11-2021

On lexicographic proof rules for probabilistic termination

Krishnendu CHATTERJEE

Ehsan Kafshdar GOHARSHADY

Petr NOVOTNÝ

Jiří ZÁREVUCKÝ

Dorde ZIKELIC
Singapore Management University, dzikelic@smu.edu.sg

Citation

CHATTERJEE, Krishnendu; GOHARSHADY, Ehsan Kafshdar; NOVOTNÝ, Petr; ZÁREVUCKÝ, Jiří; and ZIKELIC, Dorde. On lexicographic proof rules for probabilistic termination. (2021). *Proceedings of the 24th International Symposium, FM 2021 Virtual Conference, November 20-26*. 619-639.
Available at: https://ink.library.smu.edu.sg/sis_research/9070

On Lexicographic Proof Rules for Probabilistic Termination

Krishnendu Chatterjee[1], Ehsan Kafshdar Goharshady[2], Petr Novotný[3], Jiří Zárevúcky[3], and Đorđe Žikelić[1](✉)

[1] IST Austria, Klosterneuburg, Austria
{krishnendu.chatterjee,djordje.zikelic}@ist.ac.at

[2] Ferdowsi University of Mashhad, Mashhad, Iran
e.kafshdargoharshady@mail.um.ac.ir

[3] Masaryk University, Brno, Czech Republic
{petr.novotny,xzarevuc}@fi.muni.cz

Abstract. 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 LexRSM not existing even for simple terminating programs. Our contributions are twofold: First, we introduce a generalization of LexRSMs which 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.

Keywords: Probabilistic programs · Termination · Martingales

1 Introduction

The extension of classical imperative programs with randomization gives rise to probabilistic programs (PPs) [45], which are used in multitude of applications, including stochastic network protocols [7,39,56,75], randomized algorithms [32,66], security [9,10], machine learning, and planning [25,41,44,50,73,74,77]. The analysis of PPs is an active research area in formal methods [1,18,20,21,33,51,52,69,70,78]. PPs can be extended with nondeterminism to allow over-approximating program parts that are too complex for static analysis [30,59].

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 [27,28]. Such techniques typically seek a suitable *certificate* of termination. Particularly relevant certificates are *ranking functions (RFs)* [15,26,37,71,72,76] 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 multi-dimensional extensions of RFs that provide an effective approach to termination analysis [2,15–17,29,43]. The literature typically restricts to linear LexRFs for linear-arithmetic (LA) programs, as LA reasoning can be more efficiently automated compared to 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 *ranking supermartingales (RSMs)*, a probabilistic analogue of ranking functions named so due to the connection with (super)martingale stochastic processes [80]. There is a rich body of work on 1-dimensional RSMs, while the work [1] 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 [1] 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. *P-RANK*$(f, \tau) \equiv s \models G(\tau) \Rightarrow \big(\mathbb{E}^\tau[f_i(s')] \leq f_i(s) - 1$ and $\mathbb{E}^\tau[f_j(s')] \leq f_j(s)$ for all $1 \leq j < i\big)$.
2. *S-NNEG*$(f, \tau) \equiv s \models G(\tau) \Rightarrow \big(f_j(s) \geq 0$ for all $1 \leq j \leq d\big)$.

(We use the standard primed notation from program analysis, i.e. 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 [1]. Apart from the soundness proof, [1] 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.

```
$\ell_0$:  while $y \geq 0$ do
               $x := y$;
$\ell_1$:      while $x \geq 0$ do
                   $x := x - 1 + Norm(0,1)$
               od;
               $y := y - 1$
           od
```

(a)

```
$\ell_0$:  while $x \geq 0$ do
               if $y \geq 0$ then
                   $y := y + Uni[-7,1]$
               else
                   $x := x + Uni[-7,1]$;
$\ell_1$:          $y := y + Uni[-7,1]$
               fi od
```

(b)

Fig. 1. Motivating examples. $Norm(\mu, \sigma)$ samples from the normal distribution with mean μ and std. 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) so as to have one assignment per transition (a technical requirement for our approach). A formal representation of the programs via *probabilistic control flow graphs* is presented later, in Sect. 4.

Consider the program in Fig. 1a. 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, LexRSMs with strong non-negativity do not handle well programs with unbounded-support distributions.

Now consider the program in Fig. 1b. It can be again 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. But y can become arbitrarily negative within the else branch, and cannot be bounded from below by a linear function of x.

Contribution: Generalized Lexicographic RSMs. In the non-probabilistic setting, strong non-negativity can be relaxed to *partial non-negativity* (*P-NNEG*), where only the components which are to the left of the "ranking component" i (inclusive) need to be non-negative (Ben-Amram–Genaim RFs [12]). 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

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

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.

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.

Contribution: Polynomial Algorithms for Linear GLexRSMs.

1. 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. In particular, our approach proves the a.s. termination of the program in Fig. 1b.
2. Building on 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. In particular, the algorithm proves a.s. termination for our motivating example in Fig. 1a.

Related Work. Martingale-based termination literature mostly focused on 1-dimensional RSMs [18,20,21,23,36,40,42,48,60,61,63]. RSMs themselves can be seen as generalizations of Lyapunov ranking functions from control theory [14,38]. Recently, the work [49] pointed out the unsoundess of the 1-dimensional RSM-based proof rule in [36] due to insufficient lower bound conditions and provided a corrected version. On the multi-dimensional front, it was shown in [36] that requiring components of (lexicographic) RSMs to be nonnegative only at points where they are used to rank some enabled transition (analogue of Bradley-Manna-Sipma LexRFs [15]) 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 [1]. The work [24] produces an ω-regular decomposition of program's control-flow graph, with each program component ranked by a different RSM. This approach does not require a lexicographic ordering of RSMs, but each component in the decomposition must be ranked by a single-dimensional non-negative RSM. RSM approaches were also used for cost analysis [6,69,79] and additional liveness and safety properties [8,19,23].

Logical calculi for reasoning about properties of probabilistic programs (including termination) were studied in [34,35,54,55] and extended to programs with non-determinism in [46,52,58,59,70]. In particular [58,59,61] formalize RSM-like proof certificates within the *weakest pre-expectation (WPE)* calculus [64,65]. The power of this calculus allows for reasoning about complex programs [61, Sect. 5], but the proofs typically require a human input. Theoretical connections between martingales and the WPE calculus were recently explored in [47]. There is also a rich body of work on analysis of probabilistic functional programs, where the aim is typically to obtain a general type system [4,31,53,57] for reasoning about termination properties (automation for discrete probabilistic term rewrite systems was shown in [5]).

As for other approaches to a.s. termination, for *finite-state programs* with nondeterminism a sound and complete method was given in [33], while [62] considers a.s. termination proving through abstract interpretation. The work [51] 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 [11,12].

Paper Organization. The paper is split in two parts: the first one is "abstract", with mathematical preliminaries (Sect. 2) and definition and soundness proof of abstract GLexRSMs (Sect. 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 probabilistic programs: preliminaries on the program syntax and semantics (Sect. 4), a GLexRSM-based proof rule for a.s. termination (Sect. 5), and the outline of our algorithms (Sect. 6).

2 Mathematical Preliminaries

We use boldface notation for vectors, e.g. $\mathbf{x}$, $\mathbf{y}$, etc., 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 [80]. 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 $x \in \mathbb{R}$) and 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 s.t.: 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, i.e. function returning 1 for each $\omega \in A$ and 0 for each $\omega \in \Omega \setminus A$.

It is known [3] that a random variable satisfying the properties of conditional expectation exists whenever a) $\mathbb{E}[|X|] < \infty$, i.e. X is *integrable*, or b) X is real-valued and nonnegative (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 $\mathbb{E}[X \cdot Z|\mathcal{F}'] = \mathbb{E}[X|\mathcal{F}'] \cdot Z$, whenever the former conditional expectation exists [80, Theorem 9.7(j)].

A *filtration* in $(\Omega, \mathcal{F}, \mathbb{P})$ is an increasing (w.r.t. set inclusion) sequence $\{\mathcal{F}_t\}_{t=0}^{\infty}$ of sub-sigma-algebras of $\mathcal{F}$. A *stopping time* w.r.t. a filtration $\{\mathcal{F}_t\}_{t=0}^{\infty}$ is a random variable T taking values in $\mathbb{N} \cup \{\infty\}$ s.t. for every t the set $\{T = t\} = \{\omega \in \Omega \mid T(\omega) = t\}$ belongs to $\mathcal{F}_t$. Intuitively, T returns a time step in which some process should be "stopped", and 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 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 w.r.t. $\mathcal{F}$. 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* $\omega \in \cup_{j'=j}^{n} L_{j'}^t$,
   - $\mathbb{E}[\mathbf{X}_{t+1}[j] \cdot \mathbb{I}(\cup_{j'=0}^{j-1} L_{j'}^{t+1}) \mid \mathcal{F}_i](\omega) \geq 0$ *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 paper, this will be a probabilistic program run). Then, $\mathbf{X}_t$ is a vector function depending on the first t time steps (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 which "ranks" ω in step t, and the last item encodes the expected leftward non-negativity (sketched in Sect. 1). For each $1 \leq j \leq n$ and time step $t \geq 0$, the set L_j^t contains all $\omega \in \{T > t\}$ which 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 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 almost-sure termination.

Theorem 1. *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 w.r.t. $\mathcal{F}$. 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 [1], a mathematical notion of LexRSMs is defined and a result for LexRSMs analogous to our Theorem 1 is established. Thus, the first part of our proof mostly resembles the proof of Theorem 3.3. in [1], up to the point of defining the stochastic process $(Y_t)_{t=0}^{\infty}$ in Eq. (1). After that, the proof of [1] crucially relies on nonnegativity of each $\mathbf{X}_t[j]$ and Y_t at every $\omega \in \varOmega$ 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 $t \geq 0$, and then we present a very elegant argument via the Borel-Cantelli lemma [80, Theorem 2.7] which shows that this boundedness of expectation is sufficient for the theorem claim to hold.

Proof (Sketch of Proof of Theorem 1). 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 $\omega \in \varOmega$ for which the following properties hold has positive measure, i.e. $\mathbb{P}[B] > 0$: (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, see the extended version of the paper [22].

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 $D = \{\omega \in \varOmega \mid \mathbf{X}_s[k](\omega) \leq M \wedge \omega \in \cup_{j=k}^{n} L_j^s\}$. Note that D is $\mathcal{F}_t$-measurable for $t \geq s$. We define a stopping time F w.r.t. $(\mathcal{F}_t)_{t=0}^{\infty}$ 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 [22]) shows that for each $t \geq s$ we have $\mathbb{E}[Y_{t+1}] \leq \mathbb{E}[Y_t] - \mathbb{P}[L_k^t \cap D \cap \{F > t\}]$. By a simple induction we obtain:

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

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

$$\begin{aligned}
\mathbb{E}[Y_t] &= \mathbb{E}[Y_t \cdot \mathbb{I}(F = s)] + \sum_{r=s+1}^{t} \mathbb{E}[Y_t \cdot \mathbb{I}(F = r)] + \mathbb{E}[Y_t \cdot \mathbb{I}(F > t)] \\
&\quad (Y_s \geq 0 \text{ as } D \subseteq \cup_{j=k}^{n} L_j^s \text{ and } Y_t(\omega) \geq 0 \text{ whenever } F(\omega) > t) \\
&\geq \sum_{r=s+1}^{t} \mathbb{E}[Y_t \cdot \mathbb{I}(F = r)] = \sum_{r=s+1}^{t} \mathbb{E}[Y_t \cdot \mathbb{I}(\{F = r\} \cap D)] \\
&\quad (Y_t(\omega) = \mathbf{X}_{F(\omega)}[k](\omega) \text{ whenever } \omega \in D,\ t \geq s \text{ and } F(\omega) \leq t)
\end{aligned}$$

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

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

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

By the Borel-Cantelli lemma, $\mathbb{P}[L_k^r \cap D \cap \{F = \infty\}$ for infinitely many $r] = 0$. But the event $\{L_k^r \cap D \cap \{F = \infty\}$ 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 inf. 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, $B = \{L_k^r \cap D \cap \{F = \infty\}$ for infinitely many $r\}$ and $\mathbb{P}[B] = 0$, a contradiction. □

GLexRSMs would be unsound without the expected leftward nonnegativity.

Example 1. Consider a one-dimensional stochastic process $(Y_t)_{t=0}^{\infty}$ s.t. $Y_0 = 1$ 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 $(\mathcal{F}_t)_{t=0}^{\infty}$ s.t. $\mathcal{F}_t$ 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

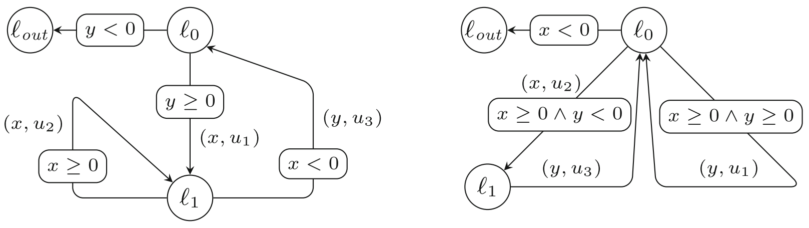

Fig. 2. The pCFGs of the programs presented in Fig. 1. Guards are shown in the rounded boxes, (absence of a box = guard is *true*). The update tuples are shown using variable aliases instead of indexes for better readability. On the left, we have $u_1 = y, u_2 = x - 1 + Norm(0, 1)$, and $u_3 = y - 1$. On the right, we have $u_1 = y + Uni[-7, 1], u_2 = x + Uni[-7, 1]$, and $u_3 = y + Uni[-7, 1]$

update $Y_{t+1} = Y_t - \frac{2}{p_t}$, but 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 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 [13] 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*, i.e. $\mathbb{E}_{X \sim d}[|X|] < \infty$. This is to ensure that 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 all variables are 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 probabilistic programs (PPs) via the standard concept of *probabilistic control flow graphs (pCFGs)* [1,21,23]. Formally, a (pCFG) is a tuple $\mathcal{C} = (L, V, \Delta, Up, G)$ where L is a finite set of *locations*; $V = \{x_1, \ldots, x_{|V|}\}$ is a finite set of *program variables*; Δ is a finite set of *transitions*, i.e. 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 2 presents the pCFGs of our two motivating examples in Fig. 1.

Transitions in Δ_{PB} correspond to the "probabilistic branching" specified by the **if prob**(p) **then**... **else**... construct in imperative-style source code [1]. A program (pCFG) is *linear* (or *affine*) if all its expressions are *linear,* i.e. 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, so they do not depend on program variable values, a common assumption in martingale-based automated approaches to a.s. termination proving [1,18,21,24,49].

A *state* of a pCFG $\mathcal{C}$ is a tuple $(\ell, \mathbf{x})$, where ℓ is a location of $\mathcal{C}$ 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 $\mathbf{x} \models G(\tau)$. 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 (see the extended version of the paper [22] for a formal definition).

A *finite path* of length k in $\mathcal{C}$ 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 $0 \le i < k$ the state $(\ell_{i+1}, \mathbf{x}_{i+1})$ is a successor of $(\ell_i, \mathbf{x}_i)$. A *run* in $\mathcal{C}$ is an infinite sequence of states whose every finite prefix is a finite path. We denote by $\mathit{Fpath}_{\mathcal{C}}$ and $\mathit{Run}_{\mathcal{C}}$ the sets of all finite paths and runs in $\mathcal{C}$, 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 $(\ell, \mathbf{x})$.

The nondeterminism 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 $\mathit{Up}(\tau) = (i, R)$ is enabled, an integrable probability distribution over R. To make the program dynamics under a given scheduler well-defined, we restrict to *measurable* schedulers. This is standard in probabilistic settings [67,68] and hence we omit the formal definition.

We use the standard Markov Decision Process (MDP) semantics of pCFGs [1,21,52]. Each pCFG $\mathcal{C}$ induces a sample space $\Omega_{\mathcal{C}} = \mathit{Run}_{\mathcal{C}}$ and the standard *Borel* sigma-algebra $\mathcal{F}_{\mathcal{C}}$ over $\Omega_{\mathcal{C}}$. Moreover, a pCFG $\mathcal{C}$ 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 $(\Omega_{\mathcal{C}}, \mathcal{F}_{\mathcal{C}}, \mathbb{P}^{\sigma}_{\ell_{init},\mathbf{x}_{init}})$ capturing the rather intuitive dynamics of the programs 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 $\mathit{Up}(\tau) = (i, u)$, 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* [3, Theorem 2.7.2]. We denote by $\mathbb{E}^{\sigma}_{\ell_{init},\mathbf{x}_{init}}$ the expectation operator in the probability space $(\Omega_{\mathcal{C}}, \mathcal{F}_{\mathcal{C}}, \mathbb{P}^{\sigma}_{\ell_{init},\mathbf{x}_{init}})$.

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 $\Omega_{\mathcal{C}}$. We say that a program represented by a pCFG $\mathcal{C}$ terminates *almost-surely (a.s.)* if for each measurable scheduler σ and each initial variable valuation $\mathbf{x}_{init}$ it holds that $\mathbb{P}^{\sigma}_{\ell_{init},\mathbf{x}_{init}}[\mathit{Terminates}] = 1$.

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 1. In what follows, let $\mathcal{C}$ be a pCFG.

Definition 2 (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 [55], was made syntactic in the Dijkstra wp-style in [64], and was extended to programs with continuous distributions in [18]. It formalizes the "one-step" expectation operator $\mathbb{E}^\tau$ we used on an intuitive level in the introduction. In the extended version of the paper [22], we generalize the definition of pre-expectation presented in [18] in order to allow taking expectation over subsets of successor states $\mathcal{C}$ (a necessity for handling the *EXP-NNEG* constraint). We say that a set S of states in $\mathcal{C}$ is *measurable*, if for each location ℓ in $\mathcal{C}$ we have that $\{\mathbf{x} \in \mathbb{R}^{|V|} \mid (\ell, \mathbf{x}) \in S\} \in \mathcal{B}(\mathbb{R}^{|V|})$, i.e. 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 non-deterministic assignments in programs and intuitively are equal to the maximal resp. minimal value of the next-step expectation over all non-deterministic choices. Let η be a 1-dimensional MM, $\tau = (\ell, \delta)$ a transition and S be a measurable set of states in $\mathcal{C}$. 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), and similarly we denote by $\text{min-pre}^{\tau}_{\eta,S}$ the *minimal pre-expectation* of η in τ given S.

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 $\mathcal{C}$ is a function I which to each location ℓ of $\mathcal{C}$ assigns a Borel-measurable set $I(\ell) \subseteq \mathbb{R}^{|V|}$ such that for any state $(\ell, \mathbf{x})$ reachable in $\mathcal{C}$ 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 $\mathsf{lev} : \mathit{\Delta} \to \{0, 1, \ldots, n\}$ a *level map.* For $\tau \in \mathit{\Delta}$ we say that $\mathsf{lev}(\tau)$ is its level. The level of a state is the largest level of any transition enabled at that state. We denote by $S^{\leq j}_{\mathsf{lev}}$ the set of states with level $\leq j$.

Definition 3 (GLexRSM Map). *Let $\boldsymbol{\eta}$ be an n-dimensional MM and I an invariant in $\mathcal{C}$. We say that $\boldsymbol{\eta}$ is a* generalized lexicographic ranking supermartingale map (GLexRSM map) *supported by I, if there is a level map $\mathsf{lev} : \mathit{\Delta} \to \{0, 1, \ldots, n\}$ such that $\mathsf{lev}(\tau) = 0$ 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. *P-RANK*$(\boldsymbol{\eta}, \tau) \equiv \mathbf{x} \in I(\ell) \cap G(\tau) \Rightarrow \big(\textit{max-pre}^{\tau}_{\eta_{\mathsf{lev}(\tau)}}(\ell, \mathbf{x}) \leq \eta_{\mathsf{lev}(\tau)}(\ell, \mathbf{x}) - 1 \wedge \textit{max-pre}^{\tau}_{\eta_j}(\ell, \mathbf{x}) \leq \eta_j(\ell, \mathbf{x})$ *for all* $1 \leq j < \mathsf{lev}(\tau)\big)$*;*
2. *P-NNEG*$(\boldsymbol{\eta}, \tau) \equiv \mathbf{x} \in I(\ell) \cap G(\tau) \Rightarrow \big(\eta_j(\ell, \mathbf{x}) \geq 0$ *for all* $1 \leq j \leq \mathsf{lev}(\tau)\big)$*;*
3. *EXP-NNEG*$(\boldsymbol{\eta}, \tau) \equiv \mathbf{x} \in I(\ell) \cap G(\tau) \Rightarrow \textit{min-pre}^{\tau}_{\eta_j, S^{\leq j-1}_{\mathsf{lev}}}(\ell, \mathbf{x}) \geq 0$ *for all* $1 \leq j \leq \mathsf{lev}(\tau)$*.*

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

Theorem 2 (Soundness of GLexRSM-maps for a.s. Termination). *Let $\mathcal{C}$ be a pCFG and I an invariant in $\mathcal{C}$. Suppose that $\mathcal{C}$ admits an n-dimensional GLexRSM map $\boldsymbol{\eta}$ supported by I, for some $n \in \mathbb{N}$. Then $\mathcal{C}$ terminates a.s.*

The previous theorem, proved in the extended version of the paper [22], instantiates Theorem 1 to probability spaces of pCFGs. The instantiation is *not* straightforward. To ensure that a scheduler cannot "escape" ranking by intricate probabilistic mixing of transitions, we prove that it is sufficient to consider *deterministic* schedulers, which do not randomization among transitions. Also, previous martingale-based certificates of a.s. termination [1,21,36,40] often impose either nonnegativity 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. This generalizes the result of [18] to PPs with nondeterminism.

Remark 1 (Comparison to [49]*).* The work [49] 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 preexpectation calculus [58,59,61].

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, and 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 $\mathcal{C}$ be a LinPP and I a linear invariant in $\mathcal{C}$.

6.1 Linear Programs with Distributions of Bounded Support

Restricting to linear arithmetic is standard in automated a.s. termination proving, allowing to encode the existence of the termination certificate into systems of linear constraints [1,18,21,24]. In the case of LinGLexRSM maps, the difficulty lies in encoding the *EXP-NNEG* condition, as it involves integrating distributions in variable updates which cannot always be done analytically. We show, however, that for LinPPs with bounded-support sampling, we can define another condition which is easier to encode and which 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 Fig. 1b has the BSP, whereas the program in Fig. 1a does not. Using the same notation as in Definition 3, we put:

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

(The 'W' stands for "weak.") Intuitively, *EXP-NNEG* requires nonnegativity of the expected value of η_j when integrated over successor states of level smaller than j, whereas the condition *W-EXP-NNEG* requires nonnegativity of the expected value of η_j when integrated over all successor states. Since η_j is nonnegative 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 existence of LinGLexRSM maps for programs with the BSP, we may w.l.o.g. replace *EXP-NNEG* by *W-EXP-NNEG* for all transitions but for those of probabilistic branching. The proof of the lemma can be found in the extended version of the paper [22].

Lemma 1. *Let $\mathcal{C}$ be a LinPP with the BSP and I be a linear invariant in $\mathcal{C}$. 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.*

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

Theorem 3. *There is a polynomial-time algorithm deciding if a tuple $(\mathcal{C}, I)$ belongs to* 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 3 is a generalization of algorithms in [1,2] finding LinLexRFs in non-probabilistic programs and LinLexRSM maps in PPs, respectively. Suppose that we are given a LinPP $\mathcal{C} = (L, V, \Delta, Up, G)$ with the BSP and a linear invariant I. Our algorithm stores a set $\mathcal{T}$ initialized to all transitions in $\mathcal{C}$. It then proceeds in iterations to compute new components of the witness. In each iteration it searches for a LEM η which is required to

1. be nonnegative on each $\tau = (\ell, \delta) \in \mathcal{T}$, i.e. $\forall \mathbf{x}.\, \mathbf{x} \in I(\ell) \cap G(\tau) \Rightarrow \eta(\ell, \mathbf{x}) \geq 0$;

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

This is done by fixing an LEM template for each location ℓ in $\mathcal{C}$, and converting the above constraints to an equivalent linear program $\mathcal{LP}_{\mathcal{T}}$ in template variables via Farkas' lemma (FL). The FL conversion (and its extension to strict inequalities [21]) is standard in termination proving and encoding conditions 1–3 and 5 above is analogous to [1,2], hence we omit the details. We show how condition 4 can be encoded via linear constraints in the extended version of the paper [22], along with the algorithm pseudocode and the proof of its correctness. 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 1). If in some iteration a new component which 1-ranks at least 1 transition in $\mathcal{T}$ cannot be found, the program does not admit a LinGLexRSM map.

We conclude by showing that our motivating example in Fig. 1b 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 2. Consider the program in Fig. 1b 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 1, $\mathcal{C}$ 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 naive solution for general LinPPs would be to "cut off" the tail events where 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*.

Algorithm 1: Algorithm for proving a.s. termination in LinPP*.

```
input : A LinPP* $\mathcal{C}$, linear invariant $I$.
output: An LEM satisfying the conditions of Lemma 2, if it exists
1  $\mathcal{T} \longleftarrow$ all transitions in $\mathcal{C}$; $d \longleftarrow 0$
2  while $\mathcal{T}$ is non-empty do
3      construct $\mathcal{LP}_{\mathcal{T}}^{\mathrm{unb}}$
4      if $\mathcal{LP}_{\mathcal{T}}^{unb}$ is feasible then
5          $d \longleftarrow d + 1$; $\eta_d \longleftarrow$ LEM defined by the optimal solution of $\mathcal{LP}_{\mathcal{T}}^{\mathrm{unb}}$
6          $\mathcal{T} \longleftarrow \mathcal{T} \setminus \{\tau \in \mathcal{T} \mid \tau$ 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 $\mathcal{LP}_{\mathcal{T}}^{\tau_0,\mathrm{unb}}$
11             if $\mathcal{LP}_{\mathcal{T}}^{\tau_0,unb}$ is feasible then
12                 $d \longleftarrow d + 1$; found $\longleftarrow$ true
13                 $\eta_d \longleftarrow$ LEM defined by the optimal solution of $\mathcal{LP}_{\mathcal{T}}^{\tau_0,\mathrm{unb}}$
14                 $\mathcal{T} \longleftarrow \mathcal{T} \setminus \{\tau \in \mathcal{T} \mid \tau$ is 1-ranked by $\eta_d\}$
15         if not found then return No LEM as in Lemma 2
16 return $(\eta_1, \ldots, \eta_d)$
```

In what follows, we consider a slight restriction on the syntax of LinPPs that we consider, and introduce a new condition on LEMs that allows the over-approximation trick mentioned above 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 Fig. 1b) which is enforced for technical reasons arising in the proof of Lemma 2. 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 3, we define the new condition *UNBOUND* as follows:

UNBOUND$(\boldsymbol{\eta}, \tau) \equiv$ if $Up(\tau) = (i, u)$ with u containing a sampling from a distribution of unbounded support, and ℓ' is the target location of τ, then the coefficient of the variable with index i in $\eta_j(\ell')$ is 0 for all $1 \le j < \mathsf{lev}(\tau)$.

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 [22].

Lemma 2. *Let $\mathcal{C}$ be a LinPP* and I be a linear invariant in $\mathcal{C}$. 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 $\mathcal{C}$ admits a piecewise linear GLexRSM map supported by I.*

Algorithm. The new algorithm shares an overall structure with the algorithm from Sect. 6.1. Thus, we only give a high level overview and focus on novel aspects. The algorithm pseudocode is presented in Algorithm 1.

The condition *UNBOUND* is encoded by modifying the templates for the new LEM components. Let $\mapsto^{\text{unb}}$ be the set of transitions in $\mathcal{C}$ 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 $\mathcal{LP}^{\text{unb}}_{\mathcal{T}}$ analogously to $\mathcal{LP}_{\mathcal{T}}$ in Sect. 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 1 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 3–6. 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 $\mathcal{LP}^{\tau_0,\text{unb}}_{\mathcal{T}}$ analogously 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 1, lines 7–15. 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 2 exists.

Theorem 4. *Algorithm 1 decides in polynomial time if a LinPP* $\mathcal{C}$ admits an LEM which satisfies all conditions of Lemma 2 and which 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 4 can be found in the extended version of the paper [22]. We conclude by showing that Algorithm 1 can prove a.s. termination of our motivating example in Fig. 1a.

Example 3. Consider the program in Fig. 1a with a linear invariant $I(\ell_0) = true$, $I(\ell_1) = y \geq 0$. 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 2 and Algorithm 1 proves a.s. termination.

7 Conclusion

In this work we present new lexicographic termination certificates for probabilistic programs. We also show how to automate the search for the new certificate within a wide class of probabilistic programs. An interesting direction of future work would be automation beyond linear arithmetic programs.

Acknowledgements. This research was partially supported by the ERC CoG 863818 (ForM-SMArt), the Czech Science Foundation grant No. GJ19-15134Y, and the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No. 665385.

References

1. Agrawal, S., Chatterjee, K., Novotný, P.: Lexicographic ranking supermartingales: an efficient approach to termination of probabilistic programs. PACMPL **2**(POPL), 34:1–34:32 (2018)
2. Alias, C., Darte, A., Feautrier, P., Gonnord, L.: Multi-dimensional rankings, program termination, and complexity bounds of flowchart programs. In: Cousot, R., Martel, M. (eds.) SAS 2010. LNCS, vol. 6337, pp. 117–133. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15769-1_8
3. Ash, R., Doléans-Dade, C.: Probability and Measure Theory. Harcourt/Academic Press, Boston (2000)
4. Avanzini, M., Dal Lago, U., Ghyselen, A.: Type-based complexity analysis of probabilistic functional programs. In: 2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), pp. 1–13 (2019). https://doi.org/10.1109/LICS.2019.8785725
5. Avanzini, M., Lago, U.D., Yamada, A.: On probabilistic term rewriting. Sci. Comput. Program. **185**, 102338 (2020). https://doi.org/10.1016/j.scico.2019.102338
6. Avanzini, M., Moser, G., Schaper, M.: A modular cost analysis for probabilistic programs. In: Proceedings of the ACM on Programming Languages, vol. 4 ((Proceedings of OOPSLA 2020)), pp. 1–30 (2020)
7. Baier, C., Katoen, J.P.: Principles of Model Checking. The MIT Press, Cambridge (2008)
8. Barthe, G., Espitau, T., Ferrer Fioriti, L.M., Hsu, J.: Synthesizing probabilistic invariants via Doob's decomposition. In: Chaudhuri, S., Farzan, A. (eds.) CAV 2016. LNCS, vol. 9779, pp. 43–61. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-41528-4_3
9. Barthe, G., Gaboardi, M., Grégoire, B., Hsu, J., Strub, P.Y.: Proving differential privacy via probabilistic couplings. In: Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, pp. 749–758, LICS 2016. ACM, New York, NY, USA (2016). https://doi.org/10.1145/2933575.2934554
10. Barthe, G., Gaboardi, M., Hsu, J., Pierce, B.: Programming language techniques for differential privacy. ACM SIGLOG News **3**(1), 34–53 (2016)
11. Ben-Amram, A.M., Genaim, S.: 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, pp. 51–62, POPL 2013. ACM, New York, NY, USA (2013). https://doi.org/10.1145/2429069.2429078
12. Ben-Amram, A.M., Genaim, S.: Complexity of Bradley-Manna-Sipma lexicographic ranking functions. In: Kroening, D., Păsăreanu, C.S. (eds.) CAV 2015. LNCS, vol. 9207, pp. 304–321. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21668-3_18
13. Billingsley, P.: Probability and Measure, 3rd edn. Wiley, New York (1995)
14. Bournez, O., Garnier, F.: Proving positive almost-sure termination. In: RTA, pp. 323–337 (2005)

15. Bradley, A.R., Manna, Z., Sipma, H.B.: Linear ranking with reachability. In: Computer Aided Verification, 17th International Conference, CAV 2005, Edinburgh, Scotland, UK, 6–10 July 2005, Proceedings, pp. 491–504 (2005). https://doi.org/10.1007/11513988_48
16. Brockschmidt, M., Cook, B., Fuhs, C.: Better termination proving through cooperation. In: Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13–19, 2013, Proceedings, pp. 413–429 (2013). https://doi.org/10.1007/978-3-642-39799-8_28
17. Brockschmidt, M., Cook, B., Ishtiaq, S., Khlaaf, H., Piterman, N.: T2: temporal property verification. In: Chechik, M., Raskin, J.-F. (eds.) TACAS 2016. LNCS, vol. 9636, pp. 387–393. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49674-9_22
18. Chakarov, A., Sankaranarayanan, S.: Probabilistic program analysis with martingales. In: CAV 2013, pp. 511–526 (2013)
19. Chakarov, A., Voronin, Y.-L., Sankaranarayanan, S.: Deductive proofs of almost sure persistence and recurrence properties. In: Chechik, M., Raskin, J.-F. (eds.) TACAS 2016. LNCS, vol. 9636, pp. 260–279. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49674-9_15
20. Chatterjee, K., Fu, H., Goharshady, A.K.: Termination analysis of probabilistic programs through Positivstellensatz's. In: CAV, pp. 3–22 (2016)
21. Chatterjee, K., Fu, H., Novotný, P., Hasheminezhad, R.: Algorithmic analysis of qualitative and quantitative termination problems for affine probabilistic programs. ACM Trans. Program. Lang. Syst. **40**(2), 7:1–7:45 (2018). https://doi.org/10.1145/3174800
22. Chatterjee, K., Goharshady, E.K., Novotný, P., Zárevúcky, J., Žikelić, D.: On lexicographic proof rules for probabilistic termination (2021). https://arxiv.org/abs/2108.02188
23. Chatterjee, K., Novotný, P., Žikelić, D.: Stochastic invariants for probabilistic termination. In: Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, pp. 145–160, POPL 2017. ACM, New York, NY, USA (2017). https://doi.org/10.1145/3009837.3009873
24. Chen, J., He, F.: Proving almost-sure termination by omega-regular decomposition. In: Proceedings of the 41st ACM SIGPLAN International Conference on Programming Language Design and Implementation, PLDI 2020, London, UK, June 15–20, 2020, pp. 869–882 (2020). https://doi.org/10.1145/3385412.3386002
25. Claret, G., Rajamani, S.K., Nori, A.V., Gordon, A.D., Borgström, J.: Bayesian inference using data flow analysis. In: Joint Meeting on Foundations of Software Engineering, pp. 92–102. ACM (2013)
26. Colón, M., Sipma, H.: Synthesis of linear ranking functions. In: Tools and Algorithms for the Construction and Analysis of Systems, 7th International Conference, TACAS 2001 Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2001 Genova, Italy, April 2–6, 2001, Proceedings, pp. 67–81 (2001). https://doi.org/10.1007/3-540-45319-9_6
27. Cook, B., Podelski, A., Rybalchenko, A.: Termination proofs for systems code. SIGPLAN Not. **41**(6), 415–426 (2006)
28. Cook, B., Podelski, A., Rybalchenko, A.: Proving program termination. Commun. ACM **54**(5), 88–98 (2011)
29. Cook, B., See, A., Zuleger, F.: Ramsey vs. lexicographic termination proving. In: Piterman, N., Smolka, S.A. (eds.) TACAS 2013. LNCS, vol. 7795, pp. 47–61. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36742-7_4

30. Cousot, P., Cousot, R.: Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: Conference Record of the Fourth ACM Symposium on Principles of Programming Languages, Los Angeles, California, USA, January 1977, pp. 238–252 (1977). https://doi.org/10.1145/512950.512973
31. Dal Lago, U., Faggian, C., Rocca, S.R.D.: Intersection types and (positive) almost-sure termination. Proc. ACM Program. Lang. **5**(POPL), 1–32 (2021). https://doi.org/10.1145/3434313
32. Dubhashi, D., Panconesi, A.: Concentration of Measure for the Analysis of Randomized Algorithms, 1st edn. Cambridge University Press, New York (2009)
33. Esparza, J., Gaiser, A., Kiefer, S.: Proving termination of probabilistic programs using patterns. In: CAV 2012, pp. 123–138 (2012)
34. Feldman, Y.A.: A decidable propositional dynamic logic with explicit probabilities. Inf. Control **63**(1), 11–38 (1984)
35. Feldman, Y.A., Harel, D.: A probabilistic dynamic logic. In: Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing, pp. 181–195. ACM (1982)
36. Fioriti, L.M.F., Hermanns, H.: Probabilistic termination: soundness, completeness, and compositionality. In: Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, January 15–17, 2015, pp. 489–501 (2015). https://doi.org/10.1145/2676726.2677001
37. Floyd, R.W.: Assigning meanings to programs. Math. Aspects Comput. Sci. **19**, 19–33 (1967)
38. Foster, F.G.: On the stochastic matrices associated with certain queuing processes. Ann. Math. Stat. **24**(3), 355–360 (1953)
39. Foster, N., Kozen, D., Mamouras, K., Reitblatt, M., Silva, A.: Probabilistic NetKAT. In: Thiemann, P. (ed.) ESOP 2016. LNCS, vol. 9632, pp. 282–309. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49498-1_12
40. Fu, H., Chatterjee, K.: Termination of nondeterministic probabilistic programs. In: Enea, C., Piskac, R. (eds.) VMCAI 2019. LNCS, vol. 11388, pp. 468–490. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-11245-5_22
41. Ghahramani, Z.: Probabilistic machine learning and artificial intelligence. Nature **521**(7553), 452–459 (2015)
42. Giesl, J., Giesl, P., Hark, M.: Computing expected runtimes for constant probability programs. In: Fontaine, P. (ed.) Automated Deduction - CADE 27, pp. 269–286. Springer, Cham (2019)
43. Gonnord, L., Monniaux, D., Radanne, G.: Synthesis of ranking functions using extremal counterexamples. In: Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation, pp. 608–618, PLDI 2015. ACM, New York, NY, USA (2015). https://doi.org/10.1145/2737924.2737976
44. Gordon, A.D., Aizatulin, M., Borgstrom, J., Claret, G., Graepel, T., Nori, A.V., Rajamani, S.K., Russo, C.: A model-learner pattern for Bayesian reasoning. ACM SIGPLAN Not. **48**(1), 403–416 (2013)
45. Gordon, A.D., Henzinger, T.A., Nori, A.V., Rajamani, S.K.: Probabilistic programming. In: Proceedings of the on Future of Software Engineering, pp. 167–181. ACM (2014)
46. Gretz, F., Katoen, J.P., McIver, A.: Operational versus weakest pre-expectation semantics for the probabilistic guarded command language. Perform. Eval. **73**, 110–132 (2014)

47. Hark, M., Kaminski, B.L., Giesl, J., Katoen, J.: Aiming low is harder: induction for lower bounds in probabilistic program verification. Proc. ACM Program. Lang. **4**(POPL), 37:1–37:28 (2020). https://doi.org/10.1145/3371105
48. Huang, M., Fu, H., Chatterjee, K.: New approaches for almost-sure termination of probabilistic programs. In: Ryu, S. (ed.) Programming Languages and Systems, pp. 181–201. Springer, Cham (2018)
49. Huang, M., Fu, H., Chatterjee, K., Goharshady, A.K.: Modular verification for almost-sure termination of probabilistic programs. Proc. ACM Program. Lang. **3**(OOPSLA), 129:1–129:29 (2019). https://doi.org/10.1145/3360555
50. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: a survey. JAIR **4**, 237–285 (1996)
51. Kaminski, B.L., Katoen, J.P., Matheja, C.: On the hardness of analyzing probabilistic programs. Acta Informatica **56**(3), 1–31 (2018)
52. Kaminski, B.L., Katoen, J., Matheja, C., Olmedo, F.: Weakest precondition reasoning for expected runtimes of randomized algorithms. J. ACM **65**(5), 30:1–30:68 (2018). https://doi.org/10.1145/3208102
53. Kobayashi, N., Lago, U.D., Grellois, C.: On the termination problem for probabilistic higher-order recursive programs. Log. Methods Comput. Sci. **16**(4), 2:1–2:57 (2020). https://lmcs.episciences.org/6817
54. Kozen, D.: Semantics of probabilistic programs. J. Comput. Syst. Sci. **22**(3), 328–350 (1981). https://doi.org/10.1016/0022-0000(81)90036-2
55. Kozen, D.: A probabilistic PDL. In: Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing, pp. 291–297, STOC 1983. ACM, New York, NY, USA (1983). https://doi.org/10.1145/800061.808758
56. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 585–591. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22110-1_47
57. Lago, U.D., Grellois, C.: Probabilistic termination by monadic affine sized typing. ACM Trans. Program. Lang. Syst. **41**(2), 10:1–10:65 (2019). https://doi.org/10.1145/3293605
58. McIver, A., Morgan, C.: Developing and reasoning about probabilistic programs in pGCL. In: PSSE, pp. 123–155 (2004)
59. McIver, A., Morgan, C.: Abstraction, Refinement and Proof for Probabilistic Systems. Monographs in Computer Science. Springer, New York (2005). https://doi.org/10.1007/b138392
60. McIver, A., Morgan, C.: A new rule for almost-certain termination of probabilistic and demonic programs. CoRR abs/1612.01091 (2016). http://arxiv.org/abs/1612.01091
61. McIver, A., Morgan, C., Kaminski, B.L., Katoen, J.: A new proof rule for almost-sure termination. PACMPL **2**(POPL), 33:1–33:28 (2018). https://doi.org/10.1145/3158121
62. Monniaux, D.: An abstract analysis of the probabilistic termination of programs. In: Cousot, P. (ed.) SAS 2001. LNCS, vol. 2126, pp. 111–126. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-47764-0_7
63. Moosbrugger, M., Bartocci, E., Katoen, J.-P., Kovács, L.: Automated termination analysis of polynomial probabilistic programs. In: ESOP 2021. LNCS, vol. 12648, pp. 491–518. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-72019-3_18
64. Morgan, C., McIver, A.: pGCL: formal reasoning for random algorithms (1999)

65. Morgan, C., McIver, A., Seidel, K.: Probabilistic predicate transformers. ACM Trans. Program. Lang. Syst. (TOPLAS) **18**(3), 325–353 (1996)
66. Motwani, R., Raghavan, P.: Randomized Algorithms. Cambridge University Press, New York (1995)
67. Neuhäußer, M.R., Katoen, J.-P.: Bisimulation and logical preservation for continuous-time Markov decision processes. In: Caires, L., Vasconcelos, V.T. (eds.) CONCUR 2007. LNCS, vol. 4703, pp. 412–427. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74407-8_28
68. Neuhäußer, M.R., Stoelinga, M., Katoen, J.-P.: Delayed nondeterminism in continuous-time Markov Decision Processes. In: de Alfaro, L. (ed.) FoSSaCS 2009. LNCS, vol. 5504, pp. 364–379. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00596-1_26
69. Ngo, V.C., Carbonneaux, Q., Hoffmann, J.: Bounded expectations: resource analysis for probabilistic programs. In: PLDI 2018, pp. 496–512 (2018)
70. Olmedo, F., Kaminski, B.L., Katoen, J.P., Matheja, C.: Reasoning about recursive probabilistic programs. In: Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, pp. 672–681, LICS 2016. ACM, New York, NY, USA (2016). https://doi.org/10.1145/2933575.2935317
71. Podelski, A., Rybalchenko, A.: A complete method for the synthesis of linear ranking functions. In: 5th International Conference on Verification, Model Checking, and Abstract Interpretation, VMCAI 2004, Venice, January 11–13, 2004, Proceedings, pp. 239–251 (2004). https://doi.org/10.1007/978-3-540-24622-0_20
72. Podelski, A., Rybalchenko, A.: Transition invariants. In: Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science, pp. 32–41, LICS 2004. IEEE Computer Society, Washington, DC, USA (2004). https://doi.org/10.1109/LICS.2004.50
73. Roy, D., Mansinghka, V., Goodman, N., Tenenbaum, J.: A stochastic programming perspective on nonparametric Bayes. In: Nonparametric Bayesian Workshop, International Conference on Machine Learning, vol. 22, p. 26 (2008)
74. Ścibior, A., Ghahramani, Z., Gordon, A.D.: Practical probabilistic programming with monads. ACM SIGPLAN Not. **50**(12), 165–176 (2015)
75. Smolka, S., Kumar, P., Foster, N., Kozen, D., Silva, A.: Cantor meets Scott: semantic foundations for probabilistic networks. In: POPL 2017, pp. 557–571 (2017)
76. Sohn, K., Gelder, A.V.: Termination detection in logic programs using argument sizes. In: Proceedings of the Tenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 29–31, 1991, Denver, Colorado, USA, pp. 216–226 (1991). https://doi.org/10.1145/113413.113433
77. Thrun, S.: Probabilistic robotics. Commun. ACM **45**(3), 52–57 (2002)
78. Wang, D., Hoffmann, J., Reps, T.W.: PMAF: an algebraic framework for static analysis of probabilistic programs. In: PLDI 2018, pp. 513–528 (2018)
79. Wang, P., Fu, H., Goharshady, A.K., Chatterjee, K., Qin, X., Shi, W.: Cost analysis of nondeterministic probabilistic programs. In: PLDI 2019, pp. 204–220 (2019)
80. Williams, D.: Probability with Martingales. Cambridge Mathematical Textbooks, Cambridge University Press, Cambridge (1991)