Deciding Termination of Simple Randomized Loops

Éléanore Meyer
RWTH Aachen University, Aachen, Germany

Jürgen Giesl
RWTH Aachen University, Aachen, Germany

Abstract

We show that universal positive almost sure termination (UPAST) is decidable for a class of simple randomized programs, i.e., it is decidable whether the expected runtime of such a program is finite for all inputs. Our class contains all programs that consist of a single loop, with a linear loop guard and a loop body composed of two linear commuting and diagonalizable updates. In each iteration of the loop, the update to be carried out is picked at random, according to a fixed probability. We show the decidability of UPAST for this class of programs, where the program's variables and inputs may range over various sub-semirings of the real numbers. In this way, we extend a line of research initiated by Tiwari in 2004 into the realm of randomized programs.

2012 ACM Subject Classification Theory of computation → Probabilistic computation; Theory of computation → Program analysis

Keywords and phrases decision procedures, randomized programs, linear loops, positive almost sure termination

Related Version The full version of this paper is published in [20]. Moreover, we provide an extended report [21] including all proofs, as well as proof sketches for all main results to help understanding the essential proof ideas.

Funding funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - DFG Research Training Group 2236 UnRAVeL

1 Introduction

We consider the problem of universal positive almost sure termination (UPAST), i.e., deciding whether a given randomized program has finite expected runtime on all inputs [5, 26]. This is a stronger property than universal almost sure termination (UAST) which requires that the probability of termination is 1. In [16], the authors showed that deciding UPAST is harder than deciding universal termination for non-randomized programs in terms of the arithmetic hierarchy. While the non-randomized problem is Π_2^0-complete, UPAST is Π_3^0-complete.

Our programs are simple randomized loops of the form

$$\texttt{while}\ \mathbf{C}\vec{x} > \vec{0} : \vec{x} \leftarrow \mathbf{A}\vec{x} \oplus_p \mathbf{B}\vec{x} \qquad (1)$$

Here, $\vec{x} = (x_1, \ldots, x_n)$ denotes the vector of program variables that range over a semiring $\mathcal{S} \subseteq \mathbb{R}$, and $\mathbf{C} \in \mathbb{R}^{m \times n}$ is a matrix representing the loop guard with m linear constraints over the program variables. In each execution of the loop body, a matrix is chosen among $\mathbf{A}, \mathbf{B} \in \mathcal{S}^{n \times n}$ according to the probability $p \in [0, 1]$ and the value $\vec{x}$ is updated accordingly.

We show that UPAST is decidable for all $\mathcal{S} \in \{\mathbb{Z}, \mathbb{Q}, \mathbb{A}\}$ when limited to loops with diagonalizable commuting matrices $\mathbf{A}$ and $\mathbf{B}$, where $\mathbb{A}$ is the set of algebraic real numbers.[1]

[1] Our approach only considers *algebraic* p, $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$, as it is not possible to represent arbitrary real numbers on computers. However, such a loop terminates for all algebraic inputs iff it terminates for all real inputs, see [20, Remark 38].

Thus, we extend previous results on the termination of linear and affine[2] non-randomized loops to the randomized setting. In addition to deciding universal termination, our approach can compute a non-termination witness $\vec{x} \in \mathcal{S}^n$, i.e., if the loop is non-terminating, then $\vec{x}$ is an input leading to an infinite expected runtime. So our programs go beyond single path loops as we might have $\mathbf{A} \neq \mathbf{B}$. Thus, for every $k \in \mathbb{N}$, there is not just a single execution of length k but instead one has a "range" of possible executions of length k where each execution occurs according to a known probability. To ensure tractability of the resulting problem we require commutativity of both updates, so that we can focus on how often each update has been selected in an execution, but we do not have to take the 2^k different orders into account in which the two updates might have been chosen. Moreover, we require diagonalizability in order to obtain closed forms of a certain shape, which allows us to analyze the behavior of a "range" of different executions at once.

To demonstrate the practical applicability of our decision procedure and the computation of non-termination witnesses, we provide a prototype implementation for the case $\mathcal{S} = \mathbb{A}$ in our tool SiRop (for "<u>Si</u>mple <u>R</u>andomized Lo<u>op</u>s"). The tool and a corresponding collection of exemplary programs can be obtained from

`https://github.com/aprove-developers/SiRop`

In particular, our tool managed to find a non-terminating algebraic input for one[3] of the only two problems from the category `C Integer` which were not solved by any tool at the 2023 Termination Competition [12],[4] the other one being the Collatz problem. While our tool only considers $\mathcal{S} = \mathbb{A}$ (whereas the problem is formulated over the integers), the constraints generated by SiRop are unsatisfiable over $\mathbb{Z}$, which implies universal termination of the program over the integers.

Related Work: We continue a line of research started in 2004 by Tiwari [27] who showed decidability of universal termination for loops with an affine guard and an affine update as its body, where the guard, updates, and inputs range over the real numbers. In his proof, Tiwari reduced the affine to the linear case. In 2006, Braverman [6] proved that the problem remains decidable for loops and inputs ranging over the rational numbers $\mathbb{Q}$, and if the guard and update are linear, then he also showed decidability over the integers $\mathbb{Z}$. Similar to Tiwari, Braverman also reduced the affine case for $\mathbb{Q}$ to the linear case. In 2015, Ouaknine et al. proved [25] that the affine case is decidable over the integers $\mathbb{Z}$ whenever the update is of the form $\vec{x} \leftarrow \mathbf{A}\vec{x} + \vec{a}$, provided that $\mathbf{A} \in \mathbb{Z}^{n\times n}$ is diagonalizable. This restriction was removed by Hosseini et al. in 2019 [15]. In a related line of work, we proved decidability of universal termination over the integers $\mathbb{Z}$ for triangular affine loops, i.e., where the matrix $\mathbf{A} \in \mathbb{Z}^{n\times n}$ is triangular [10]. Later, we extended these results to triangular weakly non-linear loops which extend triangular loops by allowing certain non-linear updates [13, 14].

The only decidability results for termination of randomized programs that we are aware of consider probabilistic vector addition systems [7] or constant probability programs [11]. The programs in [7, 11] are orthogonal to the ones considered in our approach as they can only modify the program variables by adding constants, but do not allow for multiplication. Another related area of research [3, 18, 23] deals with prob-solvable loops and moment

[2] In an affine (or non-homogeneous) loop, the guard may have the form $\mathbf{C}\vec{x} > \vec{c}$ and the update may have the form $\vec{x} \leftarrow \mathbf{A}\vec{x} + \vec{d}$ for arbitrary vectors $\vec{c}$ and $\vec{d}$.

[3] `https://github.com/TermCOMP/TPDB/blob/11.3/C_Integer/Stroeder_15/ChenFlurMukhopadhyay-SAS2012-Ex2.06_false-termination.c`

[4] This category was not part of the 2024 competition.

invariants. However, in contrast to our method, these techniques require that all variables in the loop guard may only take finitely many values. Moreover, there are many automated approaches for tackling UPAST using ranking supermartingales (RSM), e.g., [1, 2, 4, 8, 9, 19, 22, 24, 28]. To generate a suitable RSM, one often uses techniques based on affine or polynomial templates, which renders the approach incomplete.

2 Outline of our Approach

For a loop as in (1), we consider (finite) executions f corresponding to words over the alphabet $\{A, B\}$, where the i-th symbol in f indicates which update matrix was used in the i-th application of the assignment $\vec{x} \leftarrow \mathbf{A}\vec{x} \oplus_p \mathbf{B}\vec{x}$. For such executions f, $|f|_A$ and $|f|_B$ denote the number of A- and B-symbols in f, respectively. Moreover, we introduce the function $\mathcal{V}al_{\vec{x}}$ that maps finite executions f to the values $\mathcal{V}al_{\vec{x}}(f) \in \mathbb{R}^m$ of the constraints in the loop guard after executing f on a concrete input $\vec{x} \in \mathbb{R}^n$, i.e., $\mathcal{V}al_{\vec{x}}(f) = \mathbf{C} \cdot \mathbf{A}^{|f|_A} \cdot \mathbf{B}^{|f|_B} \cdot \vec{x}$, since $\mathbf{A}$ and $\mathbf{B}$ commute. Our decision procedure does not search for a non-terminating input directly, but for an *eventually* non-terminating input $\vec{x}$. An input $\vec{x}$ is eventually non-terminating if by repeated execution of the loop body on $\vec{x}$ (while ignoring the guard), a non-terminating input can eventually be reached. We show that such a loop has an eventually non-terminating input iff it also has a non-terminating input. One can then show how such an eventually non-terminating input can be lifted to an actual non-terminating input.

We introduce a mapping $\mathcal{U}$ that maps executions f to the difference between the relative number $\frac{|f|_A}{|f|}$ of times that the update matrix $\mathbf{A}$ has been chosen in f and the probability p of choosing $\mathbf{A}$, i.e., $\mathcal{U}(f) = \frac{|f|_A}{|f|} - p$. Moreover, we essentially partition the set of indices $\{1, \ldots, n\}$ of all program variables into suitable sets $\mathfrak{D}_{(\mathfrak{i},\mathfrak{o})}$ with $(\mathfrak{i}, \mathfrak{o}) \in \mathcal{I}$ for some finite set $\mathcal{I} \subsetneq \mathbb{R}^2_{>0}$. We then show that for all $c \in \{1, \ldots, m\}$, and all executions f with $|f|_A, |f|_B \geq 1$, the value $(\mathcal{V}al_{\vec{x}}(f))_c$ of the c-th constraint after executing f on $\vec{x}$ is

$$(\mathcal{V}al_{\vec{x}}(f))_c = \textstyle\sum_{(\mathfrak{i},\mathfrak{o})\in\mathcal{I}} \left(\mathfrak{i} \cdot \mathfrak{o}^{\mathcal{U}(f)}\right)^{|f|} \sum_{i\in\mathfrak{D}_{(\mathfrak{i},\mathfrak{o})}} \zeta_{i,\mathbf{A}}^{|f|_A}\, \zeta_{i,\mathbf{B}}^{|f|_B}\, \gamma_{c,i}(\vec{x}). \tag{2}$$

Here, all $\zeta_{i,\mathbf{A}}, \zeta_{i,\mathbf{B}}$ are complex numbers of modulus 1, i.e., $|\zeta_{i,\mathbf{A}}| = |\zeta_{i,\mathbf{B}}| = 1$ for all $i \in \{1, \ldots, n\}$, and the functions $\gamma_{c,i}$ are linear maps $\mathbb{R}^n \to \mathbb{C}$. The maps $\gamma_{c,i}$ and the values $\zeta_{i,\mathbf{A}}, \zeta_{i,\mathbf{B}} \in \mathbb{C}$ only depend on the matrices $\mathbf{C}$, $\mathbf{A}$, and $\mathbf{B}$, but not on the specific input $\vec{x}$. While weaker requirements would suffice to ensure that $(\mathcal{V}al_{\vec{x}}(f))_c$ has some closed form, diagonalizability of $\mathbf{A}$ and $\mathbf{B}$ guarantees that it has the form (2), which is crucial for our procedure. By a lexicographic comparison of those $(\mathfrak{i}, \mathfrak{o}) \in \mathcal{I}$ for which the inner sum in (2) is not 0 for all executions f, one can compute which of the pairs $(\mathfrak{i}, \mathfrak{o})$ is the "dominant" one. Here, a pair $(\mathfrak{i}, \mathfrak{o}) \in \mathcal{I}$ is considered dominant whenever the value of the first factor $\left(\mathfrak{i} \cdot \mathfrak{o}^{\mathcal{U}(f)}\right)^{|f|}$ of (2) grows the fastest if the execution of f is continued (i.e., if $|f| \to \infty$) and the corresponding inner sum is not 0 for all executions f. The dominant pair depends on the specific input $\vec{x}$ and on whether $\mathcal{U}(f)$ is $\mathfrak{p}$ositive or $\mathfrak{n}$egative, and correspondingly, one has to use different lexicographic comparisons to determine the dominant pair. For $d \in \{\mathfrak{n}, \mathfrak{p}\}$, let $\mathfrak{D}_{d,c,\vec{x}}$ denote the set $\mathfrak{D}_{(\mathfrak{i},\mathfrak{o})}$ where the pair $(\mathfrak{i}, \mathfrak{o})$ is dominant for input $\vec{x}$ and $c \in \{1, \ldots, m\}$ (and positive $\mathcal{U}(f)$ if $d = \mathfrak{p}$ or negative $\mathcal{U}(f)$ for $d = \mathfrak{n}$). Then, the sign of the "coefficient" $v(f) = \sum_{i\in\mathfrak{D}_{d,c,\vec{x}}} \zeta_{i,\mathbf{A}}^{|f|_A} \zeta_{i,\mathbf{B}}^{|f|_B} \gamma_{c,i}(\vec{x})$ of the dominant pair eventually determines the sign of $(\mathcal{V}al_{\vec{x}}(f))_c$, provided that $|v(f)|$ is large enough. This also indicates why an extension of our approach to programs with three instead of two update matrices would be problematic. Then instead of $\mathcal{U}(f)$ we would need a vector to express how much an execution deviates from the probabilities in the program. This would break the underlying concepts, since instead of $d \in \{\mathfrak{n}, \mathfrak{p}\}$, we would have to consider the "direction" of this deviation.

We then consider the rearrangement

$$v(f) \;=\; \underbrace{\textstyle\sum_{i\in\mathfrak{R}_{d,c,\vec{x}}} \gamma_{c,i}(\vec{x})}_{=R} \;+\; \textstyle\sum_{i\in\mathfrak{C}_{d,c,\vec{x}}} \zeta_{i,\mathbf{A}}^{|f|_A}\,\zeta_{i,\mathbf{B}}^{|f|_B}\,\gamma_{c,i}(\vec{x}) \qquad (3)$$

where $\mathfrak{R}_{d,c,\vec{x}} = \{i \in \mathfrak{D}_{d,c,\vec{x}} \mid \zeta_{i,\mathbf{A}} = \zeta_{i,\mathbf{B}} = 1\}$ and $\mathfrak{C}_{d,c,\vec{x}} = \{i \in \mathfrak{D}_{d,c,\vec{x}} \mid \{\zeta_{i,\mathbf{A}}, \zeta_{i,\mathbf{B}}\} \neq \{1\}\}$. This leads us to a necessary condition for non-termination (and hence a sufficient condition for universal termination): If $\vec{x}$ is an eventually non-terminating input, then there must be a $d \in \{\mathfrak{n}, \mathfrak{p}\}$ such that we have $R > 0$ for all constraints c, with R as in (3).

This necessary condition for non-termination is then turned into a sufficient condition. To that end, we define the set W of witnesses for eventual non-termination containing all inputs $\vec{x} \in \mathcal{S}^n$ for which there is some $d \in \{\mathfrak{n}, \mathfrak{p}\}$ such that $R > \sum_{i\in\mathfrak{C}_{d,c,\vec{x}}} |\gamma_{c,i}(\vec{x})|$ holds for all constraints c. All witnesses $\vec{x} \in W$ are eventually non-terminating. While this condition is only sufficient for non-termination, one shows that if the program admits a non-terminating input, then there is also an input in W. So the considered program is non-terminating iff $W \neq \varnothing$, i.e., the program is universally terminating iff $W = \varnothing$.

Finally, we obtain a decision procedure for the problem UPAST. Here, we use that W is semi-algebraic and thus the emptiness problem is decidable over the real algebraic numbers, i.e., if $\mathcal{S} = \mathbb{A}$. Moreover, W can be represented as a finite union of convex semi-algebraic sets. Hence, emptiness of W can also be decided over the rationals and integers [17]. Moreover, witnesses for non-termination can be obtained from eventually non-terminating inputs $\vec{x} \in W$.

References

1 Sheshansh Agrawal, Krishnendu Chatterjee, and Petr Novotný. Lexicographic ranking supermartingales: An efficient approach to termination of probabilistic programs. *Proc. ACM Program. Lang.*, 2(POPL):34:1–34:32, 2018. doi:10.1145/3158122.

2 Martin Avanzini, Georg Moser, and Michael Schaper. A modular cost analysis for probabilistic programs. *Proc. ACM Program. Lang.*, 4(OOPSLA), 2020. doi:10.1145/3428240.

3 Ezio Bartocci, Laura Kovács, and Miroslav Stankovic. Automatic generation of moment-based invariants for prob-solvable loops. In *Proc. ATVA '19*, LNCS 11781, pages 255–276, 2019. doi:10.1007/978-3-030-31784-3_15.

4 Kevin Batz, Mingshuai Chen, Sebastian Junges, Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. Probabilistic program verification via inductive synthesis of inductive invariants. In *Proc. TACAS '23*, LNCS 13994, pages 410–429, 2023. doi:10.1007/978-3-031-30820-8_25.

5 Olivier Bournez and Florent Garnier. Proving positive almost-sure termination. In *Proc. RTA '05*, LNCS 3467, pages 323–337, 2005. doi:10.1007/978-3-540-32033-3_24.

6 Mark Braverman. Termination of integer linear programs. In *Proc. CAV '06*, LNCS 4144, pages 372–385, 2006. doi:10.1007/11817963_34.

7 Tomás Brázdil, Krishnendu Chatterjee, Antonín Kucera, Petr Novotný, and Dominik Velan. Deciding fast termination for probabilistic VASS with nondeterminism. In *Proc. ATVA '19*, LNCS 11781, pages 462–478, 2019. doi:10.1007/978-3-030-31784-3_27.

8 Aleksandar Chakarov and Sriram Sankaranarayanan. Probabilistic program analysis with martingales. In *Proc. CAV '13*, LNCS 8044, pages 511–526, 2013. doi:10.1007/978-3-642-39799-8_34.

9 Krishnendu Chatterjee, Hongfei Fu, and Amir Kafshdar Goharshady. Termination analysis of probabilistic programs through positivstellensatz's. In *Proc. CAV '16*, LNCS 9779, pages 3–22, 2016. doi:10.1007/978-3-319-41528-4_1.

10 Florian Frohn and Jürgen Giesl. Termination of triangular integer loops is decidable. In *Proc. CAV '19*, LNCS 11562, pages 426–444, 2019. doi:10.1007/978-3-030-25543-5_24.

11 Jürgen Giesl, Peter Giesl, and Marcel Hark. Computing expected runtimes for constant probability programs. In *Proc. CADE '19*, LNCS 11716, pages 269–286, 2019. doi:10.1007/978-3-030-29436-6_16.

12 Jürgen Giesl, Albert Rubio, Christian Sternagel, Johannes Waldmann, and Akihisa Yamada. The termination and complexity competition. In *Proc. TACAS '19*, LNCS 11429, pages 156–166, 2019. Website of the Annual Termination Competition: https://termination-portal.org/wiki/Termination_Competition. doi:10.1007/978-3-030-17502-3_10.

13 Marcel Hark, Florian Frohn, and Jürgen Giesl. Polynomial loops: Beyond termination. In *Proc. LPAR '20*, EPiC 73, pages 279–297, 2020. doi:10.29007/nxv1.

14 Marcel Hark, Florian Frohn, and Jürgen Giesl. Termination of triangular polynomial loops. *Formal Methods in Syst. Des.*, 65(1):70–132, 2025. doi:10.1007/s10703-023-00440-z.

15 Mehran Hosseini, Joël Ouaknine, and James Worrell. Termination of linear loops over the integers. In *Proc. ICALP '19*, LIPIcs 132, 2019. doi:10.4230/LIPIcs.ICALP.2019.118.

16 Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. On the hardness of analyzing probabilistic programs. *Acta Informatica*, 56(3):255–285, 2019. URL: https://doi.org/10.1007/s00236-018-0321-1, doi:10.1007/S00236-018-0321-1.

17 Leonid Khachiyan and Lorant Porkolab. Computing integral points in convex semi-algebraic sets. In *Proc. FOCS '97*, pages 162–171, 1997. doi:10.1109/SFCS.1997.646105.

18 Andrey Kofnov, Marcel Moosbrugger, Miroslav Stankovic, Ezio Bartocci, and Efstathia Bura. Exact and approximate moment derivation for probabilistic loops with non-polynomial assignments. *ACM Trans. Model. Comput. Simul.*, 34(3):18:1–18:25, 2024. doi:10.1145/3641545.

19 Fabian Meyer, Marcel Hark, and Jürgen Giesl. Inferring expected runtimes of probabilistic integer programs using expected sizes. In *Proc. TACAS '21*, LNCS 12651, pages 250–269, 2021. doi:10.1007/978-3-030-72016-2_14.

20 Éléanore Meyer and Jürgen Giesl. Deciding termination of simple randomized loops. In *Proc. MFCS '25*, LIPIcs, 2025. To appear.

21 Éléanore Meyer and Jürgen Giesl. Deciding termination of simple randomized loops, 2025. Extended version of [20] with all proofs. URL: https://arxiv.org/abs/2506.18541, arXiv:2506.18541.

22 Marcel Moosbrugger, Ezio Bartocci, Joost-Pieter Katoen, and Laura Kovács. The probabilistic termination tool Amber. *Formal Methods Syst. Des.*, 61(1):90–109, 2022. URL: https://doi.org/10.1007/s10703-023-00424-z, doi:10.1007/S10703-023-00424-Z.

23 Marcel Moosbrugger, Miroslav Stankovic, Ezio Bartocci, and Laura Kovács. This is the Moment for Probabilistic Loops. *Proc. ACM Program. Lang.*, 6(OOPSLA2):1497–1525, 2022. doi:10.1145/3563341.

24 Van Chan Ngo, Quentin Carbonneaux, and Jan Hoffmann. Bounded expectations: Resource analysis for probabilistic programs. In *Proc. PLDI '18*, pages 496–512, 2018. doi:10.1145/3192366.3192394.

25 Joël Ouaknine, João Sousa Pinto, and James Worrell. On termination of integer linear loops. In *Proc. SODA '15*, pages 957–969, 2015. doi:10.1137/1.9781611973730.65.

26 Nasser Saheb-Djahromi. Probabilistic LCF. In *Proc. MFCS '78*, LNCS 64, pages 442–451, 1978. doi:10.1007/3-540-08921-7_92.

27 Ashish Tiwari. Termination of linear programs. In *Proc. CAV '04*, LNCS 3114, pages 70–82, 2004. doi:10.1007/978-3-540-27813-9_6.

28 Di Wang, David M. Kahn, and Jan Hoffmann. Raising expectations: Automating expected cost analysis with types. *Proc. ACM Program. Lang.*, 4(ICFP), 2020. doi:10.1145/3408992.