On Lexicographic Proof Rules for Probabilistic Termination

KRISHNENDU CHATTERJEE, IST Austria, Austria
EHSAN KAFSHDAR GOHARSHADY, Ferdowsi University of Mashhad, Iran
PETR NOVOTNÝ and JIŘÍ ZÁREVÚCKY, Masaryk University, Czech Republic
ĐORĐE ŽIKELIĆ, IST Austria, Austria

We consider the almost-sure (a.s.) termination problem for probabilistic programs, which are a stochastic extension of classical imperative programs. Lexicographic ranking functions provide a sound and practical approach for termination of non-probabilistic programs, and their extension to probabilistic programs is achieved via lexicographic ranking supermartingales (LexRSMs). However, LexRSMs introduced in the previous work have a limitation that impedes their automation: all of their components have to be non-negative in all reachable states. This might result in a LexRSM not existing even for simple terminating programs. Our contributions are twofold. First, we introduce a generalization of LexRSMs that allows for some components to be negative. This standard feature of non-probabilistic termination proofs was hitherto not known to be sound in the probabilistic setting, as the soundness proof requires a careful analysis of the underlying stochastic process. Second, we present polynomial-time algorithms using our generalized LexRSMs for proving a.s. termination in broad classes of linear-arithmetic programs.

CCS Concepts: • **Software and its engineering → Formal software verification**; **Software verification**; **Automated static analysis**; • **Mathematics of computing → Probability and statistics**; • **Theory of computation → Program analysis;**

Additional Key Words and Phrases: Probabilistic programs, termination, martingales

ACM Reference format:
Krishnendu Chatterjee, Ehsan Kafshdar Goharshady, Petr Novotný, Jiří Zárevúcky, and Đorđe Žikelić. 2023. On Lexicographic Proof Rules for Probabilistic Termination. *Form. Asp. Comput.* 35, 2, Article 11 (June 2023), 25 pages.
https://doi.org/10.1145/3585391

1 INTRODUCTION

The extension of classical imperative programs with randomization gives rise to probabilistic programs (PPs) [Gordon et al. 2014] that are used in numerous applications, including stochastic network protocols [Baier and Katoen 2008; Foster et al. 2016; Kwiatkowska et al. 2011; Smolka

The preliminary version of this article has been published in Chatterjee et al. [2021a].
This research was partially supported by the ERC CoG (grant no. 863818; ForM-SMArt), the Czech Science Foundation (grant no. GA21-24711S), and the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie Grant Agreement No. 665385.
Authors' addresses: K. Chatterjee and Đ. Žikelić, IST Austria, Am Campus 1, Klosterneuburg, 3400, Austria; emails: {krishnendu.chatterjee, djordje.zikelic}@ist.ac.at; E. K. Goharshady, Ferdowsi University of Mashhad, Azadi Square, 9177948974, Mashhad, Iran; email: e.kafshdargoharshady@mail.um.ac.ir; P. Novotný and J. Zárevúcky, Masaryk University, Žerotínovo nám. 617/9, Brno, 601 77, Czech Republic; emails: {petr.novotny, xzarevuc}@fi.muni.cz.

0934-5043/2023/06-ART11 $15.00
https://doi.org/10.1145/3585391

et al. 2017], randomized algorithms [Dubhashi and Panconesi 2009; Motwani and Raghavan 1995], security [Barthe et al. 2016b, c], machine learning, and planning [Claret et al. 2013; Ghahramani 2015; Gordon et al. 2013; Kaelbling et al. 1996; Roy et al. 2008; Ścibior et al. 2015; Thrun 2002]. The analysis of PPs is an active research area in formal methods [Agrawal et al. 2018; Chakarov and Sankaranarayanan 2013; Chatterjee et al. 2016, 2018; Esparza et al. 2012; Kaminski et al. 2018b, a; Ngo et al. 2018; Olmedo et al. 2016; Wang et al. 2018]. PPs can be extended with nondeterminism to allow over-approximating program parts that are too complex for static analysis [Cousot and Cousot 1977; McIver and Morgan 2005].

For non-probabilistic programs, the *termination* problem asks whether a given program *always* terminates. While the problem is well known to be undecidable over Turing-complete programs, many sound automated techniques that work well for practical programs have been developed [Cook et al. 2006, 2011]. Such techniques typically seek a suitable *certificate* of termination. Particularly relevant certificates are *ranking functions (RFs)* [Bradley et al. 2005; Colón and Sipma 2001; Floyd 1967; Podelski and Rybalchenko 2004a, b; Sohn and Gelder 1991], mapping program states into a well-founded domain, forcing a strict decrease of the function value in every step. The basic ranking functions are 1-dimensional, which is often insufficient for complex control-flow structures. Lexicographic ranking functions (LexRFs) are multidimensional extensions of RFs that provide an effective approach to termination analysis [Alias et al. 2010; Bradley et al. 2005; Brockschmidt et al. 2013, 2016; Cook et al. 2013; Gonnord et al. 2015]. The literature typically restricts to linear LexRFs for linear-arithmetic (LA) programs, as LA reasoning can be more efficiently automated compared with non-linear arithmetic.

For probabilistic programs, the termination problem considers aspects of the probabilistic behaviors as well. The most fundamental is the *almost-sure (a.s.)* termination problem, which asks whether a given PP terminates with probability 1. One way of proving a.s. termination is via a probabilistic analogue of ranking functions named *ranking supermartingales (RSMs)* due to the connection with (super)martingale stochastic processes [Williams 1991]. There is a rich body of work on 1-dimensional RSMs whereas the work by Agrawal et al. [2018] introduces lexicographic RSMs. In probabilistic programs, a transition τ available in some state s yields a probability distribution over the successor states. The conditions defining RSMs are formulated in terms of the expectation operator $\mathbb{E}^\tau$ of this distribution. In particular, *lexicographic ranking supermartingales* (LexRSMs) of Agrawal et al. [2018] are functions f mapping program states to $\mathbb{R}^d$, such that for each transition τ there exists a component $1 \leq i \leq d$, satisfying, for any reachable state s at which τ is enabled, the following conditions *P-RANK* and *S-NNEG* (with f_i the i-component of f and $s \models G(\tau)$ denoting the fact that s satisfies the guard of τ):

(1) $\textit{P-RANK}(f,\tau) \equiv s \models G(\tau) \Rightarrow \Big(\mathbb{E}^\tau[f_i(s')] \leq f_i(s) - 1 \text{ and } \mathbb{E}^\tau[f_j(s')] \leq f_j(s) \text{ for all } 1 \leq j < i\Big)$.
(2) $\textit{S-NNEG}(f,\tau) \equiv s \models G(\tau) \Rightarrow \Big(f_j(s) \geq 0 \text{ for all } 1 \leq j \leq d\Big)$.

(We use the standard primed notation from program analysis, that is, s' is the probabilistically chosen successor of s when performing τ.) The *P-RANK* condition enforces an expected decrease in lexicographic ordering, while *S-NNEG* stands for "strong non-negativity." Proving the soundness of LexRSMs for proving a.s. termination is highly non-trivial and requires reasoning about complex stochastic processes [Agrawal et al. 2018]. Apart from the soundness proof, Agrawal et al. [2018] also presents an algorithm for the synthesis of linear LexRSMs.

While LexRSMs improved the applicability of a.s. termination proving, their usage is impeded by the *restrictiveness of strong non-negativity* due to which a linear LexRSM might not exist even for simple a.s. terminating programs. This is a serious drawback from the automation perspective, since even if such a program admits a non-linear LexRSM, efficient automated tools that restrict to linear-arithmetic reasoning would not be able to find it.

```
ℓ0 : while y ≥ 0 do
         x := y;
ℓ1 :     while x ≥ 0 do
             x := x − 1 + Norm(0, 1)
         od;
         y := y − 1
     od
ℓout :
```

(a)

```
ℓ0 : while x ≥ 0 do
       if y ≥ 0 then
         y := y + Uni[−7, 1]
       else
         x := x + Uni[−7, 1] ;
ℓ1 :   y := y + Uni[−7, 1]
       fi od
ℓout :
```

(b)

Fig. 1. Motivating examples. $Norm(\mu, \sigma)$ samples from the normal distribution with mean μ and standard deviation σ. $Uni[a, b]$ samples uniformly from the interval $[a, b]$. Location labels are the "ℓ_i": one location per loop head and one additional location in (b) to have one assignment per transition (a technical requirement for our approach). In addition, the location label "ℓ_{out}" denotes the terminal location in each program. A formal representation of the programs via *probabilistic control flow graphs* is presented later, in Section 4.

Consider the program in Figure 1(a). By employing simple random-walk arguments, we can manually prove that the program terminates a.s. A linear LexRSM proving this needs to have a component containing a positive multiple of x at the head of the inner while-loop (ℓ_1). However, due to the sampling from the normal distribution, which has unbounded support, the value of x inside the inner loop cannot be bounded from below. Hence, the program does not admit a linear LexRSM. In general, the existing 1-dimensional variants of ranking supermartingales [Chakarov and Sankaranarayanan 2013; Huang et al. 2019] as well as LexRSMs with strong non-negativity do not handle programs with unbounded-support distributions well, as they all require their components to be non-negative in all reachable states at which some transition is enabled. The strong non-negativity condition is too restrictive for automated methods that reason over linear arithmetic, and existing methods that reason over linear arithmetic cannot prove even that the inner loop of the program in Figure 1(a) (also shown in Figure 2) terminates a.s.

Now, consider the program in Figure 1(b). Again, it can be shown that this PP terminates a.s. However, this cannot be witnessed by a linear LexRSM: to rank the "if-branch" transition, there must be a component with a positive multiple of y in ℓ_0. However, y can become arbitrarily negative within the else branch and cannot be bounded from below by a linear function of x.

Contributions. The contributions of this work are as follows:

(1) *Generalized Lexicographic RSMs.* In the non-probabilistic setting, strong non-negativity can be relaxed to *partial non-negativity* (*P-NNEG*), in which only the components to the left of the "ranking component" i (inclusive) need to be non-negative (Ben-Amram–Genaim RFs [Ben-Amram and Genaim 2015]). We show that in the probabilistic setting, the same relaxation is possible under additional *expected leftward non-negativity* constraint (*EXP-NNEG*). Formally, we say that f is a *generalized lexicographic ranking supermartingale* (GLexRSM) if for any transition τ there is $1 \leq i \leq d$ such that for any reachable state s at which τ is enabled we have $\textit{P-RANK}(f, \tau) \wedge \textit{P-NNEG}(f, \tau) \wedge \textit{EXP-NNEG}(f, \tau)$, where

$$\begin{aligned} \textit{P-NNEG}(f, \tau) \quad &\equiv \quad s \models G(\tau) \Rightarrow \Big(f_j(s) \geq 0 \text{ for all } 1 \leq j \leq i\Big) \\ \textit{EXP-NNEG}(f, \tau) \quad &\equiv \quad s \models G(\tau) \Rightarrow \Big(\mathbb{E}^{\tau}[f_j(s') \cdot \mathbb{I}_{<j}(s')] \geq 0 \text{ for all } 1 \leq j \leq i\Big), \end{aligned}$$

with $\mathbb{I}_{<j}$ being the indicator function of the set of all states in which a transition ranked by a component $< j$ is enabled.

ℓ_0 : **while** $x \geq 0$ **do**
$x := x - 1 + Norm(0, 1)$
od
ℓ_{out} :

Fig. 2. A simple loop that terminates almost-surely and that involves sampling from the standard normal distribution, which has unbounded support. However, no existing martingale-based method that reasons over linear arithmetic can prove a.s. termination of this loop.

We first formulate GLexRSMs as an abstract proof rule for general stochastic processes. We then instantiate them into the setting of probabilistic programs and define *GLexRSM maps,* which we prove to be sound for proving a.s. termination. These results are general and *not specific* to linear-arithmetic programs.

(2) *Polynomial Algorithms for Linear GLexRSMs.* We present two algorithms:
   (a) For linear-arithmetic PPs in which sampling instructions use bounded-support distributions, we show that the problem LinGLexPP of deciding whether a given PP with a given set of *linear invariants* admits a linear GLexRSM is decidable in polynomial time. Also, our algorithm computes the witnessing linear GLexRSM whenever it exists. Our approach proves the a.s. termination of the program in Figure 1(b).
   (b) Building on the results of item 1, we construct a sound polynomial-time algorithm for a.s. termination proving in PPs that *do perform* sampling from *unbounded-support* distributions. The algorithm proves a.s. termination for our motivating example in Figure 1(a).

(3) *First linear-arithmetic martingale-based method for unbounded-support distributions.* Finally, while the focus of our work is on relaxing the restrictive strong non-negativity assumption of LexRSMs, we remark that our theoretical and algorithmic results also yield the first automated method that operates over *linear arithmetic* and can prove a.s. termination in probabilistic programs in which termination depends on sampling instructions from *double-sided unbounded support* probability distributions. For instance, termination behavior of the simple loop in Figure 2 is determined by values sampled from the standard normal distribution. Any linear-arithmetic martingale-based certificate admitted by this program would need to have a vanishing linear coefficient of the sampled variable for the certificate to be non-negative. Hence, this program does not admit any martingale-based certificate that imposes the strong non-negativity condition. To the best of our knowledge, the only exception is the descent supermartingale maps (DSMs) of Huang et al. [2019], which replace the strong non-negativity assumption by the bounded difference condition that requires bounded maximal change in value at every program state. However, the program in Figure 2 does not admit a linear-arithmetic DSM either, as the sampled value from the standard normal distribution is unbounded; therefore, the linear coefficient of the sampled variable in the DSM would need to be 0. Since our method relaxes the strong non-negativity assumption while not introducing the bounded difference condition, it presents the first method that operates over linear arithmetic and can prove a.s. termination of the PP presented in Figure 2.

Related work. Martingale-based termination literature mostly focused on 1-dimensional RSMs [Chakarov and Sankaranarayanan 2013; Chatterjee et al. 2016, 2018, 2017; Fioriti and Hermanns 2015; Fu and Chatterjee 2019; Giesl et al. 2019; Huang et al. 2018; McIver and Morgan 2016; McIver et al. 2018; Moosbrugger et al. 2021]. RSMs themselves can be seen as generalizations of Lyapunov ranking functions from control theory [Bournez and Garnier 2005; Foster 1953]. Recently, the work [Huang et al. 2019] pointed out the unsoundness of the 1-dimensional RSM-based proof rule in Fioriti and Hermanns [2015] due to insufficient lower bound conditions and provided a corrected

version. On the multidimensional front, it was shown in Fioriti and Hermanns [2015] that requiring components of (lexicographic) RSMs to be non-negative only at points where they are used to rank some enabled transition (analogue of Bradley-Manna-Sipma LexRFs [Bradley et al. 2005]) is unsound for proving a.s. termination. This illustrates the intricacies of dealing with lower bounds in the design of a.s. termination certificates. Lexicographic RSMs with strong non-negativity were introduced in Agrawal et al. [2018]. The work of Chen and He [2020] produces an ω-regular decomposition of a program's control-flow graph, with each program component ranked by a different RSM. This approach does not require a lexicographic ordering of RSMs. However, each component in the decomposition must be ranked by a single-dimensional non-negative RSM. RSM approaches were also used for cost analysis [Avanzini et al. 2020b; Ngo et al. 2018; Wang et al. 2019] and additional liveness and safety properties [Barthe et al. 2016a; Chakarov et al. 2016; Chatterjee et al. 2022, 2017].

Logical calculi for reasoning about properties of PPs (including termination) were studied in Feldman [1984], Feldman and Harel [1982], and Kozen [1981, 1983] and extended to programs with non-determinism in Gretz et al. [2014], Kaminski et al. [2018b], McIver and Morgan [2004, 2005], and Olmedo et al. [2016]. McIver and Morgan [2004, 2005] and McIver et al. [2018] formalize RSM-like proof certificates within the *weakest pre-expectation (WPE)* calculus [Morgan and McIver 1999; Morgan et al. 1996]. The power of this calculus allows for reasoning about complex programs [McIver et al. 2018, Section 5]; however, the proofs typically require human input. Theoretical connections between martingales and the WPE calculus were recently explored in Hark et al. [2020]. There is also a rich body of work on analysis of probabilistic functional programs, in which the aim is typically to obtain a general type of system [Avanzini et al. 2019; Dal Lago et al. 2021; Kobayashi et al. 2020; Lago and Grellois 2019] for reasoning about termination properties (automation for discrete probabilistic term rewrite systems is shown in Avanzini et al. [2020a]).

As for other approaches to a.s. termination, for *finite-state programs* with nondeterminism a sound and complete method was given in Esparza et al. [2012], while [Monniaux 2001] considers a.s. termination proving through abstract interpretation. The work of Kaminski et al. [2018a] shows that proving a.s. termination is harder (in terms of arithmetical hierarchy) than proving termination of non-probabilistic programs.

The computational complexity of the construction of lexicographic ranking functions in non-probabilistic programs was studied in Ben-Amram and Genaim [2013, 2015].

New Material. This article is an extended version of the article [Chatterjee et al. 2021a] with the following additional material:

- The current article provides much more technical detail, with proof overviews followed by detailed proofs. We include proofs of Lemma 6.1, Theorem 6.2, and a sketch proof of Lemma 6.4 that underline the soundness of our automated approach. These proofs also show why our automated approach is restricted to linear-arithmetic programs and what would be the challenges in automating the synthesis of GLexRSMs for programs with non-linear arithmetic, which is an interesting direction of future work.
- We provide further details on constraint encoding in Section 6 that were omitted from the conference version, and we provide a pseudocode for Algorithm 1.
- We expand Section 5 with the statements of Proposition 5.5 and Definition 5.6. These are two results on probabilistic programs with non-determinism that we established in the proof of Theorem 5.4 but that were hitherto not studied in the PP analysis literature. In the conference version, we only briefly mentioned these results. Here, we state them formally and present the key ideas behind the proof of Proposition 5.5.
- Finally, we emphasize the applicability of our method to PPs with sampling instructions from double-sided unbounded support probability distributions, which was not sufficiently

discussed in the conference version, and we provide an example program in Figure 2 to demonstrate this contribution. Our algorithm for synthesizing linear GLexRSMs provides the first automated martingale-based method that operates over linear arithmetic and can prove a.s. termination of the simple loop in Figure 2.

Article organization. The article is split into two parts: the first is "abstract," with mathematical preliminaries (Section 2) and definition and soundness proof of abstract GLexRSMs (Section 3). We also present an example showing that GLexRSMs without the expected leftward non-negativity constraint are not sound. The second part covers application to PPs: preliminaries on the program syntax and semantics (Section 4), a GLexRSM-based proof rule for a.s. termination (Section 5), and the outline of our algorithms (Section 6).

2 MATHEMATICAL PRELIMINARIES

We use boldface notation for vectors, for example, $\mathbf{x}$, $\mathbf{y}$, and so on, and we denote an i-th component of a vector $\mathbf{x}$ by $\mathbf{x}[i]$. For an n-dimensional vector $\mathbf{x}$, index $1 \leq i \leq n$, and number a, we denote by $\mathbf{x}(i \leftarrow a)$ a vector $\mathbf{y}$ such that $\mathbf{y}[i] = a$ and $\mathbf{y}[j] = \mathbf{x}[j]$ for all $1 \leq j \leq n, j \neq i$. For two real numbers a and b, we use $a \cdot b$ to denote their product.

We assume familiarity with basics of probability theory [Williams 1991]. A *probability space* is a triple $(\Omega, \mathcal{F}, \mathbb{P})$, where Ω is a *sample space*, $\mathcal{F}$ is a *sigma-algebra* of measurable sets over Ω, and $\mathbb{P}$ is a *probability measure* on $\mathcal{F}$. A *random variable (r.v.)* $R : \Omega \to \mathbb{R} \cup \{\pm\infty\}$ is an $\mathcal{F}$*-measurable* real-valued function (i.e., $\{\omega \mid R(\omega) \leq x\} \in \mathcal{F}$ for all $x \in \mathbb{R}$); we denote by $\mathbb{E}[R]$ its *expected value*. A *random vector* is a vector whose every component is a random variable. We denote by $\mathbf{X}[j]$ the j-component of a random vector $\mathbf{X}$. A (discrete time) *stochastic process* in a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ is an infinite sequence of random vectors in this space. We will also use random variables of the form $R\colon \Omega \to A$ for some finite or countable set A, which easily translates to the real-valued variables.

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and let X be a random variable. A *conditional expectation* of X given a sub-sigma algebra $\mathcal{F}' \subseteq \mathcal{F}$ is any real-valued random variable Y such that (i) Y is $\mathcal{F}'$-measurable and (ii) for each set $A \in \mathcal{F}'$ it holds that $\mathbb{E}[X \cdot \mathbb{I}(A)] = \mathbb{E}[Y \cdot \mathbb{I}(A)]$. Here, $\mathbb{I}(A)\colon \Omega \to \{0, 1\}$ is an *indicator function* of A, that is, function returning 1 for each $\omega \in A$ and 0 for each $\omega \in \Omega \setminus A$.

It is known from Ash and Doléans-Dade [2000] that a random variable satisfying the properties of conditional expectation exists whenever (a) $\mathbb{E}[|X|] < \infty$, that is, X is *integrable*, or (b) X is real valued and non-negative (though these two conditions are not necessary). Moreover, whenever the conditional expectation exists, it is also known to be a.s. unique. We denote this a.s. unique conditional expectation by $\mathbb{E}[X|\mathcal{F}']$. It holds that for any $\mathcal{F}'$-measurable bounded r.v. Z, we have that $\mathbb{E}[X \cdot Z|\mathcal{F}'] = \mathbb{E}[X|\mathcal{F}'] \cdot Z$ whenever the former conditional expectation exists [Williams 1991, Theorem 9.7(j)].

A *filtration* in $(\Omega, \mathcal{F}, \mathbb{P})$ is an increasing (with regard to set inclusion) sequence $\{\mathcal{F}_t\}_{t=0}^{\infty}$ of sub-sigma-algebras of $\mathcal{F}$. A *stopping time* with regard to a filtration $\{\mathcal{F}_t\}_{t=0}^{\infty}$ is a random variable T taking values in $\mathbb{N} \cup \{\infty\}$ such that for every t the set $\{T = t\} = \{\omega \in \Omega \mid T(\omega) = t\}$ belongs to $\mathcal{F}_t$. Intuitively, T returns a timestep in which some process should be "stopped." The decision to stop is made solely on the information available at the current step.

3 GENERALIZED LEXICOGRAPHIC RANKING SUPERMARTINGALES

In this section, we introduce *generalized lexicographic ranking supermartingales (GLexRSMs):* an abstract concept that is not necessarily connected to PPs, but which is crucial for the soundness of our new proof rule for a.s. termination.

Definition 3.1 (Generalized Lexicographic Ranking Supermartingale). Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and let $(\mathcal{F}_t)_{t=0}^{\infty}$ be a filtration of $\mathcal{F}$. Suppose that T is a stopping time with regard to $\mathcal{F}$. An n-dimensional real-valued stochastic process $(\mathbf{X}_t)_{t=0}^{\infty}$ is a *generalized lexicographic ranking supermartingale for* T (GLexRSM) if:

(1) For each $t \in \mathbb{N}_0$ and $1 \leq j \leq n$, the random variable $\mathbf{X}_t[j]$ is $\mathcal{F}_t$-measurable.
(2) For each $t \in \mathbb{N}_0$, $1 \leq j \leq n$, and $A \in \mathcal{F}_{t+1}$, the conditional expectation $\mathbb{E}[\mathbf{X}_{t+1}[j] \cdot \mathbb{I}(A) \mid \mathcal{F}_t]$ exists.
(3) For each $t \in \mathbb{N}_0$, there exists a partition of the set $\{T > t\}$ into n subsets $L_1^t, \ldots, L_n^t$, all of them $\mathcal{F}_t$-measurable (i.e., belonging to $\mathcal{F}_t$), such that for each $1 \leq j \leq n$
   - $\mathbb{E}[\mathbf{X}_{t+1}[j] \mid \mathcal{F}_t](\omega) \leq \mathbf{X}_t[j](\omega)$ for each $\omega \in \cup_{j'=j}^{n} L_{j'}^t$,
   - $\mathbb{E}[\mathbf{X}_{t+1}[j] \mid \mathcal{F}_t](\omega) \leq \mathbf{X}_t[j](\omega) - 1$ for each $\omega \in L_j^t$,
   - $\mathbf{X}_t[j](\omega) \geq 0$ for each $\omega \in \cup_{j'=j}^{n} L_{j'}^t$,
   - $\mathbb{E}[\mathbf{X}_{t+1}[j] \cdot \mathbb{I}(\cup_{j'=0}^{j-1} L_{j'}^{t+1}) \mid \mathcal{F}_t](\omega) \geq 0$ for each $\omega \in \cup_{j'=j}^{n} L_{j'}^t$, with $L_0^{t+1} = \{T \leq t+1\}$.

Intuitively, we may think of each $\omega \in \Omega$ as a trajectory of process that evolves over time (in the second part of our article, this will be a PP run). Then, $\mathbf{X}_t$ is a vector function depending on the first t timesteps (each $\mathbf{X}_t[j]$ is $\mathcal{F}_t$-measurable), while T is the time at which the trajectory is stopped. Then, in point 3 of the definition, the first two items encode the expected (conditional) lexicographic decrease of $\mathbf{X}_t$, the third item encodes non-negativity of components to the left (inclusive) of the one that "ranks" ω in step t, and the last item encodes the expected leftward non-negativity (sketched in Section 1). For each $1 \leq j \leq n$ and timestep $t \geq 0$, the set L_j^t contains all $\omega \in \{T > t\}$ that are "ranked" by the component j at time t. An *instance* of an n-dimensional GLexRSM $\{\mathbf{X}_t\}_{t=0}^{\infty}$ is a tuple $(\mathbf{X}_{t=0}^{\infty}, \{L_1^t, \ldots, L_n^t\}_{t=0}^{\infty})$, where the second component is a sequence of partitions of Ω satisfying the condition in Definition 3.1. We say that $\omega \in \Omega$ has *level* j in step t of the instance $((\mathbf{X}_t)_{t=0}^{\infty}, (L_1^t, \ldots, L_n^t)_{t=0}^{\infty})$ if $T(\omega) > t$ and $\omega \in L_j^t$. If $T(\omega) \leq t$, we say that the level of ω at step t is 0.

We now state the main theorem of this section, which underlies the soundness of our new method for proving a.s. termination.

Theorem 3.2. *Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, $(\mathcal{F}_t)_{t=0}^{\infty}$ a filtration of $\mathcal{F}$, and T a stopping time with regard to $\mathcal{F}$. If there is an instance $((\mathbf{X}_t)_{t=0}^{\infty}, (L_1^t, \ldots, L_n^t)_{t=0}^{\infty})$ of a GLexRSM over $(\Omega, \mathcal{F}, \mathbb{P})$ for T, then $\mathbb{P}[T < \infty] = 1$.*

In Agrawal et al. [2018], a mathematical notion of LexRSMs is defined and a result for LexRSMs analogous to our Theorem 3.2 is established. Thus, the first part of our proof mostly resembles the proof of Theorem 3.3. in Agrawal et al. [2018], up to the point of defining the stochastic process $(Y_t)_{t=0}^{\infty}$ in Equation (1). After that, the proof of Agrawal et al. [2018] crucially relies on non-negativity of each $\mathbf{X}_t[j]$ and Y_t at every $\omega \in \Omega$ that is guaranteed by LexRSMs, and it cannot be adapted to the case of GLexRSMs. Below, we first show that, for GLexRSMs, $\mathbb{E}[Y_t] \geq 0$ for each $t \geq 0$; then, we present a very elegant argument via the Borel-Cantelli lemma [Williams 1991, Theorem 2.7] that shows that this boundedness of expectation is sufficient for the theorem claim to hold.

Proof of Theorem 3.2. We proceed by contradiction. Suppose that there exists an instance of a GLexRSM but that $\mathbb{P}[T = \infty] > 0$. First, we claim that there exists $1 \leq k \leq n$ and $s, M \in \mathbb{N}_0$ such that the set B of all $\omega \in \Omega$ for which the following properties hold has positive measure, that is, $\mathbb{P}[B] > 0$: (1) $T(\omega) = \infty$; (2) $\mathbf{X}_s[k](\omega) \leq M$; (3) for each $t \geq s$, the level of ω at step t is at least k; and (4) the level of ω equals k infinitely many times.

The claim is proved by several applications of the union bound. For each $\omega \in \Omega$, we define minlev(ω) to be the smallest $0 \leq j \leq n$ such that the level of ω is equal to j in infinitely many steps i. Let $B_k = \{\omega \in \Omega \mid T(\omega) = \infty \wedge \mathit{min\text{-}lev}(\omega) = k\}$ for each $1 \leq k \leq n$. Then,

$$\{\omega \in \Omega \mid T(\omega) = \infty\} = \bigcup_{k=1}^{n} B_k.$$

Thus, by the union bound, there exists $1 \leq k \leq n$ for which $\mathbb{P}[B_k] > 0$. We may express B_k as a union of events over the time of the last visit to some L_j^i with $j < k$. If we write $B_k^s = \{\omega \in \Omega \mid T(\omega) = \infty \wedge (i \geq s \Rightarrow \omega \in \cup_{j=k}^{n} L_j^i\}$ for each $s \in \mathbb{N}_0$, we have that $B_k = \cup_{s \geq 0} B_k^s$. As by the union bound $\mathbb{P}[B_k] \leq \sum_{s=0}^{\infty} \mathbb{P}[B_k^s]$, there exists $s \in \mathbb{N}_0$ for which $\mathbb{P}[B_k^s] > 0$. Now, for each $M \in \mathbb{N}_0$, let $B_k^{s,M}$ be defined via

$$B_k^{s,M} = \left\{\omega \in B_k^s \,\middle|\, \mathbf{X}_s[k] \leq M\right\}.$$

Then, $B_k^s = \cup_{M=0}^{\infty} B_k^{s,M}$. By the union bound, we have that $\mathbb{P}[B_k^s] \leq \sum_{M=0}^{\infty} \mathbb{P}[B_k^{s,M}]$, and there exists $M \in \mathbb{N}_0$ such that $\mathbb{P}[B_k^{s,M}] > 0$. The set $B = B_k^{s,M}$ satisfies the conditions of the claim.

Since B is defined in terms of tail properties of ω ("level is at least k *infinitely many times*"), it is not necessarily $\mathcal{F}_t$-measurable for any t. Hence, we define a stochastic process $(Y_t)_{t=0}^{\infty}$ such that each Y_t is $\mathcal{F}_t$-measurable, and which satisfies the desirable properties of $(\mathbf{X}_t[k])_{t=0}^{\infty}$ on B.

Let $D = \{\omega \in \Omega \mid \mathbf{X}_s[k](\omega) \leq M \wedge \omega \in \cup_{j=k}^{n} L_j^s\}$. Note that D is $\mathcal{F}_t$-measurable for $t \geq s$. We define a stopping time F with regard to $(\mathcal{F}_t)_{t=0}^{\infty}$ via $F(\omega) = \inf\{t \geq s \mid \omega \notin \cup_{j'=k}^{n} L_{j'}^t\}$, then a stochastic process $(Y_t)_{t=0}^{\infty}$ via

$$Y_t(\omega) = \begin{cases} 0, & \text{if } \omega \notin D, \\ M, & \text{if } \omega \in D, \text{ and } t < s, \\ \mathbf{X}_t[k](\omega), & \text{if } \omega \in D,\ t \geq s \text{ and } F(\omega) > t, \\ \mathbf{X}_{F(\omega)}[k](\omega), & \text{else.} \end{cases} \tag{1}$$

A straightforward argument (presented in the extended version of the paper [Chatterjee et al. 2021b]) shows that for each $t \geq s$ we have that $\mathbb{E}[Y_{t+1}] \leq \mathbb{E}[Y_t] - \mathbb{P}[L_k^t \cap D \cap \{F > t\}]$. By a simple induction, we obtain that

$$\mathbb{E}[Y_s] \geq \mathbb{E}[Y_t] + \sum_{r=s}^{t-1} \mathbb{P}\left[L_k^r \cap D \cap \{F > r\}\right]. \tag{2}$$

Now, we show that $\mathbb{E}[Y_t] \geq 0$ for each $t \in \mathbb{N}_0$. The claim is clearly true for $t < s$; thus, suppose that $t \geq s$. We can then expand $\mathbb{E}[Y_t]$ as follows:

$$\begin{aligned} \mathbb{E}[Y_t] &= \mathbb{E}[Y_t \cdot \mathbb{I}(F = s)] + \sum_{r=s+1}^{t} \mathbb{E}[Y_t \cdot \mathbb{I}(F = r)] + \mathbb{E}[Y_t \cdot \mathbb{I}(F > t)] \\ &\quad (Y_s \geq 0 \text{ as } D \subseteq \cup_{j=k}^{n} L_j^s \text{ and } Y_t(\omega) \geq 0 \text{ whenever } F(\omega) > t) \\ &\geq \sum_{r=s+1}^{t} \mathbb{E}[Y_t \cdot \mathbb{I}(F = r)] = \sum_{r=s+1}^{t} \mathbb{E}[Y_t \cdot \mathbb{I}(\{F = r\} \cap D)] \\ &\quad (Y_t(\omega) = \mathbf{X}_{F(\omega)}[k](\omega) \text{ whenever } \omega \in D,\ t \geq s \text{ and } F(\omega) \leq t) \\ &= \sum_{r=s+1}^{t} \mathbb{E}\left[\mathbf{X}_r[k] \cdot \mathbb{I}\left(\bigcup_{j=0}^{k-1} L_j^r\right) \cdot \mathbb{I}(\{F > r-1\} \cap D)\right] \\ &\quad (\text{properties of cond. exp. \& } \mathbb{I}(\{F > r-1\} \cap D) \text{ is } \mathcal{F}_{r-1}\text{-measurable}) \end{aligned}$$

$$= \sum_{r=s+1}^{t} \mathbb{E}\left[\mathbb{E}\left[\mathbf{X}_r[k] \cdot \mathbb{I}\left(\bigcup_{j=0}^{k-1} L_j^r\right) \middle| \mathcal{F}_{r-1}\right] \cdot \mathbb{I}(\{F > r-1\} \bigcap D)\right] \geq 0$$

$$\left(\mathbb{E}\left[\mathbf{X}_r[k] \cdot \mathbb{I}\left(\bigcup_{j=0}^{k-1} L_j^r\right) \middle| \mathcal{F}_{r-1}\right](\omega) \geq 0 \text{ for } \omega \in \{F > r-1\} \subseteq \cup_{j=k}^{n} L_j^{r-1}\right).$$

Plugging into Equation (2) that $\mathbb{E}[Y_t] \geq 0$, we get that $\mathbb{E}[Y_s] \geq \sum_{r=s}^{t-1} \mathbb{P}[L_k^r \cap D \cap \{F > r\}]$ for each $t \geq s$. By letting $t \to \infty$, we conclude that $\mathbb{E}[Y_s] \geq \sum_{r=s}^{\infty} \mathbb{P}[L_k^r \cap D \cap \{F > r\}]$. As $Y_s \leq M$ and $Y_s = 0$ outside D, we know that $\mathbb{E}[Y_s] \leq M \cdot \mathbb{P}[D]$. We get that

$$\sum_{r=s}^{\infty} \mathbb{P}\left[L_k^r \cap D \cap \{F = \infty\}\right] \leq \sum_{r=s}^{\infty} \mathbb{P}\left[L_k^r \cap D \cap \{F > r\}\right] \leq M \cdot \mathbb{P}[D] < \infty.$$

By the Borel-Cantelli lemma, $\mathbb{P}[L_k^r \cap D \cap \{F = \infty\} \text{ for infinitely many } r] = 0$. However, the event $\{L_k^r \cap D \cap \{F = \infty\} \text{ for infinitely many } r\}$ is precisely the set of all runs $\omega \in \Omega$ for which (1) $T(\omega) = \infty$ (as ω never has level zero by $\omega \in L_k^r$ for infinitely many k), (2) $\mathbf{X}_s[k](\omega) \leq M$, (3) for each $r \geq s$ the level of ω at step t is at least k, and (4) the level of ω is k infinitely many times. Hence, $B = \{L_k^r \cap D \cap \{F = \infty\} \text{ for infinitely many } r\}$ and $\mathbb{P}[B] = 0$, a contradiction. □

GLexRSMs would be unsound without the expected leftward non-negativity.

Example 3.3. Consider a one-dimensional stochastic process $(Y_t)_{t=0}^{\infty}$ such that $Y_0 = 1$ with probability 1 and then the process evolves as follows: in every step t, if $Y_t \geq 0$, then with probability $p_t = \frac{1}{4} \cdot \frac{1}{2^t}$ we put $Y_{t+1} = Y_t - \frac{2}{p_t}$ and with probability $1 - p_t$ we put $Y_{t+1} = Y_t + \frac{1}{1-p_t}$. If $Y_t < 0$, we put $Y_{t+1} = Y_t$. The underlying probability space can be constructed by standard techniques and we consider the filtration $(\mathcal{F}_t)_{t=0}^{\infty}$ such that $\mathcal{F}_t$ is the smallest sub-sigma-algebra making Y_t measurable. Finally, consider the stopping time T returning the first point in time when $Y_t < 0$. Then, $T < \infty$ if and only if the process ever performs the update $Y_{t+1} = Y_t - \frac{2}{p_t}$. However, the probability that this happens is bounded by $\frac{1}{4} + \frac{3}{4} \cdot \frac{1}{8} + \frac{3}{4} \cdot \frac{7}{8} \cdot \frac{1}{16} + \cdots < \frac{1}{4} \sum_{t=0}^{\infty} \frac{1}{2^t} = \frac{1}{2} < 1$. At the same time, putting $L_1^t = \{Y_t \geq 0\}$, we get that the tuple $((Y_t)_{t=0}^{\infty}, (L_1^t)_{t=0}^{\infty})$ satisfies all conditions of Definition 3.1 apart from the last bullet of point 3.

4 PROGRAM-SPECIFIC PRELIMINARIES

Arithmetic *expressions* in our programs are built from constants, program variables, and standard Borel-measurable [Billingsley 1995] arithmetic operators. We also allow sampling instructions to appear on right-hand sides of variable assignments as linear terms. An expression with no such terms is called *sampling free*. We allow sampling from both discrete and continuous distributions. We denote by $\mathcal{D}$ the set of distributions appearing in the program, with each $d \in \mathcal{D}$ assumed to be *integrable*, that is, $\mathbb{E}_{X \sim d}[|X|] < \infty$. We write $X \sim d$ to denote that X is a random variable with probability distribution d. This is to ensure that the expected value of each d over any measurable set is well defined and finite.

A *predicate* over a set of variables V is a Boolean combination of *atomic predicates* of the form $E \leq E'$, where E, E' are sampling-free expressions whose variables are all from V. We denote by $\mathbf{x} \models \Psi$ the fact that the predicate Ψ is satisfied by substituting values of $\mathbf{x}$ for the corresponding variables in Ψ.

We represent PPs via the standard concept of *probabilistic control flow graphs (pCFGs)* [Agrawal et al. 2018; Chatterjee et al. 2018, 2017]. Formally, a pCFG is a tuple $\mathcal{C} = (L, V, \Delta, Up, G)$, where

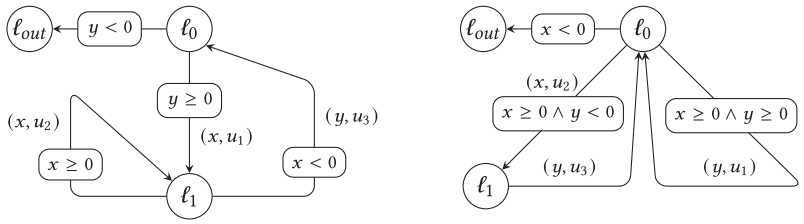

Fig. 3. The pCFGs of the programs presented in Figure 1. Guards are shown in the rounded boxes (absence of a box = guard is *true*). The update tuples are shown using variable aliases instead of indexes for better readability. On the left, we have that $u_1 = y, u_2 = x - 1 + Norm(0, 1)$, and $u_3 = y - 1$. On the right, we have that $u_1 = y + Uni[-7, 1], u_2 = x + Uni[-7, 1]$, and $u_3 = y + Uni[-7, 1]$.

L is a finite set of *locations*; $V = \{x_1, \ldots, x_{|V|}\}$ is a finite set of *program variables*; and Δ is a finite set of *transitions*, that is, tuples of the form $\tau = (\ell, \delta)$, where ℓ is a location and δ is a distribution over *successor locations*. Δ is partitioned into two disjoint sets: Δ_{PB} of probabilistic branching transitions for which $|supp(\delta)| = 2$, and Δ_{NPB} of remaining transitions for which $|supp(\delta)| = 1$. Next, *Up* is a function assigning to each transition in Δ_{NPB} either the element $\perp$ (representing no variable update) or a tuple (i, u), where $1 \leq i \leq |V|$ is a *target variable index* and u is an *update element*, which can be either an expression (possibly involving a single sampling instruction), or a bounded interval $R \subseteq \mathbb{R}$ representing a nondeterministic update. Finally, G is a function assigning a predicate (a *guard*) over V to each transition in Δ_{NPB}. Figure 3 presents the pCFGs of our two motivating examples in Figure 1.

Transitions in Δ_{PB} correspond to the "probabilistic branching" specified by the **if prob(p) then** ... **else** ... construct in imperative-style source code [Agrawal et al. 2018]. A program (pCFG) is *linear* (or *affine*) if all its expressions are *linear*, that is, of the form $b + \sum_{i=1}^{n} a_i \cdot Z_i$ for constants $a_1, \ldots, a_n, b$ and program variables/sampling instructions Z_i. We assume that parameters of distributions are constants; thus, they do not depend on program variable values, a common assumption in martingale-based automated approaches to a.s. termination proving [Agrawal et al. 2018; Chakarov and Sankaranarayanan 2013; Chatterjee et al. 2018; Chen and He 2020; Huang et al. 2019].

A *state* of a pCFG C is a tuple $(\ell, \mathbf{x})$, where ℓ is a location of C and $\mathbf{x}$ is a $|V|$-dimensional vector of *variable valuations*. A transition τ is *enabled* in $(\ell, \mathbf{x})$ if τ is outgoing from ℓ and $\mathbf{x} \models G(\tau)$.

The nondeterminism in programs is resolved via schedulers. A *scheduler* is a function σ assigning (i) to every finite path ending in a state s, a probability distribution over transitions enabled in s; and (ii) to every finite path that ends in a state in which a transition τ with a nondeterministic update $Up(\tau) = (i, R)$ is enabled, an integrable probability distribution over R. To make the program dynamics under a given scheduler well defined, we restrict ourselves to *measurable* schedulers. This is standard in probabilistic settings [Neuhäußer and Katoen 2007; Neuhäußer et al. 2009] and, hence, we omit the formal definition.

A state $c' = (\ell', \mathbf{x}')$ is a *successor* of a state $c = (\ell, \mathbf{x})$ if it can result from c by performing a transition τ enabled in c. Formally, $(\ell', \mathbf{x}')$ is a successor of $(\ell, \mathbf{x})$ under transition τ if ℓ' is the successor location of ℓ under τ and $\mathbf{x}'$ relates to $\mathbf{x}$ in one of these ways depending on $(i, u) = Up(\tau)$:

- If $u = \perp$, then $\mathbf{x}' = \mathbf{x}$.
- If u is an expression, then for the sampling instruction in the expression (if it exists) we sample from the respective distribution and replace the instruction with the sampled value. We then evaluate the resulting expression into a number A and put $\mathbf{x}' = \mathbf{x}(i \leftarrow A)$.

- If u is an interval R, then we sample a value A from the distribution prescribed by a scheduler σ for π_i and τ; we then put $\mathbf{x}' = \mathbf{x}(i \leftarrow A)$.

A *finite path* of length k in C is a finite sequence $(\ell_0, \mathbf{x}_0) \cdots (\ell_k, \mathbf{x}_k)$ of states such that $\ell_0 = \ell_{init}$ and, for each $0 \leq i < k$, the state $(\ell_{i+1}, \mathbf{x}_{i+1})$ is a successor of $(\ell_i, \mathbf{x}_i)$. A *run* in C is an infinite sequence of states whose every finite prefix is a finite path. We denote by $Fpath_C$ and Run_C the sets of all finite paths and runs in C, respectively. A state $(\ell, \mathbf{x})$ is *reachable* if there is, for some $\mathbf{x}_{init}$, a finite path starting in $(\ell_{init}, \mathbf{x}_{init})$ and ending in $(\ell, \mathbf{x})$.

We use the standard Markov Decision Process (MDP) semantics of pCFGs [Agrawal et al. 2018; Chatterjee et al. 2018; Kaminski et al. 2018b]. Each pCFG C induces a sample space $\Omega_C = Run_C$ and the standard *Borel* sigma-algebra $\mathcal{F}_C$ over Ω_C. Moreover, a pCFG C together with a scheduler σ, initial location ℓ_{init}, and initial variable valuation $\mathbf{x}_{init}$ uniquely determine a probability measure $\mathbb{P}^{\sigma}_{\ell_{init}, \mathbf{x}_{init}}$ in the probability space $(\Omega_C, \mathcal{F}_C, \mathbb{P}^{\sigma}_{\ell_{init}, \mathbf{x}_{init}})$, capturing the rather intuitive dynamics of the program's execution: we start in state $(\ell_{init}, \mathbf{x}_{init})$ and, in each step, a transition τ enabled in the current state is selected (using σ if multiple transitions are enabled). If $Up(\tau) = (i, u)$, then the value of variable x_i is changed according to u. The formal construction of $\mathbb{P}^{\sigma}_{\ell_{init}, \mathbf{x}_{init}}$ proceeds via the standard *cylinder construction* [Ash and Doléans-Dade 2000, Theorem 2.7.2]. We denote by $\mathbb{E}^{\sigma}_{\ell_{init}, \mathbf{x}_{init}}$ the expectation operator in the probability space $(\Omega_C, \mathcal{F}_C, \mathbb{P}^{\sigma}_{\ell_{init}, \mathbf{x}_{init}})$.

We stipulate that each pCFG has a special *terminal location* ℓ_{out} whose all outgoing transitions must be self-loops. We say that a run ϱ *terminates* if it contains a configuration whose first component is ℓ_{out}. We denote by *Terminates* the set of all terminating runs in Ω_C. We say that a program represented by a pCFG C terminates *almost-surely (a.s.)* if for each measurable scheduler σ and each initial variable valuation $\mathbf{x}_{init}$ it holds that $\mathbb{P}^{\sigma}_{\ell_{init}, \mathbf{x}_{init}}[Terminates] = 1$.

5 GLEXRSMS FOR PROBABILISTIC PROGRAMS

In this section, we define a syntactic proof rule for a.s. termination of PPs, showing its soundness via Theorem 3.2. In what follows, let C be a pCFG.

Definition 5.1 (Measurable Map). An n-dimensional measurable map (MM) is a vector $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_n)$, where each η_i is a function mapping each location ℓ to a real-valued Borel-measurable function $\eta_i(\ell)$ over program variables. We say that $\boldsymbol{\eta}$ is a *linear expression map* (LEM) if each η_i is representable by a linear expression over program variables.

The notion of pre-expectation was introduced in Kozen [1983], was made syntactic in the Dijkstra wp-style in Morgan and McIver [1999], and was extended to programs with continuous distributions in Chakarov and Sankaranarayanan [2013]. It formalizes the "one-step" expectation operator $\mathbb{E}^{\tau}$ we used on an intuitive level in the introduction. In what follows, we generalize the definition of pre-expectation presented in Chakarov and Sankaranarayanan [2013] in order to allow taking expectation over subsets of successor states C (a necessity for handling the *EXP-NNEG* constraint). We say that a set S of states in C is *measurable* if for each location ℓ in C we have that $\{\mathbf{x} \in \mathbb{R}^{|V|} \mid (\ell, \mathbf{x}) \in S\} \in \mathcal{B}(\mathbb{R}^{|V|})$, that is, it is in the Borel sigma-algebra of $\mathbb{R}^{|V|}$. Furthermore, we also differentiate between the *maximal* and *minimal pre-expectation*, which may differ in the case of nondeterministic assignments in programs and intuitively are equal to the maximal respective minimal value of the next-step expectation over all nondeterministic choices. Let η be a 1-dimensional MM, $\tau = (\ell, \delta)$ a transition, and S be a measurable set of states in C. We denote by $\text{max-pre}^{\tau}_{\eta, S}(s)$ the *maximal pre-expectation* of η in τ given S (i.e., the maximal expected value of η after making a step from s computed over successor states belonging to S). Similarly, we denote by $\text{min-pre}^{\tau}_{\eta, S}$ the *minimal pre-expectation* of η in τ given S.

Definition 5.2 (Pre-expectation). Let η be a 1-dimensional MM, $\tau = (\ell, \delta)$ a transition, and S be a measurable set of states in C. A *maximal pre-expectation* of η in τ given S is the function $\text{max-pre}^{\tau}_{\eta,S}$ assigning to each state $(\ell, \mathbf{x})$ the following number:

- if $\tau \in \Delta_{PB}$ is a transition of probabilistic branching, then $\text{max-pre}^{\tau}_{\eta,S}(\ell, \mathbf{x}) = \sum_{\ell' \in L} \delta(\ell') \cdot \eta(\ell', \mathbf{x}) \cdot \mathbb{I}(S)(\ell', \mathbf{x})$;
- otherwise, $\text{max-pre}^{\tau}_{\eta,S}(\ell, \mathbf{x}) = A_{\ell'}$, where ℓ' is the unique successor location of τ and $A_{\ell'}$ is defined as follows:
  (1) if $Up(\tau) = \bot$, then $A_{\ell'} = \eta(\ell', \mathbf{x}) \cdot \mathbb{I}(S)(\ell', \mathbf{x})$
  (2) if $Up(\tau) = (j, u)$ and u is an expression over program variables and sampling instructions, then $A_{\ell'} = \eta(\ell', \mathbf{x}(j \leftarrow A))$, where A is the expected value of $u \cdot \mathbb{I}(S)(\ell', u)$ under valuation $\mathbf{x}$ (i.e., integration of u is performed only over the set of states at ℓ' that belong to S; the integral is well defined and finite since $\{\mathbf{x} \in \mathbb{R}^{|V|} \mid (\ell', \mathbf{x}) \in S\} \in \mathcal{B}(\mathbb{R}^{|V|})$ and any distribution appearing in variable updates is integrable);
  (3) if $Up(\tau) = (j, u)$ and u is an interval I denoting a nondeterministic assignment, then $A_{\ell'} = \sup_{y \in I \wedge (\ell', \mathbf{x}(j \leftarrow y)) \in S} \eta(\ell', \mathbf{x}(j \leftarrow y))$.

A *minimal pre-expectation* is denoted by $\text{min-pre}^{\tau}_{\eta,S}$ and is defined analogously, with the only difference being that in point 3 we use inf instead of sup. We omit the subscript S when S is the set of all states of C.

As in the case of non-probabilistic programs, termination certificates are supported by program invariants over-approximating the set of reachable states. An *invariant* in C is a function I which assigns a Borel-measurable set $I(\ell) \subseteq \mathbb{R}^{|V|}$ to each location ℓ of C such that for any state $(\ell, \mathbf{x})$ reachable in C, it holds that $\mathbf{x} \in I(\ell)$. If each $I(\ell)$ is given by a conjunction of linear inequalities over program variables, we say that I is a *linear invariant*.

GLexRSM-Based Proof Rule for Almost-Sure Termination. Given $n \in \mathbb{N}$, we call a map $\text{lev} : \Delta \to \{0, 1, \ldots, n\}$ a *level map*. For $\tau \in \Delta$, we say that $\text{lev}(\tau)$ is its level. The level of a state is the largest level of any transition enabled at that state. We denote by $S^{\leq j}_{\text{lev}}$ the set of states with level $\leq j$.

Definition 5.3 (GLexRSM Map). Let $\boldsymbol{\eta}$ be an n-dimensional MM and I an invariant in C. We say that $\boldsymbol{\eta}$ is a *generalized lexicographic ranking supermartingale map (GLexRSM map)* supported by I, if there is a level map $\text{lev} : \Delta \to \{0, 1, \ldots, n\}$ such that $\text{lev}(\tau) = 0$ iff τ is a self-loop transition at ℓ_{out}, and for any transition $\tau = (\ell, \delta)$ with $\ell \neq \ell_{out}$ the following conditions hold:

(1) $\textit{P-RANK}(\boldsymbol{\eta}, \tau) \equiv \mathbf{x} \in I(\ell) \cap G(\tau) \Rightarrow (\text{max-pre}^{\tau}_{\eta_{\text{lev}(\tau)}}(\ell, \mathbf{x}) \leq \eta_{\text{lev}(\tau)}(\ell, \mathbf{x}) - 1 \wedge \text{max-pre}^{\tau}_{\eta_j}(\ell, \mathbf{x}) \leq \eta_j(\ell, \mathbf{x})$ for all $1 \leq j < \text{lev}(\tau))$;
(2) $\textit{P-NNEG}(\boldsymbol{\eta}, \tau) \equiv \mathbf{x} \in I(\ell) \cap G(\tau) \Rightarrow (\eta_j(\ell, \mathbf{x}) \geq 0$ for all $1 \leq j \leq \text{lev}(\tau))$;
(3) $\textit{EXP-NNEG}(\boldsymbol{\eta}, \tau) \equiv \mathbf{x} \in I(\ell) \cap G(\tau) \Rightarrow \text{min-pre}^{\tau}_{\eta_j, S^{\leq j-1}_{\text{lev}}}(\ell, \mathbf{x}) \geq 0$ for all $1 \leq j \leq \text{lev}(\tau)$.

A GLexRSM map $\boldsymbol{\eta}$ is *linear* (or LinGLexRSM map) if it is also an LEM.

Theorem 5.4 (Soundness of GLexRSM-maps for a.s. Termination). *Let C be a pCFG and I an invariant in C. Suppose that C admits an n-dimensional GLexRSM map $\boldsymbol{\eta}$ supported by I, for some $n \in \mathbb{N}$. Then, C terminates a.s.*

Theorem 5.4 instantiates Theorem 3.2 to probability spaces of pCFGs. Its proof can be found in the the extended version of the paper [Chatterjee et al. 2021b]. We stress that the instantiation is *not* straightforward. In proving it, we establish two interesting results on probabilistic programs with nondeterminism that were hitherto not studied in the probabilistic program analysis literature. In the rest of this section, we discuss these results in more detail. The proof of Theorem 5.4 uses these

results to prove that a GLexRSM map induces a GLexRSM as in Definition 3.1 in the probability space over the set of runs defined by the pCFG C.

To ensure that a scheduler cannot "escape" ranking by intricate probabilistic mixing of transitions, we show that in order to prove a.s. termination, it is sufficient to consider *deterministic* schedulers, which do not introduce randomization among transitions. This statement is formalized in the following proposition.

Proposition 5.5. *Let C be a pCFG and suppose that there exist a measurable scheduler σ and an initial configuration $(\ell_{init}, \mathbf{x}_{init})$ such that $\mathbb{P}^{\sigma}_{\ell_{init},\mathbf{x}_{init}}[Term = \infty] > 0$. Then, there exists a measurable scheduler σ^*, which is deterministic in the sense that to each finite path it assigns a single transition with probability* 1*, such that $\mathbb{P}^{\sigma^*}_{\ell_{init},\mathbf{x}_{init}}[Term = \infty] > 0$.*

The proof of Proposition 5.5 proceeds by constructing a sequence $\sigma = \sigma_0, \sigma_1, \sigma_2, \ldots$ of schedulers, where for each $i \in \mathbb{N}_0$ we have that σ_{i+1} and σ_i agree on histories of length at most $i-1$, and σ_{i+1} "refines" σ_i on histories of length i in such a way that

- σ_{i+1} is deterministic on histories of length at most i;
- σ_{i+1} is measurable; and
- $\mathbb{P}^{\sigma_{i+1}}[Term = \infty] \geq \mathbb{P}^{\sigma_i}[Term = \infty]$.

Then, we define the scheduler σ^* as $\sigma^*(\rho) = \sigma_i(\rho)$ whenever the length of a finite history ρ is i. The construction of each σ_{i+1} requires care as it needs to ensure that the newly constructed scheduler is measurable, that it satisfies $\mathbb{P}^{\sigma_{i+1}}[Term = \infty] \geq \mathbb{P}^{\sigma_i}[Term = \infty]$, and that the resulting scheduler σ^* is measurable. The fact that $\mathbb{P}^{\sigma^*}_{\ell_{init},\mathbf{x}_{init}}[Term = \infty] > 0$ follows by the application of the Monotone Convergence Theorem [Williams 1991]. The construction is highly non-trivial and uses advanced results from probability and measure theory. We provide it in the extended version of the paper [Chatterjee et al. 2021b].

Also, previous martingale-based certificates of a.s. termination [Agrawal et al. 2018; Chatterjee et al. 2018; Fioriti and Hermanns 2015; Fu and Chatterjee 2019] often impose either non-negativity or integrability of random variables defined by measurable maps in programs to ensure that their conditional expectations exist. We show that these conditional expectations exist even without such assumptions and in the presence of nondeterminism, and that they can be explicitly expressed by extending the definition of pre-expectation in Chakarov and Sankaranarayanan [2013] to also depend on a fixed scheduler that is used to resolve nondeterminism. This generalizes the result of Chakarov and Sankaranarayanan [2013] to PPs with nondeterminism.

Definition 5.6 (Pre-expectation with Respect to a Scheduler). Let S be a set of measurable states in C. The *pre-expectation with respect to a scheduler σ of an MM η* given S is a map $\mathrm{pre}_{\sigma,\eta,S} : Fpath_C \to \mathbb{R}$ defined as follows. Let $\rho \in Fpath_C$ be a finite path ending in $(\ell, \mathbf{x})$. Let $\sigma(\rho)$ denote the distribution over transitions enabled in $(\ell, \mathbf{x})$ defined by the scheduler, and for each transition $\tau = (\ell, \delta) \in supp(\sigma(\rho))$ with the update element being a nondeterministic assignment from an interval, let $d_\sigma(\rho, \tau)$ be a distribution over the interval defined by σ. Then,

$$\mathrm{pre}_{\sigma,\eta,S}(\rho) = \sum_{\tau=(l,\delta)\in supp(\sigma(\rho))} \sigma(\rho)(\tau) \cdot \mathrm{pre}^{\tau}_{\sigma,\eta,S}(\ell, \mathbf{x}),$$

where $\mathrm{pre}^{\tau}_{\sigma,\eta,S}$ is defined in the same way as the standard pre-expectation $\mathrm{pre}^{\tau}_{\eta,S}$, *except for the case* in which $\tau = (\ell, \delta)$ carries a nondeterministic assignment, that is, $u(\tau) = (j, u)$, with u being an interval. In such a case, we put $\mathrm{pre}^{\tau}_{\sigma,\eta}(\ell, \mathbf{x}) = \sum_{\ell' \in L} \delta(\ell') \cdot \eta(\ell', \mathbf{x}(j \leftarrow \mathbb{E}[d_\sigma(\rho, \tau) \cdot \mathbb{I}(S)])$.

Intuitively, $\mathrm{pre}_{\sigma,\eta,S}$ takes a finite path $\rho \in Fpath_C$ as an input, and returns the *expected value* of η in the next step when the integration is performed over program states in S given the program

run history and the choices of the scheduler σ. In the proof of Theorem 5.4, we show that, for any scheduler σ, all conditional expectations whose existence is required in order to define a GLexRSM as in Definition 3.1 in the probability space over the set of runs defined by the pCFG C and the scheduler σ indeed exist and can be expressed via pre-expectations with respect to the scheduler σ.

Remark 5.1 (Comparison with Huang et al. [2019]). The work of Huang et al. [2019] considers a modular approach. Given a loop whose body has already been proved a.s. terminating, they show that the loop terminates a.s. if it admits a 1-dimensional MM satisfying *P-RANK* for each transition in the loop, *P-NNEG* for the transition entering the loop, and the "*bounded expected difference*" property for all transitions. Hence, their approach is suited mainly for programs with incremental variable updates.

Modularity is also a feature of the approaches based on the weakest pre-expectation calculus [McIver and Morgan 2004, 2005; McIver et al. 2018].

6 ALGORITHM FOR LINEAR PROBABILISTIC PROGRAMS

We now present two algorithms for proving a.s. termination in linear probabilistic programs (LinPPs). The first algorithm considers LinPPs with sampling from bounded-support distributions. We show that the problem of deciding the existence of LinGLexRSM maps for such LinPPs is decidable. Our second algorithm extends the first algorithm into a sound a.s. termination prover for general LinPPs. In what follows, let C be a LinPP and I a linear invariant in C.

6.1 Linear Programs with Distributions of Bounded Support

Restricting to linear arithmetic is standard in automated a.s. termination proving, allowing one to encode the existence of the termination certificate into systems of linear constraints [Agrawal et al. 2018; Chakarov and Sankaranarayanan 2013; Chatterjee et al. 2018; Chen and He 2020]. In the case of LinGLexRSM maps, the difficulty lies in encoding the *EXP-NNEG* condition, as it involves integrating distributions in variable updates that cannot always be done analytically. We show that for LinPPs with bounded-support sampling, we can define another condition that is easier to encode and can replace *EXP-NNEG*. Formally, we say that a distribution $d \in \mathcal{D}$ has a *bounded support* if there exists $N(d) \geq 0$ such that $\mathbb{P}_{X\sim d}[|X| > N(d)] = 0$. Here, we use $\mathbb{P}_{X\sim d}$ to denote the probability measure induced by a random variable X with the probability distribution d. We say that a LinPP has the *bounded support property (BSP)* if all distributions in the program have bounded support. For instance, the program in Figure 1(b) has the BSP, whereas the program in Figure 1(a) does not. Using the same notation as in Definition 5.3, we put:

$$\textit{W-EXP-NNEG}(\boldsymbol{\eta}, \tau) \equiv \mathbf{x} \in I(\ell) \cap G(\tau) \Rightarrow \forall 1 \leq j \leq \mathrm{lev}(\tau)\ \text{min-pre}^{\tau}_{\eta_j}(\ell, \mathbf{x}) \geq 0.$$

(The W stands for "weak.") Intuitively, *EXP-NNEG* requires non-negativity of the expected value of η_j when integrated over successor states of level smaller than j, whereas the condition *W-EXP-NNEG* requires non-negativity of the expected value of η_j when integrated over all successor states. Since η_j is non-negative at successor states of level at least j, this new condition is weaker than *EXP-NNEG*. Nevertheless, the following lemma shows that in order to decide the existence of LinGLexRSM maps for programs with the BSP, we may without loss of generality replace *EXP-NNEG* by *W-EXP-NNEG* for all transitions but for those of probabilistic branching.

Lemma 6.1. *Let C be a LinPP with the BSP and I be a linear invariant in C. If a LEM $\boldsymbol{\eta}$ satisfies conditions P-RANK and P-NNEG for all transitions, EXP-NNEG for all transitions in Δ_{PB} and W-EXP-NNEG for all other transitions, then $\boldsymbol{\eta}$ may be increased pointwise by a constant value in order to obtain a LinGLexRSM map.*

Proof. To prove the lemma, we need to show that there exists $K > 0$ such that the LEM $\boldsymbol{\eta}'$ of the same dimension as $\boldsymbol{\eta}$, and defined via $\eta'_j(\ell, \mathbf{x}) = \eta_j(\ell, \mathbf{x}) + K$ for each component j and state $(\ell, \mathbf{x})$, is a LinGLexRSM map in C supported by I (with the level map being the same as for $\boldsymbol{\eta}$).

Note that increasing $\boldsymbol{\eta}$ pointwise by a constant $K > 0$ preserves the *P-NNEG*, *P-RANK* conditions for each transition in C, as well as *EXP-NNEG* for each transition of probabilistic branching. Hence, we are left to show that there exists $K > 0$ such that for any transition τ that is not a transition of probabilistic branching we have that

$$\textit{EXP-NNEG}(\boldsymbol{\eta}', \tau) \equiv \mathbf{x} \in I(\ell) \cap G(\tau) \Rightarrow \text{min-pre}^{\tau}_{\eta'_j, S^{\leq j-1}_{\text{lev}}}(\ell, \mathbf{x}) \geq 0 \text{ for all } 1 \leq j \leq \text{lev}(\tau).$$

Since the LinPP that induces C satisfies the BSP property and since all nondeterministic assignments are defined by closed intervals, there exists $N > 0$ such that $\mathbb{P}_{X\sim d}[|X| > N] = 0$ for each distribution $d \in \mathcal{D}$ and $[a, b] \subseteq [-N, N]$ for each interval $[a, b]$ appearing in nondeterministic assignments.

We claim that $K = 2 \cdot N \cdot \text{max-coeff}(\boldsymbol{\eta})$ satisfies the claim, where $\text{max-coeff}(\boldsymbol{\eta})$ is the maximal absolute value of a coefficient appearing in any expression $\boldsymbol{\eta}(\ell')$ for any location ℓ'.

To prove this, let τ be a transition that is not a transition of probabilistic branching, and let ℓ and ℓ_1 be its source and target location, respectively. Let $\mathbf{x} \in I(\ell) \cap G(\tau)$ and let $1 \leq j \leq \text{lev}(\tau)$. In order to prove that $\text{min-pre}^{\tau}_{\eta'_j, S^{\leq j-1}_{\text{lev}}}(\ell, \mathbf{x}) \geq 0$ holds, we distinguish between three cases:

(1) $Up(\tau) = \bot$ or $Up(\tau) = (i, u)$, where u is a linear expression with no sampling instruction. Then, $(\ell, \mathbf{x})$ has a single successor state $(\ell_1, \mathbf{x}_1)$ upon executing τ.
- If the level of $(\ell_1, \mathbf{x}_1)$ is at least j, then $S^{\leq j-1}_{\text{lev}}$ contains no successor states and we have that $\text{min-pre}^{\tau}_{\eta'_j, S^{\leq j-1}_{\text{lev}}}(\ell, \mathbf{x}) = 0$ as the integration is performed over the empty set.
- Otherwise, $S^{\leq j-1}_{\text{lev}}$ contains $(\ell_1, \mathbf{x}_1)$ and

$$\text{min-pre}^{\tau}_{\eta'_j, S^{\leq j-1}_{\text{lev}}}(\ell, \mathbf{x}) = \eta'_j(\ell_1, \mathbf{x}_1) = \eta_j(\ell_1, \mathbf{x}_1) + K = \text{min-pre}^{\tau}_{\eta_j}(\ell, \mathbf{x}) + K \geq 0,$$

where the inequality $\text{min-pre}^{\tau}_{\eta_j}(\ell, \mathbf{x}) \geq 0$ holds since *W-EXP-NNEG*$(\boldsymbol{\eta}, \tau)$.

(2) If $Up(\tau) = (i, u)$, where u is a linear expression that contains sampling from a distribution $d \in \mathcal{D}$, we may write $u = u' + X$, where u' is the linear expression part of u with no distribution samplings and $X \sim d$. Then,

$$\begin{aligned}
&\text{min-pre}^{\tau}_{\eta'_j, S^{\leq j-1}_{\text{lev}}}(\ell, \mathbf{x}) = \mathbb{E}_{X\sim d}[\eta'_j(\ell_1, \mathbf{x}[i \leftarrow u' + X]) \cdot \mathbb{I}(\text{next state has level } \leq j-1)] \\
&= \mathbb{E}_{X\sim d}[(\eta_j(\ell_1, \mathbf{x}[i \leftarrow u']) + K + \text{coeff}[i] \cdot X) \cdot \mathbb{I}(\text{next state has level } \leq j-1)] \\
&= \mathbb{E}_{X\sim d}[(\eta_j(\ell_1, \mathbf{x}[i \leftarrow u']) + \text{coeff}[i] \cdot \mathbb{E}[X]) \cdot \mathbb{I}(\text{next state has level } \leq j-1)] \\
&\quad + \mathbb{E}_{X\sim d}[(K - \text{coeff}[i] \cdot \mathbb{E}[X] + \text{coeff}[i] \cdot X) \cdot \mathbb{I}(\text{next state has level } \leq j-1)] \qquad (3) \\
&\geq \mathbb{E}_{X\sim d}[\text{min-pre}^{\tau}_{\eta_j}(\ell, \mathbf{x}) \cdot \mathbb{I}(\text{next state has level } \leq j-1)] \\
&\quad + \mathbb{E}_{X\sim d}[(K - 2 \cdot N \cdot \text{max-coeff}(\boldsymbol{\eta})) \cdot \mathbb{I}(\text{next state has level } \leq j-1)] \\
&\geq 0,
\end{aligned}$$

where $\text{min-pre}^{\tau}_{\eta_j}(\ell, \mathbf{x}) \geq 0$ holds since *W-EXP-NNEG*$(\boldsymbol{\eta}, \tau)$, and $\mathbb{E}[X] \geq -N$ holds and $X \geq -N$ holds almost-surely since $\mathbb{P}_{X\sim d}[|X| > N] = 0$ by the definition of N. Here, $\text{coeff}[i]$ denotes the linear coefficient in η_j of the i-th variable in the variable valuation $\mathbf{x}$.

ALGORITHM 1: Synthesis of LinGLexRSM Maps in LinPPs with the BSP Property

Input : A LinPP C with the BSP property, linear invariant I.
Output : LinGLexRSM map supported by I if it exists, otherwise "No LinGLexRSM map"

$\mathcal{T} \longleftarrow$ all transitions in C; $d \longleftarrow 0$
while $\mathcal{T}$ *is non-empty* **do**
  $d \longleftarrow d + 1$
  construct $\mathcal{LP}_{\mathcal{T}}$
  if $\mathcal{LP}_{\mathcal{T}}$ *is feasible* **then**
    $\eta_d \longleftarrow$ LEM defined by the optimal solution of $\mathcal{LP}_{\mathcal{T}}$
    $\mathcal{T} \longleftarrow \mathcal{T} \setminus \{\tau \in \mathcal{T} \mid \tau \text{ is 1-ranked by } \eta_d\}$
  else return No LinGLexRSM map
max $\longleftarrow$ max-coeff($\boldsymbol{\eta}$)
$N \longleftarrow$ constant such that all distributions and intervals supported in $[-N, N]$
for $1 \le j \le d$ **do**
  $\eta_j \longleftarrow \eta_j + 2 \cdot N \cdot \text{max}$
return $(\eta_1, \ldots, \eta_d)$

(3) If $Up(\tau) = (i, u)$, where u is an interval $[a, b]$ defining a nondeterministic assignment, then

$$\begin{aligned}\text{min-pre}^{\tau}_{\eta'_j, S^{\le j-1}_{\text{lev}}}(\ell, \mathbf{x}) &= \inf_{X \in [a,b] \wedge (\ell_1, \mathbf{x}(i \leftarrow X)) \in S^{\le j-1}_{\text{lev}}} \left[\eta'_j(\ell_1, \mathbf{x}[i \leftarrow X])\right] \\ &\ge \inf_{X \in [a,b]} \left[\eta'_j(\ell_1, \mathbf{x}[i \leftarrow X])\right] \\ &= \text{min-pre}^{\tau}_{\eta_j}(\ell, \mathbf{x}) + K \ge 0,\end{aligned}$$

where $\text{min-pre}^{\tau}_{\eta_j}(\ell, \mathbf{x}) \ge 0$ holds since *W-EXP-NNEG*($\boldsymbol{\eta}, \tau$) and $K \ge 0$ by definition. □

Observe that the proof of Lemma 6.1 in item (2) essentially depends on the assumption that we work over linear arithmetic. Indeed, the second expected value in the first inequality in Equation (3) might not be possible to be bounded from below by 0 for any value of $K > 0$ if we allowed non-linear arithmetic and program variable values that are not bounded.

Algorithmic Results. Let LinGLexPP$^{\text{BOUNDED}}$ be the set of pairs (C, I) of a pCFG C representing a LinPP with the BSP and a linear invariant I in C such that C admits a LinGLexRSM map supported by I.

Theorem 6.2. *There is a polynomial-time algorithm deciding whether a tuple* (C, I) *belongs to the set* LinGLexPP$^{\text{BOUNDED}}$. *Moreover, if the answer is yes, the algorithm outputs a witness in the form of a LinGLexRSM map of minimal dimension.*

The algorithm behind Theorem 6.2 is a generalization of algorithms in Alias et al. [2010] and Agrawal et al. [2018] finding LinLexRFs in non-probabilistic programs and LinLexRSM maps in PPs, respectively. The pseudocode is shown in Algorithm 1. Suppose that we are given a LinPP $C = (L, V, \Delta, Up, G)$ with the BSP and a linear invariant I. Our algorithm stores a set $\mathcal{T}$ initialized to all transitions in C. It then proceeds in iterations to compute new components of the witness. In each iteration, it searches for a LEM η that is required to

(1) be non-negative on each $\tau = (\ell, \delta) \in \mathcal{T}$, that is, $\forall \mathbf{x}.\, \mathbf{x} \in I(\ell) \cap G(\tau) \Rightarrow \eta(\ell, \mathbf{x}) \ge 0$;
(2) be unaffecting on each $\tau = (\ell, \delta) \in \mathcal{T}$, that is, $\forall \mathbf{x}.\, \mathbf{x} \in I(\ell) \cap G(\tau) \Rightarrow \text{max-pre}^{\tau}_{\eta}(\ell, \mathbf{x}) \le \eta(\ell, \mathbf{x})$;
(3) have non-negative minimal pre-expectation for each $\tau = (\ell, \delta) \in \mathcal{T} \setminus \Delta_{PB}$, that is, $\forall \mathbf{x}.\, \mathbf{x} \in I(\ell) \cap G(\tau) \Rightarrow \text{min-pre}^{\tau}_{\eta}(\ell, \mathbf{x}) \ge 0$;

(4) if S is the set of states in C whose all enabled transitions have been removed from $\mathcal{T}$ in the previous algorithm iterations, $\forall \tau = (\ell, \delta) \in \mathcal{T} \cap \Delta_{PB}, \forall \mathbf{x}.\, \mathbf{x} \in I(\ell) \cap G(\tau) \Rightarrow \mathrm{pre}^{\tau}_{\eta,S}(\ell, \mathbf{x}) \geq 0$; and
(5) 1-rank the maximal number of transitions in $\tau \in \mathcal{T}$, ithat is, $\forall \mathbf{x}.\, \mathbf{x} \in I(\ell) \cap G(\tau) \Rightarrow \text{max-pre}^{\tau}_{\eta}(\ell, \mathbf{x}) \leq \eta(\ell, \mathbf{x}) - 1$ for as many $\tau = (\ell, \delta)$ as possible.

This is done by fixing an LEM template for each location ℓ in C and converting these constraints to an equivalent linear program $\mathcal{LP}_{\mathcal{T}}$ in template variables via Farkas's lemma (FL). The FL conversion (and its extension to strict inequalities [Chatterjee et al. 2018]) is standard in termination proving, and encoding conditions 1 to 3 and 5 is analogous to Agrawal et al. [2018] and Alias et al. [2010]; hence, we omit the details. To encode condition 4, let $\tau = (\ell, \delta) \in \mathcal{T}$ be a transition of probabilistic branching. Then, $supp(\delta) = (\ell_1, \ell_2)$ and $Up(\tau) = \bot$. Thus, for any $\mathbf{x} \in I(\ell) \cap G(\tau)$, we have that

$$\mathrm{pre}^{\tau}_{\eta,S}(\ell, \mathbf{x}) = \delta(\ell_1) \cdot \eta(\ell_1, \mathbf{x}) \cdot \mathbb{I}(S)(\ell_1, \mathbf{x}) + \delta(\ell_2) \cdot \eta(\ell_2, \mathbf{x}) \cdot \mathbb{I}(S)(\ell_2, \mathbf{x}),$$

that is, we include the term $\delta(\ell_i) \cdot \eta(\ell_i, \mathbf{x})$ for $i \in \{1, 2\}$ whenever $(\ell_i, \mathbf{x}) \in S$. Hence, to encode condition 4 for τ, define $G_1 = \neg(\vee_{\tau' = (\ell_1, \_) \in \mathcal{T}} G(\tau'))$ and $G_2 = \neg(\vee_{\tau' = (\ell_2, \_) \in \mathcal{T}} G(\tau'))$, and encode the following 3 conditions:

- $\forall \mathbf{x}.\, \mathbf{x} \in I(\ell) \cap G(\tau) \cap G_1 \cap G_2 \Rightarrow \delta(\ell_1) \cdot \eta(\ell_1, \mathbf{x}) + \delta(\ell_2) \cdot \eta(\ell_2, \mathbf{x}) \geq 0$,
- $\forall \mathbf{x}.\, \mathbf{x} \in I(\ell) \cap G(\tau) \cap G_1 \cap \neg G_2 \Rightarrow \delta(\ell_1) \cdot \eta(\ell_1, \mathbf{x}) \geq 0$, and
- $\forall \mathbf{x}.\, \mathbf{x} \in I(\ell) \cap G(\tau) \cap \neg G_1 \cap G_2 \Rightarrow \delta(\ell_2) \cdot \eta(\ell_2, \mathbf{x}) \geq 0$.

Each condition can be encoded via linear constraint as in Agrawal et al. [2018] and Alias et al. [2010]. Clearly, the size of the encoding is polynomial. An important thing to note is that the negations in G_1 and G_2 might result in strict inequalities appearing in these constraints. However, it was shown in Chatterjee et al. [2018] that this is not an issue for the FL conversion. Lemma 1 in Chatterjee et al. [2018] shows that, whenever a system of linear inequalities on the left-hand side of a constraint is feasible, the strict inequalities may without loss of generality be replaced by non-strict inequalities. On the other hand, Lemma 2 in Chatterjee et al. [2018] shows that this feasibility check can be done in polynomial time.

In each algorithm iteration, all transitions that have been 1-ranked are removed from $\mathcal{T}$ and the algorithm proceeds to the next iteration. If all transitions are removed from $\mathcal{T}$, the algorithm concludes that the program admits a LinGLexRSM map (obtained by increasing the constructed LEM by a constant defined in the proof of Lemma 6.1). If in some iteration a new component that 1-ranks at least 1 transition in $\mathcal{T}$ cannot be found, the program does not admit a LinGLexRSM map.

We show that Algorithm 1 satisfies the claim of Theorem 6.2.

Proof of Theorem 6.2. We first prove that the algorithm is sound: that $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_d)$ computed by Algorithm 1 is a LinGLexRSM map supported by I and, thus, that C is a.s. terminating. We define the level map $\mathrm{lev} : \Delta \to \{0, 1 \ldots, d\}$ with the self loop at ℓ_{out} having level 0, and for any other transition τ we define $\mathrm{lev}(\tau)$ as the index of algorithm iteration in which it was removed from $\mathcal{T}$. The fact that $\boldsymbol{\eta}$ computed in lines 1 to 8 in Algorithm 1 satisfies *P-NNEG*, *P-RANK*, *EXP-NNEG* for transitions of probabilistic branching and *W-EXP-NNEG* for all other transitions then easily follows from conditions imposed by the algorithm in each iteration. From the proof of Lemma 6.1, it then follows that $\boldsymbol{\eta}$ obtained upon increasing each component by a constant term in lines 9 to 13 satisfies *EXP-NNEG* for every transition. Hence, $\boldsymbol{\eta}$ is a LinGLexRSM map supported by I. This concludes the soundness proof.

The proofs of completeness and of the minimality of dimension are similar in spirit to the completeness proofs for the algorithm of Alias et al. [2010] for computing LinLexRFs in non-probabilistic programs and for the algorithm of Agrawal et al. [2018] for computing LinLexRSMs in probabilistic programs. First, we observe that a pointwise sum of two LinGLexRSM maps supported by I is also a LinGLexRSM map supported by I. This follows by linearity of integration and, therefore, the pre-expectation operator. The argument is straightforward; thus, we omit it. However, this simple observation will be central in the rest of the proof.

Suppose first that the program admits a LinGLexRSM $\boldsymbol{\eta}' = (\eta'_1, \ldots, \eta'_m)$ supported by I. We show that Algorithm 1 then finds one such LinGLexRSM map (up to a constant term). Hence, the algorithm is complete. We prove this by contradiction. Suppose that the algorithm stops after the d-th iteration, after having computed $(\eta_1, \ldots, \eta_d)$ but with $\mathcal{T}$ still containing at least one transition. Then, $(\eta_1, \ldots, \eta_d)$ does not rank every transition in the pCFG. Thus, $\boldsymbol{\eta}'$ ranks strictly more transitions than $(\eta_1, \ldots, \eta_d)$. We distinguish two cases:

(1) There exists the smallest $1 \leq j \leq \min\{d, m\}$ such that
   - for each $1 \leq j' < j$, $\eta_{j'}$ and $\eta'_{j'}$ would rank exactly the same set of transitions if computed by the algorithm in the j'-th iteration, but
   - η'_j ranks a transition that is not ranked by η_j in the j-th iteration of the algorithm.

   Then, the algorithm could have ranked strictly more transitions by computing $\eta_j + \eta'_j$ instead of η_j, which contradicts the maximality condition for computing new components imposed by the algorithm.
(2) There is no such index. But then, since $\boldsymbol{\eta}' = (\eta'_1, \ldots, \eta'_m)$ is the LinGLexRSM supported by I, it must follow that $m > d$ and that η'_{d+1} would satisfy all of the conditions imposed by the algorithm in the $(d+1)$-st iteration and it would rank at least 1 new transition. Thus, the algorithm could not terminate after iteration d.

In both cases we reach contradiction, and the completeness claim on Algorithm 1 holds.

Minimality of dimension is proved analogously, by contradiction. If there exists a LinGLexRSM map with regard to S supported by I of dimension strictly smaller than that found by the algorithm, we can use it analogously as above to show that at some iteration the algorithm could have ranked a strictly larger number of transitions, contradicting the maximality condition for computing new components that is imposed by the algorithm. Thus, the minimality of dimension claim follows. □

We conclude by showing that our motivating example in Figure 1(b) admits a LinGLexRSM map supported by a very simple linear invariant. Thus, by completeness, our algorithm is able to prove its a.s. termination.

Example 6.3. Consider the program in Figure 1(b) with a linear invariant $I(\ell_0) = \mathit{true}$, $I(\ell_1) = x \geq -7$. Its a.s. termination is witnessed by a LEM $\boldsymbol{\eta}(\ell_0, (x, y)) = (1, x+7, y+7)$, $\boldsymbol{\eta}(\ell_1, (x, y)) = (1, x+8, y+7)$ and $\boldsymbol{\eta}(\ell_{out}, (x, y)) = (0, x+7, y+7)$. Since $\Delta_{PB} = \emptyset$ here, and since *P-RANK*, *P-NNEG* and *W-EXP-NNEG* are satisfied by $\boldsymbol{\eta}$, by Lemma 6.1, C admits a LinGLexRSM map supported by I.

6.2 Algorithm for General LinPPs

While imposing *W-EXP-NNEG* lets us avoid integration in LinPPs with the BSP, this is no longer the case if we discard the BSP.

Intuitively, the problem in imposing the condition *W-EXP-NNEG* instead of *EXP-NNEG* for LinPPs without the BSP is that the set of states of smaller level over which *EXP-NNEG* performs integration might have a very small probability. However, the value of the LinGLexRSM component on that set is negative and arbitrarily large in absolute value. Thus, a naïve solution for general

LinPPs would be to "cut off" the tail events in which the LinGLexRSM component can become arbitrarily negative and over-approximate them by a constant value in order to obtain a piecewise linear GLexRSM map. However, this might lead to the jump in maximal pre-expectation and could violate *P-RANK*.

In what follows, we consider a slight restriction on the syntax of LinPPs and introduce a new condition on LEMs that allows the over-approximation trick mentioned earlier while ensuring that the *P-RANK* condition is not violated. We consider the subclass LinPP* of LinPPs in which no transition of probabilistic branching and a transition with a sampling instruction share a target location. This is a very mild restriction (satisfied, e.g., by our motivating example in Figure 1(b)), which is enforced for technical reasons arising in the proof of Lemma 6.4. Each LinPP can be converted to satisfy this property by adding a **skip** instruction in the program's source code where necessary. Second, using the notation of Definition 5.3, we define the new condition *UNBOUND* as follows:

$$\begin{aligned} \mathit{UNBOUND}(\boldsymbol{\eta}, \tau) \equiv\ & \text{if } \mathit{Up}(\tau) = (i, u) \text{ with } u \text{ containing a sampling from a dis-} \\ & \text{tribution of unbounded support, and } \ell' \text{ is the target location of } \tau, \text{ then the} \\ & \text{coefficient of the variable with index } i \text{ in } \eta_j(\ell') \text{ is 0 for all } 1 \leq j < \mathsf{lev}(\tau). \end{aligned}$$

The following technical lemma is an essential ingredient in the soundness proof of our algorithm for programs in LinPP*. Its proof can be found in the extended version of the paper [Chatterjee et al. 2021b].

Lemma 6.4. *Let C be a LinPP* and I be a linear invariant in C. If a LEM $\boldsymbol{\eta}$ satisfies P-RANK and P-NNEG for all transitions, EXP-NNEG for all transitions of probabilistic branching, W-EXP-NNEG for all other transitions, as well as UNBOUND, then C admits a piecewise linear GLexRSM map supported by I.*

Proof. Analogously, as in the proof of Lemma 6.1, we may increase $\boldsymbol{\eta}$ by a constant term in order to ensure that all transitions satisfy *EXP-NNEG*, except for maybe those that in the variable update involve sampling from a distribution of unbounded support. Thus, without loss of generality, assume that $\boldsymbol{\eta}$ satisfies *EXP-NNEG* for all other transitions. Denote the set of all transitions in C that involve sampling from distributions of unbounded support by $\mapsto^{\mathrm{unb}}$.

As before, denote by max-coeff($\boldsymbol{\eta}$) the maximal absolute value of a coefficient appearing in $\boldsymbol{\eta}$. Also, define N analogously as in the proof of Lemma 6.1, that is, for all distributions of bounded support that appear in sampling instructions and for all bounded intervals appearing in nondeterministic assignments, we have that they are supported in $[-N, N]$. Finally, since we assume that each distribution appearing in sampling instructions is integrable, for each $d \in \mathcal{D}$ we have that $\mathbb{E}_{X\sim d}[|X|] < \infty$. Thus, by triangle inequality, we also have that $\mathbb{E}_{X\sim d}[|X - \mathbb{E}[X]|] < \infty$. Hence, as $\mathbb{E}_{X\sim d}[|X-\mathbb{E}[X]|\cdot\mathbb{I}(|X-\mathbb{E}[X]| < k)] \to \mathbb{E}_{X\sim d}[|X-\mathbb{E}[X]|]$ as $k \to \infty$ by the Monotone Convergence Theorem [Williams 1991], for each $d \in \mathcal{D}$ there exists $k(d) \in \mathbb{N}$ such that

$$\mathbb{E}_{X\sim d}[|X - \mathbb{E}[X]| \cdot \mathbb{I}(|X - \mathbb{E}[X]| \geq k)] < \frac{1}{2 \cdot \text{max-coeff}(\boldsymbol{\eta})},$$

for all $k \geq k(d)$. Define $K = \max_{d\in\mathcal{D}} k(d)$, which is finite as $\mathcal{D}$ is finite.

Next, define the set $U \subseteq \{1, 2, \ldots, \dim(\boldsymbol{\eta})\} \times L$ of pairs of indices of components of $\boldsymbol{\eta}$ and locations in C as follows:

$$U = \{(j, \ell') \mid \exists \tau \in \mapsto^{\mathrm{unb}} \text{ s.t. } \mathsf{lev}(\tau) = j \text{ and } \ell' \text{ is the target location of } \tau\}.$$

Thus, U is the set of pairs of indices of components of $\boldsymbol{\eta}$ and locations in C on which the condition *UNBOUND* imposes additional template restrictions.

ALGORITHM 2: Algorithm for Proving a.s. Termination in LinPP*

Input : A LinPP* C, linear invariant I.
Output : An LEM satisfying the conditions of Lemma 6.4, if it exists

1 $\mathcal{T} \longleftarrow$ all transitions in C; $d \longleftarrow 0$
2 **while** $\mathcal{T}$ *is non-empty* **do**
3 construct $\mathcal{LP}^{\text{unb}}_{\mathcal{T}}$
4 **if** $\mathcal{LP}^{unb}_{\mathcal{T}}$ *is feasible* **then**
5 $d \longleftarrow d + 1$; $\eta_d \longleftarrow$ LEM defined by the optimal solution of $\mathcal{LP}^{\text{unb}}_{\mathcal{T}}$
6 $\mathcal{T} \longleftarrow \mathcal{T} \setminus \{\tau \in \mathcal{T} \mid \tau \text{ is 1-ranked by } \eta_d\}$
7 **else**
8 found $\longleftarrow$ false
9 **for** $\tau_0 \in \mapsto^{unb} \cap \mathcal{T}$ **do**
10 construct $\mathcal{LP}^{\tau_0,\text{unb}}_{\mathcal{T}}$
11 **if** $\mathcal{LP}^{\tau_0,unb}_{\mathcal{T}}$ *is feasible* **then**
12 $d \longleftarrow d + 1$; found $\longleftarrow$ true
13 $\eta_d \longleftarrow$ LEM defined by the optimal solution of $\mathcal{LP}^{\tau_0,\text{unb}}_{\mathcal{T}}$
14 $\mathcal{T} \longleftarrow \mathcal{T} \setminus \{\tau \in \mathcal{T} \mid \tau \text{ is 1-ranked by } \eta_d\}$
15 **if** *not found* **then return** No LEM as in Lemma 6.4
16 **return** $(\eta_1, \ldots, \eta_d)$

We define an LEM $\boldsymbol{\eta}'$ that is of the same dimension as $\boldsymbol{\eta}$, and for each component η'_j we define

$$\eta'_j(\ell, \mathbf{x}) = \begin{cases} 0 & \text{if } (j,\ell) \in U \text{ and } \eta_j(\ell,\mathbf{x}) < -C, \\ 2 \cdot \eta_j(\ell,\mathbf{x}) + 2 \cdot C & \text{otherwise,} \end{cases}$$

where $C > 0$ is a constant to be determined. We claim that there exists $C > 0$ for which $\boldsymbol{\eta}'$ is a piecewise linear GLexRSM map supported by I (with the level map being the same as for $\boldsymbol{\eta}$). The fact that $\boldsymbol{\eta}'$ is piecewise linear for every $C > 0$ is clear from its definition. Thus, we are left to verify that there exists $C > 0$ for which *P-NNEG*$(\boldsymbol{\eta}', \tau)$, *P-RANK*$(\boldsymbol{\eta}', \tau)$, and *EXP-NNEG*$(\boldsymbol{\eta}', \tau)$ hold for each transition τ. A careful analysis, which can be found in the extended version of the paper [Chatterjee et al. 2021b], shows that $C = (2N + K) \cdot \text{max-coeff}(\boldsymbol{\eta}) + 1$ satisfies the claim. Hence, C admits a piecewise linear GLexRSM map supported by I. □

Algorithm. The new algorithm shares an overall structure with the algorithm from Section 6.1. Thus, we only give a high level overview and focus on novel aspects. The algorithm pseudocode is presented in Algorithm 2.

The condition *UNBOUND* is encoded by modifying the templates for the new LEM components. Let $\mapsto^{\text{unb}}$ be the set of transitions in C containing sampling from unbounded support distributions, and for any such transition τ, let ℓ'_τ be its target location. Then, for any set of transitions $\mathcal{T}$, construct a linear program $\mathcal{LP}^{\text{unb}}_{\mathcal{T}}$ analogous to $\mathcal{LP}_{\mathcal{T}}$ in Section 6.1, additionally enforcing that for each $\tau \in \mapsto^{\text{unb}} \cap \mathcal{T}$, the coefficient of the variable updated by τ in the LEM template at ℓ'_τ is 0. Algorithm 2 first tries to prune as many transitions as possible by repeatedly solving $\mathcal{LP}^{\text{unb}}_{\mathcal{T}}$ and removing ranked transitions from $\mathcal{T}$; see lines 3 to 6. Once no more transitions can be ranked, the algorithm tries to rank new transitions by allowing non-zero template coefficients previously required to be 0 while still enforcing *UNBOUND*. For a set of transitions $\mathcal{T}$ and for $\tau_0 \in \mapsto^{\text{unb}} \cap \mathcal{T}$, we construct a linear program $\mathcal{LP}^{\tau_0,\text{unb}}_{\mathcal{T}}$ analogous to $\mathcal{LP}^{\text{unb}}_{\mathcal{T}}$ but allowing a non-zero coefficient

of the variable updated by τ_0 at ℓ'_{τ_0}. However, we further impose that the new component 1-ranks any other transition in $\mapsto^{\text{unb}} \cap \mathcal{T}$ with the target location ℓ'_{τ_0}. This new linear program is solved for all $\tau_0 \in \mapsto^{\text{unb}} \cap \mathcal{T}$ and all 1-ranked transitions are removed from $\mathcal{T}$, as in Algorithm 2, lines 7 to 15. The process continues until all transitions are pruned from $\mathcal{T}$ or until no remaining transition can be 1-ranked, in which case no LEM as in Lemma 6.4 exists.

Theorem 6.5. *Algorithm 2 decides in polynomial time whether a LinPP* $\mathcal{C}$ admits an LEM that satisfies all conditions of Lemma 6.4 and that is supported by I. Thus, if the algorithm outputs an LEM, then $\mathcal{C}$ is a.s. terminating and admits a piecewise linear GLexRSM map supported by I.*

The proof of Theorem 6.5 uses Lemma 6.4. The completeness argument is similar to that in the proof of Theorem 6.2; thus, we omit the details. The full proof can be found in the extended version of the paper [Chatterjee et al. 2021b]. We conclude this section by showing that Algorithm 2 can prove a.s. termination of our motivating example in Figure 1(a) and the simple loop in Figure 2.

Example 6.6. Consider the program in Figure 1(a) with a linear invariant $I(\ell_0) = \mathit{true}$, $I(\ell_1) = y \geq 0$. The LEM defined via $\boldsymbol{\eta}(\ell_0, (x, y)) = (1, 2y + 2, x + 1)$, $\boldsymbol{\eta}(\ell_1, (x, y)) = (1, 2y + 1, x + 1)$ and $\boldsymbol{\eta}(\ell_{out}, (x, y)) = (0, 2y + 2, x + 1)$ satisfies *P-RANK*, *P-NNEG*, and *W-EXP-NNEG*, which is easy to check. Furthermore, the only transition containing a sampling instruction is the self-loop at ℓ_1, which is ranked by the third component of $\boldsymbol{\eta}$. As the coefficients of x of the first two components at ℓ_1 are equal to 0, $\boldsymbol{\eta}$ also satisfies *UNBOUND*. Hence, $\boldsymbol{\eta}$ satisfies all conditions of Lemma 6.4 and Algorithm 2 proves a.s. termination.

Example 6.7. Consider now the simple loop in Figure 2 with a linear invariant $I(\ell_0) = \mathit{true}$. The LEM defined via $\boldsymbol{\eta}(\ell_0, x) = (1, x + 1)$, and $\boldsymbol{\eta}(\ell_{out}, x) = (0, x + 1)$ satisfies *P-RANK*, *P-NNEG*, and *W-EXP-NNEG*, which is easy to check. The only transition containing a sampling instruction is the self-loop at ℓ_0, which is ranked by the second component of $\boldsymbol{\eta}$, and the coefficient of x of the first components at ℓ_0 is equal to 0. Hence, $\boldsymbol{\eta}$ also satisfies *UNBOUND*. Hence, $\boldsymbol{\eta}$ satisfies all conditions of Lemma 6.4 and Algorithm 2 proves a.s. termination.

7 CONCLUSION

In this work, we consider lexicographic termination certificates in probabilistic programs. We show how the strong non-negativity condition imposed by the lexicographic ranking supermartingales (LexRSMs) of Agrawal et al. [2018] can be relaxed so that it does not require all components to be non-negative, leading to generalized LexRSMs (GLexRSMs). We prove that GLexRSMs are sound for a.s. termination analysis in probabilistic programs. While such a result is standard in the literature on termination analysis in non-probabilistic programs, it was hitherto not known to hold for probabilistic programs. We then present two polynomial-time algorithms for the synthesis of linear GLexRSMs in probabilistic programs with linear arithmetic. Finally, we demonstrate that our algorithms can compute linear GLexRSMs for two linear arithmetic probabilistic programs whose a.s. termination cannot be witnessed by linear LexRSMs. In particular, we show that our GLexRSMs can even tackle programs that contain sampling instructions from probability distributions of unbounded support, whose termination analysis is not handled well by the existing 1-dimensional variants of ranking supermartingales as well as LexRSMs. There are several interesting directions of future work. The expected leftward non-negativity condition of GLexRSMs does not allow for a fully compositional a.s. termination analysis. Thus, it would be interesting to consider further relaxation of the conditions imposed by GLexRSMs in order to derive a compositional proof rule for a.s. termination. An interesting algorithmic problem would be to automate the synthesis of GLexRSMs for programs with non-linear arithmetic.

REFERENCES

Sheshansh Agrawal, Krishnendu Chatterjee, and Petr Novotný. 2018. Lexicographic ranking supermartingales: An efficient approach to termination of probabilistic programs. *PACMPL* 2, POPL (2018), 34:1–34:32.

Christophe Alias, Alain Darte, Paul Feautrier, and Laure Gonnord. 2010. Multi-dimensional rankings, program termination, and complexity bounds of flowchart programs. In *Proceedings of the 17th International Conference on Static Analysis* (Perpignan, France) *(SAS'10)*. Springer, Berlin, 117–133. http://dl.acm.org/citation.cfm?id=1882094.1882102.

R. B. Ash and C. Doléans-Dade. 2000. *Probability and Measure Theory*. Harcourt/Academic Press.

M. Avanzini, U. Dal Lago, and A. Ghyselen. 2019. Type-based complexity analysis of probabilistic functional programs. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS'19)*. 1–13. https://doi.org/10.1109/LICS.2019.8785725

Martin Avanzini, Ugo Dal Lago, and Akihisa Yamada. 2020a. On probabilistic term rewriting. *Science of Computer Programming* 185 (2020), 102338. https://doi.org/10.1016/j.scico.2019.102338

Martin Avanzini, Georg Moser, and Michael Schaper. 2020b. A modular cost analysis for probabilistic programs. *Proceedings of the ACM on Programming Languages* 4, OOPSLA (2020), 1–30. https://doi.org/10.1145/3428240

Christel Baier and Joost-Pieter Katoen. 2008. *Principles of Model Checking*. The MIT Press, Cambridge, Massachusetts.

Gilles Barthe, Thomas Espitau, Luis María Ferrer Fioriti, and Justin Hsu. 2016a. Synthesizing probabilistic invariants via Doob's decomposition. In *Proceedings of the 28th International Conference on Computer Aided Verification (CAV'16), Part I, Toronto, ON, Canada, July 17-23, 2016*, Swarat Chaudhuri and Azadeh Farzan (Eds.). Springer International Publishing, 43–61. https://doi.org/10.1007/978-3-319-41528-4_3

Gilles Barthe, Marco Gaboardi, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. 2016b. Proving differential privacy via probabilistic couplings. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science* (New York, NY, USA) *(LICS'16)*. ACM, New York, NY, 749–758. https://doi.org/10.1145/2933575.2934554

Gilles Barthe, Marco Gaboardi, Justin Hsu, and Benjamin Pierce. 2016c. Programming language techniques for differential privacy. *ACM SIGLOG News* 3, 1 (Feb. 2016), 34–53. https://doi.org/10.1145/2893582.2893591

Amir M. Ben-Amram and Samir Genaim. 2013. On the linear ranking problem for integer linear-constraint loops. In *Proceedings of the 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (Rome, Italy) *(POPL'13)*. ACM, New York, NY, 51–62. https://doi.org/10.1145/2429069.2429078

Amir M. Ben-Amram and Samir Genaim. 2015. Complexity of Bradley-Manna-Sipma lexicographic ranking functions. In *Proceedings of the 27th International Conference on Computer Aided Verification (CAV'15, San Francisco, CA, July 18-24, 2015*, Daniel Kroening and Corina S. Păsăreanu (Eds.). Springer International Publishing, 304–321. https://doi.org/10.1007/978-3-319-21668-3_18

P. Billingsley. 1995. *Probability and Measure* (3rd ed.). Wiley.

Olivier Bournez and Florent Garnier. 2005. Proving positive almost-sure termination. In *RTA*. 323–337.

Aaron R. Bradley, Zohar Manna, and Henny B. Sipma. 2005. Linear ranking with reachability. In *Proceedings of the 17th International Conference on Computer Aided Verification (CAV'05), Edinburgh, Scotland, UK, July 6-10, 2005*. 491–504. https://doi.org/10.1007/11513988_48

Marc Brockschmidt, Byron Cook, and Carsten Fuhs. 2013. Better termination proving through cooperation. In *Proceedings of the 25th International Conference on Computer Aided Verification (CAV'13), Saint Petersburg, Russia, July 13-19, 2013*. 413–429. https://doi.org/10.1007/978-3-642-39799-8_28

Marc Brockschmidt, Byron Cook, Samin Ishtiaq, Heidy Khlaaf, and Nir Piterman. 2016. T2: Temporal property verification. In *Proceedings of the 22nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems(TACAS'16), Held as Part of the European Joint Conferences on Theory and Practice of Software (ETAPS'16), Eindhoven, The Netherlands, April 2-8, 2016*, Marsha Chechik and Jean-François Raskin (Eds.). Springer, Berlin, 387–393. https://doi.org/10.1007/978-3-662-49674-9_22

Aleksandar Chakarov and Sriram Sankaranarayanan. 2013. Probabilistic program analysis with martingales. In *CAV 2013*. 511–526.

Aleksandar Chakarov, Yuen-Lam Voronin, and Sriram Sankaranarayanan. 2016. Deductive proofs of almost sure persistence and recurrence properties. In *Proceedings of the 22nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'16), Held as Part of the European Joint Conferences on Theory and Practice of Software (ETAPS'16), Eindhoven, The Netherlands, April 2-8, 2016,* Marsha Chechik and Jean-François Raskin (Eds.). Springer, Berlin, 260–279. https://doi.org/10.1007/978-3-662-49674-9_15

Krishnendu Chatterjee, Hongfei Fu, and Amir Kafshdar Goharshady. 2016. Termination analysis of probabilistic programs through positivstellensatz's. In *CAV*. 3–22.

Krishnendu Chatterjee, Hongfei Fu, Petr Novotný, and Rouzbeh Hasheminezhad. 2018. Algorithmic analysis of qualitative and quantitative termination problems for affine probabilistic programs. *ACM Transactions on Programming Languages and Systems* 40, 2 (2018), 7:1–7:45. https://doi.org/10.1145/3174800

Krishnendu Chatterjee, Amir Kafshdar Goharshady, Tobias Meggendorfer, and Dorde Zikelic. 2022. Sound and complete certificates for quantitative termination analysis of probabilistic programs. In *Proceedings of the 34th International Conference on Computer Aided Verification (CAV'22), Haifa, Israel, August 7-10, 2022, Part I (Lecture Notes in Computer Science), Vol. 13371*, Sharon Shoham and Yakir Vizel (Eds.). Springer, Berlin, 55–78. https://doi.org/10.1007/978-3-031-13185-1_4

Krishnendu Chatterjee, Ehsan Kafshdar Goharshady, Petr Novotný, Jiri Zárevúcky, and Dorde Zikelic. 2021a. On lexicographic proof rules for probabilistic termination. In *Proceedings of the 24th International Symposium on Formal Methods (FM'21), Virtual Event, November 20-26, 2021 (Lecture Notes in Computer Science), Vol. 13047*, Marieke Huisman, Corina S. Pasareanu, and Naijun Zhan (Eds.). Springer, Berlin, 619–639. https://doi.org/10.1007/978-3-030-90870-6_33

Krishnendu Chatterjee, Ehsan Kafshdar Goharshady, Petr Novotný, Jiří Zárevúcky, and Dorde Zikelic. 2021b. On Lexicographic Proof Rules for Probabilistic Termination. arXiv:2108.02188 [cs.PL] https://arxiv.org/abs/2108.02188.

Krishnendu Chatterjee, Petr Novotný, and Dorde Zikelic. 2017. Stochastic invariants for probabilistic termination. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages* (Paris, France) *(POPL'17)*. ACM, New York, NY, 145–160. https://doi.org/10.1145/3009837.3009873

Jianhui Chen and Fei He. 2020. Proving almost-sure termination by omega-regular decomposition. In *Proceedings of the 41st ACM SIGPLAN International Conference on Programming Language Design and Implementation (PLDI'20), London, UK, June 15-20, 2020*. 869–882. https://doi.org/10.1145/3385412.3386002

Guillaume Claret, Sriram K. Rajamani, Aditya V. Nori, Andrew D. Gordon, and Johannes Borgström. 2013. Bayesian inference using data flow analysis. In *Joint Meeting on Foundations of Software Engineering*. ACM, 92–102.

Michael Colón and Henny Sipma. 2001. Synthesis of linear ranking functions. In *Proceedings of the 22nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'01) Held as Part of the Joint European Conferences on Theory and Practice of Software (ETAPS'01) Genova, Italy, April 2-6, 2001*. 67–81. https://doi.org/10.1007/3-540-45319-9_6

Byron Cook, Andreas Podelski, and Andrey Rybalchenko. 2006. Termination proofs for systems code. *ACM SIGPLAN Notices* 41, 6 (June 2006), 415–426. https://doi.org/10.1145/1133255.1134029

Byron Cook, Andreas Podelski, and Andrey Rybalchenko. 2011. Proving program termination. *Communications of the ACM* 54, 5 (2011), 88–98. https://doi.org/10.1145/1941487.1941509

Byron Cook, Abigail See, and Florian Zuleger. 2013. Ramsey vs. lexicographic termination proving. In *Proceedings of the 19th International Conference on Tools and Algorithms for the Construction and Analysis of Systems* (Rome, Italy) *(TACAS'13)*. Springer, Berlin, 47–61. https://doi.org/10.1007/978-3-642-36742-7_4

Patrick Cousot and Radhia Cousot. 1977. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conference Record of the 4th ACM Symposium on Principles of Programming Languages, Los Angeles, CA, January 1977*. 238–252. https://doi.org/10.1145/512950.512973

Ugo Dal Lago, Claudia Faggian, and Simona Ronchi Della Rocca. 2021. Intersection types and (positive) almost-sure termination. *Proceedings of the ACM on Programming Languages* 5, POPL, Article 32 (Jan. 2021), 32 pages. https://doi.org/10.1145/3434313

Devdatt Dubhashi and Alessandro Panconesi. 2009. *Concentration of Measure for the Analysis of Randomized Algorithms* (1st ed.). Cambridge University Press, New York, NY.

Javier Esparza, Andreas Gaiser, and Stefan Kiefer. 2012. Proving termination of probabilistic programs using patterns. In *CAV 2012*. 123–138.

Yishai A. Feldman. 1984. A decidable propositional dynamic logic with explicit probabilities. *Information and Control* 63, 1 (1984), 11–38. https://doi.org/10.1016/S0019-9958(84)80039-X

Yishai A. Feldman and David Harel. 1982. A probabilistic dynamic logic. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing*. ACM, 181–195.

Luis María Ferrer Fioriti and Holger Hermanns. 2015. Probabilistic termination: Soundness, completeness, and compositionality. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, (POPL'15), Mumbai, India, January 15-17, 2015*. 489–501. https://doi.org/10.1145/2676726.2677001

Robert W. Floyd. 1967. Assigning meanings to programs. *Mathematical Aspects of Computer Science* 19 (1967), 19–33.

F. G. Foster. 1953. On the stochastic matrices associated with certain queuing processes. *The Annals of Mathematical Statistics* 24, 3 (1953), 355–360.

Nate Foster, Dexter Kozen, Konstantinos Mamouras, Mark Reitblatt, and Alexandra Silva. 2016. Probabilistic NetKAT. In *ESOP'16*. Springer, 282–309.

Hongfei Fu and Krishnendu Chatterjee. 2019. Termination of nondeterministic probabilistic programs. In *Proceedings of the 20th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'19), Cascais, Portugal, January 13-15, 2019 (Lecture Notes in Computer Science), Vol. 11388*, Constantin Enea and Ruzica Piskac (Eds.). Springer, Berlin, 468–490. https://doi.org/10.1007/978-3-030-11245-5_22

Zoubin Ghahramani. 2015. Probabilistic machine learning and artificial intelligence. *Nature* 521, 7553 (2015), 452–459.

Jürgen Giesl, Peter Giesl, and Marcel Hark. 2019. Computing expected runtimes for constant probability programs. In *Automated Deduction (CADE 27)*, Pascal Fontaine (Ed.). Springer International Publishing, Cham, 269–286.

Laure Gonnord, David Monniaux, and Gabriel Radanne. 2015. Synthesis of ranking functions using extremal counterexamples. In *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation* (Portland, OR) *(PLDI'15)*. ACM, New York, NY, 608–618. https://doi.org/10.1145/2737924.2737976

Andrew D. Gordon, Mihhail Aizatulin, Johannes Borgstrom, Guillaume Claret, Thore Graepel, Aditya V. Nori, Sriram K. Rajamani, and Claudio Russo. 2013. A model-learner pattern for Bayesian reasoning. *ACM SIGPLAN Notices* 48, 1 (2013), 403–416.

Andrew D. Gordon, Thomas A. Henzinger, Aditya V. Nori, and Sriram K. Rajamani. 2014. Probabilistic programming. In *Proceedings of the on Future of Software Engineering*. ACM, 167–181.

Friedrich Gretz, Joost-Pieter Katoen, and Annabelle McIver. 2014. Operational versus weakest pre-expectation semantics for the probabilistic guarded command language. *Performance Evaluation* 73 (2014), 110–132. https://doi.org/10.1016/j.peva.2013.11.004

Marcel Hark, Benjamin Lucien Kaminski, Jürgen Giesl, and Joost-Pieter Katoen. 2020. Aiming low is harder: Induction for lower bounds in probabilistic program verification. *Proceeding of the ACM on Programming Languages* 4, POPL (2020), 37:1–37:28. https://doi.org/10.1145/3371105

Mingzhang Huang, Hongfei Fu, and Krishnendu Chatterjee. 2018. New approaches for almost-sure termination of probabilistic programs. In *Programming Languages and Systems*, Sukyoung Ryu (Ed.). Springer International Publishing, Cham, 181–201.

Mingzhang Huang, Hongfei Fu, Krishnendu Chatterjee, and Amir Kafshdar Goharshady. 2019. Modular verification for almost-sure termination of probabilistic programs. *Proceeding of the ACM on Programming Languages* 3, OOPSLA (2019), 129:1–129:29. https://doi.org/10.1145/3360555

L. P. Kaelbling, M. L. Littman, and A. W. Moore. 1996. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research* 4 (1996), 237–285.

Benjamin Lucien Kaminski, Joost-Pieter Katoen, Christoph Matheja, and Federico Olmedo. 2018b. Weakest precondition reasoning for expected runtimes of randomized algorithms. *Journal of the ACM* 65, 5 (2018), 30:1–30:68. https://doi.org/10.1145/3208102

Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. 2018a. On the hardness of analyzing probabilistic programs. *Acta Informatica* (2018), 1–31.

Naoki Kobayashi, Ugo Dal Lago, and Charles Grellois. 2020. On the termination problem for probabilistic higher-order recursive programs. *Logical Methods in Computer Science* 16, 4 (2020). https://lmcs.episciences.org/6817.

Dexter Kozen. 1981. Semantics of probabilistic programs. *Journal of Computer and System Sciences* 22, 3 (1981), 328–350. https://doi.org/10.1016/0022-0000(81)90036-2

Dexter Kozen. 1983. A probabilistic PDL. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing (STOC'83)*. ACM, New York, NY, 291–297. https://doi.org/10.1145/800061.808758

Marta Z. Kwiatkowska, Gethin Norman, and David Parker. 2011. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV'11 (Lecture Notes in Computer Science)*, Ganesh Gopalakrishnan and Shaz Qadeer (Eds.). Vol. 6806, Springer, Berlin. 585–591.

Ugo Dal Lago and Charles Grellois. 2019. Probabilistic termination by monadic affine sized typing. *ACM Transactions on Programming Languages and Systems* 41, 2 (2019), 10:1–10:65. https://doi.org/10.1145/3293605

Annabelle McIver and Carroll Morgan. 2004. Developing and reasoning about probabilistic programs in *pGCL*. In *PSSE*. 123–155.

Annabelle McIver and Carroll Morgan. 2005. *Abstraction, Refinement and Proof for Probabilistic Systems*. Springer.

Annabelle McIver and Carroll Morgan. 2016. A new rule for almost-certain termination of probabilistic and demonic programs. *CoRR* abs/1612.01091 (2016). http://arxiv.org/abs/1612.01091.

Annabelle McIver, Carroll Morgan, Benjamin Lucien Kaminski, and Joost-Pieter Katoen. 2018. A new proof rule for almost-sure termination. *Proc. ACM Program. Lang.* 2, POPL (2018).

David Monniaux. 2001. An abstract analysis of the probabilistic termination of programs. In *Proceedings of the 8th International Symposium on Static Analysis (SAS'01), Paris, France, July 16-18, 2001 (Lecture Notes in Computer Science), Vol. 2126*, Patrick Cousot (Ed.). Springer, Berlin, 111–126. https://doi.org/10.1007/3-540-47764-0_7

Marcel Moosbrugger, Ezio Bartocci, Joost-Pieter Katoen, and Laura Kovács. 2021. Automated termination analysis of polynomial probabilistic programs. In *Programming Languages and Systems - Proceedings of the 30th European Symposium on Programming (ESOP'21), Held as Part of the European Joint Conferences on Theory and Practice of Software (ETAPS'21), Luxembourg City, Luxembourg, March 27 - April 1, 2021 (Lecture Notes in Computer Science), Vol. 12648*, Nobuko Yoshida (Ed.). Springer, Berlin, 491–518. https://doi.org/10.1007/978-3-030-72019-3_18

Carroll Morgan and A McIver. 1999. pGCL: Formal reasoning for random algorithms. (1999).

Carroll Morgan, Annabelle McIver, and Karen Seidel. 1996. Probabilistic predicate transformers. *ACM Transactions on Programming Languages and Systems* 18, 3 (1996), 325–353.

Rajeev Motwani and Prabhakar Raghavan. 1995. *Randomized Algorithms*. Cambridge University Press, New York, NY.

Martin R. Neuhäußer and Joost-Pieter Katoen. 2007. Bisimulation and logical preservation for continuous-time Markov decision processes. In *International Conference on Concurrency Theory (CONCUR'07)*. Springer, 412–427.

Martin R. Neuhäußer, Mariëlle Stoelinga, and Joost-Pieter Katoen. 2009. Delayed nondeterminism in continuous-time Markov decision processes. In *Proceedings of the 12th International Conference of Foundations of Software Science and Computational Structures (FOSSACS'09), Held as Part of the Joint European Conferences on Theory and Practice of Software (ETAPS'09), York, UK, March 22-29, 2009*. 364–379. https://doi.org/10.1007/978-3-642-00596-1_26

Van Chan Ngo, Quentin Carbonneaux, and Jan Hoffmann. 2018. Bounded expectations: Resource analysis for probabilistic programs. In *PLDI 2018*. 496–512.

Federico Olmedo, Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. 2016. Reasoning about recursive probabilistic programs. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science* (New York, NY) *(LICS'16)*. ACM, New York, NY, 672–681. https://doi.org/10.1145/2933575.2935317

Andreas Podelski and Andrey Rybalchenko. 2004a. A complete method for the synthesis of linear ranking functions. In *Proceedings of the 5th International Conference of Verification, Model Checking, and Abstract Interpretation (VMCAI'04), Venice, January 11-13, 2004*. 239–251. https://doi.org/10.1007/978-3-540-24622-0_20

Andreas Podelski and Andrey Rybalchenko. 2004b. Transition invariants. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science (LICS'04)*. IEEE Computer Society, Washington, DC, 32–41. https://doi.org/10.1109/LICS.2004.50

D. M. Roy, V. K. Mansinghka, N. D. Goodman, and J. B. Tenenbaum. 2008. A stochastic programming perspective on nonparametric Bayes. In *Nonparametric Bayesian Workshop, International Conference on Machine Learning*, Vol. 22. 26.

Adam Ścibior, Zoubin Ghahramani, and Andrew D. Gordon. 2015. Practical probabilistic programming with monads. *ACM SIGPLAN Notices* 50, 12 (2015), 165–176.

Steffen Smolka, Praveen Kumar, Nate Foster, Dexter Kozen, and Alexandra Silva. 2017. Cantor meets Scott: Semantic foundations for probabilistic networks. In *POPL'17*. 557–571.

Kirack Sohn and Allen Van Gelder. 1991. Termination detection in logic programs using argument sizes. In *Proceedings of the 10th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 29-31, 1991, Denver, CO*. 216–226. https://doi.org/10.1145/113413.113433

Sebastian Thrun. 2002. Probabilistic robotics. *Communications of the ACM* 45, 3 (2002), 52–57.

Di Wang, Jan Hoffmann, and Thomas W. Reps. 2018. PMAF: An algebraic framework for static analysis of probabilistic programs. In *PLDI'18*. 513–528.

Peixin Wang, Hongfei Fu, Amir Kafshdar Goharshady, Krishnendu Chatterjee, Xudong Qin, and Wenjun Shi. 2019. Cost analysis of nondeterministic probabilistic programs. In *PLDI'19*. 204–220.

D. Williams. 1991. *Probability with Martingales*. Cambridge University Press, Cambridge, UK. 251 pages.

Received 31 March 2022; revised 20 December 2022; accepted 7 February 2023