EPiC Series in Computing
Volume 73, 2020, Pages 279–297
LPAR23. LPAR-23: 23rd International Conference on Logic for Programming, Artificial Intelligence and Reasoning

Polynomial Loops: Beyond Termination*

Marcel Hark[1], Florian Frohn[2], and Jürgen Giesl[1]

[1] LuFG Informatik 2, RWTH Aachen University, Germany
[2] Max Planck Institute for Informatics and Saarland Informatics Campus, Saarbrücken, Germany

Abstract

In the last years, several works were concerned with identifying classes of programs where termination is decidable. We consider triangular weakly non-linear loops (*twn*-loops) over a ring $\mathbb{Z} \leq \mathcal{S} \leq \mathbb{R}_{\mathbb{A}}$, where $\mathbb{R}_{\mathbb{A}}$ is the set of all real algebraic numbers. Essentially, the body of such a loop is a single assignment $\left(\begin{smallmatrix} x_1 \\ \cdots \\ x_d \end{smallmatrix}\right) \leftarrow \left(\begin{smallmatrix} c_1 \cdot x_1 + pol_1 \\ \cdots \\ c_d \cdot x_d + pol_d \end{smallmatrix}\right)$ where each x_i is a variable, $c_i \in \mathcal{S}$, and each pol_i is a (possibly non-linear) polynomial over $\mathcal{S}$ and the variables $x_{i+1}, \ldots, x_d$. Recently, we showed that termination of such loops is decidable for $\mathcal{S} = \mathbb{R}_{\mathbb{A}}$ and non-termination is semi-decidable for $\mathcal{S} = \mathbb{Z}$ and $\mathcal{S} = \mathbb{Q}$ [19].

In this paper, we show that the *halting problem* is decidable for *twn*-loops over any ring $\mathbb{Z} \leq \mathcal{S} \leq \mathbb{R}_{\mathbb{A}}$. In contrast to the termination problem, where termination on *all* inputs is considered, the halting problem is concerned with termination on a *given* input. This allows us to compute *witnesses for non-termination*.

Moreover, we present the first computability results on the *runtime complexity* of such loops. More precisely, we show that for *twn*-loops over $\mathbb{Z}$ one can always compute a polynomial f such that the length of all terminating runs is bounded by $f(\|(x_1, \ldots, x_d)\|)$, where $\|\cdot\|$ denotes the 1-norm. As a corollary, we obtain that the runtime of a terminating triangular *linear* loop over $\mathbb{Z}$ is at most linear.

1 Introduction

We consider loops of the form

$$\textbf{while } \varphi \textbf{ do } \vec{x} \leftarrow \vec{u}. \tag{1}$$

Here, $\vec{x}$ is a vector[1] of pairwise different variables $x_1, \ldots, x_d$ that range over a ring $\mathbb{Z} \leq \mathcal{S} \leq \mathbb{R}_{\mathbb{A}}$, where $\leq$ denotes the subring relation and $\mathbb{R}_{\mathbb{A}}$ is the set of real algebraic numbers. The reason for the restriction to algebraic numbers is that it is unclear how to represent transcendental numbers in programs and formal languages (i.e., decision problems). Moreover, $\vec{u} \in (\mathcal{S}[\vec{x}])^d$ where $\mathcal{S}[\vec{x}]$ is the set of all polynomials over $\vec{x}$ with coefficients from $\mathcal{S}$. The condition φ is a propositional formula over the atoms $\{pol \triangleright 0 \mid pol \in \mathcal{S}[\vec{x}], \triangleright \in \{\geq, >\}\}$.[2]

*funded by DFG grant 389792660 as part of TRR 248, by the DFG Research Training Group 2236 UnRAVeL, and by DFG grant GI 274/6-2

[1] We use row- and column-vectors interchangeably to improve readability.

[2] One may also use other relations like "=" and atoms of the form "$\alpha \triangleright \beta$" with $\beta \in \mathcal{S}[\vec{x}]$ as syntactic sugar. Negation is also syntactic sugar in our setting, as, e.g., $\neg(pol > 0)$ is equivalent to $-pol \geq 0$. So w.l.o.g. φ is built from atoms, $\wedge$, and $\vee$, where the empty conjunction (resp. disjunction) represents "true" (resp. "false").

We often represent a loop (1) by the tuple $(\varphi, \vec{u})$ of the *loop condition* φ and the *update* $\vec{u} = (u_1, \ldots, u_d)$. Unless stated otherwise, $(\varphi, \vec{u})$ is always a loop on $\mathcal{S}^d$ using the variables $\vec{x} = (x_1, \ldots, x_d)$ where $\mathbb{Z} \leq \mathcal{S} \leq \mathbb{R}_{\mathbb{A}}$ throughout the paper. A loop $(\varphi, \vec{u})$ is *linear* if it only uses linear polynomial arithmetic, i.e., $\vec{u} = A \cdot \vec{x} + \vec{b}$ for some $A \in \mathcal{S}^{d \times d}$ and some $\vec{b} \in \mathcal{S}^d$, and in addition, φ is linear (i.e., it only contains linear inequations).

There exist several decidability results for the termination of linear [9, 10, 18, 24, 30, 32, 38, 43] and non-linear loops [19, 31, 45]. However, a (non-)termination proof alone is often of limited use.

In particular, non-termination proofs should be accompanied by a *witness of non-termination*, i.e., a non-terminating input that can, e.g., be used for debugging. However, most existing (semi-)decision procedures for (non-)termination do not yield such witnesses [10, 18, 19, 24, 38, 43, 45]. To close this gap, we prove the novel result that the *halting problem* for *twn*-loops is decidable (cf. Sect. 3), i.e., we show how to decide whether a loop terminates on any *fixed* input. Thus, we obtain a technique to enumerate all witnesses for non-termination of a loop.

For terminating inputs, the question *how fast* the loop in question terminates is of high interest. Thus, many automated techniques to derive bounds on the *runtime complexity* of programs have been proposed [2, 3, 12, 13, 16, 17, 23, 41, 42]. However, these techniques are usually incomplete. In contrast, we present a *complete* technique to derive polynomial upper bounds on the runtime complexity of *twn*-loops over $\mathbb{Z}$ (cf. Sect. 4). In particular, this implies that triangular *linear* loops have at most linear runtime complexity.

2 Preliminaries

In this section, we recapitulate previous results about *twn*-loops [18, 19, 27]. For any entity s, let $\mathcal{V}(s)$ be the set of all variables that occur in s. Moreover, for any variable y, let $s[y/t]$ result from s by replacing all free occurrences of y by t. Similarly, if $\vec{y} = (y_1, \ldots, y_d)$ and $\vec{t} = (t_1, \ldots, t_d)$, then $s[\vec{y}/\vec{t}] = s[y_1/t_1, \ldots, y_d/t_d]$ results from s by replacing all free occurrences of each y_i by t_i. If $\mathcal{V}(s) \subseteq \{x_1, \ldots, x_d\}$ for the specific variables $\vec{x} = (x_1, \ldots, x_d)$, then we often abbreviate $s[\vec{x}/\vec{t}]$ by $s(\vec{t})$. Def. 1 formalizes the intuitive notion of termination for a loop $(\varphi, \vec{u})$. For all $n \in \mathbb{N}$, $\vec{u}^n$ denotes the n-fold application of $\vec{u}$, i.e., for $\vec{e} \in \mathcal{S}^d$ we have $\vec{u}^0(\vec{e}) = \vec{e}$ and $\vec{u}^{n+1}(\vec{e}) = \vec{u}(\vec{u}^n(\vec{e}))$.

Definition 1 (Witness for Non-Termination). *Let $\vec{e} \in \mathcal{S}^d$. If*

$$\forall n \in \mathbb{N}.\ \varphi(\vec{u}^n(\vec{e})),$$

then $\vec{e}$ is a witness for non-termination. *Otherwise, $(\varphi, \vec{u})$* terminates *on $\vec{e}$. A loop is* terminating *if it does not have any witnesses for non-termination.*

If $\vec{u}^{n_0}(\vec{e})$ is a witness for non-termination for some $n_0 \in \mathbb{N}$, then $\vec{e}$ is called a witness for eventual non-termination. *$(E)NT_{(\varphi,\vec{u})}$ denotes the set of witnesses for (eventual) non-termination of $(\varphi, \vec{u})$ and we define $T_{(\varphi,\vec{u})} = \mathcal{S}^d \setminus NT_{(\varphi,\vec{u})}$.*

Given an assignment $\vec{x} \leftarrow \vec{u}$, the relation $\succ_{\vec{u}} \in \mathcal{V}(\vec{u}) \times \mathcal{V}(\vec{u})$ is the transitive closure of

$$\{(x_i, x_j) \mid i, j \in \{1, \ldots, d\}, i \neq j, x_j \in \mathcal{V}(u_i)\},$$

i.e., $x_i \succ_{\vec{u}} x_j$ means that x_i depends on x_j. A loop $(\varphi, \vec{u})$ is *triangular* if $\succ_{\vec{u}}$ is well founded. So the restriction to triangular loops prohibits "cyclic dependencies" of variables (e.g., where the new values of x_1 and x_2 both depend on the old values of x_1 and x_2). For example, a loop whose body consists of the assignment $\left(\begin{smallmatrix} x_1 \\ x_2 \end{smallmatrix}\right) \leftarrow \begin{pmatrix} x_1 + x_2^2 \\ x_2 + 1 \end{pmatrix}$ is triangular since $\succ = \{(x_1, x_2)\}$ is well founded, whereas a loop with the body $\left(\begin{smallmatrix} x_1 \\ x_2 \end{smallmatrix}\right) \leftarrow \begin{pmatrix} x_1 + x_2^2 \\ x_1 + 1 \end{pmatrix}$ is not triangular. Triangularity is used to compute a *closed form* for the n-fold application of the update $\vec{u}$, i.e., a vector $\vec{q}$ of d expressions over the variables $\vec{x}$ and n with $\vec{q} = \vec{u}^n$, by handling one variable after the other. From a practical point of view, the restriction to triangular loops seems quite natural. For example, in [20], 1511 polynomial loops were extracted from the *Termination Problems Data Base* [44], the benchmark collection which is used at the annual *Termination and Complexity Competition* [21], and only 26 of them were non-triangular.

Furthermore, $(\varphi, \vec{u})$ is *weakly non-linear* if there is no $1 \leq i \leq d$ such that x_i occurs in a non-linear monomial of u_i. So for example, a loop with the body $\left(\begin{smallmatrix} x_1 \\ x_2 \end{smallmatrix}\right) \leftarrow \left(\begin{smallmatrix} x_1 + x_2^2 \\ x_2 + 1 \end{smallmatrix}\right)$ is weakly non-linear, whereas a loop with the body $\left(\begin{smallmatrix} x_1 \\ x_2 \end{smallmatrix}\right) \leftarrow \left(\begin{smallmatrix} x_1 \cdot x_2 \\ x_2 + 1 \end{smallmatrix}\right)$ is not. Together with triangularity, weak non-linearity allows us to compute a closed form for the n-fold application of the update.

A *twn*-loop is triangular and weakly non-linear. So in other words, by permuting variables every *twn*-loop can be transformed to the form

$$\begin{pmatrix} x_1 \\ \cdots \\ x_d \end{pmatrix} \leftarrow \begin{pmatrix} c_1 \cdot x_1 + pol_1 \\ \cdots \\ c_d \cdot x_d + pol_d \end{pmatrix}$$

where $c_i \in \mathcal{S}$ and $pol_i \in \mathcal{S}[x_{i+1}, \ldots, x_d]$. If $(\varphi, \vec{u})$ is weakly non-linear and each c_i is non-negative, then $(\varphi, \vec{u})$ is *non-negative*. A *tnn*-loop is triangular and non-negative (and thus, also weakly non-linear).

In [18, 19] it was shown that when analyzing the termination behavior of *twn*-loops, it is enough to only consider *tnn*-loops. The reason is that *chaining* transforms *twn*- into *tnn*-loops and preserves witnesses for (eventual) non-termination.

Theorem 2 (Chaining [18, 19])**.** *If $(\varphi, \vec{u})$ is twn, then the* chained *loop $(\varphi_{\text{ch}}, \vec{u}_{\text{ch}}) = (\varphi \wedge \varphi(\vec{u}), \vec{u}(\vec{u}))$ is tnn and we have $(E)NT_{(\varphi,\vec{u})} = (E)NT_{(\varphi_{\text{ch}},\vec{u}_{\text{ch}})}$.*

When computing closed forms for the n-fold update of *tnn*-loops, one obtains so-called *poly-exponential expressions*. Poly-exponential expressions are arithmetic terms over the variables $\vec{x}$ and the additional designated variable n. In the following, for any $X \subseteq \mathbb{R}$, $k \in \mathbb{R}$, and $\triangleright \in \{\geq, >\}$, let $X_{\triangleright k} = \{x \in X \mid x \triangleright k\}$. Moreover, $Q_{\mathcal{S}}$ is the quotient field of $\mathcal{S}$, i.e., the smallest field in which $\mathcal{S}$ can be embedded.

Definition 3 (Poly-Exponential Expressions [18, 19, 27])**.** *Let $\mathcal{C}$ be the set of all finite conjunctions over the literals $n = c, n \neq c$ where n is a designated variable and $c \in \mathbb{N}$. Literals of the form $n = c$ are called* positive *and literals $n \neq c$ are* negative*. Furthermore, given a formula $\psi \in \mathcal{C}$, let $[\![\psi]\!] : \mathbb{N} \to \{0, 1\}$ be the characteristic function of ψ, i.e., we have $[\![\psi]\!]\,(c) = 1$ if $\psi[n/c]$ holds and $[\![\psi]\!]\,(c) = 0$, otherwise. The set of all* poly-exponential expressions *over $\mathcal{S}$ with the variables $\vec{x}$ is*

$$\mathbb{PE}[\vec{x}] = \left\{ \sum\nolimits_{j=1}^{\ell} [\![\psi_j]\!] \cdot \alpha_j \cdot n^{a_j} \cdot b_j^n \;\middle|\; \ell, a_j \in \mathbb{N},\ \psi_j \in \mathcal{C},\ \alpha_j \in Q_{\mathcal{S}}[\vec{x}],\ b_j \in \mathcal{S}_{>0} \right\}.$$

The set of normalized *poly-exponential expressions $\mathbb{NPE}[\vec{x}] \subseteq \mathbb{PE}[\vec{x}]$ only consists of those expressions that do not contain factors of the form $[\![\psi_j]\!]$ (i.e., where each ψ_j is* true*). The factors α_j are called the* coefficients *of a poly-exponential expression.*

So an example for a poly-exponential expression over $\mathcal{S} = \mathbb{Z}$ (where $Q_{\mathcal{S}} = \mathbb{Q}$) is

$$[\![n \neq 0 \wedge n \neq 1 \wedge n \neq 2]\!] \cdot (\tfrac{1}{2} \cdot {x_1}^2 + \tfrac{3}{4} \cdot x_2 - 1) \cdot n^3 \cdot 3^n \; + \; [\![n = 2]\!] \cdot (x_1 - x_2).$$

Theorem 4 (Closed Forms for *tnn*-Loops [18, 27])**.** *Let $(\varphi, \vec{u})$ be a tnn-loop. Then one can compute a vector $\vec{q} \in (\mathbb{PE}[\vec{x}])^d$ such that $\vec{q} = \vec{u}^n$.*

Example 5. *Consider the following loop $\mathcal{L}$:*

$$\textbf{while } x_1 > 0 \textbf{ do } \left(\begin{smallmatrix} x_1 \\ x_2 \end{smallmatrix}\right) \leftarrow \left(\begin{smallmatrix} x_1+x_2 \\ x_2+1 \end{smallmatrix}\right)$$

Here, one can compute the closed form $\vec{q} = \left(x_1^{(n)}, x_2^{(n)}\right)$ *for the* n*-fold application of the update, where*

$$x_1^{(n)} = \tfrac{1}{2} \cdot n^2 + \left(x_2 - \tfrac{1}{2}\right) \cdot n + x_1 \qquad \textit{and} \qquad x_2^{(n)} = n + x_2.$$

3 The Halting Problem

We now consider the *halting problem* for *twn*-loops. In contrast to termination, i.e., to the question whether a loop terminates for *all* $\vec{e} \in \mathcal{S}^d$, the halting problem asks whether a loop terminates for a *given* $\vec{e} \in \mathcal{S}^d$. In this section, we will prove the following theorem.

Theorem 6. *The halting problem for twn-loops is decidable.*

Clearly, if the halting problem is decidable for $\mathcal{S} = \mathbb{R}_\mathbb{A}$, then it is also decidable for all subrings of $\mathbb{R}_\mathbb{A}$. Thus, throughout this section, w.l.o.g. we restrict ourselves to loops over $\mathbb{R}_\mathbb{A}$.

In [19], we presented (semi-)decision procedures for (non-)termination of *twn*-loops. However, the approach from [19] only yields witnesses for *eventual* non-termination of a loop $(\varphi, \vec{u})$. If one applies the update $\vec{u}$ sufficiently often to $\vec{e} \in ENT_{(\varphi,\vec{u})}$, then one obtains a witness $\vec{u}^{n_0}(\vec{e})$ for non-termination. However, the number of required updates, i.e., the value of n_0, is not obvious. So in general, up to now it was unclear how to find *witnesses* for non-termination of *twn*-loops. By proving Thm. 6, i.e., by showing that the set $NT_{(\varphi,\vec{u})}$ is decidable, we fill this gap.

To prove decidability of $NT_{(\varphi,\vec{u})}$, we show how to compute a natural number n_0 such that the truth value of the loop condition φ *stabilizes*, i.e., such that $\varphi(\vec{u}^n(\vec{e})) \iff \varphi(\vec{u}^{n_0}(\vec{e}))$ for all $n \geq n_0$. Thus, given the closed form $\vec{q}$ of $\vec{u}^n$ and a witness $\vec{e}$ for eventual non-termination (which can, e.g., be computed as in [19]), a witness for non-termination can efficiently be computed by evaluating $\vec{q}[\vec{x}/\vec{e}, n/n_0]$. In this way, one can also enumerate all (possibly infinitely many) witnesses for non-termination.

We now prove Thm. 6, i.e., we show that for any $\vec{e} \in \mathbb{R}_\mathbb{A}^d$, it is decidable whether $\vec{e}$ is a witness for non-termination. As chaining preserves NT (cf. Thm. 2), we can assume that

$$(\varphi, \vec{u}) \text{ is a } \textit{tnn}\text{-loop} \qquad \text{(Simplification 1)}$$

without loss of generality. As mentioned above, our proof is based on the concept of *stabilization*.

Definition 7 (Stabilization)**.** *We say that* $(\varphi, \vec{u})$ stabilizes *on* $\vec{e} \in \mathbb{R}_\mathbb{A}^d$ *after* $n_0 \in \mathbb{N}$ *iterations if*

$$\forall n \geq n_0.\ \varphi(\vec{u}^n(\vec{e})) \iff \varphi(\vec{u}^{n_0}(\vec{e})).$$

The smallest value n_0 *for which* $(\varphi, \vec{u})$ *stabilizes on* $\vec{e}$ *is called the* stabilization threshold *of* $(\varphi, \vec{u})$ *on* $\vec{e}$ *(written* $sth_{(\varphi,\vec{u})}(\vec{e})$*).*

Clearly, $(\varphi, \vec{u})$ stabilizes on $\vec{e}$ after n_0 iterations iff

$$\forall n \geq n_0.\ \varphi(\vec{q}[\vec{x}/\vec{e}]) \iff \varphi(\vec{q}[\vec{x}/\vec{e}, n/n_0]),$$

where $\vec{q}$ is the closed form of $\vec{u}^n$. For convenience, we sometimes also say that $\varphi(\vec{q})$ stabilizes on $\vec{e}$. If $\varphi(\vec{q}) \equiv pe \triangleright 0$ for $pe \in \mathbb{PE}[\vec{x}]$ and $\triangleright \in \{\geq, >\}$, then we also just say that pe stabilizes on $\vec{e}$.

For any $pe \in \mathbb{PE}[\vec{x}]$ and $\vec{e} \in \mathbb{R}_\mathbb{A}^d$, let $sth_{pe}(\vec{e})$ be the smallest value n_0 such that

$$\forall n \geq n_0.\ \operatorname{sign}(pe[\vec{x}/\vec{e}]) = \operatorname{sign}(pe[\vec{x}/\vec{e}, n/n_0]),$$

where for any $c \in \mathbb{R}$, we define $\operatorname{sign}(c) = 1$ if $c > 0$, $\operatorname{sign}(c) = -1$ if $c < 0$, and $\operatorname{sign}(0) = 0$. So if $\varphi(\vec{q}) \equiv pe \triangleright 0$, then we have[3] $sth_{(\varphi,\vec{u})} \leq sth_{pe}$.

[3]We compare functions pointwise, i.e., $f \leq g$ iff $f(\vec{e}) \leq g(\vec{e})$ for all $\vec{e} \in \operatorname{dom}(f) = \operatorname{dom}(g)$.

The following lemma is an immediate consequence of Thm. 4 and the fact that every poly-exponential expression is weakly monotonic w.r.t. n for large enough values of n.[4]

Lemma 8. *Every tnn-loop stabilizes on each* $\vec{e} \in \mathbb{R}_{\mathbb{A}}^d$.

So for *tnn*-loops $(\varphi, \vec{u})$, stabilization generalizes eventual non-termination: For every $\vec{e} \in ENT_{(\varphi,\vec{u})}$ there exists an n_0 such that $\forall n \geq n_0.\ \varphi(\vec{u}^n(\vec{e}))$, and for every $\vec{e} \in \mathbb{R}_{\mathbb{A}}^d \setminus ENT_{(\varphi,\vec{u})}$ there exists an n_0 such that $\forall n \geq n_0.\ \neg\varphi(\vec{u}^n(\vec{e}))$.

Example 9. *The tnn-loop* $\mathcal{L}$ *from Ex. 5 stabilizes on any input. Its loop condition is* $x_1 > 0$ *and for all* $x_1, x_2 \in \mathbb{R}_{\mathbb{A}}$, *we have* $x_1^{(n)} = \frac{1}{2} \cdot n^2 + \left(x_2 - \frac{1}{2}\right) \cdot n + x_1 > 0$ *for large enough* n.

To decide the halting problem, it suffices to find a computable upper bound n_0 on $sth_{(\varphi,\vec{u})}(\vec{e})$. Then $(\varphi, \vec{u})$ diverges on $\vec{e}$ iff $\forall n \leq n_0.\ \varphi(\vec{u}^n(\vec{e}))$. Let $pol_1 \rhd_1 0, \ldots, pol_r \rhd_r 0$ be all inequations occurring in φ. Then we have

$$sth_{(\varphi,\vec{u})} \leq \max\{sth_{(pol_i \rhd_i 0,\vec{u})} \mid 1 \leq i \leq r\} \tag{2}$$

and thus we can assume

$$\varphi \equiv pol \rhd 0 \qquad \text{where } \rhd \in \{\geq, >\} \text{ and } pol \in \mathbb{R}_{\mathbb{A}}[\vec{x}]. \tag{Simplification 2}$$

So in the following, let $\varphi(\vec{q}) \equiv pol(\vec{q}) \rhd 0 \equiv pe \rhd 0$ for some $pe \in \mathbb{PE}[\vec{x}]$. Then our goal is to over-approximate the stabilization threshold of pe on $\vec{e}$. Here, we may assume $pe \in \mathbb{NPE}[\vec{x}]$ without loss of generality. To see why, let $c \in \mathbb{N}$ be larger than any constant that occurs in a factor $[\![\psi]\!]$ in pe. (If there is no such factor, then we choose $c = 0$.) Then we can compute the stabilization threshold for pe_{norm} instead of pe on $\vec{e}$, where pe_{norm} results from pe by replacing all factors $[\![\psi]\!]$ that contain positive literals with 0 and all other such factors with 1. Afterwards, we can easily lift the resulting bound on $sth_{pe_{norm}}(\vec{e})$ to a bound on $sth_{pe}(\vec{e})$. The reason is that we have $pe = pe_{norm}$ for all $n \geq c$ (since then, positive literals become false and negative literals become true). Thus,

$$sth_{(\varphi,\vec{u})}(\vec{e}) \leq sth_{pe}(\vec{e}) \leq \max\{sth_{pe_{norm}}(\vec{e}), c\}. \tag{3}$$

Hence, in the following we assume

$$\varphi(\vec{q}) \equiv pol(\vec{q}) \rhd 0 \equiv pe \rhd 0 \qquad \text{where } pe \in \mathbb{NPE}[\vec{x}]. \tag{Simplification 3}$$

To infer a bound on $sth_{pe}(\vec{e})$, we rely on the concept of *monotonicity thresholds*.

Definition 10 (Monotonicity Threshold). *Let* $\widehat{a}, \breve{a} \in \mathbb{N}$ *and* $\widehat{b}, \breve{b} \in (\mathbb{R}_{\mathbb{A}})_{>0}$ *with* $(\widehat{b}, \widehat{a}) >_{lex} (\breve{b}, \breve{a})$ *(i.e.,* $\widehat{b} > \breve{b}$ *or both* $\widehat{b} = \breve{b}$ *and* $\widehat{a} > \breve{a}$*). Moreover, let* $k \in \mathbb{R}_{\mathbb{A}}$. *Then the smallest* $n_0 \in \mathbb{N}$ *such that* $n^{\widehat{a}} \cdot \widehat{b}^n > k \cdot n^{\breve{a}} \cdot \breve{b}^n$ *for all* $n \geq n_0$ *is called the* k-monotonicity threshold *of* $(\widehat{b}, \widehat{a})$ *and* $(\breve{b}, \breve{a})$.

Example 11. *As an example, consider* $(\widehat{b}, \widehat{a}) = (1, 2) >_{lex} (1, 1) = (\breve{b}, \breve{a})$. *The* 14*-monotonicity threshold of* $(\widehat{b}, \widehat{a})$ *and* $(\breve{b}, \breve{a})$ *is* $n_0 = 15$, *as* $n^{\widehat{a}} \cdot \widehat{b}^n = n^2 \cdot 1^n > 14 \cdot n^1 \cdot 1^n = 14 \cdot n^{\breve{a}} \cdot \breve{b}^n$ *for all* $n \geq n_0 = 15$, *but* $14^{\widehat{a}} \cdot \widehat{b}^{14} = 14^2 \cdot 1^{14} = 196 \leq 196 = 14 \cdot 14^1 \cdot 1^{14} = 14 \cdot 14^{\breve{a}} \cdot \breve{b}^{14}$.

Lemma 12 (Computing Monotonicity Thresholds). *For any* $\widehat{a}, \breve{a} \in \mathbb{N}$, $\widehat{b}, \breve{b} \in (\mathbb{R}_{\mathbb{A}})_{>0}$ *with* $(\widehat{b}, \widehat{a}) >_{lex} (\breve{b}, \breve{a})$, *and any* $k \in \mathbb{R}_{\mathbb{A}}$, *the* k*-monotonicity threshold of* $(\widehat{b}, \widehat{a})$ *and* $(\breve{b}, \breve{a})$ *is computable.*

Proof. If $k \leq 0$ then the k-monotonicity threshold of $(\widehat{b}, \widehat{a})$ and $(\breve{b}, \breve{a})$ is 1. We now regard the case $k > 0$. To prove the lemma, we show that if both

$$\widehat{a} \geq \breve{a} \qquad \text{or} \qquad \left(\tfrac{m}{m+1}\right)^{\breve{a}-\widehat{a}} \cdot \tfrac{\widehat{b}}{\breve{b}} \geq 1 \qquad\qquad \text{and} \tag{4}$$

[4]In other words, for large enough n, every $pe \in \mathbb{PE}[\vec{x}]$ is weakly monotonically increasing (or decreasing), i.e., $n' \geq n$ implies validity of $pe[n/n'] \geq pe$ (or of $pe[n/n'] \leq pe$).

$$m^{\widehat{a}} \cdot \widehat{b}^m > k \cdot m^{\breve{a}} \cdot \breve{b}^m, \tag{5}$$

then m is an upper bound on the k-monotonicity threshold of $(\widehat{b}, \widehat{a})$ and $(\breve{b}, \breve{a})$. Then the claim immediately follows, as $\lim_{m\to\infty} \left(\frac{m}{m+1}\right)^{\breve{a}-\widehat{a}} \cdot \frac{\widehat{b}}{\breve{b}} > 1$ if $\widehat{a} < \breve{a}$ (and thus $\widehat{b} > \breve{b}$), i.e., (4) and (5) hold for large enough m. So to compute monotonicity thresholds, one initializes m to 0 and increments it until (4) and (5) are satisfied. As (4) and (5) are only a sufficient criterion for upper bounds on the k-monotonicity threshold, we now have $n^{\widehat{a}} \cdot \widehat{b}^n > k \cdot n^{\breve{a}} \cdot \breve{b}^n$ for all $n \geq m$, but m is not necessarily the smallest such number. Hence, one then has to decrement m again until $(m-1)^{\widehat{a}} \cdot \widehat{b}^{m-1} \leq k \cdot (m-1)^{\breve{a}} \cdot \breve{b}^{m-1}$ holds.

So assuming (4) and (5), we need to prove

$$n^{\widehat{a}} \cdot \widehat{b}^n > k \cdot n^{\breve{a}} \cdot \breve{b}^n \qquad \text{for all } n \geq m. \tag{6}$$

Since $n \geq m > 0$ due to (5), we have

$$n^{\widehat{a}} \cdot \widehat{b}^n > k \cdot n^{\breve{a}} \cdot \breve{b}^n \qquad \text{iff} \qquad n^{\widehat{a}-\breve{a}} \cdot \left(\frac{\widehat{b}}{\breve{b}}\right)^n > k.$$

Case $\widehat{a} \geq \breve{a}$: Then $n^{\widehat{a}-\breve{a}} \cdot \left(\frac{\widehat{b}}{\breve{b}}\right)^n$ is monotonically increasing and thus, (6) follows from (5).

Case $\widehat{a} < \breve{a}$: Let

$$f(n) = n^{\widehat{a}-\breve{a}} \cdot \left(\frac{\widehat{b}}{\breve{b}}\right)^n \qquad \text{and} \qquad f^{\Delta}(n) = f(n+1) - f(n).$$

We prove that (4) implies[5] $f^{\Delta}(n) \geq 0$ for all $n \geq m$. As (5) is equivalent to $f(m) > k$, our claim (6) then follows by induction on n.

For all $n \geq m$, we have

$$\begin{aligned}
& f^{\Delta}(n) \geq 0 \\
\iff\; & (n+1)^{\widehat{a}-\breve{a}} \cdot \left(\frac{\widehat{b}}{\breve{b}}\right)^{n+1} - n^{\widehat{a}-\breve{a}} \cdot \left(\frac{\widehat{b}}{\breve{b}}\right)^n \geq 0 && \text{(by definition of } f^{\Delta}) \\
\iff\; & \left(\left(\tfrac{n+1}{n}\right)^{\widehat{a}-\breve{a}} \cdot \tfrac{\widehat{b}}{\breve{b}} - 1\right) \cdot n^{\widehat{a}-\breve{a}} \cdot \left(\frac{\widehat{b}}{\breve{b}}\right)^n \geq 0 \\
\iff\; & \left(\tfrac{n+1}{n}\right)^{\widehat{a}-\breve{a}} \cdot \tfrac{\widehat{b}}{\breve{b}} - 1 \geq 0 && \left(\text{as } n^{\widehat{a}-\breve{a}} \cdot \left(\tfrac{\widehat{b}}{\breve{b}}\right)^n > 0 \text{ for all } n \geq m > 0\right) \\
\iff\; & \left(\tfrac{n}{n+1}\right)^{\breve{a}-\widehat{a}} \cdot \tfrac{\widehat{b}}{\breve{b}} - 1 \geq 0 \\
\iff\; & \left(\tfrac{n}{n+1}\right)^{\breve{a}-\widehat{a}} \cdot \tfrac{\widehat{b}}{\breve{b}} \geq 1 \\
\Longleftarrow\; & \left(\tfrac{m}{m+1}\right)^{\breve{a}-\widehat{a}} \cdot \tfrac{\widehat{b}}{\breve{b}} \geq 1 && \left(\text{as } n \geq m \text{ implies } \left(\tfrac{n}{n+1}\right)^{\breve{a}-\widehat{a}} \cdot \tfrac{\widehat{b}}{\breve{b}} \geq \left(\tfrac{m}{m+1}\right)^{\breve{a}-\widehat{a}} \cdot \tfrac{\widehat{b}}{\breve{b}}\right) \\
\iff\; & (4) && (\text{as } \widehat{a} < \breve{a})
\end{aligned}$$

□

Lemma 13 shows that Lemma 12 allows us to over-approximate the stabilization threshold of pe on $\vec{e}$. This finishes the proof of Thm. 6.

Lemma 13 (Over-Approximating Stabilization Thresholds)**.** *For any $pe \in \mathbb{NPE}[\vec{x}]$ and any $\vec{e} \in (\mathbb{R}_{\mathbb{A}})^d$, one can compute an $m \in \mathbb{N}$ with $m \geq sth_{pe}(\vec{e})$.*

[5]We use f^{Δ} instead of f's first derivative, since we consider monotonicity w.r.t. $n \in \mathbb{N}$ (as opposed to $n \in \mathbb{R}$).

Proof. Let $pe[\vec{x}/\vec{e}] = \sum_{j=1}^{\ell} k_j \cdot n^{a_j} \cdot b_j^n$ with $k_j \neq 0$ for all $1 \leq j \leq \ell$, and $(b_\ell, a_\ell) >_{lex} \ldots >_{lex} (b_1, a_1)$. If $\ell = 1$, then we trivially have $\operatorname{sign}(pe[\vec{x}/\vec{e}]) = \operatorname{sign}(k_\ell)$ for all $n \geq 1$, which implies $1 \geq sth_{pe}(\vec{e})$.

If $\ell \geq 2$, then for each addend $k_j \cdot n^{a_j} \cdot b_j^n$ with $1 \leq j < \ell - 1$, we can compute the $\frac{|k_j|}{|k_{\ell-1}|}$-monotonicity threshold mt_j of $(b_{\ell-1}, a_{\ell-1})$ and (b_j, a_j) by Lemma 12. Thus, $|k_{\ell-1}| \cdot n^{a_{\ell-1}} \cdot b_{\ell-1}^n > |k_j| \cdot n^{a_j} \cdot b_j^n$ holds for all $n \geq mt_j$. Let $mt = \max\{mt_j \mid 1 \leq j < \ell - 1\}$ (where we use the convention $\max \varnothing = 0$, i.e., we have $mt = 0$ if $\ell = 2$). Then for all $n \geq mt$ we obtain

$$\sum_{j=1}^{\ell-1} k_j \cdot n^{a_j} \cdot b_j^n \leq \sum_{j=1}^{\ell-1} |k_j| \cdot n^{a_j} \cdot b_j^n \leq \sum_{i=1}^{\ell-1} |k_{\ell-1}| \cdot n^{a_{\ell-1}} \cdot b_{\ell-1}^n = (\ell - 1) \cdot |k_{\ell-1}| \cdot n^{a_{\ell-1}} \cdot b_{\ell-1}^n.$$

Moreover, by Lemma 12 we can also compute the $\frac{(\ell-1)\cdot|k_{\ell-1}|}{|k_\ell|}$-monotonicity threshold mt' of (b_ℓ, a_ℓ) and $(b_{\ell-1}, a_{\ell-1})$. Hence, $|k_\ell| \cdot n^{a_\ell} \cdot b_\ell^n > (\ell - 1) \cdot |k_{\ell-1}| \cdot n^{a_{\ell-1}} \cdot b_{\ell-1}^n$ holds for all $n \geq mt'$. Let $m = \max\{mt, mt'\}$. Then for all $n \geq m$ we have

$$\sum_{j=1}^{\ell-1} k_j \cdot n^{a_j} \cdot b_j^n \leq (\ell - 1) \cdot |k_{\ell-1}| \cdot n^{a_{\ell-1}} \cdot b_{\ell-1}^n < |k_\ell| \cdot n^{a_\ell} \cdot b_\ell^n,$$

i.e., for all $n \geq m$ we have $\operatorname{sign}(pe[\vec{x}/\vec{e}]) = \operatorname{sign}(k_\ell)$, which implies $m \geq sth_{pe}(\vec{e})$. □

Example 14. *We show how to decide the halting problem for the loop $\mathcal{L}$ from Ex. 5 on the input $x_1 = 11$ and $x_2 = 4$. By Ex. 5, we have $\varphi(\vec{q}) \equiv x_1^{(n)} > 0$, i.e., $sth_{(\varphi,\vec{u})}(11, 4) \leq sth_{x_1^{(n)}}(11, 4)$. So our goal is to compute an upper bound on $sth_{x_1^{(n)}}(11, 4)$. We have*

$$x_1^{(n)}[x_1/11, x_2/4] = \left(\tfrac{1}{2} \cdot n^2 + \left(x_2 - \tfrac{1}{2}\right) \cdot n + x_1\right)[x_1/11, x_2/4] = \tfrac{1}{2} \cdot n^2 + \tfrac{7}{2} \cdot n + 11.$$

Thus, $\ell = 3$, $k_1 = 11$, $k_2 = \frac{7}{2}$, and $k_3 = \frac{1}{2}$. As in the proof of Lemma 13, we first compute the $\frac{|k_1|}{|k_2|}$-monotonicity threshold, i.e., the $\frac{22}{7}$-monotonicity threshold of $(1, 1)$ and $(1, 0)$, which is $mt = 4$. Next, we compute the $\frac{(\ell-1)\cdot|k_2|}{|k_3|}$-monotonicity threshold, i.e., the 14-monotonicity threshold of $(1, 2)$ and $(1, 1)$, which is $mt' = 15$, cf. Ex. 11. Hence, $\max\{mt, mt'\} = 15$ is an upper bound on $sth_{x_1^{(n)}}(11, 4)$. Thus, by verifying $\bigwedge_{n=0}^{15} \left(\frac{1}{2} \cdot n^2 + \frac{7}{2} \cdot n + 11 > 0\right)$, we conclude that $(11, 4)$ witnesses non-termination of $\mathcal{L}$.

Alg. 1 summarizes our technique to decide the halting problem for *twn*-loops. If necessary, the loop is chained in Line 1 to enforce Simplification 1. Then n_0 is initialized to 0 and afterwards, we process each inequation $pe \triangleright 0$ in $\varphi(\vec{q})$ separately, cf. Simplification 2. Line 5 implements Simplification 3. Then n_0 is set to an upper bound on $sth_{pe_{norm}}(\vec{e})$, for all inequations $pe_{norm} \triangleright 0$, cf. (2). Finally, in Line 8 and 9 we ensure that n_0 is large enough to justify normalizing pe to pe_{norm}, cf. (3).

Our results on the decidability of the halting problem can also be extended to certain loops that are not in *twn*-form by using our techniques from [19, Sect. 3], where we presented a transformation of loops which is based on $\mathbb{R}_{\mathbb{A}}$-*automorphisms*. An $\mathbb{R}_{\mathbb{A}}$-*endomorphism* of $\mathbb{R}_{\mathbb{A}}[\vec{x}]$ is a mapping $\eta : \mathbb{R}_{\mathbb{A}}[\vec{x}] \to \mathbb{R}_{\mathbb{A}}[\vec{x}]$ which is $\mathbb{R}_{\mathbb{A}}$-linear and multiplicative.[6] As usual, an $\mathbb{R}_{\mathbb{A}}$-automorphism is an *invertible* $\mathbb{R}_{\mathbb{A}}$-endomorphism η, i.e., there exists an $\mathbb{R}_{\mathbb{A}}$-endomorphism η^{-1} such that $\eta \circ \eta^{-1} = \eta^{-1} \circ \eta$ is the identity function on $\mathbb{R}_{\mathbb{A}}[\vec{x}]$. Intuitively, the transformation substitutes the variables in the program by polynomials such that the substitution can be inverted by another substitution of variables by polynomials.

[6]So we have $\eta(c \cdot pol + c' \cdot pol') = c \cdot \eta(pol) + c' \cdot \eta(pol')$, $\eta(1) = 1$, and $\eta(pol \cdot pol') = \eta(pol) \cdot \eta(pol')$ for all $c, c' \in \mathbb{R}_{\mathbb{A}}$ and all $pol, pol' \in \mathbb{R}_{\mathbb{A}}[\vec{x}]$.

Algorithm 1: Deciding the Halting Problem

```
    Input: a $twn$-loop $(\varphi, \vec{u})$ and $\vec{e} \in \mathbb{R}_{\mathbb{A}}^d$
    Result: $\top$ if $(\varphi, \vec{u})$ terminates on $\vec{e}$, $\bot$ otherwise
 1  if $(\varphi, \vec{u})$ is not $tnn$ then $(\varphi, \vec{u}) \leftarrow (\varphi \wedge \varphi(\vec{u}), \vec{u}(\vec{u}))$
 2  $\vec{q} \leftarrow$ closed form of $\vec{u}^n$ with $\vec{q} \in (\mathbb{PE}[\vec{x}])^d$
 3  $n_0 \leftarrow 0$
 4  foreach inequation $pe \rhd 0$ occurring in $\varphi(\vec{q})$ do
 5      $pe \leftarrow pe_{norm}$
 6      compute $m \geq sth_{pe}(\vec{e})$ as in the proofs of Lemma 12 and Lemma 13
 7      $n_0 \leftarrow \max\{n_0, m\}$
 8  $c \leftarrow 1 +$ the maximal constant that occurs in any factor $[\![\psi]\!]$ in $\varphi(\vec{q})$
 9  $n_0 \leftarrow \max\{n_0, c\}$
10  if $\varphi(\vec{q}[\vec{x}/\vec{e}])$ holds for all $0 \leq n \leq n_0$ then return $\bot$ else return $\top$
```

This transformation allows for transforming certain non-*twn*-loops into *twn*-form. Moreover, in [19, Cor. 15] we showed that this transformation induces a bijection $\widehat{\eta}$ on the sets of non-terminating inputs. Here, $\widehat{\eta} : \mathcal{S}^d \to \mathcal{S}^d$ maps $\vec{e}$ to $\eta(\vec{x})[\vec{x}/\vec{e}]$. Thus, given a non-*twn* loop $(\varphi, \vec{u})$, an input $\vec{e} \in \mathbb{R}_{\mathbb{A}}^d$, and a polynomial automorphism η which transforms $(\varphi, \vec{u})$ into a *twn*-loop, we can proceed as follows to decide whether $(\varphi, \vec{u})$ terminates on $\vec{e}$:

- First, we transform the loop $(\varphi, \vec{u})$ into the *twn*-loop $(\varphi', \vec{u}')$ by using η.
- Then we transform the input $\vec{e}$ into $\vec{e}\,' = \widehat{\eta}(\vec{e})$.
- Finally, we apply Alg. 1 to $(\varphi', \vec{u}')$ and $\vec{e}\,'$.

So we can not only decide the halting problem for all *twn*-loops over $\mathbb{R}_{\mathbb{A}}$, but we can decide it for all *twn-transformable* loops which can be transformed into *twn*-form by an $\mathbb{R}_{\mathbb{A}}$-automorphism. The reason is that whenever a loop is *twn*-transformable, then an automorphism which transforms it into *twn*-form is computable [19].

4 Runtime Bounds

In the following, we restrict ourselves to *twn*-loops over the *integers* and show how to obtain upper bounds on their *runtime*. Def. 15 introduces this notion formally, which is only defined for inputs where the loop terminates.

Definition 15 (Runtime). *For a loop $(\varphi, \vec{u})$ over $\mathbb{Z}$, the* runtime $rt_{(\varphi,\vec{u})} : T_{(\varphi,\vec{u})} \to \mathbb{N}$ *for $\vec{e} \in \mathbb{Z}^d$ is*

$$rt_{(\varphi,\vec{u})}(\vec{e}) = \min\{n \in \mathbb{N} \mid \neg(\varphi(\vec{u}^n(\vec{e})))\}.$$

So $(\varphi, \vec{u})$ terminates on $\vec{e} \in T_{(\varphi,\vec{u})}$ after $rt_{(\varphi,\vec{u})}(\vec{e})$ iterations. In practice, it is usually infeasible to compute the runtime exactly, so that state-of-the-art complexity analyzers rely on approximations, cf. e.g., [2, 3, 12, 13, 16, 17, 23, 41, 42]. In this spirit, we will prove that the runtime of a *twn*-loop on $\mathbb{Z}^d$ is bounded (from above) by a polynomial in the 1-norm[7]

[7]Measuring vectors of integers via the 1-norm is standard in *automated complexity analysis* where one is interested in analyzing the efficiency of *implementations*. This is orthogonal to *complexity theory*, where the size of numbers is measured logarithmically and the goal is to characterize decision problems by proving membership or hardness w.r.t. complexity classes like P or NP.

$\|\vec{x}\| = \sum_{j=1}^{d} |x_j|$ of $\vec{x}$. Moreover, this polynomial is *computable*. This constitutes the first computability result for the runtime complexity of non-linear programs that we are aware of.

Theorem 16 (Polynomial Loops Have Polynomial Runtime)**.** *Let $(\varphi, \vec{u})$ be a tnn-loop over $\mathbb{Z}$ and let $\vec{q} \in (\mathbb{PE}[\vec{x}])^d$ be the closed form of $\vec{u}^n$. Then there is a polynomial $f \in \mathbb{N}[y]$ such that*

$$rt_{(\varphi,\vec{u})}(\vec{e}) \leq f(\|\vec{e}\|) \qquad \textit{for all } \vec{e} \in T_{(\varphi,\vec{u})},$$

where $\deg(f)$ is bounded by the maximal degree of the coefficients in $\varphi(\vec{q})$.

Here, recall that the coefficients of the poly-exponential expressions in $\varphi(\vec{q})$ are polynomials from $Q_{\mathcal{S}}[\vec{x}]$. Note that Thm. 16 does not hold for loops over $\mathbb{Q}$ or $\mathbb{R}_{\mathbb{A}}$, as demonstrated by the following example.

Example 17. *Consider the loop $(x_1 \geq 0 \wedge x_2 > 0, (x_1 - x_2, x_2))$ over $\mathbb{Q}$ or $\mathbb{R}_{\mathbb{A}}$. It terminates after $\max\{0, \lceil \frac{x_1}{x_2} \rceil + 1\}$ iterations, but its runtime is not bounded by any continuous function $f : \mathbb{R} \to \mathbb{R}$ in $\|(x_1, x_2)\|$. To see this, for any $k \in \mathbb{N}$ let $\vec{e}_k = (1, \frac{1}{k})$ and let $f : \mathbb{R} \to \mathbb{R}$ be continuous. Then $\lim_{k \mapsto \infty} \|\vec{e}_k\| = \lim_{k \mapsto \infty} (|1| + |\frac{1}{k}|) = 1$ and thus, by continuity, $\lim_{k \mapsto \infty} f(\|\vec{e}_k\|) = f(1) \in \mathbb{R}$. However, for any $k \in \mathbb{N}$ we have*

$$rt_{(x_1 \geq 0 \wedge x_2 > 0, (x_1 - x_2, x_2))}(\vec{e}_k) = \max\{0, \lceil k \rceil + 1\} = k + 1.$$

Hence,

$$\lim_{k \mapsto \infty} rt_{(x_1 \geq 0 \wedge x_2 > 0, (x_1 - x_2, x_2))}(\vec{e}_k) = \lim_{k \mapsto \infty} (k+1) = \infty.$$

Thus, we have

$$rt_{(x_1 \geq 0 \wedge x_2 > 0, (x_1 - x_2, x_2))}(\vec{e}_k) > f(\|\vec{e}_k\|)$$

for large enough k. Hence, there is no continuous function $f : \mathbb{R} \to \mathbb{R}$ with

$$rt_{(x_1 \geq 0 \wedge x_2 > 0, (x_1 - x_2, x_2))}(\vec{e}_k) = k + 1 \leq f(\|\vec{e}_k\|)$$

for all $k \in \mathbb{N}$. Essentially, the problem is that measuring the "size" of non-integer inputs via the 1-norm $\|\cdot\|$ is not suitable for analyzing runtime complexity. The same problem arises with any other continuous function $\|\cdot\|' : \mathbb{R}^d \to \mathbb{R}$, e.g., any norm on $\mathbb{R}^d$, because then we have $\lim_{k \mapsto \infty} \|\vec{e}_k\|' = \|\lim_{k \mapsto \infty} (1, \frac{1}{k})\|' = \|(1, 0)\|'$ and thus, for any continuous function f, $\lim_{k \mapsto \infty} f(\|\vec{e}_k\|')$ is a constant from $\mathbb{R}$. For that reason, we restricted ourselves to loops over $\mathbb{Z}$ in this section.

While Thm. 16 only applies to *tnn*-loops, it can also be used to obtain a bound on the complexity of other *twn*-loops due to the following lemma.

Lemma 18 (Chaining Preserves Asymptotic Runtime)**.** *Let $(\varphi, \vec{u})$ be a loop over $\mathbb{Z}$ and let $(\varphi_{\text{ch}}, \vec{u}_{\text{ch}}) = (\varphi \wedge \varphi(\vec{u}), \vec{u}(\vec{u}))$ be the corresponding chained loop. We have*

$$2 \cdot rt_{(\varphi_{\text{ch}},\vec{u}_{\text{ch}})}(\vec{e}) \;\leq\; rt_{(\varphi,\vec{u})}(\vec{e}) \;\leq\; 2 \cdot rt_{(\varphi_{\text{ch}},\vec{u}_{\text{ch}})}(\vec{e}) + 1 \qquad \textit{for all } \vec{e} \in \mathbb{Z}^d.$$

Proof. Let $rt_{(\varphi,\vec{u})}(\vec{e}) = n_{\vec{e}}$ and $rt_{(\varphi_{\text{ch}},\vec{u}_{\text{ch}})}(\vec{e}) = n'_{\vec{e}}$. Then we have

$$n'_{\vec{e}} = \min\{n \in \mathbb{N} \mid \neg(\varphi_{\text{ch}}(\vec{u}_{\text{ch}}^{n}(\vec{e})))\} \qquad \text{(by definition of } rt_{(\varphi_{\text{ch}},\vec{u}_{\text{ch}})})$$

$$\iff \forall n < n'_{\vec{e}}.\ \varphi_{\text{ch}}(\vec{u}_{\text{ch}}^{n}(\vec{e})) \wedge \neg\varphi_{\text{ch}}(\vec{u}_{\text{ch}}^{n'_{\vec{e}}}(\vec{e}))$$

$$\iff \forall n < n'_{\vec{e}}.\ \left(\varphi(\vec{u}^{2\cdot n}(\vec{e})) \wedge \varphi(\vec{u}(\vec{u}^{2\cdot n}(\vec{e})))\right) \wedge \neg\left(\varphi(\vec{u}^{2\cdot n'_{\vec{e}}}(\vec{e})) \wedge \varphi(\vec{u}(\vec{u}^{2\cdot n'_{\vec{e}}}(\vec{e})))\right)$$

$$\text{(by definition of } \varphi_{\text{ch}} \text{ and } \vec{u}_{\text{ch}})$$

$$\iff \forall n < n'_{\vec{e}}.\ \left(\varphi(\vec{u}^{2\cdot n}(\vec{e})) \wedge \varphi(\vec{u}^{2\cdot n+1}(\vec{e}))\right) \wedge \neg\left(\varphi(\vec{u}^{2\cdot n'_{\vec{e}}}(\vec{e})) \wedge \varphi(\vec{u}^{2\cdot n'_{\vec{e}}+1}(\vec{e}))\right)$$

$$\iff \forall m < 2 \cdot n'_{\vec{e}}.\ \varphi(\vec{u}^{m}(\vec{e})) \wedge \left(\neg\varphi(\vec{u}^{2\cdot n'_{\vec{e}}}(\vec{e})) \vee \neg\varphi(\vec{u}^{2\cdot n'_{\vec{e}}+1}(\vec{e}))\right)$$

$$\iff 2 \cdot n'_{\vec{e}} \;\le\; rt_{(\varphi,\vec{u})}(\vec{e}) \;\le\; 2 \cdot n'_{\vec{e}} + 1.$$

□

From Thm. 16 and Lemma 18, we get the following corollary.

Corollary 19 (Linear Loops Have Linear Runtime)**.** *Let $(\varphi, \vec{u})$ be a triangular* linear *loop over $\mathbb{Z}$. Then there is a polynomial $f \in \mathbb{N}[y]$ with* $\deg(f) = 1$ *such that*

$$rt_{(\varphi,\vec{u})}(\vec{e}) \le f(\|\vec{e}\|) \qquad \textit{for all } \vec{e} \in T_{(\varphi,\vec{u})}.$$

The reason is that chaining preserves linearity, i.e., if $(\varphi, \vec{u})$ is linear, then $(\varphi_{\text{ch}}, \vec{u}_{\text{ch}})$ is linear, too. Hence, all coefficients of the closed form $\vec{q}$ of $\vec{u}^n_{\text{ch}}$ are also linear [18].

We now prove Thm. 16. As we clearly have

$$rt_{(\varphi,\vec{u})}(\vec{e}) \le sth_{(\varphi,\vec{u})}(\vec{e}) \qquad \text{for all } \vec{e} \in T_{(\varphi,\vec{u})},$$

to derive an upper bound on $rt_{(\varphi,\vec{u})}$, it suffices to find an upper bound on $sth_{(\varphi,\vec{u})}$. Moreover, due to (2), we may restrict ourselves to loops of the form $(pol \rhd 0, \vec{u})$ without loss of generality. More precisely, if $pol_1 \rhd_1 0, \ldots, pol_r \rhd_r 0$ are all inequations occurring in φ, then we compute a polynomial upper bound $f_i \in \mathbb{N}[y]$ on $sth_{(pol_i \rhd_i 0, \vec{u})}$ for each $1 \le i \le r$. Then $f_1 + \ldots + f_r$ is an upper bound on $sth_{(\varphi,\vec{u})}$, since $f_1(y) + \ldots + f_r(y) \ge \max\{f_1(y), \ldots, f_r(y)\}$ for all $y \in \mathbb{N}$.

As in Sect. 3, let $\vec{q} \in (\mathbb{PE}[\vec{x}])^d$ be the closed form of $\vec{u}^n$ and let $pe = pol(\vec{q})$. Moreover, let $c \in \mathbb{N}$ again be larger than any constant that occurs in a factor $[\![\psi]\!]$ in pe and let $pe_{norm} \in \mathbb{NPE}[\vec{x}]$ be defined as in Sect. 3. As in (3), we therefore have

$$rt_{(pol \rhd 0,\vec{u})}(\vec{e}) \le sth_{pe}(\vec{e}) \le \max\{sth_{pe_{norm}}(\vec{e}), c\}.$$

Note that the degrees of the coefficients of pe_{norm} are bounded by the degrees of the coefficients of pe. Thus, to prove Thm. 16, it suffices to find a polynomial bound f whose degree is bounded by the maximal degree of the coefficients in pe_{norm}. So from now on, we assume $pe \in \mathbb{NPE}[\vec{x}]$.

Furthermore, w.l.o.g. we may assume that

$$pe = \sum_{j=1}^{\ell} \alpha_j \cdot n^{a_j} \cdot b_j^n \qquad \text{where } \alpha_j \in \mathbb{Z}[\vec{x}] \text{ (instead of } \alpha_j \in \mathbb{Q}[\vec{x}] = Q_{\mathbb{Z}}[\vec{x}]). \quad \text{(Simplification 4)}$$

The reason is that by multiplying with the denominators of all fractions that occur in some α_j, we obtain a $pe' \in \mathbb{NPE}[\vec{x}]$ with $sth_{pe} = sth_{pe'}$ where the coefficients of pe' are elements of $\mathbb{Z}[\vec{x}]$.

Example 20. *Now the loop $\mathcal{L}$ from Ex. 5 is considered over $\mathbb{Z}$, where $\varphi(\vec{q}) \equiv x_1^{(n)} > 0$ and*

$$x_1^{(n)} = \tfrac{1}{2} \cdot n^2 + \left(x_2 - \tfrac{1}{2}\right) \cdot n + x_1.$$

To compute a bound on $sth_{x_1^{(n)}}$, we can regard the following expression over $\mathbb{Z}$ instead:

$$n^2 + (2 \cdot x_2 - 1) \cdot n + 2 \cdot x_1$$

The following lemma allows us to compute a bound on the stabilization threshold $sth_{pe}(\vec{e})$. More precisely, the lemma shows that $sth_{pe}(\vec{e})$ can be over-approximated by a polynomial in the 1-norm of $\vec{e} \in \mathbb{Z}^d$, provided that $\alpha_\ell(\vec{e}) \neq 0$. Here, we (again) use the convention $\max \varnothing = 0$.

Lemma 21 (Bound on Stabilization Thresholds)**.** *Let $pe \in \mathbb{NPE}[\vec{x}]$ with $pe = \sum_{j=1}^{\ell} \alpha_j \cdot n^{a_j} \cdot b_j^n$ and $(b_\ell, a_\ell) >_{lex} \ldots >_{lex} (b_1, a_1)$, where $\alpha_j \in \mathbb{Z}[\vec{x}]$ for all $1 \le j \le \ell$. Then there is a polynomial $f \in \mathbb{N}[y]$ with* $\deg(f) = \max\{\deg(\alpha_j) \mid 1 \le j < \ell\}$ *such that*

$$sth_{pe}(\vec{e}) \le f(\|\vec{e}\|) \qquad \textit{for all } \vec{e} \in \mathbb{Z}^d \textit{ with } \alpha_\ell(\vec{e}) \neq 0.$$

Proof. If $\ell = 1$, then the claim is trivial by choosing $f(y) = 1$. The reason is that for each $\vec{e}$ we have $pe(\vec{e}) = \alpha_1(\vec{e}) \cdot n^{a_1} \cdot b_1^n$, whose sign is the sign of $\alpha_1(\vec{e}) \neq 0$ for every $n \in \mathbb{N}$ with $n \ge 1$.

Otherwise, let

$$\begin{aligned} m &= \max\{\deg(\alpha_j) \mid 1 \leq j < \ell\}, \\ k_{\max} &= \max\{|k| \mid 1 \leq j < \ell,\ k \text{ is a coefficient of } \alpha_j\}, \text{ and} \\ \alpha_{\max}(\vec{x}) &= k_{\max} \cdot \textstyle\sum_{i=0}^{m} \|\vec{x}\|^i. \end{aligned}$$

Then for all $1 \leq j < \ell$ and all $\vec{e} \in \mathbb{Z}^d$ we have

$$\alpha_j(\vec{e}) \leq \max\{|k| \mid k \text{ is a coefficient of } \alpha_j\} \cdot \sum_{i=0}^{\deg(\alpha_j)} \|\vec{e}\|^i \leq \alpha_{\max}(\vec{e}).$$

If $\ell \geq 3$, then let

$$mt = \max\{1\text{-monotonicity threshold of } (b_{\ell-2}, a_{\ell-2}) \text{ and } (b_j, a_j) \mid 1 \leq j < \ell - 2\},$$

let mt' be the $(\ell-2)$-monotonicity threshold of $(b_{\ell-1}, a_{\ell-1})$ and $(b_{\ell-2}, a_{\ell-2})$, and let $N = \max\{mt, mt'\}$. Then

$$\sum_{j=1}^{\ell-2} n^{a_j} \cdot b_j^n \leq \sum_{j=1}^{\ell-2} n^{a_{\ell-2}} \cdot b_{\ell-2}^n = (\ell-2) \cdot n^{a_{\ell-2}} \cdot b_{\ell-2}^n < n^{a_{\ell-1}} \cdot b_{\ell-1}^n$$

holds for all $n \geq N$. If $\ell = 2$, then we define $N = 1$. Thus, we get:

$$\begin{aligned} \sum_{j=1}^{\ell-1} \alpha_j(\vec{e}) \cdot n^{a_j} \cdot b_j^n &\leq \sum_{j=1}^{\ell-1} \alpha_{\max}(\vec{e}) \cdot n^{a_j} \cdot b_j^n \\ &= \alpha_{\max}(\vec{e}) \cdot \sum_{j=1}^{\ell-1} n^{a_j} \cdot b_j^n \\ &= \alpha_{\max}(\vec{e}) \cdot \left(\sum_{j=1}^{\ell-2} n^{a_j} \cdot b_j^n + n^{a_{\ell-1}} \cdot b_{\ell-1}^n \right) \\ &< 2 \cdot \alpha_{\max}(\vec{e}) \cdot n^{a_{\ell-1}} \cdot b_{\ell-1}^n \qquad \text{for all } n \geq N \end{aligned}$$

We proceed by a case analysis.

Case $b_\ell = b_{\ell-1}$**:** As $(b_\ell, a_\ell) >_{lex} (b_{\ell-1}, a_{\ell-1})$ we have $a_\ell > a_{\ell-1}$ and hence

$$\begin{aligned} 2 \cdot \alpha_{\max}(\vec{e}) \cdot n^{a_{\ell-1}} \cdot b_{\ell-1}^n &= 2 \cdot \alpha_{\max}(\vec{e}) \cdot n^{a_{\ell-1}} \cdot b_\ell^n \\ &\leq n^{a_{\ell-1}+1} \cdot b_\ell^n \qquad \text{for all } n \geq 2 \cdot \alpha_{\max}(\vec{e}) \\ &\leq n^{a_\ell} \cdot b_\ell^n. \end{aligned}$$

Thus, we obtain

$$\sum_{j=1}^{\ell-1} \alpha_j(\vec{e}) \cdot n^{a_j} \cdot b_j^n < n^{a_\ell} \cdot b_\ell^n \qquad \text{for all } n \geq \max\{N, 2 \cdot \alpha_{\max}(\vec{e})\}. \tag{7}$$

Case $b_\ell > b_{\ell-1}$**:** Then $(b_\ell, a_\ell) >_{lex} (b_{\ell-1}, a_{\ell-1}+1)$. Let $M \in \mathbb{N}$ be the 1-monotonicity threshold of (b_ℓ, a_ℓ) and $(b_{\ell-1}, a_{\ell-1}+1)$. Then we have

$$2 \cdot \alpha_{\max}(\vec{e}) \cdot n^{a_{\ell-1}} \cdot b_{\ell-1}^n \leq n^{a_{\ell-1}+1} \cdot b_{\ell-1}^n < n^{a_\ell} \cdot b_\ell^n$$

for all $n \geq \max\{M, 2 \cdot \alpha_{\max}(\vec{e})\}$ and thus we obtain

$$\sum_{j=1}^{\ell-1} \alpha_j(\vec{e}) \cdot n^{a_j} \cdot b_j^n < n^{a_\ell} \cdot b_\ell^n \qquad \text{for all } n \geq \max\{N, M, 2 \cdot \alpha_{\max}(\vec{e})\}. \tag{8}$$

From (7) and (8), we get

$$\forall \vec{e} \in \mathbb{Z}^d, n \geq \max\{N, M, 2 \cdot \alpha_{\max}(\vec{e})\}. \quad \sum_{j=1}^{\ell-1} \alpha_j(\vec{e}) \cdot n^{a_j} \cdot b_j^n < n^{a_\ell} \cdot b_\ell^n$$

(where we define $M = 0$ if $b_\ell = b_{\ell-1}$).

Recall that we restricted ourselves to loops over $\mathbb{Z}$ (i.e., we have $\vec{e} \in \mathbb{Z}^d$) and to poly-exponential expressions where the polynomials α_j only have integer coefficients (cf. Simplification 4). Thus, if $\alpha_\ell(\vec{e}) > 0$, then $\alpha_\ell \in \mathbb{Z}[\vec{x}]$ and $e \in \mathbb{Z}^d$ imply $\alpha_\ell(\vec{e}) \geq 1$. Hence, if $n \geq \max\{N, M, 2 \cdot \alpha_{\max}(\vec{e})\}$, then $pe[\vec{x}/\vec{e}] > (\alpha_\ell(\vec{e}) - 1) \cdot n^{a_\ell} \cdot b_\ell^n \geq 0$ and hence, $\operatorname{sign}(pe[\vec{x}/\vec{e}]) = 1$. Similarly, if $\alpha_\ell(\vec{e}) < 0$, then $\alpha_\ell \in \mathbb{Z}[\vec{x}]$ and $e \in \mathbb{Z}^d$ imply $\alpha_\ell(\vec{e}) \leq -1$. So if $n \geq \max\{N, M, 2 \cdot \alpha_{\max}(\vec{e})\}$, then $pe[\vec{x}/\vec{e}] < (\alpha_\ell(\vec{e}) + 1) \cdot n^{a_\ell} \cdot b_\ell^n \leq 0$ and hence, $\operatorname{sign}(pe[\vec{x}/\vec{e}]) = -1$. This implies

$$sth_{pe}(\vec{e}) \leq \max\{N, M, 2 \cdot \alpha_{\max}(\vec{e})\} \qquad \text{for all } \vec{e} \in \mathbb{Z}^d \text{ with } \alpha_\ell(\vec{e}) \neq 0. \tag{9}$$

It remains to show that there is a polynomial $f \in \mathbb{N}[y]$ with $\deg(f) = m$ such that $f(\|\vec{e}\|) \geq \max\{N, M, 2 \cdot \alpha_{\max}(\vec{e})\}$ for all $\vec{e} \in \mathbb{Z}^d$. We define

$$f(y) = 2 \cdot k_{\max} \cdot \sum_{i=0}^{m} y^i + \max\{N, M\},$$

where we have $f \in \mathbb{N}[y]$ as $\alpha_j \in \mathbb{Z}[\vec{x}]$ implies $k_{\max} \in \mathbb{N}$. Then for any $\vec{e} \in \mathbb{Z}^d$ we get

$$\begin{aligned} f(\|\vec{e}\|) &= 2 \cdot k_{\max} \cdot \sum_{i=0}^{m} \|\vec{e}\|^i + \max\{N, M\} \\ &= 2 \cdot \alpha_{\max}(\vec{e}) + \max\{N, M\} && \text{(by definition of } \alpha_{\max}) \\ &\geq \max\{N, M, 2 \cdot \alpha_{\max}(\vec{e})\}, \end{aligned}$$

as desired. □

Example 22. *Reconsider the loop $\mathcal{L}$ from Ex. 5 over $\mathbb{Z}$. To compute a bound on its runtime, we construct a polynomial upper bound f on sth_{pe} where*

$$pe = n^2 + (2 \cdot x_2 - 1) \cdot n + 2 \cdot x_1,$$

cf. Ex. 20. Following the proof of Lemma 21, we have $\ell = 3$, $\alpha_1 = 2 \cdot x_1$, $\alpha_2 = 2 \cdot x_2 - 1$, and $\alpha_3 = 1$, i.e., $m = 1$ and $k_{\max} = 2$. Moreover, we have $mt = 0$ and $mt' = 2$ (as the 1-monotonicity threshold of $(1,1)$ and $(1,0)$ is 2). Thus, we get $N = \max\{mt, mt'\} = 2$. Furthermore, we have $M = 0$, as $b_2 = b_3 = 1$. Thus, we get

$$f(y) = 2 \cdot k_{\max} \cdot \sum_{i=0}^{m} y^i + \max\{N, M\} = 4 \cdot y + 6.$$

Note that $\alpha_\ell(\vec{e}) = \alpha_3(\vec{e}) = 1 \neq 0$ for all $\vec{e} \in \mathbb{Z}^2$. Thus, $sth_{pe}(e_1, e_2)$ is bounded by $f(\|(e_1, e_2)\|) = 4 \cdot (|e_1| + |e_2|) + 6$ for all $(e_1, e_2) \in \mathbb{Z}^2$. Consequently, the runtime of $\mathcal{L}$ is also bounded by $4 \cdot (|e_1| + |e_2|) + 6$ for all $(e_1, e_2) \in T_{\mathcal{L}}$. So for terminating inputs, $\mathcal{L}$'s runtime is at most linear.

Example 23. *The loop from Ex. 17 also shows that Lemma 21 does* not *hold for rings like $\mathbb{Q}$ or $\mathbb{R}_{\mathbb{A}}$. Here, the closed form of the n-fold update is $\vec{q} = (x_1 - n \cdot x_2, x_2)$. Thus, $\varphi(\vec{q})$ is $x_1 - n \cdot x_2 \geq 0 \land x_2 > 0$. When considering $pe = x_1 - n \cdot x_2$, then we have $sth_{pe}(\vec{e}_k) = k + 1$ for $\vec{e}_k = (1, \frac{1}{k})$. But as shown in Ex. 17, there is no continuous function $f : \mathbb{R} \to \mathbb{R}$ with $sth_{pe}(\vec{e}_k) = k + 1 \leq f(\|\vec{e}_k\|) = f(1 + \frac{1}{k})$ for all $k \in \mathbb{N}$.*

Indeed, the construction of f in the proof of Lemma 21 would no longer yield an upper bound on sth_{pe}. In our example, we have $\ell = 2$, $\alpha_1 = x_1$, and $\alpha_2 = -x_2$. Hence, $m = 1$ and $k_{\max} = 1$. Finally, since $\ell = 2$, we have $N = 1$ and since $b_1 = b_2 = 1$, we have $M = 0$. Thus, if we construct

f as in the proof of Lemma 21, then we get $f(y) = 2 \cdot k_{\max} \cdot \sum_{i=0}^{m} y^i + \max\{N, M\} = 2 \cdot y + 3$. So $f(\|\vec{e}_k\|) = 5 + \frac{2}{k}$, but $f(\|\vec{e}_5\|) = \frac{27}{5} < 6 = sth_{pe}(\vec{e}_5)$, i.e., f is not an upper bound on sth_{pe}.

According to the preconditions of Lemma 21, the resulting bound f is only valid for all $\vec{e} \in \mathbb{Z}^d$ with $\alpha_\ell(\vec{e}) \neq 0$. However, for every $\vec{e} \in \mathbb{Z}^d$, we either have $pe[\vec{x}/\vec{e}] = 0$ (and thus $sth_{pe}(\vec{e}) = 0$) or there is an $1 \leq \ell_{\vec{e}} \leq \ell$ such that $|\alpha_{\ell_{\vec{e}}}(\vec{e})| \geq 1$ and $\alpha_j(\vec{e}) = 0$ for all $\ell_{\vec{e}} < j \leq \ell$. So by applying Lemma 21 to $\sum_{j=1}^{\ell_{\vec{e}}} \alpha_j \cdot n^{a_j} \cdot b_j^n$, we obtain a polynomial $f_{\ell_{\vec{e}}}$ such that $sth_{pe}(\vec{e}) \leq f_{\ell_{\vec{e}}}(\|\vec{e}\|)$. Thus, by applying Lemma 21 to each expression $\sum_{j=1}^{\ell'} \alpha_j \cdot n^{a_j} \cdot b_j^n$ with $1 \leq \ell' \leq \ell$, we obtain polynomials $f_1, \ldots, f_\ell$ such that for each $\vec{e} \in \mathbb{Z}^d$, there is *some* f_j with $sth_{pe}(\vec{e}) \leq f_j(\|\vec{e}\|)$. Hence, for $f = \sum_{j=1}^{\ell} f_j$ we obtain $sth_{pe}(\vec{e}) \leq f(\|\vec{e}\|)$ since $f_j \in \mathbb{N}[y]$ for all $1 \leq j \leq \ell$. As the definition of f is independent of $\vec{e}$, this finishes the proof of Thm. 16.

Our technique to compute runtime bounds via Thm. 16 and Lemma 18 can be implemented as in Alg. 2. For Simplification 1, the loop is again chained in Line 1 if necessary. In that case, the difference between the complexity of the original and the chained loop is taken into account in Line 13, where the result is adapted according to Lemma 18. Next, f is initialized to the polynomial 1 and afterwards, we again process each inequation $pe \triangleright 0$ in $\varphi(\vec{q})$ separately, cf. Simplification 2. Line 5 implements Simplification 3 and Line 6 corresponds to Simplification 4. As explained after its proof, we have to apply Lemma 21 to each expression $\sum_{j=1}^{\ell'} \alpha_j \cdot n^{a_j} \cdot b_j^n$ with $1 \leq \ell' \leq \ell$ and the obtained bounds have to be added in order to get a bound on sth_{pe}. As in the proof of Lemma 21, we can skip poly-exponential expressions with just a single addend (i.e., where $\ell' = 1$). Thus, in Line 10, f is updated to the sum of its previous value and the upper bound on $\sum_{j=1}^{\ell'} \alpha_j \cdot n^{a_j} \cdot b_j^n$. By adding the bounds on sth_{pe} for all $pe \triangleright 0$ occurring in $\varphi(\vec{q})$, Line 10 also reflects the polynomial approximation of the max-expressions from (2) via a sum. Similarly, Line 12 approximates the max-expression from (3) via summation.

Algorithm 2: Computing Runtime Bounds

```
   **Input:** a $twn$-loop $(\varphi_{in}, \vec{u}_{in})$ on $\mathbb{Z}^d$
   **Output:** a polynomial $f \in \mathbb{N}[y]$ such that $rt_{(\varphi_{in},\vec{u}_{in})}(\vec{e}) \leq f(\|\vec{e}\|)$ for all $\vec{e} \in T_{(\varphi_{in},\vec{u}_{in})}$
1  **if** $(\varphi_{in}, \vec{u}_{in})$ *is tnn* **then** $(\varphi, \vec{u}) \leftarrow (\varphi_{in}, \vec{u}_{in})$ **else** $(\varphi, \vec{u}) \leftarrow (\varphi_{in} \land \varphi_{in}(\vec{u}_{in}), \vec{u}_{in}(\vec{u}_{in}))$
2  $\vec{q} \leftarrow$ closed form of $\vec{u}^n$ with $\vec{q} \in (\mathbb{PE}[\vec{x}])^d$
3  $f \leftarrow 1$
4  **foreach** *inequation* $pe \triangleright 0$ *occurring in* $\varphi(\vec{q})$ **do**
5      $pe \leftarrow pe_{norm}$
6      $pe \leftarrow pe \cdot \prod_{k \text{ occurs as denominator in } pe} k$
7      let $pe = \sum_{j=1}^{\ell} \alpha_j \cdot n^{a_j} \cdot b_j^n$ with $(b_\ell, a_\ell) >_{lex} \ldots >_{lex} (b_1, a_1)$
8      **foreach** $\ell'$ *with* $2 \leq \ell' \leq \ell$ **do**
9          compute $k_{\max}$, $N$, and $M$ for $\sum_{j=1}^{\ell'} \alpha_j \cdot n^{a_j} \cdot b_j^n$ as in the proof of Lemma 21
10         $f \leftarrow f + 2 \cdot k_{\max} \cdot \sum_{i=0}^{m} y^i + \max\{N, M\}$
11 $c \leftarrow 1 +$ the maximal constant that occurs in any factor $[\![\psi]\!]$ in $\varphi(\vec{q})$
12 $f \leftarrow f + c$
13 **if** $(\varphi_{in}, \vec{u}_{in})$ *is tnn* **then return** $f$ **else return** $2 \cdot f + 1$
```

When comparing Alg. 1 and Alg. 2, note that Alg. 1 computes an upper bound on $sth_{(\varphi,\vec{u})}(\vec{e})$ for a *given* input $\vec{e}$, whereas Alg. 2 computes an upper bound on the stabilization threshold $sth_{(\varphi,\vec{u})}$ *for all* inputs $\vec{e}$. More precisely, for twn-loops over $\mathbb{Z}$, Alg. 2 computes a polynomial f such that $sth_{(\varphi,\vec{u})}(\vec{e}) \leq f(\|\vec{e}\|)$ holds for all $\vec{e} \in \mathbb{Z}^d$. So for such loops, Alg. 2 could also be used to decide the halting problem, since $(\varphi, \vec{u})$ is non-terminating on $\vec{e}$ iff $\varphi(\vec{q}[\vec{x}/\vec{e}])$ holds for all $0 \leq n \leq f(\|\vec{e}\|)$. However, in contrast to Alg. 2, Alg. 1 can be used to decide the halting problem for loops on *arbitrary* rings $\mathbb{Z} \leq \mathcal{S} \leq \mathbb{R}_{\mathbb{A}}$. The reason is that Lemma 12 (which is the basis of Alg. 1) holds for $\mathbb{R}_{\mathbb{A}}$ (and therefore also for its subrings), whereas Lemma 21 (which is the basis of Alg. 2) only holds for the ring $\mathbb{Z}$.

As in Sect. 3, our program transformation from [19] allows us to extend our computability results for runtime bounds to certain loops that are not in *twn*-form. But while the results of Sect. 3 immediately carry over to any *twn-transformable* loop, this is not true for the results of the current section. The reason is that we now require that the coefficients of the poly-exponential expressions in $\varphi(\vec{q})$ are elements of $\mathbb{Q}[\vec{x}]$ (resp. $\mathbb{Z}[\vec{x}]$, cf. Simplification 4). This is crucial for the proof of Lemma 21.

So when applying an automorphism to a non-*twn* loop over the integers, we have to ensure that this requirement is still valid after the transformation. But when applying arbitrary $\mathbb{R}_{\mathbb{A}}$-automorphisms, this cannot be ensured, since the transformed program may contain real algebraic numbers. Thus, we have to restrict ourselves to $\mathbb{Q}$-automorphisms instead of $\mathbb{R}_{\mathbb{A}}$-automorphisms. Then the technique presented in the current section can indeed be applied to all loops that can be transformed into *twn*-form via $\mathbb{Q}$-automorphisms.[8] Here, however, the following has to be taken into account: When $(\varphi, \vec{u})$ is transformed into $(\varphi', \vec{u}')$ via a $\mathbb{Q}$-automorphism η, then the run of $(\varphi, \vec{u})$ on $\vec{e} \in \mathbb{Z}^d$ corresponds to the run of $(\varphi', \vec{u}')$ on $\widehat{\eta}(\vec{e})$ (and vice versa). Thus, if $f(\|\vec{e}\|)$ is a bound on $rt_{(\varphi',\vec{u}')}(\vec{e})$ for all $\vec{e} \in \widehat{\eta}\left(\mathbb{Z}^d\right)$, then we have $f(\|\widehat{\eta}(\vec{e})\|) \geq rt_{(\varphi',\vec{u}')}(\widehat{\eta}(\vec{e})) = rt_{(\varphi,\vec{u})}(\vec{e})$ for all $\vec{e} \in \mathbb{Z}^d$. So besides the loop under consideration, we also have to transform the resulting upper runtime bound.

5 Related Work and Conclusion

In this work, we presented a decision procedure for the halting problem of *twn*-loops, i.e., loops of the form

$$\textbf{while } \varphi \textbf{ do } \vec{x} \leftarrow \vec{u}$$

with polynomial arithmetic where the use of non-linearity in $\vec{u}$ is mildly restricted and $\vec{x} \leftarrow \vec{u}$ is a triangular system of polynomial equations. While we considered loops over $\mathbb{R}_{\mathbb{A}}$ in Sect. 3, decidability of the halting problem for loops over other rings $\mathbb{Z} \leq \mathcal{S} \leq \mathbb{R}_{\mathbb{A}}$ follows immediately. The key idea is to approximate the so-called *stabilization threshold* $sth_{(\varphi,\vec{u})}(\vec{e})$, i.e., the number of iterations after which the loop condition φ stops changing its truth value, for a *given* input $\vec{e}$. To the best of our knowledge, this is the first positive result on the decidability of the halting problem for classes of loops with *non-linear arithmetic*.

Moreover, we generalized this idea to approximate the stabilization threshold $sth_{(\varphi,\vec{u})}(\vec{x})$ for *arbitrary* inputs $\vec{x}$ of *twn*-loops over the integers. In this way we showed that for all terminating inputs $\vec{x}$, the runtime complexity of a *twn*-loop over the integers is bounded by a polynomial in the 1-norm $\|\vec{x}\|$ of the input. Moreover, we showed how to *compute* such a polynomial. As a corollary, it follows that the runtime of triangular *linear* loops is bounded by a linear polynomial in $\|\vec{x}\|$. To the best of our knowledge, this is the first positive result regarding the computability

[8]Note that the inequation (9) does not hold directly anymore, as we now have $\vec{e} \in \widehat{\eta}\left(\mathbb{Z}^d\right) \subseteq \mathbb{Q}^d$ instead of $\vec{e} \in \mathbb{Z}^d$ and thus we may have $0 < |\alpha_\ell(\vec{e})| < 1$. However, this can easily be fixed. The reason is that for each $\vec{e} \in \widehat{\eta}\left(\mathbb{Z}^d\right)$, the occurring denominators stem from the polynomials $\eta(x_1), \ldots, \eta(x_d) \in \mathbb{Q}[\vec{x}]$ and thus the least common multiple k of the denominators that occur in $\eta(x_1), \ldots, \eta(x_d)$ is a multiple of the denominators that occur in $\vec{e}$. So by multiplying the inequations $pe \triangleright 0$ in $\varphi(\vec{q})$ by k, one obtains a poly-exponential expression with coefficients α such that $\alpha(\vec{e}) \in \mathbb{Z}$ for all $\vec{e} \in \widehat{\eta}\left(\mathbb{Z}^d\right)$.

of upper bounds on the runtime complexity of polynomial integer loops.

Related Work

There exist several decidability results for the termination of linear [9, 10, 18, 24, 30, 32, 38, 43] and non-linear loops [19, 31, 45]. These works consider the question whether a polynomial loop terminates on *all* inputs, whereas we presented a decision procedure for termination on a *specific* input. Moreover, most of these works rely on the notion of *eventual* non-termination and thus they do not yield witnesses of non-termination.

For linear loops $(\varphi, \vec{u})$ where φ is a conjunction of atoms of the form $pol > 0$, *subsets* of $NT_{(\varphi,\vec{u})}$ can be computed via the techniques presented in [30, 32]. Moreover, for the same class of loops, a complete characterization of $NT_{(\varphi,\vec{u})}$ in the case of two variables is given in [15]. Here, it is also proven that in the case of more than two variables, $NT_{(\varphi,\vec{u})}$ cannot be described by a propositional formula over polynomial inequalities. Furthermore, there are certain special cases of linear loops over $\mathbb{Z}$ with conjunctive loop conditions where $\vec{u}^n$ and thus also $NT_{(\varphi,\vec{u})}$ is known to be Presburger-definable [9]. For non-linear loops, [31] shows how to compute witnesses for non-termination under certain prerequisites. In particular, the loop condition needs to characterize a closed, bounded, and connected set. In contrast, our result on the decidability of the halting problem enables us to enumerate *all* non-terminating inputs, and it allows for non-linearity and more complex loop conditions.

Obviously, decidability of the halting problem does not imply decidability of (*universal*) termination: For example, termination is undecidable for *twn*-loops over $\mathbb{Z}$ with non-linear integer arithmetic (due to undecidability of Hilbert's 10th problem: $(pol = 0, \vec{x})$ terminates over $\mathbb{Z}$ iff *pol* has no integer root). On the other hand, we showed that the halting problem is decidable for *twn*-loops over $\mathbb{Z}$ (and over any other ring $\mathbb{Z} \leq \mathcal{S} \leq \mathbb{R}_{\mathbb{A}}$).

For the converse direction, note that in contrast to Turing-complete languages, in the restricted formalism of polynomial loops, decidability of termination does not trivially imply decidability of the halting problem. The reason is that one cannot encode a fixed initial value of the program variables in a loop of the form (1).

For example, while termination of *linear* integer loops with conjunctive loop conditions is decidable [24], the halting problem for these loops is equivalent to the so-called positivity problem (cf. [39]), whose decidability is open. The positivity problem asks whether a given linear integer recurrence sequence with *fixed initial values* only takes positive values. Moreover, the closely related Skolem problem asks whether a given linear integer recurrence sequence with fixed initial values ever takes the value 0. While decidability of both the positivity and the Skolem problem is still open in the general case, there exist several partial solutions, cf. e.g., [1, 35, 36, 37]. For example, in [35] it is shown that the positivity problem is decidable for recurrences of order 5 or less, which implies decidability of the halting problem for *linear* integer loops with at most 4 variables and conjunctive loop conditions.[9] It is considered folklore that the Skolem and positivity problem are decidable if the characteristic polynomial of the recurrence sequence has real roots only [1]. For the Skolem problem, a proof of (a generalization of) this claim can be found in [22]. Hence, the halting problem is decidable for linear integer loops with conjunctive loop conditions where the update matrix has real eigenvalues only. Moreover, decidability of the halting problem for linear loops where the update matrix has only periodic rational eigenvalues[10] was shown in [27].

[9]Note that the first step of the transformation from a loop $(\varphi, A \cdot \vec{x} + \vec{b})$ into a recurrence relation is to eliminate the vector $\vec{b}$ by introducing an additional variable.

[10]A matrix has periodic rational eigenvalues if every eigenvalue λ satisfies $\lambda^m \in \mathbb{Q}$ for some $m \in \mathbb{N}_{\geq 1}$.

In contrast to these results, we showed decidability of the halting problem for *twn*-loops with *non-linear arithmetic* where the guard is an *arbitrary propositional formula* over polynomial inequations, which is orthogonal to the positivity problem.

Furthermore, there is a large body of work on *automated* complexity analysis (e.g., [2, 3, 12, 13, 16, 17, 23, 41, 42]). This line of work aims at inferring bounds on the runtime of programs with complex control flow by using incomplete techniques. The most popular approach to infer runtime bounds automatically synthesizes *lexicographic combinations of linear ranking functions*. To see the necessity of complete approaches, consider the *twn*-loop

$$\textbf{while } x_1 \geq x_2 \land x_2 \geq 1 \textbf{ do } \left(\begin{smallmatrix} x_1 \\ x_2 \end{smallmatrix}\right) \leftarrow \left(\begin{smallmatrix} 2 \cdot x_1 \\ 3 \cdot x_2 \end{smallmatrix}\right),$$

which terminates, but is beyond the capabilities of such ranking functions [8, 29]. However, with our approach, one can compute the bound $2 \cdot (|x_1| + |x_2|) + 3$ on its runtime.

There are also several works on runtime analysis for restricted program models such as vector addition systems with states (e.g., [11, 46]), bounded polynomial loops where the number of iterations of the loop is fixed in advance and does not depend on the loop body (e.g., [4, 5, 6, 7]), and max-plus automata (e.g., [14]). These models are orthogonal to the (unbounded) polynomial loops considered in the current paper.

Finally, there also exist several other approaches which exploit the existence of closed forms for similar classes of programs (like solvable loops [40]), in order to, e.g., deduce invariants (e.g., [25, 26, 27, 28, 33, 34, 40]).

References

[1] S. Akshay, N. Balaji, and N. Vyas. "Complexity of Restricted Variants of Skolem and Related Problems". In: *Proc. MFCS '17*. LIPIcs 83. 2017, 78:1–78:14. DOI: 10.4230/LIPIcs.MFCS.2017.78.

[2] E. Albert, P. Arenas, S. Genaim, and G. Puebla. "Closed-Form Upper Bounds in Static Cost Analysis". In: *Journal of Automated Reasoning* 46.2 (2011), pp. 161–203. DOI: 10.1007/s10817-010-9174-1.

[3] E. Albert, M. Bofill, C. Borralleras, E. Martin-Martin, and A. Rubio. "Resource Analysis driven by (Conditional) Termination Proofs". In: *Theory and Practice of Logic Programming* 19.5-6 (2019), pp. 722–739. DOI: 10.1017/S1471068419000152.

[4] A. M. Ben-Amram, N. D. Jones, and L. Kristiansen. "Linear, Polynomial or Exponential? Complexity Inference in Polynomial Time". In: *Proc. CiE '08*. LNCS 5028. 2008, pp. 67–76. DOI: 10.1007/978-3-540-69407-6_7.

[5] A. M. Ben-Amram and L. Kristiansen. "On the Edge of Decidability in Complexity Analysis of Loop Programs". In: *International Journal of Foundations of Computer Science* 23.7 (2012), pp. 1451–1464. DOI: 10.1142/S0129054112400588.

[6] A. M. Ben-Amram and A. Pineles. "Flowchart Programs, Regular Expressions, and Decidability of Polynomial Growth-Rate". In: *Proc. VPT@ETAPS '16*. EPTCS 216. 2016, pp. 24–49. DOI: 10.4204/EPTCS.216.2.

[7] A. M. Ben-Amram and G. W. Hamilton. "Tight Worst-Case Bounds for Polynomial Loop Programs". In: *Proc. FOSSACS '19*. LNCS 11425. 2019, pp. 80–97. DOI: 10.1007/978-3-030-17127-8_5.

[8] A. M. Ben-Amram, J. J. Doménech, and S. Genaim. "Multiphase-Linear Ranking Functions and Their Relation to Recurrent Sets". In: *Proc. SAS '19*. LNCS 11822. 2019, pp. 459–480. DOI: 10.1007/978-3-030-32304-2_22.

[9] M. Bozga, R. Iosif, and F. Konecný. "Deciding Conditional Termination". In: *Logical Methods in Computer Science* 10.3 (2014). DOI: 10.2168/LMCS-10(3:8)2014.

[10] M. Braverman. "Termination of Integer Linear Programs". In: *Proc. CAV '06*. LNCS 4144. 2006, pp. 372–385. DOI: 10.1007/11817963_34.

[11] T. Brázdil, K. Chatterjee, A. Kucera, P. Novotný, D. Velan, and F. Zuleger. "Efficient Algorithms for Asymptotic Bounds on Termination Time in VASS". In: *Proc. LICS '18*. 2018, pp. 185–194. DOI: 10.1145/3209108.3209191.

[12] M. Brockschmidt, F. Emmes, S. Falke, C. Fuhs, and J. Giesl. "Analyzing Runtime and Size Complexity of Integer Programs". In: *ACM Transactions on Programming Languages and Systems* 38.4 (2016), 13:1–13:50. DOI: 10.1145/2866575.

[13] Q. Carbonneaux, J. Hoffmann, T. W. Reps, and Z. Shao. "Automated Resource Analysis with Coq Proof Objects". In: *Proc. CAV '17*. LNCS 10427. 2017, pp. 64–85. DOI: 10.1007/978-3-319-63390-9_4.

[14] T. Colcombet, L. Daviaud, and F. Zuleger. "Size-Change Abstraction and Max-Plus Automata". In: *Proc. MFCS '14*. LNCS 8634. 2014, pp. 208–219. DOI: 10.1007/978-3-662-44522-8_18.

[15] L. Dai and B. Xia. "Non-Termination Sets of Simple Linear Loops". In: *Proc. ICTAC '12*. LNCS 7521. 2012, pp. 61–73. DOI: 10.1007/978-3-642-32943-2_5.

[16] A. Flores-Montoya. "Upper and Lower Amortized Cost Bounds of Programs Expressed as Cost Relations". In: *Proc. FM '16*. LNCS 9995. 2016, pp. 254–273. DOI: 10.1007/978-3-319-48989-6_16.

[17] F. Frohn, M. Naaf, J. Hensel, M. Brockschmidt, and J. Giesl. "Lower Runtime Bounds for Integer Programs". In: *Proc. IJCAR '16*. LNCS 9706. 2016, pp. 550–567. DOI: 10.1007/978-3-319-40229-1_37.

[18] F. Frohn and J. Giesl. "Termination of Triangular Integer Loops is Decidable". In: *Proc. CAV '19*. LNCS 11562. 2019, pp. 269–286. DOI: 10.1007/978-3-030-25543-5_24. arXiv: 1905.08664.

[19] F. Frohn, M. Hark, and J. Giesl. "On the Decidability of Termination for Polynomial Loops". In: *CoRR* abs/1910.11588 (2019). arXiv: 1910.11588.

[20] F. Frohn. "A Calculus for Modular Loop Acceleration". In: *Proc. TACAS '20*. LNCS. To appear. 2020. arXiv: 2001.01516.

[21] J. Giesl, A. Rubio, C. Sternagel, J. Waldmann, and A. Yamada. "The Termination and Complexity Competition". In: *Proc. TACAS '19*. LNCS 11429. 2019, pp. 156–166. DOI: 10.1007/978-3-030-17502-3_10.

[22] V. Halava, T. Harju, M. Hirvensalo, and J. Karhumäki. *Skolem's Problem – On the Border between Decidability and Undecidability*. Tech. rep. 683. Turku Center for Computer Science, 2005. URL: http://tucs.fi/publications/attachment.php?fname=TR683.pdf.

[23] J. Hoffmann, A. Das, and S.-C. Weng. "Towards Automatic Resource Bound Analysis for OCaml". In: *Proc. POPL '17*. 2017, pp. 359–373. DOI: 10.1145/3009837.3009842.

[24] M. Hosseini, J. Ouaknine, and J. Worrell. "Termination of Linear Loops over the Integers". In: *Proc. ICALP '19*. LIPIcs 132. 2019, 118:1–118:13. DOI: 10.4230/LIPIcs.ICALP.2019.118.

[25] A. Humenberger, M. Jaroschek, and L. Kovács. "Invariant Generation for Multi-Path Loops with Polynomial Assignments". In: *Proc. VMCAI '18*. LNCS 10747. 2018, pp. 226–246. DOI: 10.1007/978-3-319-73721-8_11.

[26] Z. Kincaid, J. Cyphert, J. Breck, and T. W. Reps. "Non-Linear Reasoning for Invariant Synthesis". In: *Proceedings of the ACM on Programming Languages* 2.POPL (2018), 54:1–54:33. DOI: 10.1145/3158142.

[27] Z. Kincaid, J. Breck, J. Cyphert, and T. W. Reps. "Closed Forms for Numerical Loops". In: *Proceedings of the ACM on Programming Languages* 3.POPL (2019), 55:1–55:29. DOI: 10.1145/3290368.

[28] L. Kovács. "Reasoning Algebraically About P-Solvable Loops". In: *Proc. TACAS '08*. LNCS 4963. 2008, pp. 249–264. DOI: 10.1007/978-3-540-78800-3_18.

[29] J. Leike and M. Heizmann. "Ranking Templates for Linear Loops". In: *Logical Methods in Computer Science* 11.1 (2015). DOI: 10.2168/LMCS-11(1:16)2015.

[30] Y. Li. "A Recursive Decision Method for Termination of Linear Programs". In: *Proc. SNC '14*. 2014, pp. 97–106. DOI: 10.1145/2631948.2631966.

[31] Y. Li. "Termination of Single-Path Polynomial Loop Programs". In: *Proc. ICTAC '16*. LNCS 9965. 2016, pp. 33–50. DOI: 10.1007/978-3-319-46750-4_3.

[32] Y. Li. "Witness to Non-Termination of Linear Programs". In: *Theoretical Computer Science* 681 (2017), pp. 75–100. DOI: 10.1016/j.tcs.2017.03.036.

[33] S. de Oliveira, S. Bensalem, and V. Prevosto. "Polynomial Invariants by Linear Algebra". In: *Proc. ATVA '16*. LNCS 9938. 2016, pp. 479–494. DOI: 10.1007/978-3-319-46520-3_30.

[34] S. de Oliveira, S. Bensalem, and V. Prevosto. "Synthesizing Invariants by Solving Solvable Loops". In: *Proc. ATVA '17*. LNCS 10482. 2017, pp. 327–343. DOI: 10.1007/978-3-319-68167-2_22.

[35] J. Ouaknine and J. Worrell. "Positivity Problems for Low-Order Linear Recurrence Sequences". In: *Proc. SODA '14*. 2014, pp. 366–379. DOI: 10.1137/1.9781611973402.27.

[36] J. Ouaknine and J. Worrell. "On the Positivity Problem for Simple Linear Recurrence Sequences," in: *Proc. ICALP '14*. LNCS 8573. 2014, pp. 318–329. DOI: 10.1007/978-3-662-43951-7_27.

[37] J. Ouaknine and J. Worrell. "Ultimate Positivity is Decidable for Simple Linear Recurrence Sequences". In: *Proc. ICALP '14*. LNCS 8573. 2014, pp. 330–341. DOI: 10.1007/978-3-662-43951-7_28.

[38] J. Ouaknine, J. S. Pinto, and J. Worrell. "On Termination of Integer Linear Loops". In: *Proc. SODA '15*. 2015, pp. 957–969. DOI: 10.1137/1.9781611973730.65.

[39] J. Ouaknine and J. Worrell. "On Linear Recurrence Sequences and Loop Termination". In: *SIGLOG News* 2.2 (2015), pp. 4–13. URL: https://dl.acm.org/citation.cfm?id=2766191.

[40] E. Rodríguez-Carbonell and D. Kapur. "Automatic Generation of Polynomial Loop Invariants: Algebraic Foundation". In: *Proc. ISSAC '04*. 2004, pp. 266–273. DOI: 10.1145/1005285.1005324.

[41] M. Sinn, F. Zuleger, and H. Veith. “Complexity and Resource Bound Analysis of Imperative Programs Using Difference Constraints”. In: *Journal of Automated Reasoning* 59.1 (2017), pp. 3–45. DOI: 10.1007/s10817-016-9402-4.

[42] A. Srikanth, B. Sahin, and W. R. Harris. “Complexity Verification Using Guided Theorem Enumeration”. In: *Proc. POPL ’17*. 2017, pp. 639–652. DOI: 10.1145/3009837.3009864.

[43] A. Tiwari. “Termination of Linear Programs”. In: *Proc. CAV ’04*. LNCS 3114. 2004, pp. 70–82. DOI: 10.1007/978-3-540-27813-9_6.

[44] *TPDB (Termination Problems Data Base)*. URL: http://termination-portal.org/wiki/TPDB.

[45] B. Xia and Z. Zhang. “Termination of Linear Programs with Nonlinear Constraints”. In: *Journal of Symbolic Computation* 45.11 (2010), pp. 1234–1249. DOI: 10.1016/j.jsc.2010.06.006.

[46] F. Zuleger. “The Polynomial Complexity of Vector Addition Systems with States”. In: *CoRR* abs/1907.01076 (2019). arXiv: 1907.01076.