On Deciding Constant Runtime of Linear Loops

Florian Frohn[1], Jürgen Giesl[1], Peter Giesl[2], and Nils Lommen[1]

[1] RWTH Aachen University, Aachen, Germany
[2] Department of Mathematics, University of Sussex, Brighton, UK

Abstract. We consider linear single-path loops of the form

$$\textbf{while} \quad \varphi \quad \textbf{do} \quad \vec{x} \leftarrow A\vec{x} + \vec{b} \quad \textbf{end}$$

where $\vec{x}$ is a vector of variables, the loop guard φ is a conjunction of linear inequations over the variables $\vec{x}$, and the update of the loop is represented by the matrix A and the vector $\vec{b}$. It is already known that termination of such loops is decidable. In this work, we consider loops where A has real eigenvalues, and prove that it is decidable whether the loop's runtime (for all inputs) is bounded by a constant if the variables range over $\mathbb{R}$ or $\mathbb{Q}$. This is an important problem in automatic program verification, since safety of linear **while**-programs is decidable if all loops have constant runtime, and it is closely connected to the existence of multiphase-linear ranking functions, which are often used for termination and complexity analysis. To evaluate its practical applicability, we also present an implementation of our decision procedure.

1 Introduction

We investigate decidability of the question whether a linear[3] single-path loop has constant runtime (i.e., whether the runtime of the loop can be bounded by the same constant for all inputs).

Example 1 (Leading Example). We consider linear single-path loops like

$$\textbf{while} \quad 0 \leq x + y \leq 10 \quad \textbf{do} \quad \underbrace{\begin{pmatrix} x \\ y \end{pmatrix}}_{\vec{x}} \leftarrow \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}}_{A} \begin{pmatrix} x \\ y \end{pmatrix} + \underbrace{\begin{pmatrix} 1 \\ 0 \end{pmatrix}}_{\vec{b}} \quad \textbf{end}$$

with linear arithmetic only, conjunctive loop guards, and without branching in the loop body. In our example, the loop body is evaluated if $0 \leq x + y \leq 10$. The value of x is increased by 1 and y is doubled in every iteration. Our goal is to decide whether the runtime of such loops is bounded by a constant. Later we will show that the loop in our example has indeed constant runtime.

Throughout this paper, the coefficients that occur in the loop are always rational numbers.[4] Regarding the domain of the variables $\vec{x}$, we focus on loops over $\mathbb{R}$, but our results also apply to the case that the variables range over $\mathbb{Q}$.

[3] In this paper, the term "linear" refers to linear *polynomials*, i.e., functions of the form $c_0 + \sum_{i=1}^{d} c_i \cdot x_i$ (in contrast to linear *transformations* of the form $\sum_{i=1}^{d} c_i \cdot x_i$).

[4] Our approach also works if the coefficients are algebraic real numbers and the variables range over $\mathbb{R}$. We restrict ourselves to rational coefficients for simplicity.

The motivation for our work is three-fold. First, it is obviously relevant in the context of automated complexity analysis, in order to prove constant runtime. Second, the question whether a linear loop admits a *multiphase-linear ranking function* (MΦRF) can be reduced to the question whether another linear loop has constant runtime w.r.t. a given set of initial states [5, Sect. 4.2]. Thus, our work is an important step towards a solution for the long-standing open problem whether the existence of MΦRFs is decidable for linear loops. MΦRFs are relevant for both automated termination and complexity analysis, since the existence of a MΦRF does not only prove termination, but it also proves that the runtime of the loop is at most linear [4, Thm. 7]. However, our approach does not consider initial conditions, so it cannot be used for the inference of MΦRFs yet. We intend to refine it to support initial conditions in future work. Third, our work is of interest for all infinite-state verification techniques that use approximations to deal with loops, like abstract interpretation, symbolic execution, recurrence analysis, loop summarization, etc. All of these techniques make use of approximations in the presence of loops to prevent divergence of the verification process, but at the price of losing precision. Instead, a loop with constant runtime c can simply be unrolled c times. Thus, reachability (and hence, safety) is decidable for all those linear **while**-programs where all loops have constant runtime.

The constant runtime problem turns out to be surprisingly challenging: Termination of linear loops is decidable over the reals [28], the rationals [7], and the integers [17]. But even for classes of loops where decidability of termination has been known for decades [28], decidability of the constant runtime problem is far from obvious. Intuitively, the reason is that all decidability results for termination of linear loops exploit *eventual monotonicity*: For loops with non-negative real eigenvalues, the variable values start to behave monotonically after finitely many loop iterations (where the loop guard is ignored), and negative eigenvalues can easily be eliminated. For the constant runtime problem, this kind of reasoning is not suitable, as one cannot disregard finitely many loop iterations.

Contributions: After introducing our notion of loops formally in Sect. 2, in Sect. 3 we prove decidability of the constant runtime problem for linear loops over $\mathbb{R}$ with real eigenvalues. Our decision procedure encodes the constant runtime problem as a validity problem, and then exploits the restriction to real eigenvalues to remove one quantifier alternation, so that the resulting formula is amenable to Fourier-Motzkin variable elimination. Our results immediately extend to loops over $\mathbb{Q}$, but not over $\mathbb{Z}$, due to the use of Fourier-Motzkin variable elimination. For loops over $\mathbb{Z}$, we show decidability of the constant runtime problem for linear loops with eigenvalues from $\{-1, 0, 1\}$ in Sect. 4. In Sect. 5, we discuss related work, and in Sect. 6 we evaluate our implementation and conclude.

2 Preliminaries

We consider loops of the form

$$\textbf{while} \quad \varphi \quad \textbf{do} \quad \vec{x} \leftarrow A\vec{x} + \vec{b} \quad \textbf{end} \qquad \text{(Loop)}$$

where $\vec{x}$ is a vector of $d \geq 1$ variables, φ is a conjunction of linear inequations over $\vec{x}$ of the form $t \sim 0$ with $\sim \in \{>, \geq\}$, $A \in \mathbb{Q}^{d \times d}$, and $\vec{b} \in \mathbb{Q}^d$. We sometimes regard a conjunction of formulas $\varphi \equiv \varphi_1 \wedge \ldots \wedge \varphi_m$ as a set $\{\varphi_1, \ldots, \varphi_m\}$ such that we can write $\varphi_i \in \varphi$ for every conjunct φ_i. For any arithmetic expression t containing variables from $\vec{x}$, we define the *update function* up of (Loop) as $\mathrm{up}(t) := t[\vec{x}/A\vec{x} + \vec{b}]$, where $[\vec{x}/A\vec{x} + \vec{b}]$ denotes the substitution which replaces each variable from $\vec{x}$ by the corresponding polynomial from $A\vec{x} + \vec{b}$.

The loop (Loop) has *constant runtime* if the following holds:

$$\exists c \in \mathbb{N}.\ \forall \vec{x} \in \mathbb{R}^d.\ \exists i \in \mathbb{N}_{\leq c}.\ \neg\varphi[\vec{x}/\mathrm{up}^i(\vec{x})] \qquad \text{(Const)}$$

Here, up^i denotes the i-fold application of up and $\mathbb{N}_{\leq c} = \{n \in \mathbb{N} \mid n \leq c\}$ for any $c \in \mathbb{N}$. Intuitively, (Const) states that $c \in \mathbb{N}$ is a bound on the runtime of (Loop) if, for every possible initial value $\vec{x} \in \mathbb{R}^d$, the loop guard is violated after $i \leq c$ iterations, i.e., $\neg\varphi[\vec{x}/\mathrm{up}^i(\vec{x})]$ holds.

As in [10], we can restrict ourselves to *non-negative loops* (where A does not have negative eigenvalues) without loss of generality. The reason is that (Loop) has constant runtime if and only if

$$\textbf{while} \quad \varphi \wedge \varphi[\vec{x}/A\vec{x} + \vec{b}] \quad \textbf{do} \quad \vec{x} \leftarrow A(A\vec{x} + \vec{b}) + \vec{b} \quad \textbf{end}$$

or, equivalently,

$$\textbf{while} \quad \varphi \wedge \varphi[\vec{x}/A\vec{x} + \vec{b}] \quad \textbf{do} \quad \vec{x} \leftarrow A^2\vec{x} + A\vec{b} + \vec{b} \quad \textbf{end} \qquad \text{(Non-Neg)}$$

has constant runtime. Here, the loop (Non-Neg) is obtained by *chaining* the loop (Loop), i.e., each iteration of (Non-Neg) corresponds to two subsequent iterations of (Loop). Clearly, if A only has real eigenvalues, then all eigenvalues of A^2 are non-negative. Now, Cor. 2 is a direct consequence of [14, Lemma 18].

Corollary 2 (Transformation to Non-Negative Loops). (Loop) *has constant runtime iff* (Non-Neg) *has constant runtime.*

3 Loops with Real Eigenvalues

From now on, we only consider loops with real eigenvalues. Then we can restrict ourselves to loops like (Loop) where all eigenvalues of A are non-negative without loss of generality due to Cor. 2. The advantage of loops like (Loop) is that they admit suitable closed forms capturing the behavior of n loop iterations by a single expression for each variable. These closed forms are heavily used in complete techniques for termination analysis, invariant generation, and complexity analysis (see, e.g., [7, 10, 14, 15, 17–19, 25, 28]).

The remainder of this section is structured as follows: First, we introduce closed forms for linear loops with real eigenvalues in Sect. 3.1. Then, we use these closed forms to reduce the constant runtime problem to a validity problem in Sect. 3.2, i.e., from that point onward, our goal is to decide validity of a first-order formula (Const_{n_0}) with a specific structure. The arithmetic expressions

in this formula contain the variables $\vec{x}$ from the loop under consideration, and an additional, specific variable that serves as a "loop counter". Next, in Sect. 3.3 we show that the number of real roots of these arithmetic expressions w.r.t. the loop counter can be bounded by a constant that does not depend on the values of $\vec{x}$. In Sect. 3.4, we exploit this fact to eliminate the innermost quantifier from ($\textsc{Const}_{n_0}$). For this step, it is crucial that all eigenvalues are non-negative real numbers. Without the innermost quantifier, the resulting formula is amenable to a variant of Fourier-Motzkin variable elimination that allows us to eliminate the variables $\vec{x}$, see Sect. 3.5 and 3.6. In this way, we obtain a univariate formula that contains both polynomial and exponential arithmetic. Finally, we show how to decide validity of such formulas in Sect. 3.7.

3.1 Closed Forms

Let $\mathbb{R}_{\mathbb{A}}$ denote the algebraic reals (i.e., $\mathbb{R}_{\mathbb{A}}$ contains every real number that is a root of a univariate, non-zero polynomial with integer coefficients). Moreover, $\mathbb{R}_{\mathbb{A}>0}$ denotes the positive algebraic reals. Similar to, e.g., [18, Thm. 3.1] and [15, Def. 3.4], for loops with non-negative real eigenvalues one can compute *poly-exponential* closed forms for the n-fold update of the loop. They have the form

$$\sum_{i=1}^{M} a_i(\vec{x}) \cdot b_i^n \cdot n^{e_i} \tag{PE}$$

where $b_i \in \mathbb{R}_{\mathbb{A}>0}$, $e_i \in \mathbb{N}$, and $a_i \in \mathbb{R}_{\mathbb{A}}[\vec{x}]$ is a linear polynomial over the variables $\vec{x}$. Here, we identify a_i and the function $\vec{x} \mapsto a_i$ where we sometimes write $a_i(\vec{x})$ to make the variables $\vec{x}$ explicit, and we use the same convention for other (vectors of) expressions. Note that the b_i are the eigenvalues of the update matrix A. We highlight that the constants in (PE) are algebraic, as it is unclear how to compute with non-algebraic numbers, and thus closed forms involving such numbers would not be suitable for our decision procedure.

To compute such a closed form, the idea is to consider the Jordan normal form $J = \mathrm{diag}(B_1, \ldots, B_N)$ of the update matrix A, where $A = S \cdot J \cdot S^{-1}$ for a suitable change-of-basis matrix $S \in \mathbb{R}_{\mathbb{A}}^{d \times d}$. Here, each Jordan block B is associated to an eigenvalue $b \in \mathbb{R}_{\mathbb{A}\geq 0}$. Note that the Jordan normal form (and all relevant computations regarding algebraic numbers) can always be computed in polynomial time [8]. Then the n-fold update is represented by the matrix $A^n = (S \cdot J \cdot S^{-1})^n = S \cdot J^n \cdot S^{-1}$ (i.e., S and S^{-1} always cancel out). Thus, one only has to compute a closed form for each Jordan block of $J^n = \mathrm{diag}(B_1^n, \ldots, B_N^n)$. For every Jordan block $B \in \mathbb{R}_{\mathbb{A}\geq 0}^{d_B \times d_B}$, this closed form is

$$B^n = \begin{pmatrix} b^n & \binom{n}{1} \cdot b^{n-1} & \cdots & \binom{n}{d_B-1} \cdot b^{n-(d_B-1)} \\ 0 & b^n & \cdots & \binom{n}{d_B-2} \cdot b^{n-(d_B-2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & b^n \end{pmatrix} \qquad \text{for all } n \in \mathbb{N}$$

if $b \neq 0$, or $B^n = 0 \in \mathbb{R}_{\mathbb{A}\geq 0}^{d_B \times d_B}$ for all $n \geq d_B$ if $b = 0$. Note that the binomial coefficients yield the polynomial factors n^{e_i} of the poly-exponential expression (PE). Furthermore, in combination with the change-of-basis matrix $S \in \mathbb{R}_{\mathbb{A}}^{d \times d}$, the coefficients in the polynomials $a_i(\vec{x})$ are determined.

Hence, for any loop of the form (Loop) with non-negative real eigenvalues only, there is a vector $\vec{cl}$ of d poly-exponential expressions of the form (PE) and a constant $n_0 \in \mathbb{N}$ such that

$$\mathrm{up}^n(\vec{x}) = \vec{cl} \qquad \text{for all } n \in \mathbb{N} \text{ with } n \geq n_0.$$

More precisely, n_0 is the dimension d_B of the largest Jordan block B which is associated to the eigenvalue $b = 0$.

Example 3 (Closed Forms, Ex. 1 cont.). For Ex. 1, we have the closed form

$$\vec{cl} = \begin{pmatrix} x+n \\ 2^n y \end{pmatrix}.$$

To see why the constant n_0 is needed for closed forms, consider the loop

$$\textbf{while} \quad x \leq 0 \quad \textbf{do} \quad \begin{pmatrix} x \\ y \end{pmatrix} \leftarrow \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \textbf{end}$$

Here, we have $n_0 = 2$, as the update matrix is already in Jordan normal form, its only eigenvalue is 0, and it has a single Jordan block of size 2. Then we obtain the closed form $\vec{cl} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$. However, for $n \in \{0, 1\}$ we have

$$\vec{cl}[n/0] = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \neq \begin{pmatrix} x \\ y \end{pmatrix} = \mathrm{up}^0(\vec{x}) \quad \text{and} \quad \vec{cl}[n/1] = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \neq \begin{pmatrix} y \\ 1 \end{pmatrix} = \mathrm{up}^1(\vec{x}).$$

3.2 From Constant Runtime to Validity

We now reduce the constant runtime problem to a validity problem. As a first step, recall the characterization of constant loops:

$$\exists c \in \mathbb{N}.\ \forall \vec{x} \in \mathbb{R}^d.\ \exists i \in \mathbb{N}_{\leq c}.\ \neg\varphi[\vec{x}/\mathrm{up}^i(\vec{x})] \tag{Const}$$

By "shifting" c by one, we obtain the equivalent formula

$$\exists c' \in \mathbb{N}.\ \forall \vec{x} \in \mathbb{R}^d.\ \exists i \in \mathbb{N}_{<c'}.\ \neg\varphi[\vec{x}/\mathrm{up}^i(\vec{x})] \tag{Const$_<$}$$

which is more suitable for our further analysis (in particular in Sect. 3.3). Thus, if (Const$_<$) holds, then $c' - 1$ is a bound on the runtime of (Loop). We now show that (Const$_<$) is equivalent to

$$\exists c \in \mathbb{N}.\ \forall \vec{x} \in \mathbb{R}^d.\ \exists i \in \mathbb{N}_{<c}.\ \neg\varphi[\vec{x}/\mathrm{up}^{n_0+i}(\vec{x})]$$

where n_0 is defined as in Sect. 3.1 This allows us to directly replace $\mathrm{up}^{n_0+i}(\vec{x})$ by its closed form, since $i \geq 0$. To this end, we define the *instantiated loop guard*

$$\psi := \varphi[\vec{x}/\vec{cl}].$$

So we show that (Const$_<$) is equivalent to

$$\exists c \in \mathbb{N}.\ \forall \vec{x} \in \mathbb{R}^d.\ \exists i \in \mathbb{N}_{<c}.\ \neg\psi[n/n_0 + i], \tag{Const$_{n_0}$}$$

and if (Const$_{n_0}$) holds, then $n_0 + c - 1$ is a bound on the runtime of (Loop).

Proving Equivalence of ($\textsc{Const}_<$) and ($\textsc{Const}_{n_0}$): To show that ($\textsc{Const}_<$) and ($\textsc{Const}_{n_0}$) are equivalent, we first prove that ($\textsc{Const}_{n_0}$) implies ($\textsc{Const}_<$). To this end, assume that ($\textsc{Const}_{n_0}$) holds, but ($\textsc{Const}_<$) does not. Hence, there exists a $c \in \mathbb{N}$ such that

$$\forall \vec{x} \in \mathbb{R}^d.\ \exists i \in \mathbb{N}_{<c}.\ \neg\psi[n/n_0 + i]. \tag{$\textsc{Const}^{in}_{n_0}$}$$

As ($\textsc{Const}_<$) does not hold, for $c' = n_0 + c$ there is an $\vec{x} \in \mathbb{R}^d$ such that

$$\forall i \in \mathbb{N}_{<n_0+c}.\ \varphi[\vec{x}/\mathrm{up}^i(\vec{x})],$$

which contradicts ($\textsc{Const}^{in}_{n_0}$), as for every $i \geq 0$ we have $\neg\psi[n/n_0 + i] \equiv \neg\varphi[\vec{x}/\mathrm{up}^{n_0+i}(\vec{x})]$.

To see why ($\textsc{Const}_<$) implies ($\textsc{Const}_{n_0}$), assume that ($\textsc{Const}_<$) holds, but ($\textsc{Const}_{n_0}$) does not. Hence, there exists a $c' \in \mathbb{N}$ such that

$$\forall \vec{y} \in \mathbb{R}^d.\ \exists i \in \mathbb{N}_{<c'}.\ \neg\varphi[\vec{x}/\mathrm{up}^i(\vec{y})]. \tag{$\textsc{Const}^{in}_{<}$}$$

As ($\textsc{Const}_{n_0}$) does not hold, for $c = c'$ there exists an $\vec{x} \in \mathbb{R}^d$ such that

$$\forall i \in \mathbb{N}_{<c'}.\ \psi[n/n_0 + i]. \tag{$\dagger$}$$

Now we set $\vec{y} = \mathrm{up}^{n_0}(\vec{x})$ in ($\textsc{Const}^{in}_{<}$), i.e., there exists an $i \in \mathbb{N}_{<c'}$ such that $\neg\varphi[\vec{x}/\mathrm{up}^i(\vec{y})]$, or equivalently, $\neg\varphi[\vec{x}/\mathrm{up}^{n_0+i}(\vec{x})]$ holds. This is a contradiction to ($\dagger$), as for every $i \geq 0$ we again have $\psi[n/n_0 + i] \equiv \varphi[\vec{x}/\mathrm{up}^{n_0+i}(\vec{x})]$.

While ($\textsc{Const}_<$) expresses that the loop guard must be violated within the first c' iterations, ($\textsc{Const}_{n_0}$) "ignores" the first n_0 iterations. This allows us to use closed forms which are only valid for $n \geq n_0$.

Example 4 (Constant Runtime to Validity, Ex. 3 cont.). For Ex. 1, it remains to decide validity of

$$\exists c \in \mathbb{N}.\ \forall x, y \in \mathbb{R}.\ \exists i \in \mathbb{N}_{<c}.\ \neg\left(0 \leq x + i + 2^i y \leq 10\right). \tag{Val}$$

Moreover, if (Val) holds, then $c - 1$ is a bound on the runtime of Ex. 1.

3.3 Bounded Number of Real Roots

Recall that ($\textsc{Const}_{n_0}$) states that there is a $c \in \mathbb{N}$ such that for all initial values $\vec{x}$, there is an iteration n with $n_0 \leq n < n_0 + c$ where the loop guard does not hold. So here, we have to consider all instantiations of n with natural numbers from n_0 up to $n_0 + c - 1$. For deciding validity of ($\textsc{Const}_{n_0}$), the problem is that c is not yet known. Our aim is to replace ($\textsc{Const}_{n_0}$) by

$$\exists m \in \mathbb{N}.\ \forall \vec{x} \in \mathbb{R}^d.\ \bigvee_{j=0}^{rb} \neg\psi[n/n_0 + j \cdot m], \tag{QE}$$

where we only check $rb + 1$ instantiations of n. Here, rb is a number that only depends on ψ, but not on the actual instantiation of $\vec{x}$. Thus, our goal is to replace

the existentially quantified formula $\exists i \in \mathbb{N}_{<c}.\ \neg\psi[n/n_0 + i]$ which depends on the (unknown) value c by the disjunction $\bigvee_{j=0}^{rb} \neg\psi[n/n_0 + j \cdot m]$, where rb is known and fixed. So to eliminate the inner existential quantifier in ($\textsc{Const}_{n_0}$), we show that it suffices to consider a fixed number of "sample points" instead of all values $i \in \mathbb{N}_{<c}$. Here, the (existentially quantified) variable m is used as the distance between our "sample points".

To determine the number rb, n is instantiated by real instead of just natural numbers. Then, the number rb is chosen in such a way that for every vector $\vec{v}$ of real numbers, the expressions in the inequations of the formula $\psi[\vec{x}/\vec{v}]$ (which only contains the variable n) have at most rb real roots. Note that the number of real roots is unbounded without the restriction to non-negative real eigenvalues.

To define rb, for any poly-exponential expression t as in (PE), we first define its $\mathsf{rootbound}$. Here, $\mathsf{rootbound}(t)$ only depends on the number M of addends and the exponents e_i, but $\mathsf{rootbound}(t)$ does not depend on $\vec{x}$. The following lemma shows that for every value $\vec{v} \in \mathbb{R}^d$, $\mathsf{rootbound}(t)$ is an upper bound on the number of real roots of the expression $t[\vec{x}/\vec{v}]$.

Lemma 5 (Bounded Number of Real Roots). *Let $\vec{v} \in \mathbb{R}^d$ and let*

$$t = \sum_{i=1}^{M} a_i(\vec{x}) \cdot b_i^n \cdot n^{e_i}$$

be a poly-exponential expression such that $t[\vec{x}/\vec{v}]$ is not the polynomial 0. Then

$$\mathsf{rootbound}(t) := M - 1 + \sum_{i=1}^{M} e_i$$

is an upper bound on the number of real roots of $t[\vec{x}/\vec{v}]$.

Proof. As in [26], for any expression s of the form

$$\sum_{i=1}^{M'} p_i(n) \cdot b_i^n, \tag{1}$$

where each b_i is unique and each $p_i(n)$ is a polynomial over n, we define

$$ord(s) := \sum_{i=1}^{M'} (\mathsf{degree}(p_i(n)) + 1) = M' + \sum_{i=1}^{M'} \mathsf{degree}(p_i(n)). \tag{2}$$

Let $\vec{v} \in \mathbb{R}^d$ be arbitrary but fixed. Then we can rewrite

$$t' = t[\vec{x}/\vec{v}] = \sum_{i=1}^{M} a_i(\vec{v}) \cdot b_i^n \cdot n^{e_i}$$

to the form (1), and thus by [26, Part V, Chap. 1, Num. 75], it has at most $ord(t') - 1$ real roots (where t' contains no other variable than n). As $M' \leq M$ and

$\sum_{i=1}^{M'} \mathsf{degree}(p_i(n)) \leq \sum_{i=1}^{M} e_i$, this implies that t' has at most $M - 1 + \sum_{i=1}^{M} e_i$ real roots due to (2). As this bound is independent from $\vec{v}$, the claim follows. □

We can now define the desired number rb.

Definition 6 (Root Bound of Loop Guard). *We define the* root bound *of the instantiated loop guard ψ as*

$$rb := \sum_{(t \sim 0) \in \psi} \mathsf{rootbound}(t),$$

where $\sim \in \{>, \geq\}$, i.e., here we consider all inequations of the form $t > 0$ and $t \geq 0$ occurring in the conjunction ψ.

Example 7 (Root Bound, Ex. 4 cont.). For Ex. 1, $\psi = \varphi[\vec{x}/\vec{cl}]$ is

$$0 \leq x + n + 2^n y \leq 10 \quad \equiv \quad x + n + 2^n y \geq 0 \ \wedge \ 10 - x - n - 2^n y \geq 0.$$

Here, we have

$$\mathsf{rootbound}(x + n + 2^n y) = \mathsf{rootbound}(10 - x - n - 2^n y) = 3$$

since $M = 3$ and $\sum_{i=1}^{3} e_i = 1$ and thus, $rb = 3 + 3 = 6$.

3.4 Eliminating the Innermost Quantifier

Now we show that ($\textsc{Const}_{n_0}$) is equivalent to

$$\exists m \in \mathbb{N}.\ \forall \vec{x} \in \mathbb{R}^d.\ \bigvee_{j=0}^{rb} \neg\psi[n/n_0 + j \cdot m], \tag{QE}$$

and if (QE) holds, then $n_0 + rb \cdot m$ is a bound on the runtime of (Loop), again provided that all eigenvalues are non-negative reals.

Recall that (QE) states that there is a distance m such that for all initial values of $\vec{x}$, the guard is violated after n_0, $n_0 + 1 \cdot m$, ..., or $n_0 + rb \cdot m$ iterations.

Proving Equivalence of ($\textsc{Const}_{n_0}$) and (QE): To see why ($\textsc{Const}_{n_0}$) and (QE) are equivalent, note that

$$\text{(QE) implies ($\textsc{Const}_{n_0}$) by choosing } c = rb \cdot m + 1. \qquad (\text{QE} \Rightarrow \textsc{Const}_{n_0})$$

Thus, $n_0 + c - 1 = n_0 + rb \cdot m$ is a bound on the runtime of (Loop).

To show that ($\textsc{Const}_{n_0}$) implies (QE), assume ($\textsc{Const}_{n_0}$), i.e., there is a $c \in \mathbb{N}$ so that $\forall \vec{x} \in \mathbb{R}^d.\ \exists i \in \mathbb{N}_{<c}.\ \neg\psi[n/n_0 + i]$. Then for every $j \in \mathbb{N}$, we have:

$$\begin{aligned} & \forall \vec{x} \in \mathbb{R}^d.\ \exists i \in \mathbb{N}_{<c}.\ \neg\psi[n/n_0 + i] && \text{(for some } c \in \mathbb{N} \text{ by ($\textsc{Const}_{n_0}$))} \\ \iff\ & \forall \vec{x} \in \mathbb{R}^d.\ \exists i \in \mathbb{N}_{<c}.\ \neg\varphi[\vec{x}/\vec{cl}][n/n_0 + i] && \text{(def. of } \psi) \end{aligned}$$

$$\begin{aligned}
&\iff \forall \vec{x} \in \mathbb{R}^d.\ \exists i \in \mathbb{N}_{<c}.\ \neg\varphi[\vec{x}/\vec{cl}[n/n_0+i]] && \text{(as } n \text{ does not occur in } \varphi\text{)}\\
&\iff \forall \vec{x} \in \mathbb{R}^d.\ \exists i \in \mathbb{N}_{<c}.\ \neg\varphi[\vec{x}/\mathrm{up}^{n_0+i}(\vec{x})] && \text{(def. of } \vec{cl}\text{, as } i \geq 0\text{)}\\
&\implies \forall \vec{x} \in \mathbb{R}^d.\ \exists i \in \mathbb{N}_{<c}.\ \neg\varphi[\vec{x}/\mathrm{up}^{n_0+j\cdot c+i}(\vec{x})] && \text{(for any } j \in \mathbb{N} \text{ (‡))}\\
&\iff \forall \vec{x} \in \mathbb{R}^d.\ \exists i \in \mathbb{N}_{<c}.\ \neg\varphi[\vec{x}/\vec{cl}[n/n_0+j\cdot c+i]] && \text{(def. of } \vec{cl}\text{, as } j\cdot c+i \geq 0\text{)}\\
&\iff \forall \vec{x} \in \mathbb{R}^d.\ \exists i \in \mathbb{N}_{<c}.\ \neg\varphi[\vec{x}/\vec{cl}][n/n_0+j\cdot c+i] && \text{(as } n \text{ does not occur in } \varphi\text{)}\\
&\iff \forall \vec{x} \in \mathbb{R}^d.\ \exists i \in \mathbb{N}_{<c}.\ \neg\psi[n/n_0+j\cdot c+i] && \text{(def. of } \psi\text{)}
\end{aligned}$$

For the step (‡), $\vec{x}$ is again universally quantified, i.e., we can instantiate $\vec{x}$ by $\mathrm{up}^{j\cdot c}(\vec{x})$, as in the proof that ($\textsc{Const}_<$) implies ($\textsc{Const}_{n_0}$), where we instantiated the universally quantified variables $\vec{y}$ by $\mathrm{up}^{n_0}(\vec{x})$. Thus, ($\textsc{Const}_{n_0}$) implies

$$\forall \vec{x} \in \mathbb{R}^d.\ \bigwedge_{j=0}^{rb} \exists i \in \mathbb{N}_{<c}.\ \neg\psi[n/n_0+j\cdot c+i]. \qquad (\textsc{InIntervals})$$

So ($\textsc{InIntervals}$) holds iff for all initial values of $\vec{x}$ and all $0 \leq j \leq rb$, the loop guard is violated after $n_0+j\cdot c$, $n_0+j\cdot c+1$, …, or $n_0+(j+1)\cdot c-1$ iterations.

To prove that ($\textsc{InIntervals}$) implies (QE), we assume that ($\textsc{InIntervals}$) holds and (QE) does not hold, and derive a contradiction. If (QE) does not hold, then there is a $\vec{v} \in \mathbb{R}^d$ such that

$$\bigwedge_{j=0}^{rb} \psi[n/n_0+j\cdot c][\vec{x}/\vec{v}] \qquad (\textsc{AtBorders})$$

by choosing $m = c$ in (QE). Then by the intermediate value theorem, for each $0 \leq j \leq rb$, there is at least one $(t_j \sim_j 0) \in \psi$ with $\sim_j \in \{>, \geq\}$ such that $t_j[\vec{x}/\vec{v}]$ has a real root in the interval

$$[n_0+j\cdot c,\ n_0+(j+1)\cdot c-1] \qquad (3)$$

due to ($\textsc{InIntervals}$), if we consider instantiations of n with real instead of just natural numbers. The reason is that ($\textsc{AtBorders}$) implies $t[\vec{x}/\vec{v}][n/n_0+j\cdot c] \sim 0$ for all $(t \sim 0) \in \psi$, and ($\textsc{InIntervals}$) implies that there exists a value $k \in [n_0+j\cdot c,\ n_0+(j+1)\cdot c-1]$ and a $(t_j \sim_j 0) \in \psi$ such that $t_j[\vec{x}/\vec{v}][n/k] \sim_j 0$ is violated. Then we must also have $t_j[\vec{x}/\vec{v}][n/k_0] = 0 \neq t_j[\vec{x}/\vec{v}][n/k_{\neq 0}]$ for some $k_0, k_{\neq 0} \in [n_0+j\cdot c,\ n_0+(j+1)\cdot c-1]$, since $t_j[\vec{x}/\vec{v}]$ is a continuous function in the variable n.[5] In other words, for every interval (3) there is a $t_j[\vec{x}/\vec{v}]$ that satisfies $t_j[\vec{x}/\vec{v}] \sim_j 0$ at the "borders" of the interval and violates $t_j[\vec{x}/\vec{v}] \sim_j 0$ inside the interval. Thus, we have $t_j[\vec{x}/\vec{v}] \neq 0$ (i.e., $t_j[\vec{x}/\vec{v}]$ is not the polynomial 0), and it must have a real root in the interval. Hence, the sum of the number of real roots of all non-zero expressions $t[\vec{x}/\vec{v}]$, $(t \sim 0) \in \psi$, is at least $rb+1$, i.e.,

$$\sum_{\substack{(t\sim 0)\in\psi \\ t[\vec{x}/\vec{v}]\neq 0}} |\mathsf{roots}(t[\vec{x}/\vec{v}])| > rb = \sum_{(t\sim 0)\in\psi} \mathsf{rootbound}(t).$$

[5] The reason for continuity of $t_j[\vec{x}/\vec{v}]$ is that all exponential expressions in $t_j[\vec{x}/\vec{v}]$ have the form b_i^n for $b_i \in \mathbb{R}_{\geq 0}$, since all eigenvalues of the loop are non-negative reals.

This contradicts the fact that $\mathsf{rootbound}(t)$ is a bound on the number of real roots of $t[\vec{x}/\vec{v}]$ by Lemma 5. Thus, as we instantiated m with c to derive (AtBorders), we showed

$$\neg(\text{QE}) \text{ and } (\text{Const}_{n_0}) \text{ yield a contradiction by choosing } m = c$$

or, equivalently,

$$(\text{Const}_{n_0}) \text{ implies } (\text{QE}) \text{ by choosing } m = c. \qquad (\text{Const}_{n_0} \Rightarrow \text{QE})$$

Example 8 (Quantifier Elimination, Ex. 4 and 7 cont.). For (Val), it remains to decide validity of

$$\exists m \in \mathbb{N}.\ \forall x, y \in \mathbb{R}.\ \bigvee_{j=0}^{6} \neg \left(0 \leq x + j \cdot m + 2^{j \cdot m} y \leq 10 \right). \qquad (\text{Val}^{\vee})$$

In our example we have $n_0 = 0$ and $rb = 6$. Thus, if $(\text{Val}^{\vee})$ holds, then this yields the bound $n_0 + rb \cdot m = 6 \cdot m$ on the runtime of the loop in Ex. 1.

The proof above yields the following corollary, which will be needed later.

Corollary 9. *Consider the subformula*

$$\forall \vec{x} \in \mathbb{R}^d.\ \bigvee_{j=0}^{rb} \neg\psi[n/n_0 + j \cdot m] \qquad (\text{QE}^{in})$$

of (QE)*, whose only free variable m ranges over $\mathbb{N}$. Then* (QE^{in}) *is either unsatisfiable, or it is valid for infinitely many different values of m.*

Proof. First assume $rb = 0$. Then (QE^{in}) does not depend on m, and thus it is either unsatisfiable, or it is valid for *all* possible values of m.

Now let $rb > 0$ and assume that (QE^{in}) is only valid for finitely many values of m. Let $v \in \mathbb{N}$ be the maximal number such that instantiating m by v satisfies (QE^{in}). Then with $(\text{QE} \Rightarrow \text{Const}_{n_0})$, it follows that instantiating c with $rb \cdot v + 1$ satisfies $(\text{Const}^{in}_{n_0})$. Next, by $(\text{Const}_{n_0} \Rightarrow \text{QE})$, it follows that instantiating m with $rb \cdot v + 1$ satisfies (QE^{in}). As $rb > 0$, we have $rb \cdot v + 1 > v$, i.e., (QE^{in}) also holds for a number larger than v, which contradicts our assumption on the maximality of v. ◻

3.5 Eventual Signs

Compared to the formula (Const_{n_0}), in (QE) we eliminated the innermost quantifier and replaced it by $rb + 1$ subformulas, where rb is fixed. The resulting formula (QE) is linear in the program variables $\vec{x}$ and poly-exponential in m.

Next, we want to get rid of $\vec{x}$ by using a variant of Fourier-Motzkin elimination. To see why Fourier-Motzkin elimination can be applied even though we have

non-linear terms in m, we disregard the outermost quantifier of (QE) for now, i.e., we assume that $m \in \mathbb{N}$ is given. Then proving (QE) amounts to proving the *validity* of a *disjunction* of inequations. In contrast, Fourier-Motzkin elimination is used to check the *satisfiability* of *conjunctions*. Thus, to prove validity of (QE), our goal is to use Fourier-Motzkin elimination to prove unsatisfiability of

$$\pi := \bigwedge_{j=0}^{rb} \psi[n/n_0 + j \cdot m],$$

which results from the negation of (QE).

However, Fourier-Motzkin's technique applies to linear inequations only. To apply it in our setting, we regard the (poly-exponential) inequations in π as linear inequations over $\vec{x}$ whose coefficients are univariate poly-exponential expressions w.r.t. m, i.e., the coefficients are of the form

$$0 \qquad \text{or} \qquad \sum_{i=1}^{M} d_i \cdot b_i^m \cdot m^{e_i} \tag{Coeff}$$

where m is the only variable, $M \geq 1$, $d_i \in \mathbb{R}_{\mathbb{A}} \setminus \{0\}$, $b_i \in \mathbb{R}_{\mathbb{A}>0}$, and $e_i \in \mathbb{N}$. Again, note that d_i and b_i are algebraic numbers, and thus they are suitable for algorithmic purposes. To apply Fourier-Motzkin elimination, the signs of these coefficients must be known, which is not the case, in general.

However, we can easily identify the asymptotically dominant addend of the expression (Coeff), which determines its sign *for large enough values of* m. Moreover, by Cor. 9, we may assume that m is sufficiently large without loss of generality: If (QE) holds for some small value of m, then there is also an arbitrarily large value of m where (QE) holds.

More formally, to determine the sign for sufficiently large m, w.l.o.g. assume that the pair (b_i, e_i) is unique for each addend in (Coeff). Then it suffices to consider the *asymptotically dominant* addend $d_j \cdot b_j^m \cdot m^{e_j}$ where $(b_j, e_j) \geq_{lex} (b_i, e_i)$ for all $1 \leq i \leq M$, i.e., (b_j, e_j) is maximal w.r.t. the lexicographic ordering. Since all eigenvalues b_i are non-negative reals, for sufficiently large m, the sign of the coefficient (Coeff) is equal to the sign of d_j or the coefficient is always 0. More precisely, for any number v, let:

$$\mathsf{sign}(v) = -1 \text{ if } v < 0 \qquad \mathsf{sign}(0) = 0 \qquad \mathsf{sign}(v) = 1 \text{ if } v > 0$$

Note that since we required $d_i \neq 0$ and uniqueness of (b_i, e_i) for each addend, when instantiating m by large enough numbers, then we have $\mathsf{sign}(\text{Coeff}) \neq 0$ whenever (Coeff) is of the form $\sum_{i=1}^{M} d_i \cdot b_i^m \cdot m^{e_i}$. In this case, we have

$$\exists m_0 \in \mathbb{N}.\ \forall m \geq m_0.\ \mathsf{sign}(\text{Coeff}) = \mathsf{sign}(d_j).$$

We define the *eventual sign* of (Coeff) as $\mathsf{esign}(\text{Coeff}) := 0$ if (Coeff) is 0 and as $\mathsf{esign}(\text{Coeff}) := \mathsf{sign}(d_j)$ otherwise.

Example 10 (Eventual Signs, Ex. 8 cont.). Reconsider (VAL$^\vee$). Here, the coefficients for the variable x are 1 and -1, and we obviously have $\mathsf{esign}(1) = 1$ and $\mathsf{esign}(-1) = -1$. The coefficients for the variable y are $2^{j \cdot m}$ and $-2^{j \cdot m}$ for $0 \leq j \leq 6$, where $\mathsf{esign}(2^{j \cdot m}) = 1$ and $\mathsf{esign}(-2^{j \cdot m}) = -1$. For a more complex example, consider $(3^m \cdot m^2 - 10 \cdot 2^m \cdot m^3) \cdot x$. As $(3, 2) \geq_{lex} (2, 3)$, we have $\mathsf{esign}(3^m \cdot m^2 - 10 \cdot 2^m \cdot m^3) = 1$, i.e., the asymptotically dominant coefficient of x eventually becomes positive.

3.6 Variable Elimination

We now show how to eliminate a single variable x in the vector $\vec{x}$ of variables from π. Then it immediately follows that *all* variables from $\vec{x}$ can be eliminated, so that the only remaining variable is m. Recall that π is defined as

$$\pi := \bigwedge_{j=0}^{rb} \psi[n/n_0 + j \cdot m].$$

So in particular, π is a conjunction of inequations. To eliminate x, consider the sets

$$\check{\pi}_x := \{p \cdot x + t \sim 0 \text{ occurs in } \pi \mid \sim \in \{>, \geq\}, \mathsf{esign}(p) > 0\}$$
$$\hat{\pi}_x := \{p \cdot x + t \sim 0 \text{ occurs in } \pi \mid \sim \in \{>, \geq\}, \mathsf{esign}(p) < 0\}$$

where p is the coefficient of x when regarding $p \cdot x + t$ as a linear polynomial w.r.t. $\vec{x}$ with coefficients of the form (COEFF). In other words, p is chosen such that t does not contain any occurrence of x. Thus, $\check{\pi}_x$ contains those inequations from π that give rise to lower bounds on x for large enough m, and $\hat{\pi}_x$ contains those inequations that give rise to upper bounds. Next, we define π_x to be the smallest set such that the following holds:

$$\begin{array}{rl} \text{If} & p_1 \cdot x + t_1 \sim_1 0 \in \check{\pi}_x \qquad \text{and} \qquad p_2 \cdot x + t_2 \sim_2 0 \in \hat{\pi}_x, \\ \text{then} & p_1 \cdot t_2 - p_2 \cdot t_1 \sim 0 \in \pi_x, \end{array}$$

where $\sim$ is $\geq$ if both $\sim_1$ and $\sim_2$ are $\geq$, and $\sim$ is $>$, otherwise. Then

$$\pi \equiv \pi' \text{ for all large enough values of } m,$$

where $\pi' := (\pi \cup \pi_x) \setminus (\check{\pi}_x \cup \hat{\pi}_x)$ and by construction, x does not occur in π'.

Example 11 (Fourier-Motzkin Elimination, Ex. 8 and 10 cont.). For the formula (VAL$^\vee$), we get the following inequations:

$$0 \leq \overbrace{1}^{p} \cdot x + \overbrace{j \cdot m + 2^{j \cdot m} y}^{t} \qquad \text{(for all } 0 \leq j \leq 6\text{)}$$
$$x + j \cdot m + 2^{j \cdot m} y \leq 10 \qquad \text{(for all } 0 \leq j \leq 6\text{)}$$

Thus, we have:

$$\check{\pi}_x = \{x + j \cdot m + 2^{j \cdot m} y \geq 0 \mid 0 \leq j \leq 6\}$$
$$\hat{\pi}_x = \{-x - j \cdot m - 2^{j \cdot m} y + 10 \geq 0 \mid 0 \leq j \leq 6\}$$

Hence, eliminating x yields

$$\begin{aligned}\pi' &= \{j \cdot m + 2^{j \cdot m} y \geq j' \cdot m + 2^{j' \cdot m} y - 10 \mid j, j' \in \{0, \ldots, 6\}\} \\ &= \{(j - j') \cdot m + (2^{j \cdot m} - 2^{j' \cdot m}) \cdot y + 10 \geq 0 \mid j, j' \in \{0, \ldots, 6\}\} \\ &= \{(j - j') \cdot m + ((2^j)^m - (2^{j'})^m) \cdot y + 10 \geq 0 \mid j, j' \in \{0, \ldots, 6\}\} \qquad (\star)\end{aligned}$$

Next, we eliminate y from π', where we get:

$$\begin{aligned}\check{\pi}'_y &\supseteq \{(4^m - 1) \cdot y + 2m + 10 \geq 0\} && ((\star) \text{ with } j = 2 \text{ and } j' = 0) \\ \hat{\pi}'_y &\supseteq \{(1 - 2^m) \cdot y - m + 10 \geq 0\} && ((\star) \text{ with } j = 0 \text{ and } j' = 1)\end{aligned}$$

Thus, the resulting formula π'' contains the conjunct

$$\begin{aligned}&(4^m - 1) \cdot (10 - m) - (1 - 2^m) \cdot (2m + 10) \geq 0 \\ \equiv\ &-4^m \cdot m + 10 \cdot 4^m + 2^m \cdot 2m + 10 \cdot 2^m - m - 20 \geq 0. \qquad (\lightning)\end{aligned}$$

Correctness of Variable Elimination: To see why this construction is correct, we show that

$$\pi \equiv \pi' \text{ for all large enough values of } m,$$

where $\pi' := (\pi \cup \pi_x) \setminus (\check{\pi}_x \cup \hat{\pi}_x)$. More precisely, we prove that $\pi[m/c]$ is equivalent to $\pi'[m/c]$ for every $c \in \mathbb{N}$ where $\mathsf{sign}(p[m/c]) = \mathsf{esign}(p)$ for all coefficients p in π.

Thus, if $m_0 \in \mathbb{N}$ such that for all $m \geq m_0$ we have $\mathsf{sign}(p) = \mathsf{esign}(p)$, then π is also equivalent to π' for all $m \geq m_0$. Hence, instead of checking unsatisfiability of π for some sufficiently large value of m, we can instead check unsatisfiability of π' for some sufficiently large value of m.

Equivalence of $\pi[m/c]$ and $\pi'[m/c]$: To show that $\pi[m/c]$ is equivalent to $\pi'[m/c]$ if $\mathsf{sign}(p[m/c]) = \mathsf{esign}(p)$ for all coefficients p in π, we can use the idea of Fourier-Motzkin elimination, since $\pi[m/c]$ is linear:

If all inequations are weak (i.e., built with $\geq$), then the inequations

$$p_1[m/c] \cdot x + t_1[m/c] \geq 0$$

in $\check{\pi}_x[m/c]$ are equivalent to $x \geq \frac{-t_1[m/c]}{p_1[m/c]}$ (since $p_1[m/c] > 0$ by definition of $\check{\pi}_x$). Similarly, the inequations

$$p_2[m/c] \cdot x + t_2[m/c] \geq 0$$

in $\hat{\pi}_x[m/c]$ are equivalent to $x \leq \frac{-t_2[m/c]}{p_2[m/c]}$ (since $p_2[m/c] < 0$ by definition of $\hat{\pi}_x$).

Hence, $\pi[m/c]$ is equivalent to the conjunction of $(\pi \setminus (\check{\pi}_x \cup \hat{\pi}_x))[m/c]$ and the inequations

$$\frac{-t_1[m/c]}{p_1[m/c]} \leq \frac{-t_2[m/c]}{p_2[m/c]} \qquad \text{(FM)}$$

for all $p_1 \cdot x + t_1 \geq 0$ from $\check{\pi}_x$ and all $p_2 \cdot x + t_2 \geq 0$ from $\hat{\pi}_x$.[6]

[6] This does not hold for discrete sets like $\mathbb{Z}$, which is the reason why the approach of Sect. 3.6 cannot be applied for loops over $\mathbb{Z}$.

Note that if at least one of the two inequations $p_1 \cdot x + t_1 \geq 0$ or $p_2 \cdot x + t_2 \geq 0$ is strict, then (FM) is also strict. Since $p_1[m/c] > 0$ and $p_2[m/c] < 0$, we get

$$\text{(FM)} \quad \equiv \quad -p_2[m/c] \cdot t_1[m/c] \geq -p_1[m/c] \cdot t_2[m/c] \quad \equiv \quad (p_1 \cdot t_2 - p_2 \cdot t_1 \geq 0)[m/c].$$

Thus, $\pi[m/c]$ is equivalent to $(\pi \setminus (\check{\pi}_x \cup \hat{\pi}_x))[m/c] \cup \pi_x[m/c] = \pi'[m/c]$.

3.7 Deciding Validity

After eliminating all variables, we obtain a conjunction of inequations of the form $p > 0$ or $p \geq 0$ where p is of the form (Coeff). Such a conjunction is unsatisfiable for large enough values of m iff there is an inequation $p > 0$ or $p \geq 0$ where $\mathsf{esign}(p) = -1$ or an inequation $p > 0$ with $\mathsf{esign}(p) = 0$. Thus, for the case where $\mathsf{esign}(p) = 0$, we have to distinguish between "$>$" and "$\geq$".

However, checking satisfiability in this way does not directly yield an instantiation of m where π is unsatisfiable. Thus, we cannot directly obtain a corresponding constant bound $n_0 + rb \cdot m$ on the runtime. However, if π is proven to be unsatisfiable, then to compute the constant bound on the runtime of the analyzed loop, the loop can simply be unrolled until its guard becomes unsatisfiable.

Example 12 (Deciding Validity, Ex. 11 finished). For the formula ($\lightning$), we have $\mathsf{esign}(\lightning) = -1$, as $-4^m \cdot m$ is the dominant addend, since we have:

$$\overbrace{(4,1)}^{-4^m \cdot m} >_{lex} \overbrace{(4,0)}^{10 \cdot 4^m} >_{lex} > \overbrace{(2,1)}^{2^m \cdot 2m} >_{lex} > \overbrace{(2,0)}^{10 \cdot 2^m} >_{lex} \overbrace{(1,1)}^{-m} >_{lex} \overbrace{(1,0)}^{-20}$$

Thus, ($\lightning$) is unsatisfiable for sufficiently large values of m, and hence (Val$^\vee$) is valid. Indeed, ($\lightning$) is contradictory for all $m \geq 11$ (but not for $0 \leq m \leq 10$). Therefore, Ex. 1 has constant runtime, and we obtain the bound $n_0 + rb \cdot m = 0 + 6 \cdot 11 = 66$, i.e., every run takes at most 66 iterations.

So in this section, we derived the following theorem.

Theorem 13 (Constant Runtime over $\mathbb{R}$). *Constant runtime of linear loops over $\mathbb{R}$ with real eigenvalues is decidable.*

Moreover, our approach can also be applied to loops over $\mathbb{Q}$ without adaptions, as shown by the following lemma.

Lemma 14 (Constant over $\mathbb{R}$ $\iff$ Constant over $\mathbb{Q}$). *A linear loop with real eigenvalues has constant runtime over $\mathbb{R}$ iff it has constant runtime over $\mathbb{Q}$.*

Proof. Constant runtime over the reals obviously implies constant runtime over the rationals. For the other direction, assume that the runtime over the rationals is bounded by the constant c. Then $\varphi^{(0..c)} := \varphi \wedge \mathrm{up}(\varphi) \wedge \cdots \wedge \mathrm{up}^c(\varphi)$ is unsatisfiable over the rationals. Note that this formula only contains linear arithmetic. As formulas with linear arithmetic are satisfiable over the reals iff they are satisfiable over the rationals, $\varphi^{(0..c)}$ is also unsatisfiable over the reals. Hence, the runtime over the reals is also bounded by the constant c. ◻

Then with Thm. 13, the following corollary is immediate.

Corollary 15 (Constant Runtime over $\mathbb{Q}$). *Constant runtime of linear loops over $\mathbb{Q}$ with real eigenvalues is decidable.*

4 Loops over $\mathbb{Z}$

We now consider loops over the integers, i.e., where the coefficients in (Loop) are from $\mathbb{Z}$, and where the variables range over $\mathbb{Z}$, too. It is easy to see that all parts of the procedure presented in Sect. 3 immediately carry over to the integer setting except for one: We cannot eliminate variables via Fourier-Motzkin's technique as in Sect. 3.6. However, instead we can exploit decidability of "one-parametric Presburger arithmetic" [6, 23]. This logic extends Presburger arithmetic with a single function $x \mapsto x \cdot m$, where m is a free variable. So the resulting formulas are non-linear polynomials w.r.t. m, but linear w.r.t. all other variables. Thus, we can apply decidability of this logic to (QE), but not directly to (Const).

In our setting, recall that the bases b_i of the exponential functions that occur in closed forms correspond to the eigenvalues of the loop. Thus, if we restrict our attention to the eigenvalues 0 and 1, then all exponential functions vanish. Moreover, if we also allow the eigenvalue -1, then after applying the simplification from Cor. 2, the only remaining eigenvalues are again 0 and 1. Thus, we obtain the following corollary.

Corollary 16 (Constant Runtime over $\mathbb{Z}$). *Constant runtime of linear loops over $\mathbb{Z}$ with eigenvalues from $\{-1, 0, 1\}$ is decidable.*

5 Related Work

Concerning related work on complete techniques, decidability of termination for linear single-path loops over $\mathbb{R}$, $\mathbb{Q}$, and $\mathbb{Z}$ was proven in [28], [7], and [17], respectively. Furthermore, [30] and [15] presented decidability results on termination of solvable loops and of triangular weakly non-linear (twn) loops, respectively. In contrast to the loops considered in our paper, such loops allow restricted forms of non-linearity. Building on [15], a technique to solve the (non-universal) halting problem (i.e., to decide termination for a *fixed* input), and to infer polynomial runtime bounds for twn-loops was introduced in [14]. The complete techniques for termination analysis and inference of upper bounds for twn-loops from [14,15] were implemented in the tool KoAT [20, 21].

Our work complements these results by providing a decision procedure for constant runtime. Similar to [21, Lemma 22], our approach can be extended to eigenvalues which are roots of reals (or of $\{-1, 0, 1\}$, in the integer case), i.e., to eigenvalues λ where $\lambda^n \in \mathbb{R}$ (or $\lambda^n \in \{-1, 0, 1\}$) for some $n \in \mathbb{N}$. In the future, it would be interesting to lift our results to update matrices with arbitrary eigenvalues (including complex ones that are no roots of reals). However, in such a setting, several steps of our technique must be revised. For example, the number

of roots of poly-exponential expressions involving complex numbers cannot be bounded by a constant as in Sect. 3.3 and the quantifier elimination of Sect. 3.4 to allow the subsequent application of Fourier-Motzkin elimination would no longer work. Similarly, it would be interesting to generalize our results for $\mathbb{Z}$.

Note that the techniques mentioned above rely on the fact that the asymptotically dominant addend of closed-form expressions like (PE) can easily be determined. In Sect. 3.5, we use a similar reasoning, but there, we are *not* interested in the asymptotic behavior of (PE), but whether an inequation eventually gives rise to a lower or upper bound on a given variable x. Thus, we are just interested in the dominant addend of x's *coefficient* when regarding (PE) as a linear polynomial with poly-exponential coefficients. Hence, our approach differs fundamentally from earlier techniques for deciding termination. In particular and in contrast to earlier approaches (and as already mentioned in Sect. 1), we cannot exploit the "eventual monotonicity" of linear loops with real eigenvalues, as the initial number of "non-monotonic" steps (when ignoring the loop guard) is often *not* bounded by a constant, even if the runtime of the loop is constant. To see this, consider the following loop with constant runtime:

$$\textbf{while} \quad 0 \leq x \leq 10 \quad \textbf{do} \quad \begin{pmatrix} x \\ y \end{pmatrix} \leftarrow \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \textbf{end}$$

Here, x behaves monotonically as soon as y becomes non-negative (as $y \geq 0$ implies $x \leq \mathrm{up}(x)$). However, the number k such that $\mathrm{up}^k(y) \geq 0$ holds is unbounded, as it depends on the initial value of y.

The complete technique for computing polynomial runtime bounds for twn-loops from [14] allows for computing a polynomial upper bound for any terminating twn-loop. However, these bounds are not necessarily tight, i.e., the resulting bound might be non-constant even if the runtime of the loop is constant. Thus, our work is orthogonal to [14].

There exist several techniques and tools for automatic complexity analysis of programs (with arbitrary multiple loops, if-then-else, non-determinism, ...) where the variables range over $\mathbb{Z}$. Most of these tools, e.g., CoFloCo [9], KoAT [20], Loopus [27], MaxCore [2], and RAML [16], infer *upper* bounds on the worst-case runtime complexity. Thus, if such a tool computes a constant runtime bound, then the program is guaranteed to terminate within a constant number of steps. In contrast, there also exist tools like LOBER [3] or LoAT [12], which infer *lower* bounds on the worst-case runtime complexity of loops or programs, respectively. Consequently, such tools can *dis*prove constant runtime.

6 Evaluation and Conclusion

Evaluation: To evaluate the practical applicability of our decision procedure, we implemented the approach from Sect. 3 in Python, using SymPy [24] for all computations with algebraic numbers. We did not yet implement the adaption to loops over $\mathbb{Z}$ from Sect. 4, as we are not aware of any implementation of a decision procedure for one-parametric Presburger arithmetic. For our evaluation, we used the benchmark set from [11] consisting of all 1495 linear single-path loops

from the category *Termination of Integer Transition Systems* of the *Termination Problems Data Base* [29], the benchmark suite used in the annual *Termination and Complexity Competition (TermComp)* [13]. After removing duplicates, we obtained 1336 loops. All these loops have only real eigenvalues, which indicates that our approach is indeed applicable to a large class of examples. 1327 of them have eigenvalues from $\{-1, 0, 1\}$, which shows that even our seemingly restrictive result for loops over $\mathbb{Z}$ (see Sect. 4) is very relevant from a practical point of view.

For our experiments, we compared the performance of our tool Loopy against all tools for complexity analysis of integer transition systems that participated in this year's *TermComp*, i.e., CoFloCo, KoAT, and LoAT. Note that these three tools consider initial values from $\mathbb{Z}$ instead of $\mathbb{R}$ or $\mathbb{Q}$. Thus, if LoAT infers a non-constant *lower* bound over the integers for a loop, then this loop also has non-constant runtime over the rationals and reals; conversely, whenever Loopy proves a constant bound over the rationals and reals, then this is also a constant *upper* bound over the integers. To the best of our knowledge, there are no other tools which analyze complexity of programs over $\mathbb{R}$ or $\mathbb{Q}$. Loopy proved constant runtime for 76 loops and non-constant runtime for the 1260 remaining loops, with an average analysis time of 1.71 seconds on an AMD Ryzen 7 3700X octa-core CPU. This shows that our decision procedure is not only of interest from a theoretical point of view, but it is also efficient on a standard benchmark set, which indicates its applicability in practice. CoFloCo and KoAT were able to prove constant runtime for five and six of these 76 loops, respectively, whereas LoAT proved non-constant runtime for 1226 loops over $\mathbb{Z}$. All examples that were classified as non-constant by LoAT were also classified as non-constant by Loopy, i.e., there are no conflicting results. Note that CoFloCo and KoAT do not implement dedicated techniques for proving constant bounds, and thus it is not surprising that they do not infer more constant runtime bounds. Hence, our experiments demonstrate that tools for upper bounds can benefit from our decision procedure to improve their precision on such loops. To download Loopy and for the detailed results of our evaluation, we refer to [22].

Conclusion: We presented the first complete approach for the constant runtime problem of linear loops. For loops over $\mathbb{R}$ and $\mathbb{Q}$, our decision procedure applies if the eigenvalues of the update matrix are real. For loops over $\mathbb{Z}$, it covers eigenvalues from $\{-1, 0, 1\}$. Our empirical evaluation shows the practicability of our approach. In future work, we will try to add support for initial conditions for loops with real eigenvalues, which would yield a partial solution to the long-standing open problem whether the existence of multiphase-linear ranking functions is decidable for linear loops [5]. Note that while termination of linear loops for fixed initial values corresponds to the positivity problem (which has been open for decades), the positivity problem is decidable for real eigenvalues [1]. Moreover, we will try to extend our approach to arbitrary complex eigenvalues (without considering initial conditions).

Data Availability Statement

An artifact including our implementation in the tool Loopy is available at [22]:

https://doi.org/10.5281/zenodo.17313899

This artifact allows to reproduce all results of our evaluation.

References

1. Akshay, S., Balaji, N., Vyas, N.: Complexity of restricted variants of Skolem and related problems. In: Proc. MFCS '17. LIPIcs 83 (2017). https://doi.org/10.4230/LIPICS.MFCS.2017.78
2. Albert, E., Bofill, M., Borralleras, C., Martín-Martín, E., Rubio, A.: Resource analysis driven by (conditional) termination proofs. Theory and Practice of Logic Programming **19**(5-6), 722–739 (2019). https://doi.org/10.1017/S1471068419000152
3. Albert, E., Genaim, S., Martín-Martín, E., Merayo, A., Rubio, A.: Lower-bound synthesis using loop specialization and Max-SMT. In: Proc. CAV '21. pp. 863–886. LNCS 12760 (2021). https://doi.org/10.1007/978-3-030-81688-9_40
4. Ben-Amram, A.M., Genaim, S.: On multiphase-linear ranking functions. In: Proc. CAV '17. pp. 601–620. LNCS 10427 (2017). https://doi.org/10.1007/978-3-319-63390-9_32
5. Ben-Amram, A.M., Doménech, J.J., Genaim, S.: Multiphase-linear ranking functions and their relation to recurrent sets. In: Proc. SAS '19. pp. 459–480. LNCS 11822 (2019). https://doi.org/10.1007/978-3-030-32304-2_22
6. Bogart, T., Goodrick, J., Woods, K.: Parametric Presburger arithmetic: logic, combinatorics, and quasi-polynomial behavior. Discrete Analysis (2017). https://doi.org/10.19086/da.1254
7. Braverman, M.: Termination of integer linear programs. In: Proc. CAV '06. pp. 372–385. LNCS 4144 (2006). https://doi.org/10.1007/11817963_34
8. Cai, J.Y.: Computing Jordan normal forms exactly for commuting matrices in polynomial time. International Journal of Foundations of Computer Science **5**(3/4), 293–302 (1994). https://doi.org/10.1142/S0129054194000165
9. Flores-Montoya, A.: Upper and lower amortized cost bounds of programs expressed as cost relations. In: Proc. FM '16. pp. 254–273. LNCS 9995 (2016). https://doi.org/10.1007/978-3-319-48989-6_16
10. Frohn, F., Giesl, J.: Termination of triangular integer loops is decidable. In: Proc. CAV '19. pp. 426–444. LNCS 11562 (2019). https://doi.org/10.1007/978-3-030-25543-5_24
11. Frohn, F.: A calculus for modular loop acceleration. In: Proc. TACAS '20. pp. 58–76. LNCS 12078 (2020). https://doi.org/10.1007/978-3-030-45190-5_4
12. Frohn, F., Giesl, J.: Proving non-termination and lower runtime bounds with LoAT (System description). In: Proc. IJCAR '22. pp. 712–722. LNCS 13385 (2022). https://doi.org/10.1007/978-3-031-10769-6_41
13. Giesl, J., Rubio, A., Sternagel, C., Waldmann, J., Yamada, A.: The termination and complexity competition. In: Proc. TACAS '19. pp. 156–166. LNCS 11429 (2019). https://doi.org/10.1007/978-3-030-17502-3_10, website of *TermComp*: https://termination-portal.org/wiki/Termination_Competition
14. Hark, M., Frohn, F., Giesl, J.: Polynomial loops: Beyond termination. In: Proc. LPAR '20. pp. 279–297. EPiC 73 (2020). https://doi.org/10.29007/NXV1

15. Hark, M., Frohn, F., Giesl, J.: Termination of triangular polynomial loops. Formal Methods in System Design **65**(1), 70–132 (2025). https://doi.org/10.1007/s10703-023-00440-z
16. Hoffmann, J., Das, A., Weng, S.: Towards automatic resource bound analysis for OCaml. In: Proc. POPL '17. pp. 359–373 (2017). https://doi.org/10.1145/3009837.3009842
17. Hosseini, M., Ouaknine, J., Worrell, J.: Termination of linear loops over the integers. In: Proc. ICALP '19. LIPIcs 132 (2019). https://doi.org/10.4230/LIPIcs.ICALP.2019.118
18. Kincaid, Z., Breck, J., Cyphert, J., Reps, T.: Closed forms for numerical loops. Proceedings of the ACM on Programming Languages **3**(POPL) (2019). https://doi.org/10.1145/3290368
19. Kovács, L.: A complete invariant generation approach for P-solvable loops. In: Proc. PSI '09. pp. 242–256. LNCS 5947 (2010). https://doi.org/10.1007/978-3-642-11486-1_21
20. Lommen, N., Meyer, F., Giesl, J.: Automatic complexity analysis of integer programs via triangular weakly non-linear loops. In: Proc. IJCAR '22. pp. 734–754. LNCS 13385 (2022). https://doi.org/10.1007/978-3-031-10769-6_43
21. Lommen, N., Giesl, J.: Targeting completeness: Using closed forms for size bounds of integer programs. In: Proc. FroCoS '23. pp. 3–22. LNCS 14279 (2023). https://doi.org/10.1007/978-3-031-43369-6_1
22. Loopy: Artifact, Evaluation, and Benchmarks. Zenodo (2025). https://doi.org/10.5281/zenodo.17313899
23. Mansutti, A., Starchak, M.R.: One-parametric Presburger arithmetic has quantifier elimination. In: Proc. MFCS '25. LIPIcs 345 (2025). https://doi.org/10.4230/LIPICS.MFCS.2025.72
24. Meurer, A., Smith, C.P., Paprocki, M., Čertík, O., Kirpichev, S.B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J.K., Singh, S., Rathnayake, T., Vig, S., Granger, B.E., Muller, R.P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., Curry, M.J., Terrel, A.R., Roučka, Š., Saboo, A., Fernando, I., Kulal, S., Cimrman, R., Scopatz, A.: SymPy: symbolic computing in Python. PeerJ Computer Science **3**, e103 (2017). https://doi.org/10.7717/peerj-cs.103
25. Ouaknine, J., Pinto, J.S., Worrell, J.: On termination of integer linear loops. In: Proc. SODA '15. pp. 957–969 (2015). https://doi.org/10.1137/1.9781611973730.65
26. Pólya, G., Szegö, G.: Problems and Theorems in Analysis II: Theory of Functions. Zeros. Polynomials. Determinants. Number Theory. Geometry. Classics in Mathematics, Springer (1976). https://doi.org/10.1007/978-3-642-61905-2
27. Sinn, M., Zuleger, F., Veith, H.: Complexity and resource bound analysis of imperative programs using difference constraints. Journal of Automated Reasoning **59**(1), 3–45 (2017). https://doi.org/10.1007/s10817-016-9402-4
28. Tiwari, A.: Termination of linear programs. In: Proc. CAV '04. pp. 70–82. LNCS 3114 (2004). https://doi.org/10.1007/978-3-540-27813-9_6
29. TPDB (Termination Problems Data Base), https://github.com/TermCOMP/TPDB
30. Xu, M., Li, Z.B.: Symbolic termination analysis of solvable loops. Journal of Symbolic Computation **50**, 28–49 (2013). https://doi.org/10.1016/j.jsc.2012.05.005