HAL open science

On the Linear Ranking Problem for Simple Floating-Point Loops

Etienne Payet, Fonenantsoa Maurica Andrianampoizinimaro, Frédéric Mesnard

▶ **To cite this version:**

Etienne Payet, Fonenantsoa Maurica Andrianampoizinimaro, Frédéric Mesnard. On the Linear Ranking Problem for Simple Floating-Point Loops. 23rd International Symposium on Static Analysis (SAS), Sep 2016, Edinburgh, United Kingdom. pp.300-316, 10.1007/978-3-662-53413-7_15 . hal-01451688

HAL Id: hal-01451688

https://hal.univ-reunion.fr/hal-01451688

Submitted on 5 Nov 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

On the linear ranking problem for simple floating-point loops

Fonenantsoa Maurica, Frédéric Mesnard, and Étienne Payet

Université de La Réunion,
Laboratoire d'Informatique et de Mathématiques,
97490 Sainte Clotilde, La Réunion, France
{fonenantsoa.maurica,frederic.mesnard,etienne.payet}@univ-reunion.fr

Abstract. Termination of loops can be inferred from the existence of linear ranking functions. We already know that the existence of these functions is PTIME decidable for simple rational loops. Since very recently, we know that the problem is coNP-complete for simple integer loops. We continue along this path by investigating programs dealing with floating-point computations. First, we show that the problem is at least in coNP for simple floating-point loops. Then, in order to work around that theoretical limitation we present an algorithm which remains polynomial by sacrificing completeness. The algorithm, based on the Podelski-Rybalchenko algorithm, can also synthesize in polynomial time the linear ranking functions it detects. To our knowledge, our work is the first adaptation of this well-known algorithm to floating-points.

Keywords: Termination analysis, Linear ranking functions, Floating-point numbers

1 Introduction

Termination analysis of programs is a research topic that has already produced many remarkable results. This work is a continuation of a series of connected results concerning simple loops. [24] first showed that termination of loops of the form `while` $(Dx \leq d)$ `do` $x' = Vx + v$ `done` where the column vector of n variables x ranges over $\mathbb{R}^{n\times 1}$ is decidable. Then, [7] showed that the problem is also decidable when $x \in \mathbb{Q}^{n\times 1}$ and even when $x \in \mathbb{Z}^{n\times 1}$ for the homogeneous case where $d = 0$. Followingly, [6] investigated the more general case of the non-deterministic loops of the form `while` $(Dx \leq d)$ `do` $V\left(x\ x'\right)^T \leq v$ `done` where $\left(x\ x'\right)^T = \begin{pmatrix} x \\ x' \end{pmatrix}$ denotes the column vector of $2n$ elements obtained by concatenating x with x'. Termination of such loops was proved to be undecidable when $x, x' \in \mathbb{Z}^{n\times 1}$. The existence of *linear* ranking functions for such loops is however decidable and the problem is shown to be coNP-complete [4, 5]. That result applies even when the variables range over a finite set $E \subset \mathbb{Z}$ with $|E| \geq 2$, like the case of machine integers. We now study in this paper the case where

```
rational x := 100;
while(x >= 0)
  x := x - 1;
```

Fig. 1. A simple program, $\mathcal{P}_{simple}$

the variables are of floating-point type. In addition to being a prolongation of these previous works, ours can also be seen as a part of the current efforts in analyzing bit-vector programs instead of purely mathematical ones with error-free computations. Indeed, the rounding errors inherent to floating-point arithmetic render invalid most of the results obtained for programs using real or rational variables.

The paper is organized as follows. Section 2 briefly introduces the prerequisites. Sections 3 and 4 expose our contribution. We first show that there is no polynomial algorithm that can decide the existence of linear ranking functions for simple floating-point loops as the problem is at least in coNP. Then, we work around that theoretical limitation by presenting a sufficient but not necessary condition, thus incomplete but checkable in polynomial time, that ensures the existence of these functions. Section 5 presents the related work. Section 6 concludes.

2 Preliminaries

In this section we introduce the notions and notations we use in the paper.

Definition 1 (Programs as transition systems). *We formalize a program $\mathcal{P}$ as a transition system $\langle \mathcal{X}, \mathcal{I}, \mathcal{R} \rangle$ where $\mathcal{X}$ is a set of states, $\mathcal{I} \subseteq \mathcal{X}$ is the set of initial states and $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{X}$ is a transition relation between a state and its possible successors.*

Example 1. The program $\mathcal{P}_{simple}$ presented in Figure 1 can be formalized as the transition system $\langle \mathcal{X}_{simple}, \mathcal{I}_{simple}, \mathcal{R}_{simple} \rangle$ where $\mathcal{X}_{simple} = \mathbb{Q}$, $\mathcal{I}_{simple} = \{100\}$ and $\mathcal{R}_{simple} = \{\langle x, x' \rangle \in \mathbb{Q}^2 | x \geq 0 \wedge x' = x - 1\}$.

Definition 2 (Sequences). *Given a set $\mathcal{X}$ of states X_i, we say that s is an $\mathcal{X}$-sequence if $s = X_1 X_2 \ldots$ A finite sequence will have a last index last(s).*

Let $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{X}$. A finite sequence s is said to be permitted by $\mathcal{R}$ if and only if $\forall i \in 1 \ldots last(s) - 1 : X_i \mathcal{R} X_{i+1}$. An infinite sequence s is permitted by $\mathcal{R}$ if and only if $\forall i : i > 0 \implies X_i \mathcal{R} X_{i+1}$.

Proposition 1 (Termination characterization, ranking functions). *A program $\mathcal{P} = \langle \mathcal{X}, \mathcal{I}, \mathcal{R} \rangle$, where $\mathcal{X}$ is a set of states X_i, terminates if and only if there exists a function f from $(\mathcal{X}, \mathcal{R})$ to a well-founded set $(\mathcal{W}, <)$ such that $\forall i : X_i \mathcal{R} X_{i+1} \implies f(X_i) > f(X_{i+1})$. The function f is called a ranking function for $\mathcal{P}$.*

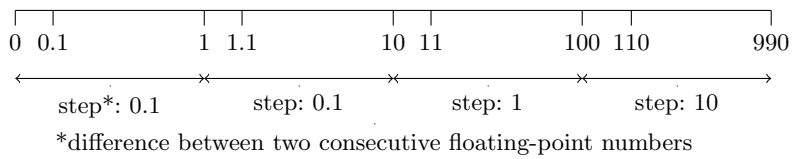


Fig. 2. A personalized floating-point type, $\mathbb{F}_{simple}$. Symmetry to the origin for the negatives.

By choosing $(\mathbb{Q}, <_{\delta_0,\delta}), \delta_0 \in \mathbb{Q}, \delta \in \mathbb{Q}_+^*$ such that $\forall x, x' \in \mathbb{Q} : x' <_{\delta_0,\delta} x \iff x \geq \delta_0 \wedge x \geq \delta + x'$ as well-founded set, a ranking function f is a function such that: $\forall X, X' \in \mathcal{X} : X\mathcal{R}X' \implies f(X) \geq \delta_0 \wedge f(X) \geq \delta + f(X')$.

Definition 3 (LRF). *A linear ranking function, that we shorten LRF, for a program having the column vector $X \in \mathbb{Q}^{n\times 1}$ as variables is a ranking function f of the form $f(X) = CX$ where $C \in \mathbb{Q}^{1\times n}$ is a constant row vector.*

Definition 4 ($\mathbb{F}_{\beta,p,e_{min},e_{max}}$). *A real number $x \in \mathbb{R}$ is said to be approximated by the floating-point number $\hat{x} \in \mathbb{F}_{\beta,p,e_{min},e_{max}}$ if $x = (-1)^s m\beta^e$ and $\hat{x} = (-1)^s \hat{m}\beta^e$ where $s \in \{0,1\}$ is the sign, $\beta \in \mathbb{N}$ such that $\beta \geq 2$ is the radix, $e \in \mathbb{N}$ such that $e_{min} \leq e \leq e_{max}$ is the exponent and $\hat{m}$ is the mantissa. The mantissa $\hat{m}$ is a fractional number approximating in $p \in \mathbb{N}^*$ digits the real number m and is such that $\hat{m} = \Diamond(\frac{m}{\beta^{-p+1}})\beta^{-p+1}$ where $\Diamond$ is an approximation function that is defined according to the rounding mode $\Box$. The number p is called precision.*

If $|x| \geq \beta^{e_{min}}$, $\hat{x}$ is called a normal floating-point number and $1 \leq \hat{m} < \beta$. Otherwise, $\hat{x}$ is called a subnormal floating-point number and $0 \leq \hat{m} < 1$.

Example 2. The floating-point type $\mathbb{F}_{simple} = \mathbb{F}_{\beta=10,p=2,e_{min}=-1,e_{max}=2}$ is presented in Figure 2. Notice that the more we go away from zero, the more the distance between two consecutive floating-point numbers increases.

Using $\mathbb{F}_{simple}$, the real number $x = 10\pi = 31.4159\cdots = (-1)^s m\beta^e$ where $s = 0, m = \pi, \beta = 10$ and $e = 1$ is represented by $\hat{x} = (-1)^s \hat{m}\beta^e$ where $\hat{m} = \Box(\pi)$.

We are addressing floating-point types that can be freely parameterized. This allows our result to be applied not only to programs using the IEEE-754 floating-point types, but also to programs using personalized ones like with the MPFR library [14].

Definition 5 (Rounding modes). *Given a real number x and the two floating-point numbers $\hat{x}_1$ and $\hat{x}_2$ adjacent to x such that $\hat{x}_1 \leq x \leq \hat{x}_2$, the rounding mode $\Box$ defines the choice to make between $\hat{x}_1$ and $\hat{x}_2$ when approximating x.*

In the rounding mode to-nearest-ties-to-even, denoted o, we choose the one closest to x. In case of tie, we choose the one having the last digit in the mantissa even. In the rounding mode to-zero, denoted $\uparrow^0$, we choose the one closest to 0.

Definition 6 (Correct rounding). *Given a rounding mode $\square$, a real arithmetic operation $\star$, its floating-point equivalent $\circledast$ and 2 floating-point numbers $\hat{x}_1, \hat{x}_2$, we say that $\circledast$ is correctly rounded if $\hat{x}_1 \circledast \hat{x}_2 = \square(\hat{x}_1 \star \hat{x}_2)$.*

More details on floating-point computations can be found in [15].

Definition 7 (Simple rational loops). *We call simple rational loop a loop of the form* `while` $(Dx \leq d)$ `do` $V \left(x\ x'\right)^T \leq v$. *The column vector $x = \left(x_1 \ldots x_n\right)^T \in \mathbb{Q}^{n \times 1}$ represents the variables at the beginning of an iteration. The primed equivalent x' represents the variables at the end of an iteration, when they have been updated. D, d, V, v are rational matrices of appropriate dimensions. The operations are done in the rationals.*

A simple rational loop can be viewed as the rational convex polyhedron described by the set of inequalities $A'' \left(x\ x'\right)^T \leq b$ obtained by conjuncting the loop condition with the update constraints.

Definition 8 (Simple floating-point loops). *Similarly to simple rational loops, simple floating-point loops are loops described by the set of inequalities $A'' \odot \left(x\ x'\right)^T \leq b$ in which x and x' are column vectors of floating-point variables. The floating-point matrix multiplication $\odot$ is similar to the rational matrix multiplication with the difference that the operations are done within the set of the floating-point numbers, that is:*

$$\begin{cases} a_{11} \odot x_1 \ \oplus\ a_{12} \odot x_2 \ \oplus\ \ldots \ \oplus\ a'_{1n} \odot x'_n \ \leq\ b_1 \\ a_{21} \odot x_1 \ \oplus\ a_{22} \odot x_2 \ \oplus\ \ldots \ \oplus\ a'_{2n} \odot x'_n \ \leq\ b_2 \\ \ldots \\ a_{m1} \odot x_1 \ \oplus\ a_{m2} \odot x_2 \ \oplus\ \ldots \ \oplus\ a'_{mn} \odot x'_n \ \leq\ b_m \end{cases}$$

where a_{ij}, x_i, x'_i and b_i are respectively elements of A'', x, x' and b. In absence of parenthesis, the order of operations is such that floating-point multiplications are performed before floating-point additions. Then, additions are performed from left to right.

We would like to note that simple rational loops, as we defined them, are named in multiple ways in the literature. With only slight differences, they are for example designated as Single-path Linear-Constraints loops in [4, 5], Linear Simple Loops in [9] and Linear Arithmetic Simple While loops in [21]. In an attempt to have a uniform naming for both the rational and the floating-point case, we chose the denomination *simple loop* as floating-point operations are non-linear, making all the previous names inappropriate.

We also find interesting to note that any unnested loop, using rational variables and rational arithmetic operations, having sequential assignments in its body, can be described by a simple rational loop of Definition 7, as explained in [8, Section 3.1]. This result is not valid when dealing with floating-point numbers due to the non-associativity of the floating-point arithmetic operations. Generally, floating-point arithmetic expressions cannot be simplified like the rational

ones and should be treated as-is. We chose to study the floating-point loops of Definition 8 to give general ideas that can be adapted to other form of floating-point loops.

Definition 9 (LinRF). *The decision problem of the existence of a LRF, denoted LinRF, is defined as follows. Its instance is a simple loop. The question it tries to answer is whether there exists a LRF for the simple loop. The decision problem is denoted by* $LinRF(\mathbb{Q})$ *and* $LinRF(\mathbb{F})$ *when the variables respectively range over rationals and floating-point numbers from any floating-point type* $\mathbb{F}_{\beta,p,e_{min},e_{max}}$ *(and any rounding mode* $\square$*) that could be defined. More generally, we denote by* $LinRF(E)$ *the decision problem when the variables range over* E*.*

In the same way, we define the more general problem of universal termination.

Definition 10 (Halt). *The decision problem of universal termination, denoted Halt, is defined as follows. Its instance is a simple loop. The question it tries to answer is whether the simple loop terminates for every possible input. We denote by* $Halt(E)$ *the decision problem when the variables range over* E*.*

3 Complexity of $LinRF(\mathbb{F})$

In this section, we show that there is no polynomial algorithm that can decide $LinRF(\mathbb{F})$. We start by studying the decidability of $Halt(\mathbb{F})$ and $LinRF(\mathbb{F})$.

Theorem 1. *Halt(*$\mathbb{F}$*) is decidable.*

Proof. A program $\mathcal{P} = \langle \mathcal{X}, \mathcal{I}, \mathcal{R} \rangle$ terminates if and only if every possible $\mathcal{X}$-sequence permitted by $\mathcal{R}$ is finite. As the variables range over some finite set $\mathbb{F}_{\beta,p,e_{min},e_{max}}$, then for any given sequence $s = X_1 X_2 \ldots$, we can check whether or not it is finite.

If the last element $X_{last(s)}$ of s has no more successor with respect to $\mathcal{R}$, s is a finite sequence. If there exists an element X_i that appears in s at index i and that already appeared at an index j, $j < i$, then there exists a subsequence $X_j \ldots X_i$ that will be repeated infinitely, causing s to be infinite. $\square$

Theorem 2. *LinRF(*$\mathbb{F}$*) is decidable.*

Proof. Recall the proof of Theorem 1. We reason similarly with the difference that in addition to checking the finiteness of all the sequences, we also check the existence of LRFs which are valid with respect to all of them.

We say that $f(X) = CX$ is a valid LRF with respect to a sequence $s = X_1 X_2 \ldots X_{last(s)}$ if and only if the following system of constraints Φ_s is satisfiable:

$$\begin{cases} CX_1 \geq \delta_0 \\ CX_1 \geq \delta + CX_2 \\ \ldots \\ CX_{last(s)-1} \geq \delta_0 \\ CX_{last(s)-1} \geq \delta + CX_{last(s)} \\ \delta > 0 \end{cases} \tag{1}$$

$\Phi = \bigwedge_{\text{all } s} \Phi_s$ is the system of constraints of validity of f with respect to all the sequences. If Φ is satisfiable, the space of all the valid LRFs is described by $f(X) = CX \wedge \Phi$. Otherwise, there is no valid LRF. $\square$

It is worth mentioning that the algorithms we just presented for deciding $Halt(\mathbb{F})$ and $LinRF(\mathbb{F})$ can be applied to any program using variables ranging over finite sets. In other words, they can be applied not only to simple loops using floating-point variables but also to any implementable program on machines. Indeed, machines all have a finite amount of memory forcing the variables to range over finite sets.

Let us now focus on $LinRF(\mathbb{F})$. We intuitively understand that the decision algorithm we proposed is highly costly. Indeed, if there are n variables in the program, that is $\mathcal{X} = \mathbb{F}^n_{\beta,p,e_{min},e_{max}}$, and if there are N floating-points numbers that can be taken as values, that is $|\mathbb{F}_{\beta,p,e_{min},e_{max}}| = N$, then in the very worst case, we need to build the system of constraints of validity Φ from the $|\mathcal{X}|! = n^N!$ possible $\mathcal{X}$-sequences.

The question now arises whether we can have a more efficient algorithm for deciding $LinRF(\mathbb{F})$. Notably, we would like to know if a polynomial one exists. For that purpose, we rely on a result presented in [5, Theorem 3.1] regarding the complexity of $LinRF(\mathbb{Z})$.

Lemma 1. *$LinRF(\mathbb{Z})$ is coNP-hard. Even for a finite set $E \subset \mathbb{Z}$, $LinRF(E)$ is still coNP-hard.*

Proof. The proof consists in reducing $LinRF(\mathbb{Z})$ to the well-known coNP-hard problem of deciding whether a rational convex polyhedron contains integer points.

Theorem 3. *$LinRF(\mathbb{F})$ is at least in coNP.*

Proof. The proof consists in showing that for some specific floating-point types $\mathbb{F}_{\beta,p,e_{min},e_{max}}$ and a rounding mode $\square$, the problem is at least in coNP. As we are studying the worst case complexity, that gives us a lower bound for the complexity of the generalization for any possible floating-point type.

Consider the finite set $Z_M \subset \mathbb{Z}$ defined as $Z_M = \{z \in \mathbb{Z} | -M \leq z \leq M\}$. For all $M \in \mathbb{N}^*$, we can construct the floating-point type $\mathbb{F}_M$ defined by the parameters $\beta = M$, $p = 1$, $e_{min} = 0$ and $e_{max} = 1$ for which $Z_M = \mathbb{F}_M$. Both Z_M and $\mathbb{F}_M$ have the same elements. Moreover, if the rounding mode for the operations in $\mathbb{F}_M$ is to-zero, that is if $\square = \uparrow^0$, then the operations in both Z_M and $\mathbb{F}_M$ are performed identically.

Hence, $\forall M \in \mathbb{N}^*, LinRF(\mathbb{F}_M) = LinRF(\mathbb{Z}_M)$. As $LinRF(\mathbb{Z}_M)$ is at least in coNP by Lemma 1, so is $LinRF(\mathbb{F}_M)$. $\square$

As we already know [23], the coNP class contains problems that are at least as hard as the NP class. Indeed, the complement of an NP-complete problem, which is in coNP, admits a polynomial decision algorithm only if P = NP, implying P = coNP. Thus, by conjecturing that P $\neq$ NP, we derive the following corollary.

Corollary 1. *There is no polynomial algorithm for deciding LinRF($\mathbb{F}$).*

Although that theoretical limitation may be discouraging, it is important to note that it applies to the problem of finding *one* general algorithm for *all* the possible instances of $LinRF(\mathbb{F})$. There may be special cases for which polynomial decision algorithms may exist. We could also have correct algorithms that are polynomial but not complete. In other words, we could have algorithms that detect in polynomial time only part of the space of the existing LRFs. We investigate this idea in the next section.

4 A sufficient condition for inferring LRFs in polynomial time

In this section, we present a novel technique for inferring in polynomial time the existence of LRFs for simple floating-point loops. The idea is to adapt to $LinRF(\mathbb{F})$ the well-known Podelski-Rybalchenko algorithm, that we shorten PR algorithm, which solves $LinRF(\mathbb{Q})$ in polynomial time and which is complete. This is achieved by means of sound over-approximations, that we shorten approximations for simplicity.

Firstly, section 4.1 studies from a termination analysis aspect the links between a program and its possible approximations. Secondly, section 4.2 presents a floating-point version of the PR algorithm obtained by using the approximation by maximal absolute error. Lastly, section 4.3 discusses the results obtained with other approximations.

4.1 Program approximation and termination

Definition 11 (Approximation). *We say that the program $\mathcal{P}^\# = \langle \mathcal{X}, \mathcal{I}^\#, \mathcal{R}^\# \rangle$ is an approximation of the program $\mathcal{P} = \langle \mathcal{X}, \mathcal{I}, \mathcal{R} \rangle$ if $\mathcal{I} \subseteq \mathcal{I}^\#$ and $\mathcal{R}^\#$ is such that $\forall X_1, X_2 \in \mathcal{X} : X_1 \mathcal{R} X_2 \implies X_1 \mathcal{R}^\# X_2$.*

Example 3. Consider the program $\mathcal{P}_{simple}$ presented in Example 1 and the program $\mathcal{P}^\#_{simple}$ presented in Figure 3 in which `:<=` is a non-deterministic assignment. $\mathcal{P}^\#_{simple}$ can be formalized as the transition system $\langle \mathcal{X}_{simple}, \mathcal{I}^\#_{simple}, \mathcal{R}^\#_{simple} \rangle$ where $\mathcal{I}^\#_{simple} = \{x \in \mathbb{Q} | x \leq 1000\}$ and $\mathcal{R}^\#_{simple} = \{\langle x, x' \rangle \in \mathbb{Q}^2 | x \geq -1 \wedge x' \leq x - 0.5\}$. We can easily verify that $\mathcal{I}_{simple} \subseteq \mathcal{I}^\#_{simple}$ and $\forall x, x' \in \mathcal{X}_{simple} : x \mathcal{R}_{simple} x' \implies x \mathcal{R}^\#_{simple} x'$. Hence, $\mathcal{P}^\#_{simple}$ is an approximation of $\mathcal{P}_{simple}$.

We now study the link between the termination of a given program and the termination of one of its approximations.

Theorem 4. *Given a program $\mathcal{P}$ and a corresponding approximation $\mathcal{P}^\#$, if $\mathcal{P}^\#$ terminates, so does $\mathcal{P}$.*

```
rational x :<= 1000;
while(x  >= -1) {
  x :<= x - 0.5;
}
```

Fig. 3. An approximation of $\mathcal{P}_{simple}$, $\mathcal{P}^{\#}_{simple}$

Proof. Differently stated, if $(\mathcal{X}, \mathcal{R}^{\#})$ is well-founded, so is $(\mathcal{X}, \mathcal{R})$. In order to prove that, let us reason by contradiction.

Suppose $(\mathcal{X}, \mathcal{R}^{\#})$ is well-founded while $(\mathcal{X}, \mathcal{R})$ is not. There is an infinite $\mathcal{X}$-sequence s_∞ permitted by $\mathcal{R}$: $s_\infty = X_0 \mathcal{R} X_1 \mathcal{R} \ldots$ By definition of approximation (Definition 11), $X_i \mathcal{R} X_{i+1} \implies X_i \mathcal{R}^{\#} X_{i+1}$. Consequently, there exists $s^{\#}_\infty = X_0 \mathcal{R}^{\#} X_1 \mathcal{R}^{\#} \ldots$ corresponding to s_∞ which contradicts our hypothesis as an infinite sequence is permitted by $\mathcal{R}^{\#}$. □

Though Theorem 4 is already enough to infer termination of a program through one of its possible approximations, it gives no information about the termination argument. Notably, we would like to have the deeper knowledge of the link between the space of ranking functions, that we denote by SRF.

Theorem 5. *Given a program $\mathcal{P}$ and a corresponding approximation $\mathcal{P}^{\#}$, we have $SRF(\mathcal{P}^{\#}) \subseteq SRF(\mathcal{P})$.*

Proof. Let us reason by contradiction. Suppose $SRF(\mathcal{P}^{\#}) \not\subseteq SRF(\mathcal{P})$ which means that there exists a ranking function f for $\mathcal{P}^{\#}$ which is not one for $\mathcal{P}$.

As f is not a ranking function for $\mathcal{P}$, $\exists X_1, X_2 \in \mathcal{X} : X_1 \mathcal{R} X_2 \implies f(X_1) \not> f(X_2)$. However, by definition of approximation (Definition 11), $X_1 \mathcal{R} X_2 \implies X_1 \mathcal{R}^{\#} X_2$ and as f is a ranking function for $\mathcal{P}^{\#}$, $X_1 \mathcal{R}^{\#} X_2 \implies f(X_1) > f(X_2)$ which leads to a contradiction. □

Differently stated, Theorem 5 says that any ranking function for a given program is also a ranking function for any program it approximates. For our particular case of interest, we derive the following corollary regarding the space of LRFs, that we denote by $SLRF$.

Corollary 2. *Given a program $\mathcal{P}$ and a corresponding approximation $\mathcal{P}^{\#}$, we have $SLRF(\mathcal{P}^{\#}) \subseteq SLRF(\mathcal{P})$: if f is a LRF for $\mathcal{P}^{\#}$, then f is also a LRF for $\mathcal{P}$.*

Example 4. Consider the program $\mathcal{P}_{simple}$ presented in Example 1 and the corresponding approximation $\mathcal{P}^{\#}_{simple}$ presented in Example 3. The space of LRFs of $\mathcal{P}^{\#}_{simple}$ is described by $\rho(x) = ax, a > 0$. By Corollary 2, ρ also describes LRFs for $\mathcal{P}_{simple}$.

We point out that it is a misunderstanding of the previous results to deduce from them that all termination analysis based on approximations are doomed to be incomplete. It would be indeed the case if $SRF(\mathcal{P}^{\#}) \subset SRF(\mathcal{P})$. However, it is always possible to refine the approximation until having $SRF(\mathcal{P}^{\#}) =$

$SRF(\mathcal{P})$. A way to do so for the particular case of floating-point loops is for example presented in [18].

Thus, in our search for LRFs for simple floating-point loops, it is always possible to find approximations that are precise enough to ensure completeness. However, due to Corollary 1, it cannot be achieved by any polynomial algorithm. In the technique we will now present, we trade completeness for complexity: the algorithm may fail at detecting all the possible LRFs, but it answers in polynomial time.

4.2 A floating-point version of the Podelski-Rybalchenko algorithm

Our technique is based on the well-known PR algorithm [21] which provides sufficient and necessary conditions, hence making the algorithm complete, for the existence of LRFs for simple rational loops. In addition to deciding the existence of LRFs, the algorithm can also synthesize them. It proceeds by reducing the decision/synthesizing problem to a linear programming problem which is long known to be solvable in polynomial time [17].

Theorem 6 (Podelski-Rybalchenko [21]). *Given a simple rational loop $\mathcal{L}_{\mathbb{Q}}$ described by $A''\left(x\ x'\right)^T \leq b$ such that $A'' \in \mathbb{Q}^{m\times 2n}$, $b \in \mathbb{Q}^{m\times 1}$ and $x, x' \in \mathbb{Q}^{n\times 1}$, let $A'' = (A\ A')$ where $A, A' \in \mathbb{Q}^{m\times n}$. A LRF exists for $\mathcal{L}_{\mathbb{Q}}$ if and only if there exist $\lambda_1, \lambda_2 \in \mathbb{Q}^{1\times m}$ such that:*

$$\begin{cases} \lambda_1, \lambda_2 \geq 0 \\ \lambda_1 A' = 0 \\ (\lambda_1 - \lambda_2)A = 0 \\ \lambda_2(A + A') = 0 \\ \lambda_2 b < 0 \end{cases}$$

The synthesized LRFs are of the form $f(x) = \mu x$ with $\mu = \lambda_2 A'$ and are such that for all x,x':

$$\begin{cases} f(x) \geq \delta_0, \delta_0 = -\lambda_1 b \\ f(x) \geq f(x') + \delta, \delta = -\lambda_2 b, \delta > 0 \end{cases}$$

Proof. Omitted as there are already various papers discussing it in various ways. Interested readers can find in-depth study of the PR algorithm in [1].

Example 5. Consider the program $\mathcal{P}_{ilog37q}$ having $n = 2$ variables presented in Figure 4. The loop of $\mathcal{P}_{ilog37q}$ can be expressed in the matrix form $\left(A_q\ A'_q\right)\left(x\ x'\right)^T \leq b$ by letting

$$A_q = \begin{pmatrix} -1 & 0 \\ 0 & -1 \\ -1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & -1 \end{pmatrix}, A'_q = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 37 & 0 \\ -37 & 0 \\ 0 & -1 \\ 0 & 1 \end{pmatrix}, b = \begin{pmatrix} -37 \\ -1 \\ 0 \\ 0 \\ -1 \\ 1 \end{pmatrix} \tag{2}$$

```
rational x1 = input(), x2 = 1;
while(x1  >= 37 & x2 >=1) {
  x1 := x1 / 37;
  x2 := x2 + 1;
}
```

Fig. 4. A program that computes and stores in `x2` the integer base-37 logarithm of `x1`, $\mathcal{P}_{ilog37q}$. A similar program with variables ranging over the integers is studied in [1].

The corresponding linear system given by PR is satisfiable and the row vectors $\lambda_1 = \left(\lambda_1^1\ \lambda_1^2\ \lambda_1^3\ \lambda_1^4\ \lambda_1^5\ \lambda_1^6\right)$ and $\lambda_2 = \left(\lambda_2^1\ \lambda_2^2\ \lambda_2^3\ \lambda_2^4\ \lambda_2^5\ \lambda_2^6\right)$ are such that:

$$\begin{cases} \lambda_1^1 = -\lambda_1^2 + \lambda_1^4 + \lambda_2^1 + \lambda_2^3 - \lambda_2^4 \\ \lambda_1^2 = -\lambda_2^5 + \lambda_2^6 \\ \lambda_1^3 = \lambda_1^4 \\ \lambda_1^4 \geq 0 \\ \lambda_1^5 \geq 0 \\ \lambda_1^6 = \lambda_1^5 \end{cases} \quad \text{and} \quad \begin{cases} \lambda_2^1 = 36\lambda_2^3 - 36\lambda_2^4 \\ \lambda_2^2 = 0 \\ \lambda_2^3 > 0 \\ 0 \leq \lambda_2^4 < \lambda_2^3 \\ \lambda_2^5 \geq 0 \\ \lambda_2^5 \leq \lambda_2^6 < 1332\lambda_2^3 - 1332\lambda_2^4 \end{cases} \tag{3}$$

Thus, $SLRF(\mathcal{P}_{ilog37q})$ is completely described by $f(x_1, x_2) = \mu_1 x_1 + \mu_2 x_2$ such that $\mu_1 > 0$ and $0 \leq \mu_2 < 36\mu_1$.

Every time a LRF exists for a given simple rational loop, the PR algorithm *does* find it in polynomial time. Unfortunately, it cannot be applied to simple floating-point loops. Indeed, the rounding errors cause the floating-point operations to be non-associative, making invalid most of the mathematical results leading to the PR algorithm.

We work around that issue by means of rational approximations. These approximations must be linear and must be defined in a single piece so that the PR algorithm can be used. Detailed explanations for the need for these constraints are given in Section 4.3. An example of such approximation is the approximation by maximal absolute error, as used in various ways in various works [3],[20].

Definition 12 (Absolute error). *The absolute error $\mathcal{A}(x)$ of the approximation of $x \in \mathbb{R}$ by $\hat{x} \in \mathbb{F}_{\beta,p,e_{min},e_{max}}$ using a rounding mode $\square$ is defined as $\mathcal{A}(x) = |x - \hat{x}|$.*

Theorem 7 (Approximation by maximal absolute error). *Given a real arithmetic operation $\star$, its floating-point equivalent $\circledast$ and 2 floating-point numbers $\hat{x}_1, \hat{x}_2 \in \mathbb{F}_{\beta,p,e_{min},e_{max}}$, if we use the rounding mode to-nearest-ties-to-even and if no overflow occurs then the following holds:*

$$\hat{x}_1 \star \hat{x}_2 - \mathcal{A}_{max} \leq \hat{x}_1 \circledast \hat{x}_2 \leq \hat{x}_1 \star \hat{x}_2 + \mathcal{A}_{max}$$

where $\mathcal{A}_{max} = \frac{\beta^{e_{max}-p+1}}{2}$ is the maximal absolute error.

Proof. Recall that by property of correct rounding, as presented in Definition 6, $\hat{x}_1 \circledast \hat{x}_2 = o(\hat{x}_1 \star \hat{x}_2)$. Let $t = \hat{x}_1 \star \hat{x}_2$ and $\hat{t} = o(t) = \hat{x}_1 \circledast \hat{x}_2$, $t \in \mathbb{R}$ and $\hat{t} \in \mathbb{F}_{\beta,p,e_{min},e_{max}}$.

From the definition of the absolute error given in Definition 12, it can be easily derived that $t - \mathcal{A}(t) \leq \hat{t} \leq t + \mathcal{A}(t)$. In the worst case, t is located right in the middle of $\hat{t}$ and the nearest floating-point number $\hat{t}_a$ adjacent to $\hat{t}$. Thus we have $\mathcal{A}(t) \leq \frac{|\hat{t}-\hat{t}_a|}{2}$. By writing $\hat{t}$ in standardized notation as presented in Definition 4, that is $\hat{t} = (-1)^s \hat{m}\beta^e$, we have $\hat{t}_a = (-1)^s(\hat{m} \pm \beta^{-p+1})\beta^e$. Hence, in the worst case: $\mathcal{A}(t) \leq \frac{\beta^{e-p+1}}{2}$. As $e \leq e_{max}$, $\mathcal{A}(t) \leq \mathcal{A}_{max}$ from which the theorem follows. □

Example 6. Consider the floating-point type $\mathbb{F}_{simple}$ presented in Example 2. If the rounding mode to-nearest-ties-to-even is used, then for any arithmetic operation $\star$, its floating-point equivalent $\circledast$ and 2 floating-point numbers $\hat{x}_1, \hat{x}_2 \in \mathbb{F}_{simple}$, the following holds: $\hat{x}_1 \star \hat{x}_2 - 5 \leq \hat{x}_1 \circledast \hat{x}_2 \leq \hat{x}_1 \star \hat{x}_2 + 5$.

We now present an adaptation of the PR algorithm using the approximation by maximal absolute error.

Theorem 8 (General floating-point version of Podelski-Rybalchenko). *Consider the floating-point type $\mathbb{F}$ which has such parameters and which uses such rounding mode that $\mathcal{A}_{max}$ is the maximal absolute error. Consider also the simple floating-point loop $\mathcal{L}_{\mathbb{F}}$ described by $A'' \odot \left(x\ x'\right)^T \leq b$ such that $A'' \in \mathbb{F}^{m\times 2n}$, $b \in \mathbb{F}^{m\times 1}$ and $x, x' \in \mathbb{F}^{n\times 1}$. By letting $A'' = (A\ A')$ where $A, A' \in \mathbb{F}^{m\times n}$ and if no overflow occurs, a LRF exists for $\mathcal{L}_{\mathbb{F}}$ if there exist $\lambda_1, \lambda_2 \in \mathbb{Q}^{1\times m}$ such that:*

$$\begin{cases} \lambda_1, \lambda_2 \geq 0 \\ \lambda_1 A' = 0 \\ (\lambda_1 - \lambda_2)A = 0 \\ \lambda_2(A + A') = 0 \\ \lambda_2 c < 0 \end{cases}$$

where $c \in \mathbb{Q}^{m\times 1}$ and $c = b + colvect^m((4n-1)\mathcal{A}_{max})$. Here, $colvect^m(e)$ denotes the column vector of m elements, all elements equaling e.

The synthesized LRFs are of the form $f(x) = \mu x$ with $\mu = \lambda_2 A'$ and are such that for all x,x':

$$\begin{cases} f(x) \geq \delta_0, \delta_0 = -\lambda_1 c \\ f(x) \geq f(x') + \delta, \delta = -\lambda_2 c, \delta > 0 \end{cases}$$

Proof. Let us call l_k the first member of the k-th inequality, that is:

$$l_k = a_{k1} \odot x_1 \ \oplus\ a_{k2} \odot x_2 \ \oplus\ \ldots \ \oplus\ a'_{kn} \odot x'_n \tag{4}$$

$$l_k \leq b_k \tag{5}$$

By approximating by below each operation using the approximation by maximal absolute error (Theorem 7) we get a linear lower bound for l_k:

$$\begin{aligned}
(a_{k1}x_1 - \mathcal{A}_{max}) \oplus a_{k2}\odot x_2 \quad & \oplus \ldots \oplus a'_{kn}\odot x'_n \leq l_k \\
a_{k1}x_1 \oplus (a_{k2}x_2 - \mathcal{A}_{max}) \quad & \oplus \ldots \oplus a'_{kn}\odot x'_n - \mathcal{A}_{max} \leq l_k \\
(a_{k1}x_1 + a_{k2}x_2 - \mathcal{A}_{max}) \quad & \oplus \ldots \oplus a'_{kn}\odot x'_n - 2\mathcal{A}_{max} \leq l_k \\
& \ldots \\
a_{k1}x_1 + a_{k2}x_2 \quad & + \cdots + a'_{kn}x'_n - (4n-1)\mathcal{A}_{max} \leq l_k \quad (6)
\end{aligned}$$

Combining that result with the upper bound b_k of l_k as shown in (5), we get:

$$\begin{aligned}
a_{k1}x_1 + a_{k2}x_2 + \ldots + a'_{kn}x'_n - (4n-1)\mathcal{A}_{max} \quad & \leq l_k \leq b_k \\
a_{k1}x_1 + a_{k2}x_2 + \ldots + a'_{kn}x'_n - (4n-1)\mathcal{A}_{max} \quad & \leq b_k \\
a_{k1}x_1 + a_{k2}x_2 + \ldots + a'_{kn}x'_n \quad & \leq b_k + (4n-1)\mathcal{A}_{max} \quad (7)
\end{aligned}$$

Hence, the floating-point loop $\mathcal{L}_{\mathbb{F}}$ described by $A''\odot\left(x\ x'\right)^T \leq b$ is approximated by the rational loop $\mathcal{L}_{\mathbb{Q}}$ described by $A''\left(x\ x'\right)^T \leq c$ where $c = b + colvect^m((4n-1)\mathcal{A}_{max})$.

By applying the PR algorithm presented in Theorem 6 on $\mathcal{L}_{\mathbb{Q}}$, if a LRF f is found, then f is also a LRF for $\mathcal{L}_{\mathbb{F}}$ by Corollary 2. □

Example 7. Consider the program $\mathcal{P}_{ilog37q}$ presented in Example 5. We are interested in its floating-point version $\mathcal{P}_{ilog37f}$ in which variables are from the type $\mathbb{F}_{simple}$ presented in Example 2. The rounding mode used is to-nearest-ties-to-even.

As presented in Example 6, the maximal absolute error for any arithmetic floating-point operation $\circledast$ done in $\mathbb{F}_{simple}$ is $\mathcal{A}_{max} = 5$. Thus, $\mathcal{P}_{ilog37f}$ is approximated by the rational loop $\mathcal{P}^{\#}_{ilog37f}$ described by $\left(A_f\ A'_f\right)\left(x\ x'\right)^T \leq c$ such that:

$$A_f = A_q, A'_f = A'_q, c = b + colvect^6(35) = c_2 = \left(-2\ 34\ 35\ 35\ 34\ 36\right)^T \quad (8)$$

By applying the PR algorithm, LRFs for $\mathcal{P}_{ilog37f}$ exist and are of the form $f(x_1, x_2) = \mu_1 x_1 + \mu_2 x_2$ such that $\mu_1 > 0$ and $0 \leq \mu_2 < \frac{\mu_1}{36}$.

It is important to observe that this floating-point version of the PR algorithm applies only if no overflow occurs during the computation. Indeed, by using the approximation by maximal absolute error, we assumed that the results of the operations all laid in the authorized range. Thus, the initial ranges of the values of the variables that do not eventually lead to an overflow should be determined beforehand. It can be achieved for example using techniques from the framework of Abstract Interpretation [11].

As we already discussed in the Preliminaries section of the paper, floating-point expressions are to be studied as-is and our results are only general. For example, we can have a better approximation of $\mathcal{P}_{ilog37f}$ by noticing that there

are less than $(4n-1)$ operations per line. Moreover, by only approximating the operations that are not known to be exactly computed, we get the approximation $\mathcal{P}^{\#}_{ilog37f2}$ described by $\left(A_{f2}\ A'_{f2}\right)\left(x\ x'\right)^T \leq c_2$ such that:

$$A_{f2} = A_q, A'_{f2} = A'_q, c_2 = \left(-37\ -1\ 10\ 10\ 4\ 6\right)^T \tag{9}$$

By applying the PR algorithm, LRFs for $\mathcal{P}_{ilog37f2}$ exist and are of the form $f(x_1, x_2) = \mu_1 x_1 + \mu_2 x_2$ such that $\mu_1 > 0$ and $0 \leq \mu_2 < \frac{661\mu_1}{111}$. We easily verify that $SLRF(\mathcal{P}^{\#}_{ilog37f}) \subset SLRF(\mathcal{P}^{\#}_{ilog37f2})$, that is we detect a wider range of LRFs with $\mathcal{P}^{\#}_{ilog37f2}$.

4.3 Other approximations

At this point of the paper, the reader may wonder why we specifically chose the approximation by maximal absolute error instead of another approximation. Indeed, the justification we proposed was that the approximation needed to be linear and defined in a single piece but we gave no further explanation. Now, we give more details.

Recall the proof for the general floating-point version of the PR algorithm (Theorem 8). It consisted in transforming the simple floating-point loop into a simple rational loop. Indeed, the PR algorithm works on loops described by linear systems, making the linearity of the approximation straightforward.

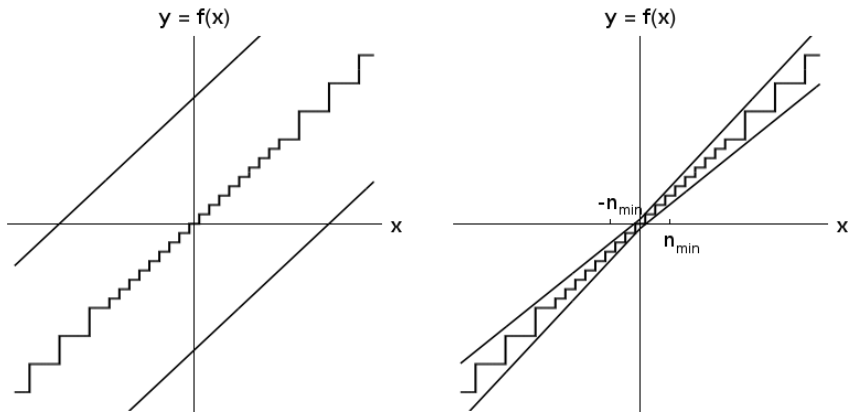
It remains to justify why the approximation needs to be defined in one piece. Indeed, we can have piecewise linear approximations having precisions that increase with the number of pieces, as shown in [18]. For example, the approximation defined in 3 pieces presented in Theorem 9 is better than the approximation by maximal absolute error. Indeed, the surface of the area enclosed between the upper and lower approximation functions is smaller in the approximation in 3 pieces, as illustrated in Figure 4.3.

Theorem 9 (Approximation in 3 pieces). *Given a real arithmetic operation $\star$, its floating-point equivalent $\circledast$ and 2 floating-point numbers $\hat{x}_1, \hat{x}_2 \in \mathbb{F}_{\beta,p,e_{min},e_{max}}$, if we use the rounding mode to-nearest-ties-to-even and if no overflow occurs then the following holds:*

$$\begin{cases} (1-\mathcal{R}^n_{max})t \leq \hat{x}_1 \circledast \hat{x}_2 \leq (1+\mathcal{R}^n_{max})t & \text{if} \quad n_{min} < t \leq n_{max} \\ t - \mathcal{A}^s_{max} \leq \hat{x}_1 \circledast \hat{x}_2 \leq t + \mathcal{A}^s_{max} & \text{if} \quad -n_{min} \leq t \leq n_{min} \\ 1+\mathcal{R}^n_{max})t \leq \hat{x}_1 \circledast \hat{x}_2 \leq 1-\mathcal{R}^n_{max})t & \text{if} \quad -n_{max} \leq t < -n_{min} \end{cases}$$

where $t = \hat{x}_1 \star \hat{x}_2$, $n_{min} = \beta^{e_{min}}$ is the smallest positive normal number, $n_{max} = (\beta - \beta^{-p+1})\beta^{e_{max}}$ is the biggest positive normal number, $\mathcal{A}^s_{max} = \frac{\beta^{e_{min}-p+1}}{2}$ is the maximal absolute error for the subnormal numbers and $\mathcal{R}^n_{max} = \frac{\beta^{-p+1}}{2+\beta^{-p+1}}$ is the maximal relative error for the normal numbers.

Proof. For the case where $-n_{min} \leq t \leq n_{min}$, that is t is a subnormal number, the proof is similar to that for the approximation with maximal absolute error (Theorem 7). Only the value of the maximal absolute error differs.



(a) Approximation in 1 piece or approximation by maximal absolute error

(b) Approximation in 3 pieces. The partitioning chosen here separates the normals from the subnormals.

Fig. 5. Piecewise linear approximations of $\hat{x}$ zoomed around the origin

For the case where $n_{min} \leq |t| \leq n_{max}$, that is t is a normal number, we use an approximation based on the maximal relative error. The relative error of the approximation of t is $\mathcal{R}(t) = \frac{|t-\hat{t}|}{|t|}$. From that definition, it can be easily derived that $(1-\mathcal{R}(t))|t| \leq |\hat{t}| \leq (1+\mathcal{R}(t))|t|$. As $\mathcal{R}(t) \leq \frac{\beta^{-p+1}}{2+\beta^{-p+1}}$ for the case of the normal numbers, as shown in [16], the theorem follows. □

As another example of approximation, we can combine the bound for the absolute error for the subnormals with the bound for the relative error for the normals. It results in using approximation functions defined in 2 pieces as it distinguishes the positives from the negatives [18].

Using these piecewise linear approximations, a simple floating-point loop will be transformed into a set of simple rational loops, designated as *Multi-path Linear-Constraint* loops or MLC loops in [4, 5]. The existence of LRFs in MLC loops are known to be also decidable in polynomial time [13],[19]. However, the size of the obtained MLC loop is an exponential function in the size of the simple floating-point loop it approximates, as shown in the following theorem.

Theorem 10 (From simple floating-point loop to MLC loop). *Consider the floating-point type* $\mathbb{F}$*. Consider also the simple floating-point loop* $\mathcal{L}_{\mathbb{F}}$ *described by* $A'' \bigodot \begin{pmatrix} x & x' \end{pmatrix}^T \leq b$ *such that* $A'' \in \mathbb{F}^{m \times 2n}$*,* $b \in \mathbb{F}^{m \times 1}$ *and* $x, x' \in \mathbb{F}^{n \times 1}$*. If we use a linear approximation defined in* k *pieces, then* $\mathcal{L}_{\mathbb{F}}$ *is approximated by a MLC loop composed of* $k^{(4n-1)m}$ *simple rational loops.*

Proof. The proof is similar to that for the general floating-point version of the PR algorithm (Theorem 8). The difference is that when approximating by below one operation, there are k cases to take into account. As there are $(4n-1)m$ operations, the theorem follows. □

Thus, only linear approximations defined in one piece preserve the transformations from becoming exponential. Without any information on the range of the variables, the approximation by maximal absolute error is the best approximation having these properties. That motivated our choice.

5 Related work

To the best of our knowledge, there is only limited work on the termination of floating-point programs. One of the first work addressing that problem is [22] which is an extension of the adornments-based approach. It consists in transforming the logic program to analyze in a way that we can use techniques originally developed for analysis of numerical computations.

Recently, techniques have been developed for treating the case of bit-vector programs. These techniques belong to the family of Model Checking [2]. [10] presents several novel algorithms to generate ranking functions for relations over machine integers: a method based on a reduction to Presburger arithmetic, and a template-matching approach for predefined classes of ranking functions based on reduction to SAT- and QBF-solving. In a similar way, [12] reduces the termination problem for programs using machine integers or floating-point numbers to a second-order satisfiability problem. Both methods are complete but are highly costly.

A different approach is presented in [18]. It consists in translating the floating-point programs into rational ones by means of sound approximations. Hence, it bridges the gap between termination analysis of floating-point loops and rational loops. The floating-point expressions are approximated using piecewise linear functions that can be parameterized depending on the precision desired/required.

6 Conclusion

We have studied the hardness of termination proofs for simple loops when the variables are of floating-point type. We have focused on termination inferred from the existence of linear ranking functions and showed that the problem is at least in coNP. This is a very valuable information as it dissuades us from looking for a decision algorithm that is both polynomial and complete. The problem of deciding the existence of linear ranking functions for simple integer and machine integer loops was studied in depth very recently and was shown to be coNP-complete. To the best of our knowledge, our work is the first attempt at providing a similar result for the floating-points.

To design a polynomial algorithm, we have traded completeness for complexity. We have proposed the first adaptation of the Podelski-Rybalchenko algorithm for simple floating-point loops. This is achieved by means of linear approximations defined in one piece. We have suggested the use of the approximation by maximal absolute error. A possible improvement would be to use more precise linear approximations defined in one piece based on the ranges of the variables. As we have provided a sufficient but not necessary condition for inferring the

existence of linear ranking functions, experimentations have yet to be conducted in order to get a practical evaluation of the technique.

References

1. R. Bagnara, F. Mesnard, A. Pescetti, and E. Zaffanella. A new look at the automatic synthesis of linear ranking functions. *Information and Computation*, 215:47–67, 2012.
2. C. Baier and J. Katoen. *Principles of model checking*. MIT Press, 2008.
3. M. S. Belaid, C. Michel, and M. Rueher. Boosting local consistency algorithms over floating-point numbers. In M. Milano, editor, *Proc. of the 18th International Conference on Principles and Practice of Constraint Programming (CP'12)*, volume 7514 of *Lecture Notes in Computer Science*, pages 127–140. Springer, 2012.
4. A. M. Ben-Amram. Ranking functions for linear-constraint loops. In A. Lisitsa and A. P. Nemytykh, editors, *Proc. of the 1st International Workshop on Verification and Program Transformation (VPT'13)*, volume 16 of *EPiC Series*, pages 1–8. EasyChair, 2013.
5. A. M. Ben-Amram and S. Genaim. Ranking functions for linear-constraint loops. *Journal of the ACM*, 61(4):26:1–26:55, 2014.
6. A. M. Ben-Amram, S. Genaim, and A. N. Masud. On the termination of integer loops. *ACM Transactions on Programming Languages and Systems*, 34(4):16, 2012.
7. M. Braverman. Termination of integer linear programs. In T. Ball and R. B. Jones, editors, *Proc. of the 18th International Conference on Computer Aided Verification (CAV'06)*, volume 4144 of *Lecture Notes in Computer Science*, pages 372–385. Springer, 2006.
8. H. Y. Chen. *Program analysis: termination proofs for linear simple loops*. PhD thesis, Louisiana State University, 2012.
9. H. Y. Chen, S. Flur, and S. Mukhopadhyay. Termination proofs for linear simple loops. *Software Tools for Technology Transfer*, 17(1):47–57, 2015.
10. B. Cook, D. Kroening, P. Rümmer, and C. M. Wintersteiger. Ranking function synthesis for bit-vector relations. *Formal Methods in System Design*, 43(1):93–120, 2013.
11. P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In R. M. Graham, M. A. Harrison, and R. Sethi, editors, *Proc. of the 4th ACM Symposium on Principles of Programming Languages (POPL'77)*, pages 238–252. ACM, 1977.
12. C. David, D. Kroening, and M. Lewis. Unrestricted termination and non-termination arguments for bit-vector programs. In J. Vitek, editor, *Proc. of the 24th European Symposium on Programming (ESOP'15)*, volume 9032 of *Lecture Notes in Computer Science*, pages 183–204. Springer, 2015.
13. P. Feautrier. Some efficient solutions to the affine scheduling problem. I. One-dimensional time. *International Journal of Parallel Programming*, 21(5):313–347, 1992.
14. L. Fousse, G. Hanrot, V. Lefèvre, P. Pélissier, and P. Zimmermann. MPFR: A multiple-precision binary floating-point library with correct rounding. *ACM Transactions on Mathematical Software*, 33(2):13, 2007.
15. D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1):5–48, 1991.

16. C.-P. Jeannerod and S. M. Rump. On relative errors of floating-point operations: optimal bounds and applications. Preprint, 2014.
17. L. Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53 – 72, 1980.
18. F. Maurica, F. Mesnard, and E. Payet. Termination analysis of floating-point programs using parameterizable rational approximations. In *Proc. of the 31st ACM Symposium on Applied Computing (SAC'16)*, 2016.
19. F. Mesnard and A. Serebrenik. Recurrence with affine level mappings is p-time decidable for CLP(R). *Theory and Practice of Logic Programming*, 8(1):111–119, 2008.
20. A. Miné. Relational abstract domains for the detection of floating-point run-time errors. *Computing Research Repository*, abs/cs/0703077, 2007.
21. A. Podelski and A. Rybalchenko. A complete method for the synthesis of linear ranking functions. In B. Steffen and G. Levi, editors, *Proc. of the 5th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'04)*, volume 2937 of *Lecture Notes in Computer Science*, pages 239–251. Springer, 2004.
22. A. Serebrenik and D. D. Schreye. Termination of floating-point computations. *Journal of Automated Reasoning*, 34(2):141–177, 2005.
23. M. Sipser. *Introduction to the theory of computation*. PWS Publishing Company, 1997.
24. A. Tiwari. Termination of linear programs. In R. Alur and D. A. Peled, editors, *Proc. of the 16th International Conference on Computer Aided Verification (CAV'04)*, volume 3114 of *Lecture Notes in Computer Science*, pages 70–82. Springer, 2004.