Termination of Nondeterministic Probabilistic Programs

Hongfei Fu[1(✉)] and Krishnendu Chatterjee[2]

[1] Shanghai Jiao Tong University, Shanghai, China
fuhf@cs.sjtu.edu.cn
[2] IST Austria, Klosterneuburg, Austria
krish.chat@ist.ac.at

Abstract. We study the termination problem for nondeterministic probabilistic programs. We consider the bounded termination problem that asks whether the supremum of the expected termination time over all schedulers is bounded. First, we show that ranking supermartingales (RSMs) are both sound and complete for proving bounded termination over nondeterministic probabilistic programs. For nondeterministic probabilistic programs a previous result claimed that RSMs are not complete for bounded termination, whereas our result corrects the previous flaw and establishes completeness with a rigorous proof. Second, we present the first sound approach to establish lower bounds on expected termination time through RSMs.

1 Introduction

In this work we consider nondeterministic probabilistic programs and the termination analysis problem for them. We present results that show how martingale-based approaches provide sound and complete method for such analysis, and can also derive quantitative bounds related to termination time. We first present probabilistic programs, then the termination problems, next the previous results, and finally our contributions.

Probabilistic Programs. Probabilistic aspects in computation is becoming increasingly important, and analysis of programs with probabilistic aspects has received significant attention in the recent years [5,6,12,15,21,24,28,52,55]. In probabilistic programs the classical imperative programs are extended with *random value generators*. The random value generators produce random values according to some desired probability distribution. Probabilistic programs provide a flexible framework to model a wide variety of applications, such as analysis of stochastic network protocols [4,41], robot planning [35], etc. The formal analysis of probabilistic systems and probabilistic programs is an active research topic across different disciplines, such as probability theory and statistics [22,33,39,49,51], formal methods [4,41], artificial intelligence [35,36], and programming languages [5,6,12,15,21,23,24,28,52,55].

C. Enea and R. Piskac (Eds.): VMCAI 2019, LNCS 11388, pp. 468–490, 2019.
https://doi.org/10.1007/978-3-030-11245-5_22

Termination Problems. The most basic and fundamental notion of liveness for static analysis of programs is the *termination* problem. In the absence of probabilistic behavior, the termination problem asks whether a program always terminates. For nonprobabilistic programs, the proof of termination in finite time coincides with the construction of *ranking functions* [25]. Many different approaches exist for construction of ranking functions for termination analysis of nonprobabilistic programs [9,18,50,54]. In the presence of probabilistic behavior the notion of termination problem needs to be extended. Two natural and basic extensions are as follows: first, the *almost-sure* termination question asks whether the program terminates with probability 1; second, the *bounded* termination question asks whether the expected termination time of the program is bounded. While the bounded termination implies almost-sure termination, the converse is not true in general. In this work we focus on the bounded termination problem.

Previous Results: Nonrecursive Probabilistic Programs. We describe the most relevant previous results for termination of probabilistic nonrecursive programs.

- *Finite probabilistic choices.* First, in [43,44] quantitative invariants were used to establish termination for probabilistic programs with nondeterminism, but restricted only to finite probabilistic choices.
- *Infinite probabilistic choices without nondeterminism.* The approach of [43,44] was extended in [12] to *ranking supermartingales* to obtain a sound (but not complete) approach for almost-sure termination for infinite-state probabilistic programs with infinite-domain random variables. The above approach was for probabilistic programs without nondeterminism. The connection between termination of probabilistic programs without nondeterminism and *Lyapunov ranking functions* was considered in [8]. For probabilistic programs with countable state space and without nondeterminism, the Lyapunov ranking functions provide a sound and complete method to prove bounded termination [8,26].
- *Infinite probabilistic choices with nondeterminism.* In the presence of nondeterminism for bounded termination, the Lyapunov-ranking-function/ranking-supermartingale method were claimed to be incomplete, and a partial completeness results has been established for subclass of ranking supermartingales [24]. A proof-rule based approach has also been proposed for probabilistic programs with loops [38]. Automated approaches for synthesizing linear and polynomial ranking supermartingales have been established in [12,14,15]. A martingale based approach for high probability termination and nontermination has also been considered [16].

The problem of deciding termination of probabilistic programs is undecidable, and its precise undecidability characterization has been studied in [37].

Important Open Questions. Given the many important results established in the literature, there are still several fundamental open questions.

- A sound and complete martingale-based approach for bounded termination for nondeterministic probabilistic programs is an important open question.

– While ranking supermartingales can provide upper bounds for expected termination time, whether they can be used to derive lower bounds remains open.

We address these fundamental questions in this work.

Our Results. We consider probabilistic programs with nondeterminism. Our main contributions are as follows.

– *Bounded termination: soundness and completeness.* We show that a ranking-supermartingale based approach is both sound and complete for the bounded termination problem for probabilistic programs with nondeterminism. Note that [24, Theorem 5.7] claimed that ranking supermartingales are incomplete for probabilistic programs with nondeterminism. A counterexample was used as the witness for the incompleteness claim in [24]. We present an explicit ranking supermartingale for the counterexample (see Example 2) to show that the counterexample is invalid, and establish completeness for nondeterministic probabilistic programs for bounded termination. The significance of our result is as follows: it presents a sound and complete approach for nondeterministic probabilistic programs, thus clarifies the understanding of ranking-supermartingale approach in the presence of nondeterminism. Moreover, we also show that our results extend even in the presence of recursion (i.e., for recursive probabilistic programs with nondeterminism).
– *Quantitative bounds.* We present the first sound approach to obtain lower bounds on expected termination time of nondeterministic probabilistic programs using ranking supermartingales. In detail, we show that lowerly-bounded ranking supermartingales present the above sound approach, and demonstrate the necessity of the lowerly-bounded condition.

We note that previous works, such as [12,14,15], present algorithmic approaches to construct special classes (e.g. linear, polynomial) of ranking supermartingales. Thus ranking supermartingales often lead to automated approaches, and we establish that in general such approaches are both sound and complete.

2 Preliminaries

We first introduce some basic concepts in probability theory, and then present the syntax and semantics of nondeterministic probabilistic programs.

2.1 Basic Notations and Concepts

We denote by $\mathbb{N}$, $\mathbb{N}_0$, $\mathbb{Z}$, and $\mathbb{R}$ the sets of all positive integers, nonnegative integers, integers, and real numbers, respectively.

Probability Space. A *probability space* is a triple $(\Omega, \mathcal{F}, \mathbb{P})$, where Ω is a nonempty set (the so-called *sample space*), $\mathcal{F}$ is a *sigma-algebra* over Ω (i.e., a collection of subsets of Ω that contains the empty set $\emptyset$ and is closed under

complementation and countable union), and $\mathbb{P}$ is a *probability measure* on $\mathcal{F}$, i.e., a function $\mathbb{P}\colon \mathcal{F} \to [0,1]$ such that (i) $\mathbb{P}(\Omega) = 1$ and (ii) for all set-sequences $A_1, A_2, \cdots \in \mathcal{F}$ that are pairwise-disjoint (i.e., $A_i \cap A_j = \emptyset$ whenever $i \neq j$) it holds that $\sum_{i=1}^{\infty} \mathbb{P}(A_i) = \mathbb{P}\left(\bigcup_{i=1}^{\infty} A_i\right)$. Elements $A \in \mathcal{F}$ are usually called *events*. An event $A \in \mathcal{F}$ is said to hold *almost surely* (a.s.) if $\mathbb{P}(A) = 1$.

Random Variables. A *random variable* X from a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ is an $\mathcal{F}$-measurable function $X\colon \Omega \to \mathbb{R} \cup \{-\infty, +\infty\}$, i.e., a function satisfying the condition that for all $d \in \mathbb{R} \cup \{-\infty, +\infty\}$, the set $\{\omega \in \Omega \mid X(\omega) < d\}$ belongs to $\mathcal{F}$. By convention, we abbreviate $+\infty$ as ∞.

Expectation. The *expected value* of a random variable X from a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, denoted by $\mathbb{E}(X)$, is defined as the Lebesgue integral of X w.r.t $\mathbb{P}$, i.e., $\mathbb{E}(X) := \int X \,\mathrm{d}\mathbb{P}$; the detailed definition of Lebesgue integral is somewhat technical and is omitted here (cf. [57, Chap. 5] for a formal definition). In the case that *range* $X = \{d_0, d_1, \ldots, d_k, \ldots\}$ is countable with distinct d_k's, we have that $\mathbb{E}(X) = \sum_{k=0}^{\infty} d_k \cdot \mathbb{P}(X = d_k)$.

Characteristic Random Variables. Given random variables $X_0, \ldots, X_n$ from a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ and a predicate Φ over $\mathbb{R} \cup \{-\infty, +\infty\}$, we denote by $\mathbf{1}_{\Phi(X_0,\ldots,X_n)}$ the random variable such that $\mathbf{1}_{\Phi(X_0,\ldots,X_n)}(\omega) = 1$ if $\Phi\left(X_0(\omega), \ldots, X_n(\omega)\right)$ holds, and $\mathbf{1}_{\Phi(X_0,\ldots,X_n)}(\omega) = 0$ otherwise. By definition, $\mathbb{E}\left(\mathbf{1}_{\Phi(X_0,\ldots,X_n)}\right) = \mathbb{P}\left(\Phi(X_0, \ldots, X_n)\right)$. Note that if Φ does not involve any variable, then $\mathbf{1}_{\Phi}$ can be deemed as a constant whose value depends only on whether Φ holds or not.

Filtrations and Stopping Times. A *filtration* of a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ is an infinite sequence $\{\mathcal{F}_n\}_{n \in \mathbb{N}_0}$ of sigma-algebras over Ω such that $\mathcal{F}_n \subseteq \mathcal{F}_{n+1} \subseteq \mathcal{F}$ for all $n \in \mathbb{N}_0$. A *stopping time* (from $(\Omega, \mathcal{F}, \mathbb{P})$) w.r.t $\{\mathcal{F}_n\}_{n \in \mathbb{N}_0}$ is a random variable $R : \Omega \to \mathbb{N}_0 \cup \{\infty\}$ such that for every $n \in \mathbb{N}_0$, the event $R \leq n$ belongs to $\mathcal{F}_n$.

Conditional Expectation. Let X be any random variable from a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ such that $\mathbb{E}(|X|) < \infty$. Then given any σ-algebra $\mathcal{G} \subseteq \mathcal{F}$, there exists a random variable (from $(\Omega, \mathcal{F}, \mathbb{P})$), conventionally denoted by $\mathbb{E}(X|\mathcal{G})$, such that

(E1) $\mathbb{E}(X|\mathcal{G})$ is $\mathcal{G}$-measurable, and
(E2) $\mathbb{E}\left(|\mathbb{E}(X|\mathcal{G})|\right) < \infty$, and
(E3) for all $A \in \mathcal{G}$, we have $\int_A \mathbb{E}(X|\mathcal{G}) \,\mathrm{d}\mathbb{P} = \int_A X \,\mathrm{d}\mathbb{P}$.

The random variable $\mathbb{E}(X|\mathcal{G})$ is called the *conditional expectation* of X given $\mathcal{G}$. The random variable $\mathbb{E}(X|\mathcal{G})$ is a.s. unique in the sense that if Y is another random variable satisfying (E1)–(E3), then $\mathbb{P}(Y = \mathbb{E}(X|\mathcal{G})) = 1$. We refer to [57, Chap. 9] for more details.

Discrete-Time Stochastic Processes. A *discrete-time stochastic process* is a sequence $\Gamma = \{X_n\}_{n \in \mathbb{N}_0}$ of random variables where the X_n's are all from some probability space (say, $(\Omega, \mathcal{F}, \mathbb{P})$); we say that Γ is *adapted to* a filtration

$\{\mathcal{F}_n\}_{n\in\mathbb{N}_0}$ of sub-sigma-algebras of $\mathcal{F}$ if for all $n \in \mathbb{N}_0$, the random variable X_n is $\mathcal{F}_n$-measurable.

Difference-Boundedness. A discrete-time stochastic process $\Gamma = \{X_n\}_{n\in\mathbb{N}_0}$ is *difference-bounded* if there is $c \in [0, \infty)$ such that for every $n \in \mathbb{N}_0$, $|X_{n+1} - X_n| \leq c$ almost-surely.

Stopping Time Z_Γ. Given a discrete-time stochastic process $\Gamma = \{X_n\}_{n\in\mathbb{N}_0}$ adapted to a filtration $\{\mathcal{F}_n\}_{n\in\mathbb{N}_0}$, we define the random variable Z_Γ by $Z_\Gamma(\omega) := \min\{n \mid X_n(\omega) \leq 0\}$ where $\min \emptyset := \infty$. Note that by definition, Z_Γ is a stopping time w.r.t $\{\mathcal{F}_n\}_{n\in\mathbb{N}_0}$.

Martingales. A discrete-time stochastic process $\Gamma = \{X_n\}_{n\in\mathbb{N}}$ adapted to a filtration $\{\mathcal{F}_n\}_{n\in\mathbb{N}_0}$ is a *martingale* (resp. *supermartingale*, *submartingale*) if for every $n \in \mathbb{N}_0$, $\mathbb{E}(|X_n|) < \infty$ and it holds almost-surely that $\mathbb{E}(X_{n+1}|\mathcal{F}_n) = X_n$ (resp. $\mathbb{E}(X_{n+1}|\mathcal{F}_n) \leq X_n$, $\mathbb{E}(X_{n+1}|\mathcal{F}_n) \geq X_n$). We refer to [57, Chap. 10] for more details.

In this paper, we construct super/submartingales Γ from probabilistic programs and use them to prove termination and lower bound properties of the programs. In our setting, Z_Γ will correspond to termination time of a probabilistic program.

Discrete Probability Distributions over Countable Support. A *discrete probability distribution* over a countable set U is a function $q : U \to [0, 1]$ such that $\sum_{z\in U} q(z) = 1$. The *support* of q is defined as $\mathrm{supp}(q) := \{z \in U \mid q(z) > 0\}$. We use discrete probability distributions for samplings of values.

2.2 The Syntax for Nondeterministic Probabilistic Programs

Due to page limit, we present a brief description of our syntax. Our programming language involves two types of variables: *program variables* and *sampling variables*. Program variables are normal variables, while each sampling variable is bound to a discrete probability distribution. Statements in our language are similar to C programming language: assignment statements are indicated by ':='; '**skip**' is the special statement that does nothing; if-branches (resp. while-loops) are indicated by '**if**' (resp. '**while**') together with a logical formula (as the condition) and possibly '**then**' and '**else**' branches; demonic nondeterministic branches are indicated by '**if**', the special symbol '$\star$' and the two nondeterministic '**then**' and '**else**' branches. The detailed syntax that also covers recursion can be found in [13, p. 13].

Remark 1. The syntax of our programming language is quite general and covers major features of imperative probabilistic programming. For example, our syntax is the same as [37] considered for studying theoretical complexity on termination of probabilistic programs, as well as similar to the popular probabilistic programming language from [30] (the only difference is that the language of [30] has extra observe statements). Finally, we use standard control-flow graphs (CFGs) as the basis of our semantics and then present results directly on CFGs (see Sect. 2.3).

Thus our results are not specific to any syntax, but applicable to all probabilistic imperative programs with CFG-based semantics. □

2.3 The Semantics for Nondeterministic Probabilistic Programs

We use control-flow graphs (CFGs) and discrete-time Markov decision processes (MDPs) to specify the operational semantics of nondeterministic probabilistic programs. To avoid measurability issues arising from real-valued variables, for simplicity we only consider integer-valued variables in our semantics.

We first introduce the notions of *valuations* and *propositional arithmetic predicates* as follows.

Valuations and Propositional Arithmetic Predicates. Let V be a finite set of variables. A *valuation* over V is a function ν from V into $\mathbb{Z}$. The set of valuations over V is denoted by Val_V. A *propositional arithmetic predicate* (over V) is a logical formula ϕ built from (i) atomic formulae of the form $\mathfrak{e} \bowtie \mathfrak{e}'$ where $\mathfrak{e}, \mathfrak{e}'$ are arithmetic expressions over V and $\bowtie \in \{<, \leq, >, \geq\}$, and (ii) logical connectives such as $\vee, \wedge, \neg$. The satisfaction relation $\models$ between a valuation ν and a propositional arithmetic predicate ϕ is defined through evaluation and standard semantics of logical connectives such that $\nu \models \phi$ holds iff ϕ holds when all variables in V are replaced by theirs corresponding values in ν.

Then we describe the notion of control-flow graphs (CFGs).

Definition 1 (Control-Flow Graphs (CFGs)). *A* control-flow graph *(CFG) is a tuple which takes the form*

$$(L, \ell_{\text{in}}, \ell_{\text{out}}, V_{\text{p}}, V_{\text{r}}, \rightarrow) \tag{1}$$

where:

- V_{p} *(resp.* V_{r}*) is a finite set of* program variables *(resp.* sampling variables*);*
- L *is a finite set of* labels *partitioned into the set* L_{b} *of* branching *labels, the set* L_{a} *of* assignment *labels and the set* L_{d} *of* nondeterministic *labels;*
- ℓ_{in} *(resp.* ℓ_{out}*) is the* initial label *(resp.* terminal label*);*
- $\rightarrow$ *is a relation whose every member is a triple of the form* (ℓ, α, ℓ') *for which* $\ell \in L$ *(resp.* $\ell' \in L$*) is the source label (resp. target label) of the triple and* α *is either a propositional arithmetic predicate* ϕ *over* V_{p} *if* $\ell \in L_{\text{b}}$*, or an* update function $u : Val_{V_{\text{p}}} \times Val_{V_{\text{r}}} \rightarrow Val_{V_{\text{p}}}$ *if* $\ell \in L_{\text{a}}$*, or the nondeterminism symbol* $\star$ *if* $\ell \in L_{\text{d}}$*.*

Informally, a CFG specifies how labels (program counters) and values for program variables change along the execution of a program. In addition, it is intuitively clear that every nondeterministic probabilistic program can be equivalently transformed into a CFG. Transformation from programs to CFGs can be found in e.g. [14,15].

Below we illustrate an example for probabilistic programs and CFGs.

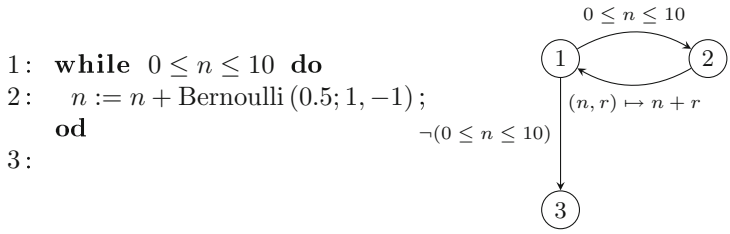

Fig. 1. A probabilistic program (left) with its control-flow graph (right)

Example 1. Consider the program depicted in the left part of Fig. 1, where n is a program variable and Bernoulli $(0.5; 1, -1)$ is a sampling variable. Its CFG is given in the right part of the figure. We use "$0 \leq n \leq 10$" for a shorthand of "$n \geq 0 \wedge n \leq 10$". The semantics of Bernoulli $(0.5; 1, -1)$ is a sampling from the two-point probability distribution q such that $q(1) = q(-1) = \frac{1}{2}$. Basically, the program executes around the value held by n. First, the program starts at the program counter 1. Second, if the value of n falls in $[0, 10]$ then the program enters the while loop (the arc from 1 to 2 in the CFG), otherwise the program terminates. Third, in the while loop, the value of n is incremented by a random value that observes the probability distribution of the sampling variable r, then the program goes back to the start (the program counter 1). □

Based on CFGs, we illustrate the semantics of nondeterministic probabilistic programs as follows. Below we fix a nondeterministic probabilistic program W with its CFG taking the form (1). We first define the notion of *configurations*.

Definition 2 (Configurations $\mathcal{C}$). *A* configuration $\mathfrak{c}$ *is a pair* (ℓ, ν) *where* $\ell \in L$ *and* $\nu \in Val_{V_\mathrm{p}}$. *We say that the configuration* $\mathfrak{c}$ *is* terminal *if* $\ell = \ell_{\mathrm{out}}$, *and* nondeterministic *if* $\ell \in L_\mathrm{d}$. *The set of configurations is denoted by* $\mathcal{C}$.

To demonstrate the formal semantics of probabilistic programs, we also need to assign exact probability distributions to sampling variables. To this purpose, we introduce the notion of sampling functions. A *sampling function* Υ is a function assigning to every sampling variable $r \in V_\mathrm{r}$ a discrete probability distribution over $\mathbb{Z}$; the associated joint discrete probability distribution $\overline{\Upsilon}$ over Val_{V_r} is then defined by $\overline{\Upsilon}(\mu) := \prod_{r \in V_\mathrm{r}} \Upsilon(r)(\mu(r))$ for $\mu \in Val_{V_\mathrm{r}}$.

Now given a sampling function Υ, the semantics of a probabilistic program W is described by a Markov decision process (MDP) (cf. [4, Chap. 10]) $\mathcal{M}_W = (S_W, Act, \mathbf{P}_W)$ as follows. Informally, $\mathcal{M}_W$ describes the probabilistic execution of the program W such that (i) the states S_W of $\mathcal{M}_W$ are configurations reflecting both the current program counter and the values for program variables, (ii) the actions Act are either normal (i.e., absence of non-determinism) or the **then**/**else**-branch that refers to the choice at a non-deterministic label, and (iii) the probabilistic transition function $\mathbf{P}_W$ describes the probabilistic

transitions between configurations. Due to lack of space, we put the detailed definition of the MDP $\mathcal{M}_W$ in [13, Definition 5].

Nondeterminism in MDPs are resolved by *schedulers*. To introduce the notion of schedulers, we first describe the notion of histories upon which schedulers make decisions.

Definition 3 (Histories). *A* history *is a finite sequence* $\rho = \mathfrak{c}_0 \dots \mathfrak{c}_n$ $(n \geq 0)$ *of configurations such that for all* $0 \leq k < n$*, we have* $\mathbf{P}_W(\mathfrak{c}_k, a, \mathfrak{c}_{k+1}) > 0$ *for some* $a \in Act$*. We denote the ending configuration* $\mathfrak{c}_n$ *of the history* ρ *by* $\rho{\downarrow}$*.*

Below we present the standard notion of schedulers. Informally, a scheduler resolves nondeterminism at nondeterministic configurations by choosing a discrete probability distribution over actions that specifies the probabilities to take each action. In this paper, the schedulers are considered *demonic* in the sense that they always try to make the expected termination time longer.

Definition 4 (Schedulers). *A* scheduler σ *is a function which maps every history* ρ *to a discrete probability distribution* $\sigma(\rho)$ *over all possible successor configurations of* $\rho{\downarrow}$*.*

The Final Semantics. Based on schedulers, applying a scheduler σ to $\mathcal{M}_W$ yields an infinite-state discrete-time Markov chain $\mathcal{M}_{W,\sigma}$ where the state space is the set of all histories and the probability transition function is determined by the counterpart from $\mathcal{M}_W$ and the scheduler σ. With an *initial configuration* $\mathfrak{c}$, the semantics of the MDP $\mathcal{M}_W$ is then defined as the probability space $(\Lambda, \mathcal{H}, \mathbb{P}^\sigma_{\mathfrak{c}})$ induced by the Markov chain $\mathcal{M}_{W,\sigma}$ where (i) the sample space Λ consists of all infinite sequences $\{\rho_n\}_{n\in\mathbb{N}_0}$ of histories such that (a) $\rho_0 = \mathfrak{c}$ and (b) for all $k \in \mathbb{N}_0$, there is some configuration $\mathfrak{c}'$ such that $\rho_{k+1} = \rho_k \cdot \mathfrak{c}'$, (ii) the sigma-algebra $\mathcal{H}$ is generated by all *cylinder sets* for which a cylinder set consists of all infinite sequences of histories sharing a common prefix, and (iii) the probability measure $\mathbb{P}^\sigma_{\mathfrak{c}}$ is uniquely determined by the initial configuration $\mathfrak{c}$ and the probability transition function from the Markov chain $\mathcal{M}_{W,\sigma}$ (which in turn depends on the MDP $\mathcal{M}_W$ and the scheduler σ). See [4, Chap. 10] for details.

Infinite Runs. We shall call elements from Λ *infinite runs*. By definition, each infinite run $\{\rho_n\}_{n\in\mathbb{N}_0}$ can be equivalently expressed as the unique infinite sequence $\{\mathfrak{c}_n\}_{n\in\mathbb{N}_0}$ of configurations such that $\rho_n = \mathfrak{c}_0 \dots \mathfrak{c}_n$ for all $n \geq 0$. For the sake of technical convenience, we treat each infinite run $\{\rho_n\}_{n\in\mathbb{N}_0}$ as its corresponding infinite sequence $\{\mathfrak{c}_n\}_{n\in\mathbb{N}_0}$ of configurations. Intuitively, such a sequence $\{\mathfrak{c}_n\}_{n\in\mathbb{N}_0}$ describes an execution of the probabilistic program in the sense that the nth configuration in the execution is $\mathfrak{c}_n$.

Expectation for $(\Lambda, \mathcal{H}, \mathbb{P}^\sigma_{\mathfrak{c}})$. We use the notation $\mathbb{E}^\sigma_{\mathfrak{c}}(-)$ to denote expectation for random variables over elements from Λ w.r.t the probability measure $\mathbb{P}^\sigma_{\mathfrak{c}}$ (with the initial configuration $\mathfrak{c}$ and the scheduler σ).

Finally, we discuss other operational semantics for probabilistic programs that will be crucial to our completeness result.

Remark 2. There are two possible operational semantics for probabilistic programs.

– *Standard MDP Semantics.* In the MDP semantics, the probability space is defined over infinite runs. Moreover, each scheduler defines a probability measure. In this setting, there is only one termination time random variable T (cf. Definition 5), but each scheduler σ defines a probability measure $\mathbb{P}^{\sigma}_{\mathfrak{c}}$.
– *Alternative Semantics.* In an alternative semantics (cf. [24]) the probability space is defined directly over sampled values. In this setting, there is only one probability measure $\mathbb{P}$ (generated by the samplings) but many termination time random variables T^{σ} (each corresponds to a scheduler σ).

Although these two semantics seems similar, they are different. For example, in the standard MDP semantics, the assertion "the program terminates within a bounded amount of time" can be expressed as $\sup_{\sigma} \mathbb{E}^{\sigma}(T) < \infty$, while in the alternative semantics it is expressed as $\mathbb{E}(\sup_{\sigma} T^{\sigma}) < \infty$. In general, $\sup_{\sigma} \mathbb{E}^{\sigma}(T)$ can be smaller than $\mathbb{E}(\sup_{\sigma} T^{\sigma})$ and it is possible that the former is finite and the latter is infinite. The standard MDP semantics is also more applicable as it preserves the local information of nondeterminism by assigning to each scheduler a probability measure. In this work we follow the standard MDP semantics. □

3 Termination Problems

In this section, we define the notions of finite and bounded termination over nondeterministic probabilistic programs. Below we fix a probabilistic program W with its associated CFG in the form (1) and a sampling function Υ. We recall the probability spaces (i.e., $(\Lambda, \mathcal{H}, \mathbb{P}^{\sigma}_{\mathfrak{c}})$'s) defined in the previous section where σ is a scheduler and $\mathfrak{c}$ is an initial configuration.

We first present two definitions of termination times of a probabilistic program.

Definition 5 (Termination-Time Random Variable and Function). *The* termination-time random variable T *is a random variable on Λ defined by:*

$$T(\{(\ell_n, \nu_n)\}_{n\in\mathbb{N}_0}) := \min\{n \in \mathbb{N}_0 \mid \ell_n = \ell_{\text{out}}\}$$

for any infinite sequence $\{(\ell_n, \nu_n)\}_{n\in\mathbb{N}_0}$ of configurations (as an infinite run), where $\min \emptyset := \infty$ (this case corresponds to program nontermination where no ℓ_n is ℓ_{out}). The termination-time function $\overline{T} : \mathcal{C} \to [0, \infty]$ *is given by $\overline{T}(\mathfrak{c}) := \sup_{\sigma} \mathbb{E}^{\sigma}_{\mathfrak{c}}(T)$ for all configurations $\mathfrak{c}$, where σ ranges over all schedulers.*

Thus, T is the random variable that measures the amount of computational steps until termination, while $\overline{T}$ is the function that takes the supremum of expected termination times over all schedulers. Below we further define the notion of finite and bounded termination.

Definition 6 (Finite and Bounded Termination). *We say that the program W is:* finitely terminating *from an initial configuration $\mathfrak{c}$ if $\mathbb{E}^{\sigma}_{\mathfrak{c}}(T) < \infty$ for all schedulers σ; furthermore, it is* boundedly terminating *from an initial configuration $\mathfrak{c}$ if $\overline{T}(\mathfrak{c}) = \sup_{\sigma} \mathbb{E}^{\sigma}_{\mathfrak{c}}(T) < \infty$.*

Remark 3 (Finite vs Bounded Termination). We note that there is an important conceptual difference between finite and bounded termination. While finite termination requires the expected termination time to be finite for all schedulers, the bounded termination requires the supremum of the expected termination time to be finite. In other words, the bounded termination is the uniform bounded version of finite termination. Bounded termination implies finite termination, but not vice-versa. For example, there can be schedulers $\sigma_1, \sigma_2, \sigma_3, \ldots$, such that for σ_i the expected termination time is i; thus for every scheduler the expected termination time is finite, but the supremum is unbounded. For an explicit example see the example in the first paragraph, right column on Page 2 in [24]. Finite termination is also called *positive a.s. termination* in the literature. However, to clarify the important difference between finite vs bounded version of expected termination time we refer to them as finite and bounded termination, respectively. In this work we focus on bounded termination problem, consider ranking supermartingales (a special class of stochastic processes) and show that they are sound and complete for bounded termination. □

4 Bounded Termination: Soundness and Completeness

In this section, we consider the notion of ranking supermartingales (RSMs) for proving bounded termination of probabilistic programs with nondeterminism. Our contributions for this section, which are the main results of this work, are two-fold: (i) we show that RSMs in general form are sound for proving bounded termination; (ii) we prove that RSMs in general form are complete for proving bounded termination, in contrast to the previous claim from [24]. In the whole section, we fix a nondeterministic probabilistic program W together with its CFG taking the form (1) and a sampling function Υ. We define $Val^{\mathrm{r}} := \mathrm{supp}(\overline{\Upsilon})$.

4.1 The Soundness Result

We first recall the notion of RSMs. Intuitively, an RSM is a nonnegative stochastic process with decreasing conditional expectation until the value of the process becomes zero.

Definition 7 (Ranking Supermartingales [12,15,24]). *A discrete-time stochastic process $\Gamma = \{X_n\}_{n\in\mathbb{N}_0}$ adapted to a filtration $\{\mathcal{F}_n\}_{n\in\mathbb{N}_0}$ is a* ranking supermartingale *(RSM) if there exists $\epsilon \in (0,\infty)$ such that for all $n \in \mathbb{N}_0$, the following conditions hold:*

- *(*integrability*)* $\mathbb{E}(|X_n|) < \infty$*;*
- *(*nonnegativity*) it holds a.s. that* $X_n \geq 0$*;*
- *(*ranking*) it holds a.s. that* $\mathbb{E}(X_{n+1}|\mathcal{F}_n) \leq X_n - \epsilon \cdot \mathbf{1}_{X_n>0}$.

Thus, an integrable stochastic process Γ is an RSM if it is nonnegative and its values decrease in conditional expectation when the step n increases. The following known result relates RSMs Γ with the bounded-terminating behaviour of the stopping times Z_Γ. It serves as an extension of Foster's Theorem [8,26].

Theorem 1 ([24, Lemma 5.5]). *Let $\Gamma = \{X_n\}_{n\in\mathbb{N}_0}$ be a ranking supermartingale adapted to a filtration $\{\mathcal{F}_n\}_{n\in\mathbb{N}_0}$ with ϵ given as in Definition 7. Then $\mathbb{P}(Z_\Gamma < \infty) = 1$ and $\mathbb{E}(Z_\Gamma) \leq \frac{\mathbb{E}(X_0)}{\epsilon}$.*

To apply Theorem 1, one needs to embed RSMs into probabilistic programs. To resolve this issue, the notion of linear/polynomial *ranking-supermartingale maps* (RSM-maps) (see [12, Definition 6] and [15, Definition 8]) plays a key role. Below we generalize linear/polynomial RSM-maps to RSM-maps in general form.

Definition 8 (RSM-maps). *A* ranking-supermartingale map *(RSM-map) is a function $h : \mathcal{C} \to [0,\infty]$ satisfying that there exists $\epsilon \in (0,\infty)$ such that for all configurations (ℓ,ν), the following conditions hold:*

(B1) if $\ell = \ell_{\text{out}}$, then we have $h(\ell,\nu) = 0$;
(B2) if $\ell \in L_{\text{a}} \setminus \{\ell_{\text{out}}\}$ and (ℓ,u,ℓ') is the only triple in $\to$ with source label ℓ and update function u, then we have $\epsilon + \sum_{\mu\in Val^r} \overline{\Upsilon}(\mu)\cdot h(\ell',u(\nu,\mu)) \leq h(\ell,\nu)$;
(B3) if $\ell \in L_{\text{b}} \setminus \{\ell_{\text{out}}\}$ and $(\ell,\phi,\ell_1),(\ell,\neg\phi,\ell_2)$ are the two triples in $\to$ with source label ℓ and propositional arithmetic predicate ϕ, then we have $\mathbf{1}_{\nu\models\phi}\cdot h(\ell_1,\nu) + \mathbf{1}_{\nu\models\neg\phi}\cdot h(\ell_2,\nu) + \epsilon \leq h(\ell,\nu)$;
(B4) if $\ell \in L_{\text{d}} \setminus \{\ell_{\text{out}}\}$ and $(\ell,\star,\ell_1),(\ell,\star,\ell_2)$ are the two triples in $\to$ with source label ℓ, then we have $\max\{h(\ell_1,\nu),h(\ell_2,\nu)\} + \epsilon \leq h(\ell,\nu)$.

($d\cdot\infty := \infty$ for $d \in (0,\infty]$, $0\cdot\infty := 0$ by convention.)

Intuitively, an RSM-map is a function whose expected values decrease by a positive stepwise amount ϵ along the execution of a probabilistic program. For example, the condition (B2) means that the expected value taken by h after the execution of an assignment statement decreases by at least ϵ compared with the current value taken by h; (B3) means that the value taken by h after the conditional branch decreases by at least ϵ; (B4) means that the value taken by h after the nondeterministic branch decreases by at least ϵ no matter which branch is taken. We incorporate the infinity ∞ to cover the situation that the supremum of the expected termination time may not be finite.

Below we demonstrate the soundness of RSMs for bounded termination through RSM-maps. There is also a related soundness result established in [24], see Remark 5 below.

Lemma 1 (Soundness). *For all RSM-maps h with ϵ given as in Definition 8 and for all configurations $\mathfrak{c}$, we have that $\overline{T}(\mathfrak{c}) \leq \frac{h(\mathfrak{c})}{\epsilon}$.*

Proof (Proof Sketch for Lemma 1). We generalize the proof idea in [15]. Informally, the soundness result holds as the existence of an RSM-map leads to the existence of an RSM that witnesses the bounded termination of the program. First, we define the random variables lb_n, val_n^x for $n \in \mathbb{N}_0$, $x \in V_{\text{p}}$ so that given any infinite run ω, $\mathsf{lb}_n(\omega)$ represents the label (i.e. the program counter) at the nth step, while $\mathsf{val}_n^x(\omega)$ represents the value of the program variable x at the nth

step. We write $\mathsf{val}_n(\omega)$ for the valuation which maps every program variable x to $\mathsf{val}_n^x(\omega)$. Then we consider any RSM-map h with ϵ given as in Definition 8 and any initial configuration $\mathfrak{c} = (\ell, \nu)$. The case when $h(\mathfrak{c}) = \infty$ is straightforward. So the non-trivial case is that of $h(\mathfrak{c}) < \infty$. Let σ be any scheduler. Define the stochastic process $\Gamma = \{X_n\}_{n \in \mathbb{N}_0}$ by:

$$X_n(\omega) := h\left(\mathsf{lb}_n(\omega),\ \mathsf{val}_n(\omega)\right) \tag{2}$$

for all n and all infinite runs ω. By Definition 8, we have that for all ω, $X_n(\omega) > 0$ iff $\mathsf{lb}_n(\omega) \neq \ell_{\text{out}}$; it follows immediately that $T = Z_\Gamma$. Thus, once we show that Γ is an RSM (under $\mathbb{P}_{\mathfrak{c}}^{\sigma}$), we can apply Lemma 1 and obtain the result. Intuitively, we have that Γ is an RSM since conditions (B2)–(B4) somehow specify the ranking condition of an RSM (see Definition 7). Then by applying Lemma 1, we obtain that $\mathbb{E}_{\mathfrak{c}}^{\sigma}(T) = \mathbb{E}_{\mathfrak{c}}^{\sigma}(Z_\Gamma) \leq \frac{\mathbb{E}_{\mathfrak{c}}^{\sigma}(X_0)}{\epsilon} = \frac{h(\mathfrak{c})}{\epsilon}$. Thus, $\overline{T}(\mathfrak{c}) \leq \frac{h(\mathfrak{c})}{\epsilon}$ by the arbitrary choice of σ. Note that each X_n is nonnegative so the integrability is equivalent to saying that $\mathbb{E}_{\mathfrak{c}}^{\sigma}(X_n) < \infty$. The integrability proof follows from an inductive argument saying that $\mathbb{E}(X_{n+1}) \leq \mathbb{E}(X_n)$, which in turn is derived from the decreasing amount ϵ. Then, we obtain directly from $\mathbb{E}(X_0) = h(\mathfrak{c}) < \infty$ that every X_n is integrable. The detailed proof that also works for recursion is available in [13, Lemma 1]. □

Remark 4 (RSM-maps and Bounded Termination). From Lemma 1 it follows that to prove that the program W is boundedly terminating from an initial configuration $\mathfrak{c}$, it suffices to construct an RSM-map h (possibly in a specific form) satisfying that $h(\mathfrak{c}) < \infty$. □

Remark 5 (Novelty of Our Soundness Result). In previous works such as [12,15], RSM-maps linear or polynomial in program variables serve as a sound approach for proving bounded termination of probabilistic programs. We present a general result to show that RSM-maps in general form are also sound for bounded termination. The main novelty is to ensure that general RSM-maps induce integrable stochastic processes, which naturally holds for linear and polynomial RSM-maps. We also note that there is a soundness result established in [24], however, it is orthogonal to our results as we follow different semantics (see Remark 2). □

4.2 The Completeness Result

Lemma 2 (Completeness). *$\overline{T}$ is an RSM-map with corresponding $\epsilon = 1$ (cf. Definition 8).*

Informally, our completeness result says that the termination time function itself is an RSM-map. This has the following consequence: if the program W is boundedly terminating from some initial configuration $\mathfrak{c}$ (i.e. $\overline{T}(\mathfrak{c}) < \infty$), then one can always find an RSM-map h (in general form) satisfying $h(\mathfrak{c}) < \infty$ by taking h simply to be $\overline{T}$. Note that the existence of such an RSM-map witnesses the bounded-terminating behaviour of the program W (cf. Remark 4). In this sense, Lemma 2 shows that RSMs are complete for proving bounded termination of probabilistic programs.

Proof. By the definition of RSM-maps, we need to prove that the function $\overline{T}$ satisfies the properties (B2)–(B4) for which $\epsilon = 1$. (Note that (B1) directly holds for $\overline{T}$ from definition.) This follows directly from the MDP semantics. In detail, the fact that (B2)–(B4) hold follows from the one-step properties of MDPs. For example, the condition (B2) holds since it is intuitive that the expected termination time is an averaged sum over all successor configurations; (B4) holds since the supremum expected termination time should be the maximum among the two nondeterministic branches. □

Our completeness result is non-trivial as it corrects a previous incompleteness claim from [24, Theorem 5.7]. Below we compare our results with the previous incompleteness claim.

Remark 6 (Comparison with [24]). The result of [24, Theorem 5.7] claims that under the standard MDP semantics, RSMs are not complete for bounded termination over nondeterministic probabilistic programs. We proved that the claim is wrong and established the completeness of RSMs for bounded termination. Note that a relative completeness result is established in [24, Theorem 5.8] under their alternative semantics (cf. Remark 2). Since we consider the MDP semantics, this relative completeness result is orthogonal to the focus of our work. □

In [24], the incompleteness claim was supported by a "counterexample". In the following example, we present however an explicit RSM-map for the "counterexample". This invalidates the incompleteness claim.

Example 2. Consider the probabilistic program depicted in Fig. 2 which results from adapting the example in [24, Figure 1] by using the parameters in the second paragraph on Page 2, right column of [24]. In the figure, n, i, c are program variables and Bernoulli (0.5) is a sampling variable that samples to either 0 or 1 both with probability $\frac{1}{2}$. In [24, Theorem 5.7], this program is used as the counterexample to witness the incompleteness of RSMs for proving bounded termination of nondeterministic probabilistic programs under standard MDP semantics. In contrast, we present an exponential RSM (as an RSM-map h) with corresponding $\epsilon = 1$ for this program in Fig. 3. In the table, the column "Invariant" specifies logical formulae at labels that reachable valuations satisfy when the program starts from label 1, while "The RSM-map h" presents an RSM-map h label by label, e.g., the RSM-map at the label 5 is specified by $h(5, n, i, c) = 2^{n+1} + 2 \cdot n + 13$. In the invariant column, we abbreviate "$i \geq 0 \wedge n \geq 0$" as "$i, n \geq 0$". With the help of the "invariant" column, one can verify from definition that h is an RSM-map. For example, at label 3 we have from the invariant $i = 0 \wedge n = 0$ that $16 = h(4, 0, 0, 0) + 1 \leq h(3, 0, 0, c) = 17$ which fulfills the (B2) condition; at label 6, we have $1 + 0.5 \cdot (2^{n+2} + 2 \cdot n + 18) + 0.5 \cdot (2 \cdot n + 4) \leq 2^{n+1} + 2 \cdot n + 12$ which fulfills the (B2) condition that $1 + 0.5 \cdot h(7, n, i, 0) + 0.5 \cdot h(7, n, i, 1) \leq h(6, n, i, c)$; from label 8 to label 4, we have $1 + h(4, n + 1, i, 0) \leq h(8, n, i, 0)$ for (B2); at label 4 the condition (B3) holds directly from the function at label 5, 12 in the table. Note that although we replace uniform distribution (in the original program) by Bernoulli

distribution to fit our integer setting, the RSM-map given in Fig. 3 remains to be effective for the original program as it preserves probability value for the guard $c < 0.5$. □

```
1:  n := 0;  2:  i := 0;
3:  c := 0;
4:  while c = 0 do
5:    if ⋆ then
6:      c := Bernoulli(0.5);
7:      if c = 0 then
8:        n := n + 1
        else
9:        i := n
        fi
      else
10:     i := 2^n;  11:  c := 1
      fi od;
12:  while i > 0 do
13:     i := i − 1 od
14:
```

Fig. 2. The counterexample in [24]

Label	Invariant	The RSM-map h
1	true	19
2	$n = 0$	18
3	$i = 0 \wedge n = 0$	17
4	$i, n \geq 0$	$\mathbf{1}_{c=1} \cdot (2 \cdot i + 2) +$ $\mathbf{1}_{c=0} \cdot (2^{n+1} + 2 \cdot n + 14)$
5	$i, n \geq 0 \wedge c = 0$	$2^{n+1} + 2 \cdot n + 13$
6	$i, n \geq 0 \wedge c = 0$	$2^{n+1} + 2 \cdot n + 12$
7	$i, n \geq 0$	$\mathbf{1}_{c=0} \cdot (2^{n+2} + 2 \cdot n + 18) +$ $\mathbf{1}_{c=1} \cdot (2 \cdot n + 4)$
8	$i, n \geq 0 \wedge c = 0$	$2^{n+2} + 2 \cdot n + 17$
9	$i, n \geq 0 \wedge c = 1$	$2 \cdot n + 3$
10	$i, n \geq 0 \wedge c = 0$	$2^n + 4$
11	$i, n \geq 0 \wedge c = 0$	$2 \cdot i + 3$
12	$i, n \geq 0$	$2 \cdot i + 1$
13	$i \geq 1 \wedge n \geq 0$	$2 \cdot i$
14	$i = 0 \wedge n \geq 0$	0

Fig. 3. The RSM-map for Example 2

We summarize our soundness and completeness result as follows.

Theorem 2 (Soundness and Completeness). *RSM-maps are sound and complete for bounded termination over non-deterministic probabilistic programs.*

Remark 7 (Decidability). A sound and complete approach does not imply decidability as it only guarantees the existence of RSM-maps in general form for bounded termination. While termination of probabilistic programs is undecidable in general [37], yet RSMs present a sound and complete approach. □

5 Quantitative Results on Bounded Termination

In this section, we present results showing how RSMs can establish quantitative results related to termination of probabilistic programs. Our main contribution of this section shows that *lowerly-bounded* RSMs, a subclass of RSMs, can be used to derive lower bounds on expected termination time.

Lower Bounds on Expected Termination Time. We first present a result (Proposition 1) which shows that difference-bounded RSMs Γ can derive lower bound on the expected value of the stopping time Z_Γ. This proposition serves

as the theoretical backbone for deriving a lower bound for expected termination time. Moreover, we present an example (Example 3) showing that the difference-boundedness condition of Proposition 1 is necessary, and without such condition the desired result does not hold. Then we define the notion of lowerly-bounded RSM-maps, and in Theorem 3 show that they can derive tight lower bounds on expected termination time. Finally we present an example (Example 4) to illustrate the application of Theorem 3.

Proposition 1. *Consider any difference-bounded ranking supermartingale* $\Gamma = \{X_n\}_{n\in\mathbb{N}_0}$ *adapted to a filtration* $\{\mathcal{F}_n\}_{n\in\mathbb{N}_0}$ *with* ϵ *given as in Definition 7. If*

- *for every* $n \in \mathbb{N}_0$*, it holds for all* ω *that* $X_n(\omega) = 0$ *implies* $X_{n+1}(\omega) = 0$*, and*
- The Lower Bound Condition. *there exists* $\delta \in (0, \infty)$ *such that for all* $n \in \mathbb{N}_0$*, it holds a.s. that* $\mathbb{E}(X_{n+1}|\mathcal{F}_n) \geq X_n - \delta \cdot \mathbf{1}_{X_n>0}$*,*

then $\mathbb{E}(Z_\Gamma) \geq \frac{\mathbb{E}(X_0)}{\delta}$.

Note that by definition, we have that $\delta \geq \epsilon$. Below we sketch the proof ideas for Proposition 1.

Proof (Proof Sketch for Proposition 1). The key idea is to construct a difference-bounded submartingale from Γ and apply Optional Stopping Theorem. We first define the stochastic process $\{Y_n\}_{n\in\mathbb{N}_0}$ by: $Y_n = X_n + \delta \cdot \min\{n, Z_\Gamma\}$, and prove that it is a difference-bounded submartingale from the Lower Bound condition. Then, we apply Optional Stopping Theorem to the supermartingale $\{-Y_n\}_{n\in\mathbb{N}_0}$ and the stopping time Z_Γ, and obtain that $-\mathbb{E}\left(X_{Z_\Gamma} + \delta \cdot Z_\Gamma\right) = \mathbb{E}(-Y_{Z_\Gamma}) \leq \mathbb{E}(-Y_0) = \mathbb{E}(-X_0)$. It follows from $X_{Z_\Gamma} = 0$ a.s. that $\mathbb{E}(Z_\Gamma) \geq \frac{\mathbb{E}(X_0)}{\delta}$. □

In Proposition 1 the difference-bounded condition is a prerequisite, and in the following example we show that the prerequisite is a necessary condition.

Example 3 (Necessity of Difference-boundedness). The difference-bounded condition in Proposition 1 cannot be dropped. Consider the family $\{Y_n\}_{n\in\mathbb{N}_0}$ of independent random variables defined by: $Y_0 := 3$ and each Y_n $(n \geq 1)$ satisfies that $\mathbb{P}\left(Y_n = 2^{n-1}\right) = \frac{1}{2}$ and $\mathbb{P}\left(Y_n = -2^{n-1} - 2\right) = \frac{1}{2}$. Let the stochastic process $\Gamma = \{X_n\}_{n\in\mathbb{N}_0}$ be inductively defined by: $X_0 := Y_0$ and for all $n \in \mathbb{N}_0$, we have $X_{n+1} := \mathbf{1}_{X_n>0} \cdot (X_n + Y_{n+1})$. Let $\{\mathcal{F}_n\}_{n\in\mathbb{N}_0}$ be the filtration such that each $\mathcal{F}_n$ is the smallest sigma-algebra that makes all $Y_0, \ldots, Y_n$ measurable, so that Γ is adapted to $\{\mathcal{F}_n\}_{n\in\mathbb{N}_0}$. Then we obtain that for all $n \in \mathbb{N}_0$, we have $\mathbb{E}(X_{n+1}|\mathcal{F}_n) - X_n = -\mathbf{1}_{X_n>0}$. Moreover, for all n and ω, we have $X_n(\omega) = 0 \Rightarrow X_{n+1}(\omega) = 0$. However, $\mathbb{E}(Z_\Gamma) = 2 < \frac{\mathbb{E}(X_0)}{1}$. □

Now we introduce the notion of lowerly-bounded RSM-maps which serve as the main technical notion for proving lower bounds on expected termination time.

Definition 9 (Lowerly-bounded RSM-maps). *An RSM-map* $h : \mathcal{C} \to [0, \infty]$ *is* lowerly-bounded RSM-map *if there exist* $\delta, \zeta \in (0, \infty)$ *such that for all configurations* (ℓ, ν) *satisfying* $h(\ell, \nu) < \infty$*, the following conditions hold:*

(B5) *if* $\ell \in L_{\mathrm{a}} \setminus \{\ell_{\mathrm{out}}\}$ *and* (ℓ, u, ℓ') *is the only triple in* $\to$ *with source label* ℓ *and update function* u*, then we have that* $\delta + \sum_{\mu \in Val^{\mathrm{r}}} \overline{T}(\mu) \cdot h(\ell', u(\nu, \mu)) \geq h(\ell, \nu)$*, and* $|h(\ell', u(\nu, \mu)) - h(\ell, \nu)| \leq \zeta$ *for all* $\mu \in Val^{\mathrm{r}}$*;*

(B6) *if* $\ell \in L_{\mathrm{b}} \setminus \{\ell_{\mathrm{out}}\}$ *and* $(\ell, \phi, \ell_1), (\ell, \neg\phi, \ell_2)$ *are the two triples in* $\to$ *with source label* ℓ *and propositional arithmetic predicate* ϕ*, then we have that* $\mathbf{1}_{\nu \models \phi} \cdot h(\ell_1, \nu) + \mathbf{1}_{\nu \models \neg\phi} \cdot h(\ell_2, \nu) + \delta \geq h(\ell, \nu)$*;*

(B7) *if* $\ell \in L_{\mathrm{d}} \setminus \{\ell_{\mathrm{out}}\}$ *and* $(\ell, \star, \ell_1), (\ell, \star, \ell_2)$ *are the two triples in* $\to$ *with source label* ℓ*, then we have that* $\max\{h(\ell_1, \nu), h(\ell_2, \nu)\} + \delta \geq h(\ell, \nu)$*.*

Informally, an RSM-map h is lowerly-bounded if (i) its change on values between the current step and the next step is bounded by some real number ζ along all possible program executions and (ii) the gap between the expected value of the next step and the current value is no greater than δ. The constant ζ guarantees that the RSM induced by h is difference-bounded, while the constant δ is related to the Lower Bound condition in Proposition 1. For example, the condition (B5) means that the difference between the current value $h(\ell, \nu)$ and the next value $h(\ell', u(\nu, \mu))$ is bounded by ζ for any sampled values μ, while the gap between the expected next-step value $\sum_{\mu \in Val^{\mathrm{r}}} \overline{T}(\mu) \cdot h(\ell', u(\nu, \mu))$ and the current value $h(\ell, \nu)$ should be no greater than δ. In (B6), (B7) we only have the gap condition as the bound on value changes follows implicitly from the conditions (B3), (B4) for RSM-maps.

By Theorem 1, RSM-maps serve as a sound approach for proving bounded termination (cf. Lemma 1). Below we prove through Proposition 1 that lowerly-bounded RSM-maps serve as a sound approach for proving lower bounds on expected termination time. This extends the metering functions proposed in [29].

Theorem 3 (Lower Bounds on Expected Termination Time). *For any lowerly-bounded RSM-map* h *with* δ, ζ *given as in Definition 9,* $\overline{T}(\mathfrak{c}) \geq \frac{h(\mathfrak{c})}{\delta}$ *for all configurations* $\mathfrak{c}$ *such that* $h(\mathfrak{c}) < \infty$*.*

Proof (Proof Sketch). Fix an initial configuration $\mathfrak{c}$ such that $h(\mathfrak{c}) < \infty$. Consider any lowerly-bounded RSM-map h with corresponding δ, ζ. Define the stochastic process $\Gamma = \{X_n\}_{n \in \mathbb{N}_0}$ as in (2). From the proof of Lemma 1, we have that Γ is an RSM with $T = Z_\Gamma$ for any schedulers, thus $\overline{T}(\mathfrak{c}) \leq \frac{h(\mathfrak{c})}{\epsilon} < \infty$. By the bound ζ, we further have that Γ is difference-bounded under any schedulers. Pick a scheduler σ that always choose the nondeterministic branch ℓ' satisfying $h(\ell, \nu) = h(\ell', \nu)$ for any history ending in a nondeterministic configuration (ℓ, ν). Then from the constant δ and the conditions (B5)–(B7), we obtain that $\mathbb{E}(X_{n+1} | \mathcal{F}_n) \geq X_n - \delta \cdot \mathbf{1}_{X_n > 0}$. It follows that Γ satisfies the prerequisites of Proposition 1. Hence by applying Proposition 1, we obtain that $\mathbb{E}^\sigma_{\mathfrak{c}}(T) = \mathbb{E}^\sigma_{\mathfrak{c}}(Z_\Gamma) \geq \frac{\mathbb{E}^\sigma_{\mathfrak{c}}(X_0)}{\delta} = \frac{h(\mathfrak{c})}{\delta}$. Since $\overline{T}(\mathfrak{c}) \geq \mathbb{E}^\sigma_{\mathfrak{c}}(T)$, we have $\overline{T}(\mathfrak{c}) \geq \mathbb{E}^\sigma_{\mathfrak{c}}(T) \geq \frac{h(\mathfrak{c})}{\delta}$. □

Remark 8. We consider the lower bound of $\overline{T}(\mathfrak{c})$ (i.e., the supremum of all expected termination times) instead of the infimum. This is because we have demonic nondeterminism which always tries to make the program nonterminating or the termination time longer. □

Below we illustrate an example on how one can derive lower bounds on expected termination time. This example shows that we can derive *exact* lower bounds from Theorem 3.

Example 4. Consider the program in Example 1. To derive a lower bound on expected termination time, we construct a lowerly-bounded RSM-map h in Table 1. First, one can verify that h is an RSM-map with $\epsilon = 1$ (see Definition 8). This can be observed from the facts that (i) the condition (B3) holds at the label 1 as the loop guard is $0 \leq n \leq 10$, while (ii) the condition (B2) holds at the label 2 as for $0 \leq n \leq 10$, it holds that $1+0.5\cdot h(1,n+1)+0.5\cdot h(1,n-1) = h(2,n)$. (For $n \notin [0,10]$, the value $h(2,n)$ is not relevant and we simply let this value to be the infinity.) Second, from the equality that $1 + 0.5 \cdot h(1, n+1) + 0.5 \cdot h(1, n-1) = h(2,n)$ for $0 \leq n \leq 10$, we can also set $\delta = 1$ for h in Definition 9. Finally, since the interval $[0, 10]$ is bounded, we can choose a sufficiently large ζ that fulfills the conditions in Definition 9. Thus, h is also lowerly-bounded. Hence by Theorem 3 and Lemma 1, we have that given any initial configuration $\mathfrak{c} = (1, n)$, $\frac{h(\mathfrak{c})}{\epsilon} \geq \overline{T}(\mathfrak{c}) \geq \frac{h(\mathfrak{c})}{\delta}$. Since $\delta = \epsilon$, we have $\overline{T}(\mathfrak{c}) = h(\mathfrak{c})$. Thus, our lower bound through Theorem 3 is tight on this example. □

Table 1. A Lowerly-Bounded RSM-map h for Example 1

Label	The RSM-map h
1	$1 + \mathbf{1}_{-1 \leq n \leq 11} \cdot 2 \cdot (n+1) \cdot (11-n)$
2	$1 + \mathbf{1}_{0 \leq n \leq 10} \cdot (2 \cdot (n+1) \cdot (11-n) - 1) + \mathbf{1}_{n \leq -1 \vee n \geq 11} \cdot \infty$
3	0

Remark 9 (Significance of lower bounds). While RSMs provide upper bounds on expected termination time (see Lemma 1), there has been no RSM-based techniques for lower bounds for expected termination time. The significance of lower bounds is that together with upper bounds they provide guarantees on expected termination time. In particular, if the lower and upper bounds are asymptotically same, then they provide tight (i.e., asymptotically optimal) bounds on expected termination time. Example 4 illustrates that there are examples where our approach provides such tight bounds for expected termination time. The lower bounds for expected termination time has been considered in other approaches (see Remark 10 below), and we present the first RSM-based approach for lower bounds for expected termination time for probabilistic programs. □

Remark 10 (Proof-rule Based Approaches). As far as we know, the only known approach to derive lower bounds on expected termination time is based on the notion of proof rules [38,48]. Compared with their approaches, our approach based on RSMs. □

6 Extensions: Continuous Sampling and Recursion

In this section we show that our soundness and completeness results extend in several directions. In particular we show extension to probabilistic programs with (a) continuous sampling and (b) recursion.

Continuous Sampling. The extension of our results to continuous sampling is simple and straightforward, but technical, has following three steps.

1. First, we use general state space Markov chains [46] as our semantics. To this end, we need to define *measurable* schedulers so that a scheduler is measurable if it is a measurable function from histories into actions (see e.g. standard textbooks [7, Chap. 2] for measurability). Then given any initial configuration and a measurable scheduler, we need to define measurable *kernel* functions that specify the probabilities from a current configuration to a region of next configurations. These kernel functions will determine a unique general state space Markov chain. This is standard measurability aspects for Markov chains.
2. Second, for soundness we need to define measurable RSM-maps so that an RSM-map is measurable if it is a measurable function from the measurable space of configurations into the Borel measurable space of real numbers, and change the infinite summation $\sum_{\mu \in Val^r} \overline{T}(\mu) \cdot h\left(\ell', u(\nu, \mu)\right)$ in (B2) by the Lebesgue integral $\int h\left(\ell', u(\nu, \mu)\right) \mathrm{d}\mu$, where the $\mathrm{d}\mu$ here is the probability measure for samplings. By adopting this definition, our soundness proof directly extends to continuous sampling, for which we use Fubini's Theorem [7, Chap. 3] (for multidimensional integrals) to show that (E3) holds, i.e., the conditional expectation of $h\left(\ell', u(\nu, \mu)\right)$ is $\int h\left(\ell', u(\nu, \mu)\right) \mathrm{d}\mu$. Thus the only extension is to switch from sum to integrals, which is technical rather than conceptual.
3. Finally, for completeness we simply need to prove that the measurability of the termination time function $\overline{T}$, while the fact that $\overline{T}$ still satisfies the conditions (B1)–(B4) follows from applying Fubini's Theorem at (B2). This can be observed as follows. By definition, $\overline{T}(\mathfrak{c}) = \sup_\sigma \mathbb{E}^\sigma_{\mathfrak{c}}(T)$. Then there exists a sequence of measurable schedulers $\sigma_1, \sigma_2, \ldots$ such that $\overline{T}(\mathfrak{c}) = \lim_{n \to \infty} \mathbb{E}^{\sigma_n}_{\mathfrak{c}}(T)$. Note that for each measurable scheduler σ, the function $\mathfrak{c} \mapsto \mathbb{E}^\sigma_{\mathfrak{c}}(T)$ is measurable as we have (i) each function $\mathfrak{c} \mapsto \mathbb{E}^\sigma_{\mathfrak{c}}(\min\{m, T\})$ $(m \in \mathbb{N}_0)$ is measurable since the function computes expected values in a bounded horizon (i.e., m), and (ii) $\lim_{m \to \infty} \mathbb{E}^\sigma_{\mathfrak{c}}(\min\{m, T\}) = \mathbb{E}^\sigma_{\mathfrak{c}}(T)$ (from Monotone Convergence Theorem). Thus we have $\overline{T}$ is measurable as measurability is preserved under limit. Again the extension is technical but straightforward application of standard results.

Theorem 4. *RSM-maps are sound and complete for bounded termination over nondeterministic probabilistic programs, even with continuous sampling.*

Recursion. Our results also extend to nondeterministic recursive probabilistic programs. We consider value passing recursive probabilistic programs (i.e., we

have call-by-value setting and no return statements). In the presence of recursion, we extend RSM maps to recursive RSM maps. Intuitively, recursive RSM maps extend RSM maps by adding an additional condition that sum the values taken for the function calls. The soundness and completeness result also extends to the recursive case (detailed demonstration in [13, Theorem 1]).

Theorem 5. *Recursive RSM-maps are sound and complete for bounded termination over nondeterministic probabilistic programs with recursion.*

7 Related Works

Termination Approaches. In [53] the termination of concurrent probabilistic programs with finite state space was considered as a fairness problem, and the precise probabilities did not play a role in termination. A sound and complete method for proving termination of weakly finite state programs was given in [23]. The above approaches do not apply to programs with countable state space in general. For countable state space and almost-sure termination a characterization through fixed-point theory was presented in [31], which is irrelevant to our approach. The analysis of nonprobabilistic program and the termination problem has also been extensively studied [9,10,18,19,29,42,50,54].

Supermartingale Based Approach. The most relevant works related to supermartingale based approach and their predecessor, such as [8,12,14–16,24,43,44], has been discussed in the introduction (Sect. 1). Compared with those results, the most significant difference is that our result considered completeness of ranking supermartingales for proving bounded termination. Besides bounded termination, special classes of (ranking) supermartingales have also been considered as a sound approach for almost-sure termination [1,45], while a potential-function based sound approach similar to ranking supermartingales has also been proposed in [47] to derive upper bounds on expected cost.

Proof-Rule Based Approach. In this work we consider the supermartingale based approach for probabilistic programs. An alternative approach for termination analysis is based on the notion of proof rules [32,34,38,48]. For example, [38] presents a complete proof-rule based approach for probabilistic while loops, but no recursion, and [48] presents sound proof rules for probabilistic programs with recursion, but no completeness result. Both these works do not consider continuous sampling variables. In contrast, our completeness result extends to probabilistic programs with recursion and continuous variables. The proof-rule and martingale-based approaches complement each other, and has their own advantages. A detailed comparison is as follows. The proof-rule based approach itself does not depend on invariants (cf. e.g. [17,20]) and synthesize quantitative invariants, whereas the supermartingale approach usually require invariants (generated with approaches such as [12,14,15]). In contrast the advantage of the supermartingale-based approach are as follows: (a) the supermartingale-based approach leads to algorithmic results, such as polynomial-time algorithms [14,15], and (b) the supermartingale-based approach also yield results

to reason about tail-bound on probabilities of termination [12,15,40]. A deep investigation of combining proof-rule based and supermartingale based approach is an interesting direction for future work.

Comparison with the Recent Work [2]. A recent work [2] studies termination for probabilistic term rewriting systems in the same principle of RSMs. The work considers only discrete probability setting and the bounded expected derivation height which is specialized for term rewriting systems. Instead our results can handle (i) both probability distribution over countable variables, as well as continuous variables, and (ii) the program termination problem. Moreover, our results that handle countable variables as well as recursion have already been announced in January, 2017 [13, Lemmas 1, 2 and Theorem 1] (before the results of [2] were announced in [3]), which subsumes the results of [2].

8 Conclusion and Future Work

In this work we studied termination of nondeterministic probabilistic programs. We show that RSMs are sound and complete for proving bounded termination; in particular, the completeness result corrects a previous incompleteness claim in [24]. Then, we showed that under additional restrictions, ranking supermartingales can serve as a sound approach for deriving lower bounds on expected termination time. An interesting direction is the deep investigation of combining the proof-rule based approach and the RSM based approach.

Acknowledgements. This work is partially funded by the National Natural Science Foundation of China (NSFC) Grant no. 61802254, Austrian Science Fund (FWF) grant S11407-N23 (RiSE/SHiNE) and Vienna Science and Technology Fund (WWTF) project ICT15-003.

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. Avanzini, M., Dal Lago, U., Yamada, A.: On probabilistic term rewriting. In: Gallagher, J.P., Sulzmann, M. (eds.) FLOPS 2018. LNCS, vol. 10818, pp. 132–148. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-90686-7_9
3. Avanzini, M., Lago, U.D., Yamada, A.: On probabilistic term rewriting. CoRR abs/1802.09774 (2018). http://arxiv.org/abs/1802.09774
4. Baier, C., Katoen, J.P.: Principles of Model Checking. MIT Press, Cambridge (2008)
5. Barthe, G., Espitau, T., Grégoire, B., Hsu, J., Strub, P.: Proving expected sensitivity of probabilistic programs. PACMPL **2**(POPL), 57:1–57:29 (2018)
6. Barthe, G., Grégoire, B., Hsu, J., Strub, P.: Coupling proofs are probabilistic product programs. In: Castagna and Gordon [11], pp. 161–174 (2017)
7. Billingsley, P.: Probability and Measure, 3rd edn. Wiley, Hoboken (1995)

8. Bournez, O., Garnier, F.: Proving positive almost-sure termination. In: RTA, pp. 323–337 (2005)
9. Bradley, A.R., Manna, Z., Sipma, H.B.: Linear ranking with reachability. In: CAV, pp. 491–504 (2005)
10. Bradley, A.R., Manna, Z., Sipma, H.B.: The polyranking principle. In: ICALP, pp. 1349–1361 (2005)
11. Castagna, G., Gordon, A.D. (eds.): Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, 18–20 January 2017. ACM (2017)
12. Chakarov, A., Sankaranarayanan, S.: Probabilistic program analysis with martingales. In: CAV, pp. 511–526 (2013)
13. Chatterjee, K., Fu, H.: Termination of nondeterministic recursive probabilistic programs. CoRR abs/1701.02944 (2017). http://arxiv.org/abs/1701.02944
14. Chatterjee, K., Fu, H., Goharshady, A.K.: Termination analysis of probabilistic programs through Positivstellensatz's. In: CAV, pp. 3–22 (2016)
15. Chatterjee, K., Fu, H., Novotný, P., Hasheminezhad, R.: Algorithmic analysis of qualitative and quantitative termination problems for affine probabilistic programs. In: POPL, pp. 327–342 (2016)
16. Chatterjee, K., Novotný, P., Zikelic, D.: Stochastic invariants for probabilistic termination. In: Castagna and Gordon [11], pp. 145–160 (2017)
17. Colón, M.A., Sankaranarayanan, S., Sipma, H.B.: Linear invariant generation using non-linear constraint solving. In: Hunt, W.A., Somenzi, F. (eds.) CAV 2003. LNCS, vol. 2725, pp. 420–432. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45069-6_39
18. Colón, M., Sipma, H.: Synthesis of linear ranking functions. In: TACAS, pp. 67–81 (2001)
19. Cook, B., See, A., Zuleger, F.: Ramsey vs. lexicographic termination proving. In: TACAS, pp. 47–61 (2013)
20. Cousot, P.: Proving program invariance and termination by parametric abstraction, lagrangian relaxation and semidefinite programming. In: Cousot, R. (ed.) VMCAI 2005. LNCS, vol. 3385, pp. 1–24. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30579-8_1
21. Cusumano-Towner, M., Bichsel, B., Gehr, T., Vechev, M.T., Mansinghka, V.K.: Incremental inference for probabilistic programs. In: Foster and Grossman [27], pp. 571–585 (2018)
22. Durrett, R.: Probability: Theory and Examples, 2nd edn. Duxbury Press, Boston (1996)
23. Esparza, J., Gaiser, A., Kiefer, S.: Proving termination of probabilistic programs using patterns. In: CAV, pp. 123–138 (2012)
24. Fioriti, L.M.F., Hermanns, H.: Probabilistic termination: soundness, completeness, and compositionality. In: POPL, pp. 489–501 (2015)
25. Floyd, R.W.: Assigning meanings to programs. Math. Asp. Comput. Sci. **19**, 19–33 (1967)
26. Foster, F.G.: On the stochastic matrices associated with certain queuing processes. Ann. Math. Stat. **24**(3), 355–360 (1953)
27. Foster, J.S., Grossman, D. (eds.): Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2018, Philadelphia, PA, USA, 18–22 June 2018. ACM (2018)
28. Foster, N., Kozen, D., Mamouras, K., Reitblatt, M., Silva, A.: Probabilistic netKAT. In: Thiemann [56], pp. 282–309 (2016)

29. Frohn, F., Naaf, M., Hensel, J., Brockschmidt, M., Giesl, J.: Lower runtime bounds for integer programs. In: Olivetti, N., Tiwari, A. (eds.) IJCAR 2016. LNCS (LNAI), vol. 9706, pp. 550–567. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40229-1_37
30. Gordon, A.D., Henzinger, T.A., Nori, A.V., Rajamani, S.K.: Probabilistic programming. In: Herbsleb, J.D., Dwyer, M.B. (eds.) Proceedings of the on Future of Software Engineering, FOSE 2014, Hyderabad, India, 31 May–7 June 2014, pp. 167–181. ACM (2014)
31. Hart, S., Sharir, M.: Concurrent probabilistic programs, or: How to schedule if you must. SIAM J. Comput. **14**(4), 991–1012 (1985)
32. Hesselink, W.H.: Proof rules for recursive procedures. Formal Asp. Comput. **5**(6), 554–570 (1993)
33. Howard, H.: Dynamic Programming and Markov Processes. MIT Press, Cambridge (1960)
34. Jones, C.: Probabilistic non-determinism. Ph.D. thesis, The University of Edinburgh (1989)
35. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artif. Intell. **101**(1), 99–134 (1998)
36. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: a survey. JAIR **4**, 237–285 (1996)
37. Kaminski, B.L., Katoen, J.-P.: On the hardness of almost–sure termination. In: Italiano, G.F., Pighizzini, G., Sannella, D.T. (eds.) MFCS 2015. LNCS, vol. 9234, pp. 307–318. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48057-1_24
38. Kaminski, B.L., Katoen, J., Matheja, C., Olmedo, F.: Weakest precondition reasoning for expected run-times of probabilistic programs. In: Thiemann [56], pp. 364–389 (2016)
39. Kemeny, J., Snell, J., Knapp, A.: Denumerable Markov Chains. D. Van Nostrand Company, New York City (1966)
40. Kura, S., Urabe, N., Hasuo, I.: Tail probabilities for randomized program runtimes via martingales for higher moments. CoRR abs/1811.06779 (2018). http://arxiv.org/abs/1811.06779
41. 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
42. Lee, C.S., Jones, N.D., Ben-Amram, A.M.: The size-change principle for program termination. In: POPL, pp. 81–92 (2001)
43. McIver, A., Morgan, C.: Developing and reasoning about probabilistic programs in pGCL. In: PSSE, pp. 123–155 (2004)
44. McIver, A., Morgan, C.: Abstraction, Refinement and Proof for Probabilistic Systems. Monographs in Computer Science. Springer, Heidelberg (2005). https://doi.org/10.1007/b138392
45. 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)
46. Meyn, S., Tweedie, R.: Markov Chains and Stochastic Stability. Cambridge University Press, Cambridge (2009)
47. Ngo, V.C., Carbonneaux, Q., Hoffmann, J.: Bounded expectations: resource analysis for probabilistic programs. In: Foster and Grossman [27], pp. 496–512 (2018)
48. Olmedo, F., Kaminski, B.L., Katoen, J.P., Matheja, C.: Reasoning about recursive probabilistic programs. In: LICS, pp. 672–681 (2016)

49. Paz, A.: Introduction to Probabilistic Automata. Computer Science and Applied Mathematics. Academic Press, Cambridge (1971)
50. Podelski, A., Rybalchenko, A.: A complete method for the synthesis of linear ranking functions. In: VMCAI, pp. 239–251 (2004)
51. Rabin, M.: Probabilistic automata. Inf. Control. **6**, 230–245 (1963)
52. Sankaranarayanan, S., Chakarov, A., Gulwani, S.: Static analysis for probabilistic programs: inferring whole program properties from finitely many paths. In: PLDI, pp. 447–458 (2013)
53. Sharir, M., Pnueli, A., Hart, S.: Verification of probabilistic programs. SIAM J. Comput. **13**(2), 292–314 (1984)
54. Sohn, K., Gelder, A.V.: Termination detection in logic programs using argument sizes. In: PODS, pp. 216–226 (1991)
55. Staton, S., Yang, H., Wood, F.D., Heunen, C., Kammar, O.: Semantics for probabilistic programming: higher-order functions, continuous distributions, and soft constraints. In: Grohe, M., Koskinen, E., Shankar, N. (eds.) Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2016, 5–8 July 2016, pp. 525–534. ACM, New York (2016)
56. Thiemann, P. (ed.): ESOP 2016. LNCS, vol. 9632. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49498-1
57. Williams, D.: Probability with Martingales. Cambridge University Press, Cambridge (1991)