reduction to linear programming. We further give a classification of the growth rates of satisfiable systems in this format and are now able to draw conclusions about resource bounds for programs that involve lists and also arbitrary data structures if we make the additional restriction that their resource annotations are generated by an infinite list (rather than an infinite tree as in the most general case). Decidability of the tree case which was also part of the original formulation by Hofmann and Rodriguez still remains an open problem.

1 Introduction

Motivation. In this paper we study linear inequalities with variables ranging over infinite sequences of numbers in the set $\mathbb{D} := \mathbb{Q}_0^+ \cup \{\infty\}$. This is a way to generalize linear programming to infinitely many variables.

The inclusion of infinity is motivated from resource analysis; it represents availability of unlimited resources in certain places. It also simplifies and streamlines the mathematical development. However, our results remain true if we remove infinity, as we discuss in section 8.

An example for the inequalities in list notation is $\mathbf{tl}(\mathbf{x}) \geq \mathbf{x} + 3\mathbf{tl}(\mathbf{tl}(\mathbf{y}))$, which is understood pointwise. These systems were introduced in [1] but only a heuristic approach for solving them

*DFG Graduiertenkolleg 1480, with Technical University of Munich

based on Fourier-Motzkin was given there. In this paper we show hardness of the general case as formulated in loc. cit. for the Skolem-Mahler-Lech problem, which makes it unlikely to be decidable and in any case very hard. However, we have identified a subcase covering in particular an important application, namely resource analysis of object-oriented programs, and provide polynomial-time algorithms for deciding satisfiability and also for the growth rate of minimal solutions. We remark that this is the first nontrivial decidability result for linear inequalities with infinitely many variables. Before we go into detail we will briefly recall the motivation from resource analysis.

Linear arithmetic constraints were used by Hofmann and Jost in the context of automatic type-based amortized resource analysis by the potential method [2] where it was applied to first-order functional programs. In this approach the available resources of expressions are captured in type annotations (resource types). The idea is to automatically generate constraints on the resource types from a program such that their solutions allow to compute the resource usage of the program. The constraint systems appearing in this analysis have finitely many variables and can be reduced to linear programming. The same was true for subsequent extensions to higher-order functions and more complex potential functions [3, 4, 5, 6, 7]. The extension of this method to object-oriented programs in a language RAJA (Resource Aware JAva, a fragment of Java with inheritance, recursive methods and extended with resource annotations in the types) [8, 9, 10, 1] led to the use of constraints involving infinite lists or trees whose entries are numerical variables. In this case, it is no longer a direct option to solve the constraints with linear programming. A heuristic procedure was developed [10], which allowed to find solutions in many cases, but the general question of decidability of these infinitary constraint systems remained open. It is precisely this question that we treat in this paper.

The contributions of this paper are the following:

- We prove that the list constraint problem is computationally hard.
- We identify a subclass which captures all instances from program analysis,
- give a polynomial decision procedure for this subclass and
- determine bounds on the growth rates for the minimal solutions of systems of constraints.

With this knowledge and the ability to solve the constraints, it is now possible to analyse the resource usage of list programs in RAJA.

Related work. Although our list constraint problem has a very simple and natural formulation (see section 2), it has not yet been studied extensively and it is not subsumed by existing decidability or growth rate results, such as [11] and [12]. The authors of these papers use difference bound constraints on infinite sequences. Our constraints contain more conditions than difference bounds, and further the array logic used in [11] does not allow to add entire lists pointwise, just comparison of two lists and constant differences are allowed. We could cover this with our constraints, but on the other hand there are operations we do not have (modulo constraints and relations between indices). Other related publications are [1] and [13] and both of them also notice that there exists only very few research on constraint systems with infinitely many variables. The first paper [1] introduced the problem we are interested in and the second paper [13] is about satisfiability of constraints with Boolean variables and higher-dimensional index arithmetic, which leads to undecidability in most cases.

Organization. The remainder of this paper is organized as follows: In section 2 we introduce the syntax and semantics of list constraints. In section 3 we establish a reduction from the Skolem-Mahler-Lech problem to the general case which makes it unlikely to be decidable. In section 4 we identify a hitherto unexplored fragment of the general problem which is still sufficient for the purposes of amortized resource analysis and prove in section 5 that satisfiability is decidable for that fragment. Section 6 describes estimates on the growth rate of solutions to satisfiable systems. In particular, we provide nontrivial upper bounds on such solutions. However, we cannot at this point prove tightness of such bounds in all cases and thus the question whether a given constraint system admits a solution that lies below a fixed bound remains open. In section 7, we further explain the connection to resource analysis and the applications of our results. We conclude in section 8.

2 Syntax and Semantics

The constraint systems are described as follows. One has finitely many unknowns ranging over infinite lists (sequences) of nonnegative rational numbers including ∞. Let us denote this latter set by $\mathbb{D} := \mathbb{Q}_0^+ \cup \{\infty\}$. Terms and constraints are built from those unknowns by addition ($+$) and comparison ($\geq$) understood pointwise, and the tail function ($\mathbf{tl}$) that removes the first element of a sequence. Furthermore, constraint systems may contain ordinary linear arithmetic constraints involving the heads of the lists as given by the function $\mathbf{hd}$.

An example of such a constraint is $\mathbf{tl}(\mathbf{tl}(\mathbf{x})) \geq \mathbf{tl}(\mathbf{x}) + \mathbf{x}$, which states that the unknown list $\mathbf{tl}(\mathbf{tl}(\mathbf{x}))$ is growing at least as fast as the Fibonacci sequence (depending on the initial variables $\mathbf{hd}(\mathbf{x})$ and $\mathbf{hd}(\mathbf{tl}(\mathbf{x}))$). To illustrate our list notation further, we view infinite lists as arrays, and this constraint becomes $\mathbf{x}[i] \geq \mathbf{x}[i-1] + \mathbf{x}[i-2] \forall i \geq 2$ in array notation. The pointwise inequalities state not only that the i'th position of the list $\mathbf{x}$ has to be greater than or equal to the i'th position of the Fibonacci list with starting variables $\mathbf{hd}(\mathbf{x})$ and $\mathbf{hd}(\mathbf{tl}(\mathbf{x}))$ for all i, but that in each position i the entry there has to be greater than or equal to the sum of the previous two entries. For instance, the list $1, 1, 2, 99, 5, 8, 13, \ldots$ is greater than a Fibonacci sequence with initial values 1 and 1. However, it does not fulfill the constraint since 5 is not greater than 2+99. The minimal continuation of this list beginning at index 5 is $101, 200, 301, 501, 802, \ldots$.

Example 1. *The following constraint system is satisfiable:*

Arithmetic Constraints	*List Constraints*
$\mathbf{hd}(\mathbf{x}) = \mathbf{hd}(\mathbf{z}) = 2$	$\mathbf{tl}(\mathbf{y}) \geq \mathbf{y}$
$\mathbf{hd}(\mathbf{tl}(\mathbf{x})) = 5$	$\mathbf{z} \geq \mathbf{tl}(\mathbf{z}) + \mathbf{tl}(\mathbf{z})$
$\mathbf{hd}(\mathbf{y}) \geq 1$	$\mathbf{tl}(\mathbf{tl}(\mathbf{x}))) \geq \mathbf{tl}(\mathbf{x}) + \mathbf{x} + \mathbf{tl}(\mathbf{y}) + \mathbf{tl}(\mathbf{y})$

and has the equivalent formulation in array notation

Arithmetic Constraints	*List Constraints*
$\mathbf{x}[0] = \mathbf{z}[0] = 2$	$\mathbf{y}[i+1] \geq \mathbf{y}[i] \forall i \geq 0$
$\mathbf{x}[1] = 5$	$\mathbf{z}[i] \geq \mathbf{z}[i+1] + \mathbf{z}[i+1] \forall i \geq 0$
$\mathbf{y}[0] \geq 1$	$\mathbf{x}[i+2] \geq \mathbf{x}[i+1] + \mathbf{x}[i] + \mathbf{y}[i+1] + \mathbf{y}[i+1] \forall i \geq 0$

It has the (minimal) solutions $\mathbf{y}$ *a constant list,* $\mathbf{z}$ *a list with first element 2 and the rest zero and* $\mathbf{x}$ *a "Fibonacci list" as above, with an additional summand.*

$$\mathbf{y} = 1, 1, \ldots, \ \mathbf{z} = 2, 0, 0, \ldots,$$
$$\mathbf{x} = 2, 5, 9, 16, 27, 45, .., \mathbf{x}_{n-1}, \mathbf{x}_n, \mathbf{x}_n + \mathbf{x}_{n-1} + 2, \ldots \ .$$

If we add the constraint $\mathbf{x} \leq \mathbf{z}$*, (i.e.* $\mathbf{x}[i] \leq \mathbf{z}[i], \forall i \geq 0$*) the constraints become unsatisfiable, since a maximal solution to the list constraint on* $\mathbf{z}$ *is an exponentially decreasing list*

$$\mathbf{z} = 2, 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots, \frac{1}{2^n}, \ldots \ .$$

We remark that tree constraints without arithmetic constraints are always satisfiable with infinite lists of zeros. In the sequel, we will only use the list notation with **hd** and **tl**.

3 Hardness of the Problem for General Linear List Constraints

In order to establish the announced hardness and also for the subsequent development it is useful to draw a connection between list constraints and recurrences and rational generating functions as for example in [14, 15]. In this section, we use this connection to show the hardness of the list constraint problem. We consider the two decision problems in Figure 3 and establish a reduction between them.

Figure 1: Decision problems

- **Skolem-Mahler-Lech Problem (SML)**

  Given: A homogeneous linear recurrent sequence of degree k with initial values $b_1, \ldots, b_k$ and constant rational coefficients of the form

  $$\begin{aligned} &x_n = a_1 x_{n-1} + \cdots + a_k x_{n-k}, n > k \\ &x_1 = b_1, \ldots, x_k = b_k \\ &a_i, b_i \in \mathbb{Q}, b_i \geq 0, \text{ for all } i = 1, \ldots, k, a_k \neq 0. \end{aligned}$$

  Asked: Is there an index n such that $x_n = 0$?

- **List Constraint Satisfiability Problem (LC)**

  Given: A finite system C of list constraints and arithmetic constraints.

  Asked: Is there a set of lists, which simultaneously satisfies all constraints of the system in $\mathbb{D}$?

The SML problem is not known to be decidable and is believed to be computationally very hard (see [16, 17, 18]).

Theorem 1. *The list constraint satisfiability problem is hard for the Skolem-Mahler-Lech-Problem.*

We give an example for the translation between recurrences and list constraints. With this we can express the question

$$\text{Is } x_i \geq 0 \text{ for all } i? \quad (\text{SML}')$$

in terms of the constraints. Now C is satisfiable if and only if (SML′) for $\mathbf{x}$ has a positive answer and, according to [16, 19], SML can be reduced to SML′.

Example 2. *Consider the recurrence*

$$x_1 = x_2 = \cdots = x_{10} = 1,$$
$$x_n = -\frac{1}{2}x_{n-1} + 2x_{n-3} + x_{n-10}, n > 10.$$

The constraint system C consisting of the constraints

$$\mathbf{tl}^{10}(\mathbf{x}) + \mathbf{tl}^{10}(\mathbf{x}) + \mathbf{tl}^{9}(\mathbf{x}) \; R_i \; \mathbf{tl}^{7}(\mathbf{x}) + \mathbf{tl}^{7}(\mathbf{x}) + \mathbf{tl}^{7}(\mathbf{x}) + \mathbf{tl}^{7}(\mathbf{x}) + \mathbf{x} + \mathbf{x}, i = 1, 2$$
$$\mathbf{hd}(\mathbf{x}) = \mathbf{hd}(\mathbf{tl}(\mathbf{x})) = \mathbf{hd}(\mathbf{tl}^2(\mathbf{x})) = \cdots = \mathbf{hd}(\mathbf{tl}^9(\mathbf{x})) = 1$$

for $R_1 =$ " $\geq$ " and $R_2 =$ " $\leq$ " describes the list with entries exactly the elements of the recurrence, since the inequality in the constraint holds pointwise for all entries of the list after position 10.

Corollary 1. *The problem LC is NP-hard.*

This follows from the NP-hardness of the SML problem [17] and from the fact that the reduction was polynomial many-to-one. The size of the constraint system C is linear in the length of the encoding for the recurrence.

Further, we remark that certain behaviors cannot arise as unique solutions (i.e. solutions described by a recurrence relation) to type inference problems (described in terms of list constraints). A concrete example is the harmonic sequence $(\frac{1}{n})_n$, because it has a logarithmic generating function [20], whereas all linear recurrence sequences have a rational generating function. Clearly, the harmonic sequence is a non-unique solution, for instance to the constraint $\mathbf{x} \geq \mathbf{tl}(\mathbf{x})$.

4 Unilateral Subcase

Unilateral constraints are sufficient for amortized analysis. Fortunately, the hardness construction does not carry over to type inference in RAJA. Indeed, we were able to identify a well-delineated subcase of LC which contains the image of the translation from type-based resource inference and, as we will show, enjoys decidable satisfiability.

We call a list constraint *unilateral*, if it has only one summand on the greater side of the inequality. This means it is of the form

$$\mathbf{x} \geq \mathbf{y}_1 + \cdots + \mathbf{y}_m,$$

where $\mathbf{x}, \mathbf{y}_i$ are terms of the form $\mathbf{tl}^j(\mathbf{z})$ for some variable $\mathbf{z}$ and $j \in \mathbb{N}_0$. Note that with this definition we do not allow negative coefficients, in contrast to the general case. This causes a significant decrease of complexity, as we will see in section 5.

Definition 1. *ULC is the special case of LC where all list constraints are unilateral.*

For space reasons, we will neither show the constraint generation nor the proof that unilateral constraints are sufficient for resource inference, but we give a short informal argument. In RAJA, variables are refined with resource annotations and their resources can be used to "pay" for costs arising from field update, use as parameter, recursive calls, etc. Inequalities between these variables then come from subtyping (subtypes have at least the same potential as their supertypes, recursive calls can just be done with subtypes of the type the method belongs to)

and aliasing (i.e. multiple use of the same variable), where the potential of the variable itself has to be at least the sum of the potential of the aliases. There is no rule in the constraint generation that yields inequalities such that a sum of lists that is greater than another term. Accordingly, it is not allowed to collect potential from several lists infinitely many times to pay for operations. Note that in contrast to list constraints, the arithmetic constraints underlie no restriction.

Normal form for unilateral constraints. We have developed a procedure, which enables us to analyze the class of list programs by deciding ULC. Before we present this result, we make another simplification. With $\mathbf{x}^{(i)}$ we denote $\mathbf{tl}^i(\mathbf{x})$. We further write $n\mathbf{x}$ for the repeated addition $\mathbf{x} + \cdots + \mathbf{x}$. Let an unilateral constraint system be given. We apply the elimination procedure in [1] to these constraints in order to bring it in a suitable form for our algorithm. This procedure essentially eliminates variables, which appear only on one side of the inequalities in the style of Fourier-Motzkin elimination [21] without affecting satisfiability.

One can show that after the procedure, the constraints of the system take the form (which we will call normal form in the sequel)

$$\mathbf{x}^{(i)} \geq \mathbf{lin}(\mathbf{x}) + \mathbf{R}, \tag{1}$$

where $\mathbf{lin}(\mathbf{x})$ is a *nonempty* sum of terms $\mathbf{tl}^j(\mathbf{x})$ and $\mathbf{R}$ is a (possibly empty) sum of other terms of the form $\mathbf{tl}^j(\mathbf{y})$ with $\mathbf{x} \neq \mathbf{y}$. Additionally, we may have constraints on the same variables (i.e. those variables for which a constraint of the form (1) exists): $\mathbf{x}^{(i)} \geq \mathbf{R}$. In other words, the constraints in normal form consist of inequalities between *non-eliminable* (w.r.t. the procedure in [1]) variables, namely those which appear on two sides of an inequality.

For example, the constraint system

$$\mathbf{x}^{(3)} \geq \mathbf{y}^{(3)}, \mathbf{y}^{(3)} \geq \mathbf{x}^{(2)} + \mathbf{z}^{(1)}, \mathbf{z}^{(1)} \geq \mathbf{x}^{(1)}.$$

yields after two applications of the elimination procedure (eliminating $\mathbf{y}$ and $\mathbf{z}$) the inequality $\mathbf{x}^{(3)} \geq \mathbf{x}^{(2)} + \mathbf{x}^{(1)}$ and is now in normal form and the elimination procedure can no longer be applied. Since the constraints hold pointwise, this implies $\mathbf{x}^{(3+i)} \geq \mathbf{x}^{(2+i)} + \mathbf{x}^{(1+i)}$, for all i.

Example 3. *The constraints*

$$\mathbf{x}^{(2)} \geq \mathbf{x}^{(1)} + \mathbf{y}^{(3)}, \tag{2}$$
$$\mathbf{x}^{(2)} \geq 2\mathbf{x}^{(1)} + 3\mathbf{y}, \tag{3}$$
$$\mathbf{y}^{(2)} \geq 4\mathbf{y}^{(1)} + \mathbf{x}^{(1)}, \tag{4}$$
$$\mathbf{hd}(\mathbf{y}^{(1)}) = 1, \tag{5}$$
$$\mathbf{hd}(\mathbf{x}^{(2)}) \leq 2. \tag{6}$$

form a constraint system from ULC with two constraints on the variable $\mathbf{x}$*. It is unsatisfiable in* $\mathbb{D}$*, since the constraints (2) and (4) imply*

$$\mathbf{x}^{(2)} \geq \mathbf{x}^{(1)} + 4\mathbf{y}^{(2)} + \mathbf{x}^{(2)},$$

which means $\mathbf{hd}(\mathbf{x}^{(1)})$ *and* $\mathbf{hd}(\mathbf{y}^{(2)})$ *have to vanish or* $\mathbf{hd}(\mathbf{x}^{(2)}) = \infty$*. If* $\mathbf{hd}(\mathbf{y}^{(2)}) = 0$*, then (4) and (5), which together imply* $\mathbf{hd}(\mathbf{y}^{(2)}) \geq 4$ *(or (6) if* $\mathbf{hd}(\mathbf{x}^{(2)}) = \infty$*, respectively) cannot be satisfied.*

5 Decidability of ULC

We now show our decidability result for unilateral list constraints. Let us call an infinite list $x_0, x_1, x_2, \ldots$ *periodic* if there exist $k \geq 0, l > 0$ such that $x_i = x_{i+l}$ holds for all $i \geq k$.

Lemma 1. *If there are two constraints of the form*

$$\mathbf{x}^{(i)} \geq \mathbf{x}^{(j)} + \mathbf{R}, j < i \qquad (LB)$$
$$\mathbf{x}^{(i)} \geq \mathbf{x}^{(k)} + \mathbf{R}', i < k \qquad (UB)$$

with $\mathbf{R}, \mathbf{R}'$ arbitrary sums of terms $\mathbf{tl}^j(\mathbf{z})$, every satisfying solution maps $\mathbf{x}$ to a periodic list.

Moreover, the period length is bounded by $\mathrm{lcm}(i-j, k-i)$ and the periodicity starts after at most i steps.

Proof. Since $\mathbf{R}$ is nonnegative, we have $\mathbf{x}^{(i)} \geq \mathbf{x}^{(j)}$, and thus we know that $\mathbf{x}$ is non-decreasing in each $(i-j)$th position beginning from entry number j. We also have $\mathbf{x}^{(i)} \geq \mathbf{x}^{(k)}$, which means it is non-increasing in each $(k-i)$th position from position i on; thus it is periodic with the period length at most $\mathrm{lcm}(i-j, k-i)$. □

Remark 1.

- *We will call constraints of the forms (LB) and (UB) lower and upper bounds. Upper bounds have as solutions at most constant lists and lower bounds have solution lists that are at least constant.*
- *The upper bound on the period length can be sharpened by applying the theorem of Fine and Wilf [22], but for our purposes it is not necessary.*

The importance of Lemma 1 lies in the fact that we can replace a list variable with finitely many arithmetic variables. Note that the premise to Lemma 1 may be satisfied by a single constraint which is simultaneously a lower and upper bound, as $\mathbf{x}^{(2)} \geq \mathbf{x}^{(1)} + \mathbf{x}^{(3)}$. In many cases such periodic constraints with two summands on the smaller side of the inequality are unsatisfiable. Nevertheless, we first exploit the periodicity by replacing the lists by arithmetic variables. If the constraints are unsatisfiable, the constraints on these arithmetic variables will be contradictory, too. This will become clearer in the proof of Theorem 2 .

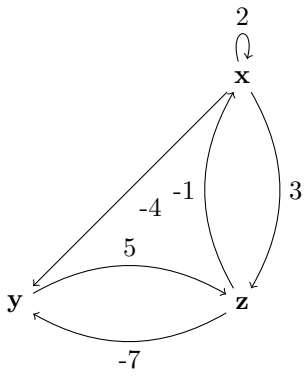
Our aim is now to deduce periodic constraints (i.e. list constraints as in Lemma 1) in list constraint systems. For that, we introduce a weighted graph G for the list constraints by creating a node for each variable. Note that we have w.l.o.g. only terms of the form $\mathbf{v}^{(n)}$ on the left hand side of the inequalities, with $\mathbf{v}$ a variable and n the same number for all variables (possibly after applications of $\mathbf{tl}$, e.g. the constraints $\mathbf{x}^{(2)} \geq \mathbf{x}^{(1)}, \mathbf{y}^{(3)} \geq \mathbf{y}^{(2)}$ become $\mathbf{x}^{(3)} \geq \mathbf{x}^{(2)}, \mathbf{y}^{(3)} \geq \mathbf{x}^{(2)}$ and a new constraint $\mathbf{hd}(\mathbf{x}^{(2)}) \geq \mathbf{hd}(\mathbf{x}^{(1)})$ is added to the set of arithmetic constraints). We create an edge weighted with $i \in \mathbb{Z}$ from node $\mathbf{x}$ to node $\mathbf{y}$ if in a constraint for $\mathbf{x}^{(n)}$ a term $\mathbf{y}^{(n+i)}$ appears. For instance, if we start with $\mathbf{x}^{(20)}$ for the constraints in Fig. 2, we get bounds in both directions:

$$\mathbf{x}^{(20)} \geq \max(\mathbf{x}^{(22)}, \mathbf{x}^{(18)})$$

by using either the cycle of weight 2 in the graph or by visiting the nodes in the order $\mathbf{x}, \mathbf{z}, \mathbf{y}, \mathbf{z}, \mathbf{y}, \mathbf{z}, \mathbf{x}$. Note that if we have cycles with zero weight, as for instance $\mathbf{x}^{(20)} \geq \mathbf{x}^{(20)} + \mathbf{R}$, then all lists in $\mathbf{R}$ have to vanish.

We use this graph to show the following.

$$\mathbf{x}^{(10)} \geq \mathbf{x}^{(12)} + \mathbf{y}^{(6)} + \mathbf{z}^{(13)}$$
$$\mathbf{y}^{(10)} \geq \mathbf{z}^{(15)}$$
$$\mathbf{z}^{(10)} \geq \mathbf{x}^{(9)} + \mathbf{y}^{(3)}$$



Step 1	+ cycles	- cycles
x	1	0
y	0	1
z	1	1

Step 2	+ cycles	- cycles
x	1	1
y	1	1
z	1	1

Figure 2: Example: from the constraints to the graph

Lemma 2. *We can compute in polynomial time the following data:*

- *a partition of the variables* $X = X_p \cup X_{p'} \cup X_0 \cup X_\infty$
- *natural numbers* s *and* d

such that any solution S *will map the variables in* X_p *("necessarily periodic variables") to periodic lists with period* d *starting at* s *and such that changing (in* S*) the valuations of the variables in* $X_{p'}$ *("periodic variables") to periodic lists and the valuations of* X_0 *("zero variables") to* 0_s *(only zeros after* s*) and those of the variables in* X_∞ *("infinity variables") to* ∞_s *(infinity after* s*) yields a solution as well.*

Proof. We divide the list variables into those with positive cycles, those with negative cycles and those which have both. We use the Bellman-Ford algorithm to detect negative cycles in the graph G beginning at node $\mathbf{x}$. We can also find positive cycles by multiplying the labels with -1. Negative cycles correspond to lower bounds and positive cycles to upper bounds on $\mathbf{x}$. Thus we have a periodic constraint if there are both positive and negative cycles.

We set up a table with rows labeled with the variables and columns labeled with "+" and "-". We will have 1 in row $\mathbf{x}$ and column "+" (resp. "-") if there is a positive (resp. negative) cycle beginning from $\mathbf{x}$, else we fill in 0. We start with filling in the cycles that contain each variable only once (Step 1 in Figure 2) and update the table in the following way (Step 2 in Figure 2): If a positive (resp. negative) cycle from variable $\mathbf{x}$ contains a variable $\mathbf{y}$ (i.e. they are in the same strongly connected component) and if $\mathbf{y}$ has a negative (resp. positive) cycle, we can conclude that $\mathbf{x}$ has a negative (resp. positive) cycle. This follows from the fact that we can run around this additional cycle from $\mathbf{y}$ as many times as are necessary to change the sign of the weight of the resulting cycle.

The lists with 1 in both columns of the table in Fig. 2 belong to X_p. Now only those variables remain for which we have either only positive or only negative cycles.

We now say that a variable $\mathbf{x}$ has a lower (resp. upper) bound if it has a negative (resp. positive) cycle *or* if it is greater (resp. less) than another variable $\mathbf{y}$. For X_0 and X_∞ we can

argue by induction on the position of the variables in G for the constraint system. If there are only lower bounds for some list $\mathbf{x}$, which means the node $\mathbf{x}$ in the graph has only negative cycles and no incoming edges from other variables, then we set $\mathbf{x}^{(n)}$ to the list ∞_s, which consists only of infinity. If there are only upper bounds for a list $\mathbf{x}^{(n)}$, which means that the node $\mathbf{x}$ has only positive cycles and no outgoing edges to other variables, we set $\mathbf{x}^{(n)}$ to the zero list 0_s.

If there are both upper and lower bounds on a list $\mathbf{x}^{(n)}$, which do not fit into the situation from Lemma 1, then there are three possible cases.

Case 1: (see Fig. 3): there is a negative cycle in $\mathbf{x}$ and $\mathbf{x}$ is not reachable from other variables $\mathbf{y}$ with a positive cycle. This means $\mathbf{x}$ is bounded from above only by other variables $\mathbf{z}$ that are all *unbounded* from above. Then we set also $\mathbf{x}^{(n)}$ to ∞_s.

Figure 3: Case 1

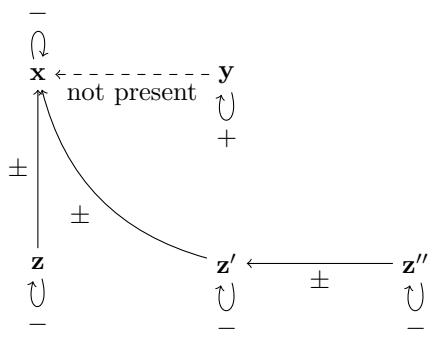
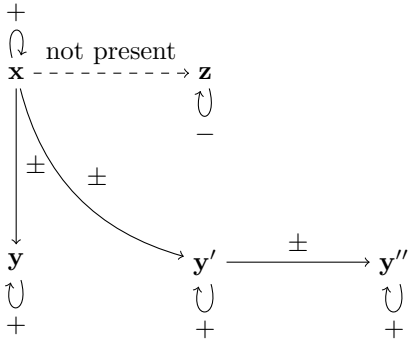


Figure 4: Case 2



Case 2 (see Fig. 4): In this case $\mathbf{x}$ has a positive cycle and no other variables $\mathbf{y}$ with negative cycle are reachable from $\mathbf{x}$. This means that $\mathbf{x}$ is only bounded from below by variables that have no lower bound. In this case we intend to set $\mathbf{x}$ to a list that finally vanishes. But we cannot simply set $\mathbf{x}^{(n)}$ to 0_s, because there may be a constraint on $\mathbf{x}$ like

$$\mathbf{x}^{(n)} \geq \mathbf{x}^{(n+j)} + \mathbf{y}^{(i)}, i < n \tag{7}$$

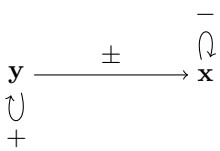
and, for instance, an arithmetic constraint $\mathbf{hd}(\mathbf{y}^{(i)}) \geq 1$.(Recall that until position $n-1$ we can have arithmetic constraints.) We observe that, if we have a path of weight $-j$ from $\mathbf{x}$ to $\mathbf{y}$, as in (7), we cannot have a path with weight less than j from $\mathbf{y}$ to $\mathbf{x}$ because then we would have a negative cycle in $\mathbf{x}$.

In the example (7) we can set $\mathbf{y}^{(n)} = 0_s$ and $\mathbf{x}^{(2n)} = 0_s$. According to the observation, all constraints on $\mathbf{y}$ only contain terms of $\mathbf{x}$ with index at least $2n$ and from this position on $\mathbf{x}$ can be set to 0_s, so the problem does not occur any more.

In order to generalize this to more variables, we now build a new directed graph G', which has no negative cycles. As nodes, we take all variables with only positive cycles. The edges of G' are the negative weighted edges in the original graph G. We begin with the variables that have only incoming edges and set all of them to lists vanishing from the n'th position. All other nodes $\mathbf{x}$ have edges to variables $\mathbf{y}_1, \ldots \mathbf{y}_k$. If all $\mathbf{y}_i^{(n')}$ have been set to 0_s in the previous steps, we set also $\mathbf{x}^{(n+n')}$ to 0_s.

Case 3 (see Fig. 5): Now only the case remains where we have a variable $\mathbf{x}$ with a negative cycle which is reachable from $\mathbf{y}$, which has a positive cycle. This means that a list with lower bound constraint is less or equal to a list with upper bound constraint.

Figure 5: Case 3



We argue that then there is a solution with $\mathbf{x}$ and $\mathbf{y}$ periodic lists. More precisely, if there is a constraint that enforces one of the variables to be strictly increasing or decreasing, then the system is unsatisfiable.

We first describe the situation where this happens: Constraints that admit as solution only strictly falling lists can have no solutions for which a lower bound constraint exists (except that this bound is zero and thus periodic). The reason for this is that lists with lower bounds are at least periodic. Thus these lists must fall in case 2 and can be set to 0_s preserving satisfiability. Similarly, constraints that admit as solutions only strictly increasing lists are either of the form $\mathbf{x}^{(n)} \geq \mathbf{x}^{n-j} + \mathbf{R}$, with $\mathbf{R}$ having a lower bound (i.e. being at least periodic, which means of constant growth) or $\mathbf{x}^{(n)} \geq \mathbf{R}$, with $\mathbf{R}$ strictly increasing. One observes that at least for one variable in the set $\{\mathbf{v}|\mathbf{v} \text{ variable in } \mathbf{R}\} \cup \{\mathbf{x}\}$ a constraint of the first form must hold, and thus $\mathbf{x}$ is at least linear.

Now we go back to the general situation of case 3 without the strictness assumption. Due to the normal form of the constraint syntax, which allows sums only on the smaller side of the inequality, $\mathbf{y}$ has the constraint (rewritten with the greatest index on the left) $\mathbf{y}^{n+j} \leq \mathbf{y}^{(n)} - \mathbf{R}$ with $\mathbf{R}$ nonnegative. Thus $\mathbf{y}$ is at most periodic. Similarly, $\mathbf{x}$ is at least periodic.

Thus $\mathbf{y}$ can only be greater than $\mathbf{x}$ if $\mathbf{x}$ is bounded from above by a constant a and $\mathbf{y}$ is bounded from below by a constant $b \geq a$. According to the above facts, then there must be a solution with both lists periodic.

Let n' be the maximum of the lengths of the aperiodic initial parts given by case 2 and the arithmetic constraints and k the least common multiple of all index differences that appear. With the index differences of a constraint $\mathbf{x}^{(n)} \geq \mathbf{y_1}^{(l_1)} + \cdots + \mathbf{y_m}^{(l_m)}$ we denote the numbers $|n - y_j|$ for all j. Let d be the least common multiple of k and all period lengths of the lists that have been set to periodic list so far. Further, let h be the number of variables that remain. For all these remaining variables $\mathbf{x}$ we set $\mathbf{x}^{(hn')} = \mathbf{x}^{(hn'+d)}$. Note that this is a large overestimation of aperiodic initial parts (by length $s = hn'$) and period lengths d, but it simplifies the construction. □

Theorem 2. *The satisfiability problem for our list constraint systems in $\mathbb{D}$ is decidable in polynomial time.*

Proof. Given a list constraint system, we assume that there is a solution as in Lemma 2 and replace the thus obtained lists by finitely many variables (namely polynomially many variables in the number of lists). Then we translate the constraints in inequalities between these arithmetic variables (see Example 4). Recall that we may have more than one constraint per variable that we must take all into account. More precisely, for each constraint C of the form

$$\mathbf{x} \geq \mathbf{y}_1 + \cdots + \mathbf{y}_k,$$

where some of the $\mathbf{y}_j$ may be terms $\mathbf{x}^{(i)}$, we have the equivalent formulation

$$\begin{aligned} x_0x_1\dots x_m(x'_1\dots x'_l)^\omega \geq & y_{1,0}y_{1,1}\dots y_{1,m_1}(y'_{1,1}\dots y'_{1,l_1})^\omega + \dots \\ & + y_{k,0}y_{k,1}\dots y_{k,m_k}(y'_{k,1}\dots y'_{k,l_k})^\omega. \end{aligned}$$

For this inequality, we iteratively carry out the following procedure to translate it into arithmetic constraints. We add an arithmetic constraint

$$\begin{aligned} \mathbf{hd}(x_0x_1\dots x_m(x'_1\dots x'_l)^\omega) \geq & \mathbf{hd}(y_{1,0}y_{1,1}\dots y_{1,m_1}(y'_{1,1}\dots y'_{1,l_1})^\omega) + \dots \\ & + \mathbf{hd}(y_{k,0}y_{k,1}\dots y_{k,m_k}(y'_{k,1}\dots y'_{k,l_k})^\omega), \end{aligned}$$

and then drop the first element of all lists and continue with the constraint

$$\begin{aligned} \mathbf{tl}(x_0x_1\dots x_m(x'_1\dots x'_l)^\omega) \geq & \mathbf{tl}(y_{1,0}y_{1,1}\dots y_{1,m_1}(y'_{1,1}\dots y'_{1,l_1})^\omega) + \dots \\ & + \mathbf{tl}(y_{k,0}y_{k,1}\dots y_{k,m_k}(y'_{k,1}\dots y'_{k,l_k})^\omega) \Leftrightarrow \\ x_1\dots x_m(x'_1\dots x'_l)^\omega) \geq & y_{1,1}\dots y_{1,m_1}(y'_{1,1}\dots y'_{1,l_1})^\omega) + \dots \\ & + y_{k,1}\dots y_{k,m_k}(y'_{k,1}\dots y'_{k,l_k})^\omega), \end{aligned}$$

This procedure stops after at most $\max_i(m, m_i + \mathrm{lcm}_j(l, l_j)) + 1$ steps, when no new constraints are derived any more, and yields a finite set of arithmetic constraints. If the resulting purely arithmetic system is satisfiable then clearly the original system is as well.

Conversely, if there is a solution S at all then its modification as in Lemma 2 yields a solution of the arithmetic system. □

Example 4. *Consider the constraints*

$$\mathbf{x}^{(4)} \geq \mathbf{x}^{(2)} + 2\mathbf{x}^{(3)}, \mathbf{x}^{(4)} \geq \mathbf{x}^{(7)}.$$

According to Lemma 1, we can conclude that we have a periodic list of period starting at position 4 and of length at most $6 = \mathrm{lcm}(7-4, 4-2)$, $\mathbf{x}^{(4)} = \mathbf{x}^{(10)}$. *Thus we have*

$$\begin{aligned} \mathbf{x} = x_1x_2x_3x_4(abcdef)^\omega, \mathbf{x}^{(2)} = x_3x_4(abcdef)^\omega, \\ \mathbf{x}^{(3)} = x_4(abcdef)^\omega, \mathbf{x}^{(4)} = (abcdef)^\omega, \end{aligned}$$

with $x_1, x_2, x_3, x_4, a, b, c, d, e, f$ *arithmetic variables. If we translate the list constraints into arithmetic constraints for these variables, we obtain finitely many different inequalities, namely for the first constraint*

$$\begin{aligned} a \geq x_3 + 2x_4, b \geq x_4 + 2a, c \geq a + 2b, d \geq b + 2c, \\ e \geq c + 2d, f \geq d + 2e, a \geq e + 2f, b \geq f + 2a. \end{aligned}$$

and for the second

$$a \geq d, b \geq e, c \geq f, d \geq a, e \geq b, f \geq c.$$

In this example the only solution is a list with x_1, x_2 *arbitrary and the rest zero or infinity. For all nontrivial lists we have a contradiction since the first constraint ensures exponential growth and the second is an upper bound.*

6 Estimation of Growth Rates

So far we have studied the problem of satisfiability. In practical applications it will also be of importance to get information about the growth rate of solutions or, even better, to compute solutions of minimal growth rate. For the general list constraints (those that are hard for SML) the two problems are closely related: In order to know whether $\mathbf{x}$ is less than, say, a given polynomial or exponential function $f(n)$, just add a variable $\mathbf{y}$, the constraint $\mathbf{y} \geq \mathbf{x}$ and constraints ensuring that $\mathbf{y}^{(n)} = \Theta(f(n))$ and check whether the resulting system is still satisfiable. However, such constraints do not fall into the fragment ULC. On the other hand, ULC has the property that in presence of nontrivial upper bounds (i.e. constraints of the form (UB) in section 5), all growth behaviors are bounded by a constant. This also means that if there are such upper bounds and strictly increasing lower bounds on the same variables, the system immediately becomes unsatisfiable (see proof of Theorem 2).

Proposition 1. *Let C be a ULC problem and write C' for C with all upper bounds removed, such that only constraints of the following form remain:*

$$\mathbf{x}^{(n)} \geq \mathbf{x}^{(i_1)} + \cdots + \mathbf{x}^{(i_m)} + \mathbf{R}, \forall l : i_l < n \tag{8}$$

(with $\mathbf{R}$ restricted not to contain $\mathbf{y}^{(j)}$ for which constraints $\mathbf{y}^{(j)} \geq \mathbf{x}^{(k)}, k \geq n$ can be obtained as in Figure 2). If C is satisfiable then the minimal solutions of C and C' coincide.

Now we draw a connection between matrix recurrences and constraint systems. For a system of lower bound constraints we consider the associated recurrence relations, which consists of all constraints with inequality replaced by equality. Homogeneous recurrence relations $(x_n)_n$ of order d, that correspond to constraints with only one list variable $\mathbf{x}$, can be rewritten using their *companion matrix* [23]. This matrix contains the coefficients of the recurrence in its first row, 1 on the lower minor diagonal and the rest zero. It yields the vector $(x_n, \ldots, x_{n-d})$ when applied to the vector $(x_{n-1}, \ldots, x_{n-d-1})$. The rational expansion theorem for linear recurrences [20] ensures that, if the greatest root (w.r.t. the modulus) of the matrix is simple and positive, the sequence becomes monotone with exponential growth rate and is in particular not oscillating between negative and positive values [17, 16]. Now we generalize the companion matrix to constraint systems. The following example shows the idea.

Example 5. *For the constraints*

$$\mathbf{x}^{(n)} \geq \mathbf{x}^{(n-2)} + 2\mathbf{y}^{(n-1)},$$
$$\mathbf{y}^{(n)} \geq 3\mathbf{y}^{(n-1)} + 2\mathbf{x}^{(n-1)}$$

the matrix version is

$$\vec{\mathbf{x}}_n = \begin{pmatrix} \mathbf{hd}(\mathbf{x}^{(n)}) \\ \mathbf{hd}(\mathbf{x}^{(n-1)}) \\ \mathbf{hd}(\mathbf{y}^{(n)}) \end{pmatrix} = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 0 \\ 2 & 0 & 3 \end{pmatrix} \begin{pmatrix} \mathbf{hd}(\mathbf{x}^{(n-1)}) \\ \mathbf{hd}(\mathbf{x}^{(n-2)}) \\ \mathbf{hd}(\mathbf{y}^{(n-1)}) \end{pmatrix} = A\vec{\mathbf{x}}_{n-1} \ .$$

In general, where we have more than one constraint per variable, the associated recurrence for a constraint system contains in addition to + also the max operator:

$$\vec{\mathbf{x}}_n = \max(A_1\vec{\mathbf{x}}_{n-1}, \ldots, A_k\vec{\mathbf{x}}_{n-1}),$$

where $\vec{\mathbf{x}}_n$ is a d-dimensional vector and the A_i are $d \times d$ matrices with nonnegative integers as entries. We first treat the case, where $k = 1$ and where we have no maximum.

6.1 One Constraint per Variable

In case that we have only one constraint per variable, the companion matrix for the system is unique and the following holds.

Let us fix a constraint set with one constraint per variable and also fix a particular variable in that set. We are interested in determining the asymptotically least solution for that variable. Let us write $\mathbb{L}$ for the set of solutions to that variable. Also let A be the companion matrix to the constraint system.

Lemma 3. *There is $d \in \mathbb{N}$ and polynomials $p_i(n)$ and complex numbers λ_i for $i = 1 \dots d$ and a d-dimensional convex set $\mathcal{C}$ specified by finitely many linear inequalities such that*

- *For each $(v_1, \dots, v_d) \in \mathcal{C}$ the list $(x(n))_n$ with $x(n) = \sum_{i=1}^{d} v_i p_i(n) \lambda_i^n$ is in $\mathbb{L}$*
- *For each $(x(n))_n \in \mathbb{L}$ there exists $(v_1, \dots, v_d) \in \mathcal{C}$ such that the list $(x(n))_n$ with $x(n) \geq \sum_{i=1}^{d} v_i p_i(n) \lambda_i^n$.*

Proof. Clearly, every solution is bounded below by a solution to the associated system of linear recurrences obtained by replacing inequality by equality. These solutions can be written in the form $x(n) = e^T A^n v$ where A is the companion matrix, e is a unit vector singling out the variable we're focusing on and v is a start vector which must satisfy the arithmetic constraints. Since the latter form a convex set we can conclude using the standard formula [24] for powers of a Jordan normal form. We also notice that the λ_i are among the eigenvalues of A. □

Lemma 4. *Let $\lambda_1, \dots \lambda_d \in \mathbb{C}$ be pairwise distinct and for all i let $|\lambda_i| = 1$. Let the numbers $v_1, \dots, v_d$ not all be 0. Then there is a constant c, such that*

$$\left| \sum_i v_i \lambda_i^n \right| \geq c \text{ for infinitely many } n$$

Proof. Let $z_n = \sum_i v_i \lambda_i^n$. Not all of the $z_0, z_1, \dots z_{d-1}$ can be zero. This can be seen using the Vandermonde matrix and the formula

$$\begin{pmatrix} z_0 \\ \vdots \\ z_{d-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ \lambda_1 & \lambda_2 & \dots & \lambda_d \\ \lambda_1^2 & \lambda_2^2 & \dots & \lambda_d^2 \\ & \dots & & \\ \lambda_1^{d-1} & \lambda_2^{d-1} & \dots & \lambda_d^{d-1} \end{pmatrix} \begin{pmatrix} v_1 \\ \vdots \\ v_d \end{pmatrix}$$

where the matrix determinant is nonzero given that the λ_i are distinct. Thus let $z_j \neq 0$. If the arguments of all λ_i are rational multiples of 2π, then let N be the least common multiple of the denominators and we have $z_{j+kN} = z_j, k = 0, 1, 2, 3, 4, \dots$. In the other case, $c = |z_j|$ may not be reached again, but since we can approximate irrational arguments with arbitrary precision by rationals, we know that at least $\frac{c}{2}$ will be exceeded infinitely often. □

Lemma 4 then yields the following result.

Theorem 3. *Let $\mathbb{L}$ be the set of solutions for a particular variable in a list constraint system with one constraint per variable. We can effectively find (in polynomial time) $k \in \mathbb{N}$ and $r \geq 0$ and $c > 0$ such that*

- *for all $\mathbf{x} \in \mathbb{L}$*

$$x(n) \geq c n^k r^n \text{ for infinitely many } n.$$

- *there exist a constant* $c' \in \mathbb{R}$ *and* $\mathbf{x} \in \mathbb{L}$ *with*

$$x(n) \leq c' n^k r^n.$$

Thus the asymptotic growth of the minimal solution is $n^k r^n$.

Proof. Write $x_1(n), \ldots, x_d(n)$ for the functions $x_i(n) = p_i(n)\lambda_i^n$ from Lemma 3 and let $\mathcal{C}$ be the convex set of start vectors from that lemma. We assume w.l.o.g. that the x_i are sorted in a way that those x_i which belong to an eigenvalue of greater modulus have a greater index than those x_i with a smaller eigenvalue and also that those x_i with equal eigenvalue are grouped together and those x_i with same eigenvalue are sorted by the degree of their polynomial p_i.

Now we want to find out, whether there are initial values $v \in \mathcal{C}$ such that in the representation of v as a linear combination of the x_i the "greatest" x_i (i.e. those that belong to the greatest eigenvalue λ_{x_i}), do not appear. For instance, if x_d und x_{d-1} belong to the eigenvalue λ with $|\lambda| > \max(\lambda_{x_{d-2}}, \ldots\ \lambda_{x_1})$, then we check whether there is $v \in \mathcal{C}$ with $(0 \ldots 001)v = 0$ and $(0 \ldots 010)v = 0$. We continue this way until we have found $r \in \mathbb{R}$, such that there is necessarily an x_i with corresponding eigenvalue λ in the linear combinations for all $v \in \mathcal{C}$ with $|\lambda| = r$, but none of modulus greater than r.

Let us now consider all x_i with corresponding eigenvalue of modulus r. Those x_i with smaller corresponding eigenvalue can be neglected. Now we try, by an iteration that is similar to the one above, to find $v \in \mathcal{C}$ such that the coefficient of the x_i whose polynomial has maximal degree, is zero. For example, let λ and μ be two eigenvalues of modulus r and x_i functions $ax^m + \mathcal{O}(x^{m-1}\lambda^n)$ (respectively $bx^m + \mathcal{O}(x^{m-1}\lambda^n)$ and $ex^m + \mathcal{O}(x^{m-1}\mu^n)$) and all other x_i with polynomials of degree less than m. Let v_a, v_b, v_e be the components of v for the functions x_i. We then check if there is a $v \in \mathcal{C}$ with $av_a + bv_b = 0$ and $v_e = 0$. If yes, the $\deg(k) < \deg(m)$; if no, it is exactly k, according to Lemma 4.

□

6.2 At least two Constraints for one Variable

If there are several constraints on a variable, we have no longer tight asymptotic bounds but an approximation by a range between two functions, where the minimal solution must lie. There are k constraints on $\mathbf{x}$, namely

$$\begin{aligned}
\mathbf{x}^{(n)} &\geq a_{1,1}\mathbf{x}^{(n-1)} + \cdots + a_{1,m}\mathbf{x}^{(n-m)}, \\
&\cdots \\
\mathbf{x}^{(n)} &\geq a_{k,1}\mathbf{x}^{(n-1)} + \cdots + a_{k,m}\mathbf{x}^{(n-m)},
\end{aligned}$$

where m is the maximum order of the constraints and without loss of generality all indices on the left hand side equal n. Then we could build several companion matrices, one for each of the $m = \prod_{\mathbf{x}} k_{\mathbf{x}}$ combinations of constraints, $A_1, \ldots, A_m$, and calculate in each step of the recurrence the maximal value for $\vec{\mathbf{x}}_n$, such that $\vec{\mathbf{x}}_n \geq \max_i(A_i \vec{\mathbf{x}}_{n-1})$. But it is possible that the matrices where the maximum is taken are always changing, such that we do not obtain a formula that can be efficiently computed. An example are the matrices

$$A_1 = \begin{pmatrix} 2 & 3 \\ 4 & 1 \end{pmatrix}, A_2 = \begin{pmatrix} 3 & 2 \\ 4 & 1 \end{pmatrix}.$$

These matrices belong to a system of two constraints $\mathbf{x}^{(n)} \geq 2\mathbf{x}^{(n-1)} + 3\mathbf{y}^{(n-1)}$ (resp. $\mathbf{x}^{(n)} \geq 3\mathbf{x}^{(n-1)} + 2\mathbf{y}^{(n-1)}$) on the first variable $\mathbf{x}$ and only one constraint $\mathbf{y}^{(n)} \geq 4\mathbf{x}^{(n-1)} + \mathbf{y}^{(n-1)}$ for

the second variable $\mathbf{y}$. We have for all initial vectors v

$$Av > Bv \Rightarrow BAv > AAv \wedge Av < Bv \Rightarrow ABv > BBv.$$

This can be seen by a case distinction. If and only if $v = (v_1, v_2)^t$ and $v_1 > v_2$, $A_1 v$ is less than $A_2 v$. Thus in this step A_2 will be the chosen matrix. After application of A_2, the resulting vector $v' = (v'_1, v'_2)^t$ will have the second component v'_2 greater than the first v'_1 and thus in the next step A_1 will be chosen and so on. If we list the matrices where the maximum is taken in each step, there are sequence which are not finally constant (as in the example), and other examples suggest that we do not know yet if they are even finally periodic.

In case there is in total only one variable (i.e. we have two or more homogeneous recurrences $r_1, \ldots r_n$), we can already prove that if the maximum is obtained for the same matrix A as many times as the degree d_1 of the recurrence r_1 for A is, then it will always be taken for A. Assume for any B

$$A^i \vec{\mathbf{x}}_n \geq BA^{i-1}\vec{\mathbf{x}}_n, i = 1, ..., d_1 \ .$$

Consider the characteristic polynomial of $x^{d_1} - \sum_{i=0}^{d_1-1} a_i x^i$, of which A is a zero by Cayley-Hamilton, and calculate

$$A^{d_1+1}\vec{\mathbf{x}}_n = \sum_{i=1}^{d_1} a_{i-1} A^i \vec{\mathbf{x}}_n \geq \sum_{i=1}^{d_1} a_{i-1} BA^{i-1}\vec{\mathbf{x}}_n = B \sum_{i=0}^{d_1-1} a_i A^i \vec{\mathbf{x}}_n = BA^{d_1}\vec{\mathbf{x}}_n \ .$$

The above considerations lead us from linear list constraints to the field of (possibly inhomogeneous) positive recurrences with maximums. We give the following heuristic to approximate the behavior of optimal solutions. Replacing the multiple constraints on $\mathbf{x}$ by the new constraint

$$\mathbf{x}^{(n)} \geq \max(a_{1,1}, \ldots a_{k,1})\mathbf{x}^{(n-1)} + \cdots + \max(a_{1,m}, \ldots, a_{k,m})\mathbf{x}^{(n-m)}, \tag{9}$$

delivers an upper bound on the optimal solution. Note that this is only necessary, if for each two constraints there are indices i, i', j, k such that $a_{i,j} < a_{i',j}$ and $a_{i,k} > a_{i',k}$. If there are other variables than $\mathbf{x}$, we take analogously the maximal coefficients of the additional terms. This allows us to find a nontrivial upper bound on the systems growth.

7 Applications to Resource Analysis

In previous work, Hofmann and Jost gave a method for analyzing the heap space consumption of functional programs.

In functional programs it is not possible to build circular data structures, whereas in object oriented language there may be arbitrary and unpredictable chains of data structures where one is an attribute of the other (such as a list denoted by tail(x) is an attribute of a list x).

Thus Rodriguez and Hofmann needed to take into account also infinite data structures for their linear list constraints when considering the object oriented language RAJA. In the general setting, objects are represented as infinite trees, where the objects themselves are nodes and the degree of a node is the number of attributes the object has and its children are exactly these attributes, which may again be objects. Now a program is correctly typeable (with resource annotated types) if and only if the linear constraints derived by the type inference algorithm are satisfiable. A heuristic procedure was developed ([10]) which allowed to find solutions in many cases but the general question of decidability of these infinitary constraint systems remained

open. It is precisely this question that we have solved in section 4, enabling us to analyse the entire class of list programs.

To put the problem in perspective we consider a piece of code in the Java-like language RAJA (as defined in [10]), which implements the sieve of Eratosthenes and which can not be analyzed with the methods presented in [10]. Here the classes `List`, `Cons` and `Nil` are defined as usual, such that `List` is a superclass of `Nil` (the empty list) and `Cons` (a nonempty list) and the methods `eratos` and `filter` of the class `List` are defined in the two subclasses.

```
class Cons extends List
{
  int elem; List next;
  //RAJA syntax: let _ = a <- b is the same as a = b
  List filter(int a)
  {
    let l = new Cons in
    let _ = this.elem <- this.elem - (a * this.elem / a) in
    if (this.elem == 0) then
      let _ = l.next<-this.next.filter(a) in
      return l.next
    else
      let _ = l.elem <- this.elem in
      let _ = l.next<-this.next.filter(a) in
      return l;
  }
  List eratos()
  {
    let l = new Cons in
    let _ = l.next<-this.next.filter(this.elem).eratos() in
    return l;
  }
}
```

In [6], this program was implemented in the functional language RAML, and analyzed with the result that its resource consumption is quadratic. In our case, the existing RAJA tool cannot handle the appearing constraints because they do not admit regular solutions over $\mathbb{R}_0^+$, i.e. solutions representing linear potential and hence space usage.

Now we illustrate how the constraint generation works. This is explained more precisely in [1]. Since the constraints generated by the RAJA type inference algorithm for this program are hundreds in number and the analysis is very involved, we decided to pick simpler (schematic) program fragments to explain the constraint generation. Although they may seem artificial, there are imaginable cases, where one wants to write such code. However, the aim of these programs is to be as simple as possible and to generate constraints of a certain format. We start with an example that has a cascade of recursive calls, which results in constraints with solution an exponentially increasing list and exponential heap space consumption (in the size of the input list):

```
class Cons extends List
{
  List next;
  int elem;
```

```
  List m(int i)
  {
    let res = new Cons in
    let res'= new Cons in
    // ... some calculation ...
    let _ = res.next <- this.next.m(i) in
    let _ = res'.next <- this.next.m(i+1) in
    return res;
  }
}
```

The resulting constraints include

$$\mathbf{x}.\mathbf{next} \geq \mathbf{x} + \mathbf{x}.$$

where $\mathbf{x}$ ranges over infinite lists of nonnegative rational numbers (or infinity) and $\mathbf{x}.\mathbf{next}$ denotes the tail of $\mathbf{x}$. Addition $(+)$ is understood pointwise. Constraints of that form do not admit regular solutions (except the list consisting only of infinity), but can be treated by the method we have described.

We now explain the meaning of the above constraint. The list $\mathbf{x} = x_0, x_1, x_2, \ldots$ models the potential assigned to the argument, i.e., `this`: x_0 "dollars" for `this` itself, x_1 dollars for `this.next` (if present), etc.. The constraint then models the sharing of potential of `this.next` between its two occurrences $\texttt{this.next}_i, i = 1, 2$ in the let-expressions, which means that their potential sums to the potential of `this.next` itself. Since we invoke the method `m` of `this` recursively with $\texttt{this.next}_i$, they must be subtypes of the type of `this`. This means that they have at least the same potential as `this`.

One can modify this program by replacing one of the recursive calls by a call to another method of the `this` object. This would result in a constraint of the form

$$\mathbf{x}.\mathbf{next} \geq \mathbf{x} + \mathbf{y},$$

where the potential requirements of the other method are captured in $\mathbf{y}$. If the method requires nontrivial (e.g. polynomial) potential, we obtain a $\mathbf{x}$ list with polynomially growing entries, where the degree has increased by one.

Our new algorithm allows us to treat a wider class of programs than was the case previously. We can decide satisfiability (i.e. the possibility of correct typing refined with potential) and in many cases we are able to detect minimal solutions with respect to the heap space consumption. In these cases we can guarantee that the program can execute successfully with a certain amount of memory as a function of it input size. Unlike in [10] this function does not have to be linear.

8 Summary and Future Work

We have studied a constraint satisfaction problem arising from the amortized resource analysis for object-oriented programs but which due to its simple and natural formulation may be of interest in its own right.

We show that this problem is computationally hard in general but identified a natural fragment which still covers the entire range of the resource analysis problems for list programs and for which we demonstrate feasibility in the sense that both satisfiability and estimation of growth rates can be decided in polynomial time. This special format encompasses systems

whose minimal solutions exhibit polynomial or exponential growth, a case that was not within the scope of the heuristic method proposed earlier [10]. Our method uses a novel combination of ideas from constraint satisfaction, linear programming, and matrix theory.

Our results can be used to extend the existing amortized analysis, in particular such that type inference becomes decidable for all kinds of list programs in RAJA and such that the tool can derive resource bounds from the minimal solutions.

We now discuss what happens if we consider $\mathbb{Q}_0^+$ instead of $\mathbb{D}$. Regarding satisfiability, it remains true that once we have found periodic constraints we can replace them by finitely many arithmetic variables as we did and solve the resulting constraints with linear programming. Variables with lower bounds can not be instantiated with constant lists consisting of infinity, but a solution can be found by reading them as recurrence relations, cf. section 6. This section works without changes. One reviewer asked what happens if variables range over (nonnegative) integers or reals. The use of real numbers is not motivated by the application, so we have not considered it, but due to the linear nature of the constraints we expect no changes at all. Regarding integers, the polynomial time result no longer holds, as integer linear programming is NP-complete. Beyond that, we expect no further complications. For the amortized approach to resource analysis this kind of integrality constraints do not arise.

We remark that as a consequence of the decision procedure in case of unsatisfiability the contradiction can be proved with a finite number of **hd** and **tl** applications and translation of the resulting constraints into arithmetic variables. This defines a complete proof procedure. The number of steps is not known a priori, but it is known after carrying out the decision procedure and thus can be used as a certificate for unsatisfiability in the style of Farkas' lemma.

The natural next step will be the generalization of our method to decide satisfiability of more general constraint systems involving $\mathbb{D}$-labeled trees; our current investigations based on results of [25, 26] suggest that this problem cannot be reduced to the case of lists. Further, it involves complicated word combinatorics because of the different branchings. But with our algorithm we can already now treat programs which involve trees with only one nonzero infinite path or with all nodes on the same level equal. These special cases can directly be reduced to the list case.

On a more practical side, we plan to implement our algorithms and to integrate them into the existing RAJA tool.

References

[1] Martin Hofmann and Dulma Rodriguez. Linear constraints over infinite trees. In *Proceedings of the 18th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, LPAR'12, pages 343–358, Berlin, Heidelberg, 2012. Springer-Verlag.

[2] Martin Hofmann and Steffen Jost. Static prediction of heap space usage for first-order functional programs. In *Proceedings of the 30th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '03, pages 185–197, New York, NY, USA, 2003. ACM.

[3] Martin Hofmann and Dulma Rodriguez. *Automatic Type Inference for Amortised Heap-Space Analysis*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[4] Jan Hoffmann and Martin Hofmann. Amortized resource analysis with polymorphic recursion and partial big-step operational semantics. In *APLAS*, pages 172–187, 2010.

[5] Jan Hoffmann and Martin Hofmann. *Amortized Resource Analysis with Polynomial Potential*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[6] Jan Hoffmann, Klaus Aehlig, and Martin Hofmann. Multivariate amortized resource analysis. *ACM Trans. Program. Lang. Syst.*, 34(3):14:1–14:62, November 2012.

[7] Jan Hoffmann. *Types with Potential: Polynomial Resource Bounds via Automatic Amortized Analysis*. PhD thesis, University of Munich, 2011.

[8] Martin Hofmann and Dulma Rodriguez. Automatic type inference for amortised heap-space analysis. In *ESOP: 22nd European Symposium on Programming*, 2013.

[9] Martin Hofmann and Dulma Rodriguez. Efficient type-checking for amortised heap-space analysis. In *CSL: 18th EACSL Annual Conference on Computer Science Logic*. LNCS, Springer-Verlag, 2009.

[10] Dulma Rodriguez. *Amortized Analysis for Object Oriented Programs*. PhD thesis, University of Munich, 2012.

[11] Peter Habermehl, Radu Iosif, and Tomáš Vojnar. *What Else Is Decidable about Integer Arrays?* Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[12] Marius Bozga, Radu Iosif, and Filip Konečný. *Fast Acceleration of Ultimately Periodic Relations*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[13] Stefan Dantchev and Frank D Valencia. On infinite csps. *Modelling and Reformulating Constraint Satisfaction Problems*.

[14] Pieter Moree. *Recurrence Sequences (Mathematical Surveys and Monographs 104) By Graham Everest, Alf van der Poorten, Igor Shparlinksi and Thomas Ward: 318 pp., ISBN 0-8218-3387-1 (American Mathematical Society, Providence, RI, 2003)* -, volume 36. Cambridge University Press, Cambridge, UK, 05 2004.

[15] Herbert S. Wilf. In *Generatingfunctionology*. Academic Press, 1990.

[16] Joël Ouaknine and James Worrell. Positivity problems for low-order linear recurrence sequences. *CoRR*, abs/1307.2779, 2013.

[17] Vincent D. Blondel and Natacha Portier. The presence of a zero in an integer linear recurrent sequence is NP-hard to decide. *Linear Algebra and its Applications*, 351:91–98, 2002.

[18] Vesa Halava, Tero Harju, Mika Hirvensalo, and Juhani Karhumäki. Skolem's problem - on the border between decidability and undecidability. Technical report, 2005.

[19] Joël Ouaknine and James Worrell. *Decision Problems for Linear Recurrence Sequences*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[20] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 1994.

[21] Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1986.

[22] N. J. Fine and H. S. Wilf. Uniqueness theorems for periodic functions. *Proceedings of the American Mathematical Society*, 16(1):109–114, 1965.

[23] Amer Abu-Omar and Fuad Kittaneh. Estimates for the numerical radius and the spectral radius of the frobenius companion matrix and bounds for the zeros of polynomials. *Ann. Funct. Anal.*, 5(1):56–62, 2014.

[24] Christopher S. Withers and Saralees Nadarajah. The nth power of a matrix and approximations for large n. Technical report, 2008.

[25] Christian Choffrut and Juhani Karhumäki. *Combinatorics of Words*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997.

[26] Pál Dömösi and Masami Ito. *Context-free languages and primitive words*. World Scientific, 2014.