Termination Analysis of Probabilistic Programs Through Positivstellensatz's

Krishnendu Chatterjee[1(✉)], Hongfei Fu[1,2], and Amir Kafshdar Goharshady[1]

[1] IST Austria, Vienna, Austria
{krishnendu.Chatterjee,hongfei.fu,goharshady}@ist.ac.at
[2] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, People's Republic of China

Abstract. We consider nondeterministic probabilistic programs with the most basic liveness property of termination. We present efficient methods for termination analysis of nondeterministic probabilistic programs with polynomial guards and assignments. Our approach is through synthesis of polynomial ranking supermartingales, that on one hand significantly generalizes linear ranking supermartingales and on the other hand is a counterpart of polynomial ranking-functions for proving termination of nonprobabilistic programs. The approach synthesizes polynomial ranking-supermartingales through Positivstellensatz's, yielding an efficient method which is not only sound, but also semi-complete over a large subclass of programs. We show experimental results to demonstrate that our approach can handle several classical programs with complex polynomial guards and assignments, and can synthesize efficient quadratic ranking-supermartingales when a linear one does not exist even for simple affine programs.

1 Introduction

Probabilistic Programs. Classic imperative programs extended with *random-value generators* give rise to probabilistic programs. Probabilistic programs provide the appropriate framework to model applications ranging from randomized algorithms [17,38], to stochastic network protocols [5,34], to robot planning [30,33], etc. Nondeterminism plays a crucial role in modeling, such as, to model behaviors over which there is no control, or for abstraction. Thus nondeterministic probabilistic programs are crucial in a huge range of problems, and hence their formal analysis has been studied across disciplines, such as probability theory and statistics [18,28,32,39,42], formal methods [5,34], artificial intelligence [30,31], and programming languages [10,19,21,43].

Basic Termination Questions. Besides safety properties, the most basic property for analysis of programs is the liveness property. The most basic and widely used notion of liveness for programs is *termination*. In absence of probability (i.e., for

A full version is available in [11].

S. Chaudhuri and A. Farzan (Eds.): CAV 2016, Part I, LNCS 9779, pp. 3–22, 2016.
DOI: 10.1007/978-3-319-41528-4_1

nonprobabilistic programs), the synthesis of *ranking functions* and proof of termination are equivalent [22], and numerous approaches exist for synthesis of ranking functions for nonprobabilistic programs [8,13,40,48]. The most basic extension of the termination question for probabilistic programs is the *almost-sure termination* question which asks whether a program terminates with probability 1. Another fundamental question is about *finite termination* (aka positive almost-sure termination [7,21]) which asks whether the expected termination time is finite. The next interesting question is the *concentration* bound computation problem that asks to compute a bound M such that the probability that the termination time is below M is concentrated, or in other words, the probability that the termination time exceeds the bound M decreases exponentially.

Previous Results. We discuss the relevant previous results for termination analysis of probabilistic programs.

- *Probabilistic Programs.* First, quantitative invariants was introduced to establish termination of discrete probabilistic programs with demonic nondeterminism [35,36], This was extended in [10] to *ranking supermartingales* resulting in a sound (but not complete) approach to prove almost-sure termination of probabilistic programs without nondeterminism but with integer- and real-valued random variables from distributions like uniform, Gaussian, and Poison, etc. For probabilistic programs with countable state-space and without nondeterminism, the Lyapunov ranking functions provide a sound and complete method for proving finite termination [7,23]. Another sound method is to explore bounded-termination with exponential decrease of probabilities [37] through abstract interpretation [15]. For probabilistic programs with nondeterminism, a sound and complete characterization for finite termination through ranking-supermartingale is obtained in [21]. Ranking supermartingales thus provide a very powerful approach for termination analysis of probabilistic programs.
- *Ranking Functions/Supermartingales Synthesis.* Synthesis of linear ranking-functions/ranking-supermartingales has been studied extensively in [10,12,13, 40]. In context of probabilistic programs, the algorithmic study of synthesis of linear ranking supermartingales for probabilistic programs (cf. [10]) and probabilistic programs with nondeterminism (cf. our previous result [12]) has been studied. The major technique adopted in these results is Farkas' Lemma [20] which serves as a complete reasoning method for linear inequalities. Beyond linear ranking functions, polynomial ranking functions have also been considered. Heuristic synthesis method of polynomial ranking-functions is studied in [4,9]: Babic *et al.* [4] checked termination of deterministic polynomial programs by detecting divergence on program variables and Bradley *et al.* [9] extended to nondeterministic programs through an analysis on finite differences over transitions. More general methods for deterministic polynomial programs are given by [14,47] where Cousot [14] uses Lagrangian Relaxation, and Shen *et al.* [47] use Putinar's Positivstellensatz [41]. Complete methods of synthesizing polynomial ranking-functions for nondeterministic programs are studied by Yang *et al.* [50], where a complete method through root

classification/real root isolation of semi-algbebraic systems and quantifier elimination is proposed.

To summarize, while many different approaches has been studied, the algorithmic study of synthesis of ranking supermartingales for probabilistic programs has only been limited to linear ranking supermartingales (cf. [10,12]). Hence there is no algorithmic approach to handle nonlinear ranking supermartingales even for probabilistic programs without nondeterminism.

Our Contributions. Our contributions are as follows:

1. *Polynomial Ranking Supermartingales.* First, we extend the notion of linear ranking supermartingales (LRSM) to polynomial ranking supermartingales (pRSM). We show (by a straightforward extension of LRSM) that pRSM implies both almost-sure as well as finite termination.
2. *Positivstellensatz's.* Second, we conduct a detailed investigation on the application of Positivstellensatz's (German for "positive-locus-theorem" which is related to polynomials over semialgebraic sets) (cf. Sect. 5.1) to synthesis of pRSMs over nondeterministic probabilistic programs. To the best of our knowledge, this is the first result which demonstrates the synthesis of a polynomial subclass of ranking supermartingales through Positivstellensatz's.
3. *New Approach for Non-probabilistic Programs.* Our results also extend existing results for nonprobabilistic programs. We present the first result that uses Schmüdgen's Positivstellensatz [45] and Handelman's Theorem [25] to synthesize polynomial ranking-functions for nonprobabilistic programs.
4. *Efficient Approach.* The previous complete method [50] suffers from high computational complexity due to the use of quantifier elimination. In contrast, our approach (sound but not complete) is efficient since the synthesis can be accomplished through linear or semi-definite programming, which can mostly be solved in polynomial time in the problem size [24]. In particular, our approach does not require quantifier elimination, and works for nondeterministic probabilistic programs.
5. *Experimental Results.* We demonstrate the effectiveness of our approach on several classical examples. We show that on classical examples, such as Gambler's Ruin, and Random Walk, our approach can synthesize a pRSM efficiently. For these examples, LRSMs do not exist, and many of them cannot be analysed efficiently by previous approaches.

In summary, while Farkas' Lemma and Motzkin's Transposition Theorem are standard techniques to linear ranking functions or linear ranking supermartingales, they are not sufficient for synthesizing polynomial ranking-supermartingales. To address this problem, we study the use of Positivstellensatz's for the first time to synthesize polynomial ranking-supermartingales for probabilistic programs, for some of them even the first time for nonprobabilistic programs, and show that how they can be used for efficient termination analysis over programs. Due to space restrictions, some technical details are available only in the full version [11].

2 Probabilistic Programs

2.1 Basic Notations and Concepts

For a set A, we denote by $|A|$ the cardinality of A. We denote by $\mathbb{N}$, $\mathbb{N}_0$, $\mathbb{Z}$, and $\mathbb{R}$ the sets of all positive integers, non-negative integers, integers, and real numbers, respectively. We use boldface notation for vectors, e.g. $\boldsymbol{x}$, $\boldsymbol{y}$, etc., and we denote an i-th component of a vector $\boldsymbol{x}$ by $\boldsymbol{x}[i]$.

Polynomial Predicates. Let X be a finite set of variables endowed with a fixed linear order under which we have $X = \{x_1, \ldots, x_{|X|}\}$. We denote the set of real-coefficient polynomials by $\mathfrak{R}\big[x_1, \ldots, x_{|X|}\big]$ or $\mathfrak{R}[X]$. A *polynomial constraint* over X is a logical formula of the form $g_1 \bowtie g_2$, where g_1, g_2 are polynomials over X and $\bowtie \in \{<, \leq, >, \geq\}$. A *propositional polynomial predicate* over X is a propositional formula whose all atomic propositional literals are either *true*, *false* or polynomial constraints over X. The validity of the satisfaction assertion $\boldsymbol{x} \models \phi$ between a vector $\boldsymbol{x} \in \mathbb{R}^{|X|}$ (interpreted in the way that the value for x_j $(1 \leq j \leq |X|)$ is $\boldsymbol{x}[j]$) and a propositional polynomial predicate ϕ is defined in the standard way w.r.t polynomial evaluation and normal semantics for logical connectives. The satisfaction set of a propositional polynomial predicate ϕ is defined as $[\![\phi]\!] := \{\boldsymbol{x} \in \mathbb{R}^{|X|} \mid \boldsymbol{x} \models \phi\}$. For more on polynomials (e.g., polynomial evaluation and arithmetic over polynomials), we refer to the textbook [29, Chapter 3].

Probability Space. A *probability space* is a triple $(\Omega, \mathcal{F}, \mathbb{P})$, where Ω is a non-empty set (so-called *sample space*), $\mathcal{F}$ is a *σ-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 $\sum_{i=1}^{\infty} \mathbb{P}(A_i) = \mathbb{P}\left(\bigcup_{i=1}^{\infty} A_i\right)$.

Random Variables and Filtrations. A *random variable* X in 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 $\{\omega \in \Omega \mid X(\omega) \leq d\}$ belongs to $\mathcal{F}$. The *expected value* of a random variable X, denote by $\mathbb{E}(X)$, is defined as the Lebesgue integral of X with respect to $\mathbb{P}$, i.e., $\mathbb{E}(X) := \int X \, \mathrm{d}\mathbb{P}$; the precise definition of Lebesgue integral is somewhat technical and is omitted here (cf. [6, Chapter 5] for a formal definition). A *filtration* of a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ is an infinite sequence $\{\mathcal{F}_n\}_{n \in \mathbb{N}_0}$ of σ-algebras over Ω such that $\mathcal{F}_n \subseteq \mathcal{F}_{n+1} \subseteq \mathcal{F}$ for all $n \in \mathbb{N}_0$.

2.2 Probabilistic Programs

The Syntax. The class of probabilistic programs we consider encompasses basic programming mechanisms such as assignment statement (indicated by ':='), while-loop, if-branch, basic probabilistic mechanisms such as probabilistic branch (indicated by 'prob') and random sampling, and demonic nondeterminism indicated by '$\star$'. Variables (or identifiers) of a probabilistic program

are of *real* type, i.e., values of the variables are real numbers; moreover, variables are classified into *program* and *sampling* variables, where program variables receive their values through assignment statements and sampling variables do through random samplings. We consider that each sampling variable r is *bounded*, i.e., associated with a one-dimensional cumulative distribution function Υ_r and a non-empty bounded interval supp_r such that any random variable z which respects Υ_r satisfies that z lies in the bounded interval with probability 1. Due to space restriction, details (e.g., grammar) are relegated to the full version [11]. An example probabilistic program is illustrated in Example 1.

Example 1. Consider the running example depicted in Fig. 1, where r is a sampling variable with the two-point distribution $\{1 \mapsto 0.5, -1 \mapsto 0.5\}$ where the probability to take values 1 and -1 are both 0.5. The probabilistic program models a scenario of Gambler's Ruin where the gambler has initial money x and repeats gambling until he wins more than 10 or loses all his money. The result of a gamble is nondeterministic: either win 1 with probability 0.5 (nondeterministic branch); or lose with probability 0.51 (the probabilistic branch). The numbers 1–7 on the left are the program counters for the program, where 1 is the initial program counter and 7 the terminal program counter.

```
1: while 1 ≤ x ∧ x ≤ 10 do
2:     if ⋆ then
3:         x := x + r
       else
4:         if prob(0.51) then
5:             x := x − 1
           else
6:             x := x + 1
           fi
       fi
   od
7:
```

Fig. 1. Running example: Gambler Ruin

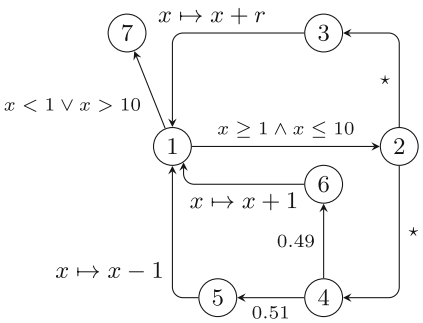

Fig. 2. The CFG of the running example

The Semantics. We use control flow graphs to capture the semantics of probabilistic programs, which we define below.

Definition 1 (Control Flow Graph). *A* control flow graph *(CFG) is a tuple* $\mathcal{G} = (L, \bot, (X, R), \mapsto)$ *with the following components:*

– *L is a finite set of* labels *partitioned into four pairwise-disjoint subsets L_d, $L_\mathrm{p}, L_\mathrm{c}$ and L_a of demonic, probabilistic, conditional-branching (branching for short) and assignment labels, resp.; and $\bot$ is a special label not in L called the* terminal label*;*

– X *and* R *are disjoint finite sets of real-valued* program *and* sampling variables *respectively;*
– $\mapsto$ *is a* transition relation *in which every member (called* transition*) is a tuple of the form* (ℓ, α, ℓ') *for which* ℓ *(resp.* ℓ'*) is the* source label *(resp.* target label*) in* L *and* α *is either a real number in* $(0, 1)$ *if* $\ell \in L_{\mathrm{p}}$*, or* $\star$ *if* $\ell \in L_{\mathrm{d}}$*, or a propositional polynomial predicate if* $\ell \in L_{\mathrm{c}}$*, or an* update function $f \colon \mathbb{R}^{|X|} \times \mathbb{R}^{|R|} \to \mathbb{R}^{|X|}$ *if* $\ell \in L_{\mathrm{a}}$.

W.l.o.g, we assume that $L \subseteq \mathbb{N}_0$. Intuitively, labels in L_{d} correspond to demonic statements indicated by '$\star$'; labels in L_{p} correspond to probabilistic-branching statements indicated by '**prob**'; labels in L_{c} correspond to conditional-branching statements indicated by some propositional polynomial predicate; labels in L_{a} correspond to assignments indicated by ':='; and the terminal label $\bot$ denotes the termination of a program. The transition relation $\mapsto$ specifies the transitions between labels together with the additional information specific to different types of labels. The update functions are interpreted as follows: we first fix two linear orders on X and R so that $X = \{x_1, \ldots, x_{|X|}\}$ and $R = \{r_1, \ldots, r_{|R|}\}$, interpreting each vector $\boldsymbol{x} \in \mathbb{R}^{|X|}$ (resp. $\boldsymbol{r} \in \mathbb{R}^{|R|}$) as a *valuation* of program (resp. sampling) variables in the sense that the value of x_j (resp. r_j) is $\boldsymbol{x}[j]$ (resp. $\boldsymbol{r}[j]$); then each update function f is interpreted as a function which transforms a valuation $\boldsymbol{x} \in \mathbb{R}^{|X|}$ before the execution of an assignment statement into $f(\boldsymbol{x}, \boldsymbol{r})$ after the execution of the assignment statement, where $\boldsymbol{r}$ is the valuation on R obtained from a sampling before the execution of the assignment statement.

It is intuitively clear that any probabilistic program can be naturally transformed into a CFG. Informally, each label represents a program location in an execution of a probabilistic program for which the statement of the program location is the next to be executed (see Fig. 2).

In the rest of the section, we fix a probabilistic program P with the set $X = \{x_1, \ldots, x_{|X|}\}$ of program variables and the set $R = \{r_1, \ldots, r_{|R|}\}$ of sampling variables, and let $\mathcal{G} = (L, \bot, (X, R), \mapsto)$ be its associated CFG. We also fix ℓ_0 and resp. $\boldsymbol{x}_0$ to be the label corresponding to the first statement to be executed in P and resp. the initial valuation of program variables.

The Semantics. A *configuration* (for P) is a tuple $(\ell, \boldsymbol{x})$ where $\ell \in L \cup \{\bot\}$ and $\boldsymbol{x} \in \mathbb{R}^{|X|}$. A *finite path* (of P) is a finite sequence of configurations $(\ell_0, \boldsymbol{x}_0), \cdots, (\ell_k, \boldsymbol{x}_k)$ such that for all $0 \leq i < k$, either (i) $\ell_{i+1} = \ell_i = \bot$ and $\boldsymbol{x}_i = \boldsymbol{x}_{i+1}$ (i.e., the program terminates); or (ii) there exist $(\ell_i, \alpha, \ell_{i+1}) \in \mapsto$ and $\boldsymbol{r} \in \{\boldsymbol{r}' \mid \forall r \in R.\ \boldsymbol{r}'(r) \in \mathrm{supp}_r\}$ such that one of the following conditions hold: (a) $\ell_i \in L_{\mathrm{p}} \cup L_{\mathrm{d}}$ and $\boldsymbol{x}_i = \boldsymbol{x}_{i+1}$ (probabilistic or demonic transitions), (b) $\ell_i \in L_{\mathrm{c}}$, $\boldsymbol{x}_i = \boldsymbol{x}_{i+1}$ and $\boldsymbol{x}_i \models \alpha$ (conditional-branch transitions), (c) $\ell_i \in L_{\mathrm{a}}$ and $\boldsymbol{x}_{i+1} = \alpha(\boldsymbol{x}_i, \boldsymbol{r})$ (assignment transitions). A *run* (of P) is an infinite sequence of configurations whose all finite prefixes are finite paths over P. A configuration $(\ell, \boldsymbol{x})$ is *reachable* from the initial configuration $(\ell_0, \boldsymbol{x}_0)$ if there exists a finite path $(\ell_0, \boldsymbol{x}_0), \cdots, (\ell_k, \boldsymbol{x}_k)$ such that $(\ell, \boldsymbol{x}) = (\ell_k, \boldsymbol{x}_k)$.

The probabilistic feature of P can be captured by constructing a suitable probability measure over the set of all its runs. However, before this can be done, nondeterminism in P needs to be resolved by some *scheduler*.

Definition 2 (Scheduler). *A* scheduler *(for P) is a function which assigns to every finite path $(\ell_0, \boldsymbol{x}_0), \ldots, (\ell_k, \boldsymbol{x}_k)$ with $\ell_k \in L_{\mathrm{d}}$ a transition in $\mapsto$ with source label ℓ_k.*

The behaviour of P under a scheduler σ is standard: at each step, P first samples a real number for each sampling variable and then evolves to the next step according to its CFG or the scheduler choice. In this way, the scheduler and random choices/samplings produce a run over P. Moreover, each scheduler σ induces a unique probability measure $\mathbb{P}^\sigma$ over the runs of P. In the sequel, we will use $\mathbb{E}^\sigma(\cdot)$ to denote the expected values of random variables under $\mathbb{P}^\sigma$.

Random Variables and Filtrations over Runs. We define the following (vectors of) random variables on the set of runs of P: $\{\theta_n^P\}_{n\in\mathbb{N}_0}$, $\{\overline{\boldsymbol{x}}_n^P\}_{n\in\mathbb{N}_0}$ and $\{\overline{\boldsymbol{r}}_n^P\}_{n\in\mathbb{N}_0}$: each θ_n^P is the random variable representing the (integer-valued) label at the n-th step; each $\overline{\boldsymbol{x}}_n^P$ is the vector of random variables such that each $\overline{\boldsymbol{x}}_n^P[i]$ is the random variable representing the value of the program variable x_i at the n-th step; and each $\overline{\boldsymbol{r}}_n^P[i]$ is the random variable representing the sampled value of the sampling variable r_i at the n-th step. The filtration $\{\mathcal{H}_n^P\}_{n\in\mathbb{N}_0}$ is defined such that each σ-algebra $\mathcal{H}_n^P$ is the smallest σ-algebra that makes all random variables in $\{\theta_k^P\}_{0\le k\le n}$ and $\{\overline{\boldsymbol{x}}_k^P\}_{0\le k\le n}$ measurable. We will omit the superscript P in all the notations above if it is clear from the context.

Remark 1. Under the condition that each sampling variable is bounded, using an inductive argument it follows that each $\overline{\boldsymbol{x}}_n$ is a vector of bounded random variables. Thus $\mathbb{E}^\sigma(|\overline{\boldsymbol{x}}_n[i]|)$ exists for each random variable $\overline{\boldsymbol{x}}_n[i]$.

Below we define the notion of *polynomial invariants* which logically captures all reachable configurations. A polynomial invariant may be obtained through abstract interpretation [15].

Definition 3 (Polynomial Invariant). *A* polynomial invariant *(for P) is a function I assigning a propositional polynomial predicate over X to every label in $\mathcal{G}$ such that for all configurations $(\ell, \boldsymbol{x})$ reachable from $(\ell_0, \boldsymbol{x}_0)$ in $\mathcal{G}$, it holds that $\boldsymbol{x} \models I(\ell)$.*

3 Termination over Probabilistic Programs

In this section, we first define the notions of almost-sure/finite termination and concentration bounds over probabilistic programs, and then describe the computational problems studied in this paper. Below we fix a probabilistic program P with its associated CFG $\mathcal{G} = (L, \bot, (X, R), \mapsto)$ and an initial configuration $(\ell_0, \boldsymbol{x}_0)$ for P.

Definition 4 (Termination [7,12,21]). *A run $\omega = \{(\ell_n, \boldsymbol{x}_n)\}_{n\in\mathbb{N}_0}$ over P is* terminating *if $\ell_n = \bot$ for some $n \in \mathbb{N}_0$. The* termination time *of P is a random variable T_P such that for each run $\omega = \{(\ell_n, \boldsymbol{x}_n)\}_{n\in\mathbb{N}_0}$, $T_P(\omega)$ is the least number n such that $\ell_n = \bot$ if such n exists, and ∞ otherwise. The program P is said to be* almost-sure terminating *(resp.* finitely terminating*) if $\mathbb{P}^\sigma(T_P < \infty) = 1$ (resp. $\mathbb{E}^\sigma(T_P) < \infty$) for all schedulers σ (for P).*

Note that $\mathbb{E}^\sigma(T_P) < \infty$ implies that $\mathbb{P}^\sigma(T_P < \infty) = 1$, but the converse does not necessarily hold (see [10, Example 5] for an example). To measure the expected values of the termination time under all (demonic) schedulers, we further define the quantity $\mathsf{ET}(P) := \sup_\sigma \mathbb{E}^\sigma(T_P)$.

Definition 5 (Concentration on Termination Time [12,37]). *A* concentration bound *for P is a non-negative integer M such that there exist real constants $c_1 \geq 0$ and $c_2 > 0$, and for all $N \geq M$ we have $\mathbb{P}(T_P > N) \leq c_1 \cdot e^{-c_2 \cdot N}$.*

Informally, a concentration bound characterizes exponential decrease of probability values of non-termination beyond the bound. On one hand, it can be used to give an upper bound on probability of non-termination beyond a large step; and on the other hand, it leads to an algorithm that approximates $\mathsf{ET}(P)$ (cf. [12, Theorem 5]).

In this paper, we consider the algorithmic analysis of the following problems:

- **Input:** a probabilistic program P, a polynomial invariant I for P and an initial configuration $(\ell_0, \boldsymbol{x}_0)$ for P;
- **Output (Almost-Sure/Finite Termination):** "*yes*" if the algorithm finds that P is almost-sure/finite terminating and "*fail*" otherwise;
- **Output (Concentration on Termination):** a concentration bound if the algorithm finds one and "*fail*" otherwise.

4 Polynomial Ranking-Supermartingale

In this section, we develop the notion of polynomial ranking-supermartingale which is an extension of linear ranking-supermartingale [10,12]. We fix a probabilistic program P, a polynomial invariant I for P and an initial configuration $(\ell_0, \boldsymbol{x}_0)$ for P. Let $\mathcal{G} = (L, \bot, (X, R), \mapsto)$ be the associated CFG of P, with $X = \{x_1, \ldots, x_{|X|}\}$ and $R = \{r_1, \ldots, r_{|R|}\}$. We first present the general notion of *ranking supermartingale*, and then define *polynomial ranking supermartingale*.

Definition 6 (Ranking Supermartingale [12,21]). *A discrete-time stochastic process $\{X_n\}_{n \in \mathbb{N}_0}$ w.r.t a filtration $\{\mathcal{F}_n\}_{n \in \mathbb{N}_0}$ is a* ranking supermartingale *(RSM) if there exist $K < 0$ and $\epsilon > 0$ such that for all $n \in \mathbb{N}_0$, we have $\mathbb{E}(|X_n|) < \infty$ and it holds almost surely (with probability 1) that $X_n \geq K$ and $\mathbb{E}(X_{n+1} \mid \mathcal{F}_n) \leq X_n - \epsilon \cdot \mathbf{1}_{X_n \geq 0}$, where $\mathbb{E}(X_{n+1} \mid \mathcal{F}_n)$ is the conditional expectation of X_{n+1} given $\mathcal{F}_n$ (cf. [49, Chapter 9]).*

Informally, a polynomial ranking-supermartingale over P is a polynomial instantiation of an RSM through certain function $\eta : (L \cup \{\bot\}) \times \mathbb{R}^{|X|} \to \mathbb{R}$ which satisfies that each $\eta(\ell, \cdot)$ (for all $\ell \in L \cup \{\bot\}$) is essentially a polynomial function over X. Given such a function η, the intuition is to have conditions that make the stochastic process $X_n = \eta(\theta_n, \overline{\boldsymbol{x}}_n)$ an RSM. To ensure this, we consider the conditional expectation $\mathbb{E}^\sigma(X_{n+1} \mid \mathcal{H}_n)$; this is captured by an extension of *pre-expectation* [10,12] from the linear to the polynomial case. Below we define $L_\bot := L \cup \{\bot\}$. For a function $g : \mathbb{R}^{|X|} \times \mathbb{R}^{|R|} \to \mathbb{R}$, we let $\mathbb{E}_R(g, \cdot) : \mathbb{R}^{|X|} \to \mathbb{R}$

be the function such that each $\mathbb{E}_R(g, \boldsymbol{x})$ is the expected value $\mathbb{E}(g(\boldsymbol{x}, \hat{\boldsymbol{r}}))$, where $\hat{\boldsymbol{r}}$ is any vector of independent random variables such that each $\hat{\boldsymbol{r}}[i]$ is a random variable that respects the cumulative distribution function Υ_{r_i}.

Definition 7 (Pre-Expectation). *Let $\eta : L_\perp \times \mathbb{R}^{|X|} \to \mathbb{R}$ be a function such that each $\eta(\ell, \cdot)$ (for all $\ell \in L_\perp$) is a polynomial function over X. The function $\mathrm{pre}_\eta : L_\perp \times \mathbb{R}^{|X|} \to \mathbb{R}$ is defined by:*

- $\mathrm{pre}_\eta(\ell, \boldsymbol{x}) := \sum_{(\ell, z, \ell') \in \mapsto} z \cdot \eta(\ell', \boldsymbol{x})$ *if $\ell \in L_\mathrm{p}$ (probabilistic transitions);*
- $\mathrm{pre}_\eta(\ell, \boldsymbol{x}) := \max_{(\ell, \star, \ell') \in \mapsto} \eta(\ell', \boldsymbol{x})$ *if $\ell \in L_\mathrm{d}$ (nondeterministic transitions);*
- $\mathrm{pre}_\eta(\ell, \boldsymbol{x}) := \eta(\ell', \boldsymbol{x})$ *if $\ell \in L_\mathrm{c}$ and (ℓ, ϕ, ℓ') is the only transition in $\mapsto$ such that $\boldsymbol{x} \models \phi$ (conditional transitions);*
- $\mathrm{pre}_\eta(\ell, \boldsymbol{x}) := \mathbb{E}_R(g, \boldsymbol{x})$ *if $\ell \in L_\mathrm{a}$, where g is the function such that $g(\boldsymbol{x}, \boldsymbol{r}) = \eta(\ell', f(\boldsymbol{x}, \boldsymbol{r}))$ and (ℓ, f, ℓ') is the only transition in $\mapsto$ (assignment transitions); and*
- $\mathrm{pre}_\eta(\ell, \boldsymbol{x}) := \eta(\ell, \boldsymbol{x})$ *if $\ell = \perp$ (terminal location).*

The following lemma establishes the relationship between pre-expectation and conditional expectation.

Lemma 1. *Let $\eta : L_\perp \times \mathbb{R}^{|X|} \to \mathbb{R}$ be a function such that each $\eta(\ell, \cdot)$ (for all $\ell \in L_\perp$) is a polynomial function over X, and σ be any scheduler. Let the stochastic process $\{X_n\}_{n \in \mathbb{N}_0}$ be defined by: $X_n := \eta(\theta_n, \overline{\boldsymbol{x}}_n)$. Then for all $n \in \mathbb{N}_0$, we have $\mathbb{E}^\sigma(X_{n+1} \mid \mathcal{H}_n) \leq \mathrm{pre}_\eta(\theta_n, \overline{\boldsymbol{x}}_n)$.*

Example 2. Consider the running example in Example 1 with CFG in Fig. 2. Let η be the function specified in the second and fifth column of Table 1, where $g(x) := (x-1)(10-x)$. Then pre_η is given in the third and sixth column of Table 1. Note that the case for $i = 2$ is obtained from $\mathrm{pre}_\eta(2, x) = \max\{g(x) + 9.6, g(x) + 9.6\}$, and the case for $i = 3$ is from $\mathrm{pre}_\eta(3, x) = \mathbb{E}_R(h, x)$, where h is the function $h(y, r) = g(y) - (2y - 11)r - r^2 + 10$.

We now define the notion of polynomial ranking-supermartingale. The intuition is that we encode the RSM-difference condition as a logical formula, treat zero as the threshold between terminal and non-terminal labels, and use the invariant I to over-approximate the set of reachable configurations at each label. Below for each $\ell \in L_\mathrm{c}$, we define $\mathsf{PP}(\ell)$ to be the propositional polynomial predicate $\bigvee_{(\ell, \phi, \ell') \in \mapsto, \ell' \neq \perp} \phi$; and for $\ell \in L \backslash L_\mathrm{c}$, we let $\mathsf{PP}(\ell) := \mathit{true}$.

Table 1. η and pre_η for Example 1 and Fig. 2

i	$\eta(i, x)$	$\mathrm{pre}_\eta(i, x)$	i	$\eta(i, x)$	$\mathrm{pre}_\eta(i, x)$
1	$g(x) + 10$	$\mathbf{1}_{1 \leq x \leq 10} \cdot (g(x) + 9.8)$ $+ \mathbf{1}_{x < 1 \vee x > 10} \cdot (-0.2)$	5	$g(x) + 2x - 1.8$	$g(x) + 2x - 2$
2	$g(x) + 9.8$	$g(x) + 9.6$	6	$g(x) - 2x + 20.2$	$g(x) - 2x + 20$
3	$g(x) + 9.6$	$g(x) + 9$	7	-0.2	-0.2
4	$g(x) + 9.6$	$g(x) + 0.04x + 8.98$			

Definition 8 (Polynomial Ranking-Supermartingale). *A* d-degree polynomial ranking-supermartingale map *(in short,* d*-pRSM) w.r.t* (P, I) *is a function* $\eta : L_\perp \times \mathbb{R}^{|X|} \to \mathbb{R}$ *satisfying that there exist* $\epsilon > 0$ *and* $K \leq -\epsilon$ *such that for all* $\ell \in L_\perp$ *and all* $\boldsymbol{x} \in \mathbb{R}^{|X|}$*, the conditions (C1-C4) hold:*

- C1: *the function* $\eta(\ell, \cdot) : \mathbb{R}^{|X|} \to \mathbb{R}$ *is a polynomial over* X *of order at most* d*;*
- C2: *if* $\ell \neq \perp$ *and* $\boldsymbol{x} \models I(\ell)$*, then* $\eta(\ell, \boldsymbol{x}) \geq 0$*;*
- C3: *if* $\ell = \perp$*, then* $\eta(\ell, \boldsymbol{x}) = K$*;*
- C4: *if* $\ell \neq \perp$ *and* $\boldsymbol{x} \models I(\ell) \wedge \mathsf{PP}(\ell)$*, then* $\mathrm{pre}_\eta(\ell, \boldsymbol{x}) \leq \eta(\ell, \boldsymbol{x}) - \epsilon$*.*

Note that C2 and C3 together separate non-termination and termination by the threshold 0, and C4 is the *RSM difference* condition which is intuitively related to the ϵ difference in the RSM definition (cf. Definition 6). By generalizing our previous proofs in [12] (from LRSM to pRSM), we establish the soundness of pRSMs w.r.t both almost-sure and finite termination.

Theorem 1. *If there exists a* d*-pRSM* η *w.r.t* (P, I) *with constants* ϵ, K *(cf.* Definition 8*), then* P *is a.s. terminating and* $\mathsf{ET}(P) \leq \mathsf{UB}(P) := \frac{\eta(\ell_0, \boldsymbol{x}_0) - K}{\epsilon}$*.*

Example 3. Consider the running example (cf. Example 1) and the function η given in Example 2. Assuming that the initial valuation satisfies $1 \leq x \wedge x \leq 10$, we assign the trivial invariant I such that $I(1) = 0 \leq x \wedge x \leq 11$, $I(j) = 1 \leq x \wedge x \leq 10$ for $2 \leq j \leq 6$ and $I(7) = x < 1 \vee x > 10$. It is straightforward to verify that η is a 2-pRSM with $\epsilon = 0.2$ and $K = -0.2$ (cf. Definition 8 for ϵ, K). Hence by Theorem 1, the program in Example 1 terminates almost-surely under any scheduler and its expected termination time is at most $5 \cdot (x_0 - 1) \cdot (10 - x_0) + 51$, given the initial value x_0.

Remark 2. The running example (cf. Example 1) does not admit a linear (i.e. 1-) pRSM since $\mathbb{E}_R(r) = 0$ at label 3. This indicates that linear pRSMs may not exist even over simple affine programs like Example 1. Thus, this motivates the study of pRSMs even for simple affine programs.

Remark 3. The non-strict inequality symbol '$\geq$' in C2 can be replaced by its strict counterpart '$>$' since $\eta + c$ ($c > 0$) remains to be a pRSM if η is a pRSM and K (in C3) is sufficiently small. (By definition, $\mathrm{pre}_{\eta+c} = \mathrm{pre}_\eta + c$.) Moreover, the non-strict inequality symbol '$\leq$' in C4 can be replaced by '$<$' since a pRSM η and a constant K (for C3) can be scaled by a constant factor (e.g. 1.1) so that strict inequalities are ensured. Moreover, one can also assume that $K = -1$ and $\epsilon = 1$ in Definition 8. This is because one can first scale a pRSM with constants ϵ, K by a positive scalar to ensure that $\epsilon = 1$, and then safely set $K = -1$ due to C2.

Theorem 1 answers the questions of almost-sure and finite termination in a unified fashion. Generalizing our approach in [12], we show that by restricting a pRSM to have *bounded difference*, we also obtain concentration results.

Definition 9 (Difference-Bounded pRSM). *A* d*-pRSM* η *is* difference-bounded *w.r.t a non-empty interval* $[a, b] \subseteq \mathbb{R}$ *if the following conditions hold:*

- *for all* $\ell \in L_{\mathrm{d}} \cup L_{\mathrm{p}}$ *and* $(\ell, \alpha, \ell') \in \mapsto$, *and for all* $\boldsymbol{x} \in [\![I(\ell)]\!]$, *it holds that* $a \leq \eta(\ell', \boldsymbol{x}) - \eta(\ell, \boldsymbol{x}) \leq b$;
- *for all* $\ell \in L_{\mathrm{c}}$ *and* $(\ell, \phi, \ell') \in \mapsto$, *and for all* $\boldsymbol{x} \in [\![I(\ell) \wedge \phi]\!]$, *it holds that* $a \leq \eta(\ell', \boldsymbol{x}) - \eta(\ell, \boldsymbol{x}) \leq b$;
- *for all* $\ell \in L_{\mathrm{a}}$ *and* $(\ell, f, \ell') \in \mapsto$, *for all* $\boldsymbol{x} \in [\![I(\ell)]\!]$ *and for all* $\boldsymbol{r} \in \{\boldsymbol{r'} \mid \forall r \in R.\ \boldsymbol{r'}[r] \in \mathrm{Supp}_r\}$, *it holds that* $a \leq \eta(\ell', f(\boldsymbol{x}, \boldsymbol{r})) - \eta(\ell, \boldsymbol{x}) \leq b$.

Note that if a d-pRSM η with constants ϵ, K (cf. Definition 8) is difference-bounded w.r.t $[a, b]$, then from definition $a \leq -\epsilon$; one can further assume that $-\epsilon \leq b$ since otherwise one can reset $\epsilon := -b$. By definition, the stochastic process $X_n := \eta(\theta_n, \overline{\boldsymbol{x}}_n)$ defined through a difference-bounded pRSM w.r.t $[a, b]$ satisfies that $a \leq X_{n+1} - X_n \leq b$; then using Hoeffding's Inequality [12,26], we establish a concentration bound.

Theorem 2. *Let* η *be a difference-bounded* d*-pRSM w.r.t* $[a, b]$ *with constants* ϵ *and* K. *For all* $n \in \mathbb{N}$, *if* $\epsilon(n-1) > \eta(\ell_0, \boldsymbol{x}_0)$, *then* $\mathbb{P}(T_P > n) \leq e^{-\frac{2(\epsilon(n-1)-\eta(\ell_0,\boldsymbol{x}_0))^2}{(n-1)(b-a)^2}}$.

From Theorem 2, a difference-bounded d-pRSM η implies a concentration bound $\frac{\eta(\ell_0,\boldsymbol{x}_0)}{\epsilon} + 2$.

Example 4. Consider again our running example in Example 1 with invariant given in Example 3. Let η be the function illustrated in Table 1. One can verify that the interval $[-10.2, 8.6]$ satisfies the conditions specified in Definition 9 for η, as the following hold:

- for all $x \in [1, 10]$, $\eta(2, x) - \eta(1, x) = -0.2$;
- for all $x \in [0, 1) \cup (10, 11]$, $-10.2 \leq \eta(7, x) - \eta(1, x) \leq -0.2$;
- for all $x \in [1, 10]$ and $i \in \{3, 4\}$, $\eta(i, x) - \eta(2, x) = -0.2$;
- for all $x \in [1, 10]$ and $i \in \{5, 6\}$, $-9.4 \leq \eta(i, x) - \eta(4, x) \leq 8.6$;
- for all $x \in [1, 10]$, $\eta(1, x - 1) - \eta(5, x) = -0.2$;
- for all $x \in [1, 10]$, $\eta(1, x + 1) - \eta(6, x) = -0.2$;
- for all $x \in [1, 10]$ and $r \in \{-1, 1\}$, $-9.6 \leq \eta(1, x + r) - \eta(3, x) \leq 8.4$.

Then by Theorem 2, assuming that the program have initial value $x_0 = 5$, one can deduce that $\mathbb{P}(T_P > 50000) \leq e^{-\frac{2\cdot(0.2\cdot 49999-30)^2}{49999\cdot 18.8^2}} \approx 1.3016 \cdot 10^{-5}$.

We end this section with a result stating that whether a (difference-bounded) d-pRSM exists can be decided (using quantifier elimination).

Theorem 3. *For any fixed natural number* $d \in \mathbb{N}$, *the problem whether a (difference-bounded)* d*-pRSM w.r.t an input pair* (P, I) *exists is decidable.*

5 The Synthesis Algorithm

In this section, we present an efficient algorithmic approach for solving almost-sure/finite termination and concentration questions through synthesis of pRSMs. Instead of computationally-expensive quantifier elimination (cf. Theorem 3) we use Positivstellensatz, which is sound but not complete. Note that by Theorem 1, the existence of a pRSM implies both almost-sure and finite termination of a probabilistic program.

The General Framework. To synthesize a pRSM, the algorithm first sets up a polynomial template with unknown coefficients. Next, the algorithm finds values for the unknown coefficients, ϵ, K (cf. Definition 8) and $[a, b]$ (cf. Definition 9) so that C2-C4 in Definition 8 and concentration conditions in Definition 9 are satisfied. Note that from Definition 7, each $\mathrm{pre}_\eta(\ell, \cdot)$ is a (piecewise) polynomial over X whose coefficients are *linear combinations* of unknown coefficients from the polynomial template. Instead of using quantifier elimination (cf. e.g. [50] or Theorem 3), we use Positivstellensatz's [44]. We observe that each universally-quantified formula described in C2, C4 and Definition 9 can be decomposed (through disjunctive normal form of propositional polynomial predicate or transformation of max in Definition 7 into two conjunctive clauses) into a conjunction of formulae of the following pattern (†)

$$\forall \boldsymbol{x} \in \mathbb{R}^{|X|}. \left[(g_1(\boldsymbol{x}) \geq 0 \wedge \cdots \wedge g_m(\boldsymbol{x}) \geq 0) \rightarrow g(\boldsymbol{x}) > 0\right] \qquad (\dagger)$$

where each g_i is a polynomial with constant coefficients and g is one with unknown coefficients from the polynomial template. In the pattern, we over-approximate any possible '$g_j(\boldsymbol{x}) > 0$' by '$g_j(\boldsymbol{x}) \geq 0$'. By Remark 3, the difference between '$g(\boldsymbol{x}) > 0$' and '$g(\boldsymbol{x}) \geq 0$' does not matter.

Example 5. Consider again the program in Example 1 with its CFG. Consider the invariant specified in Example 3. The instances of the pattern for termination of this program are listed as follows, where each instance is represented by a pair (Γ, g) where Γ and g corresponds to $\{g_1, \ldots, g_m\}$ and resp. g described in (†).

- (C4, label 1) $(\{x-1, 10-x, x, 11-x\}, \eta(1,x) - \eta(2,x) - \epsilon)$;
- (C4, label 2) $(\{x-1, 10-x\}, \eta(2,x) - \eta(3,x) - \epsilon)$ and $(\{x-1, 10-x\}, \eta(2,x) - \eta(4,x) - \epsilon)$;
- (C4, label 3) $(\{x-1, 10-x\}, \eta(3,x) - \mathbb{E}_R((y,r) \mapsto \eta(1, y+r), x) - \epsilon)$;
- (C4, label 4) $(\{x-1, 10-x\}, \eta(4,x) - 0.51\eta(5,x) - 0.49\eta(6,x) - \epsilon)$;
- (C4, label 5) $(\{x-1, 10-x\}, \eta(5,x) - \eta(1, x-1) - \epsilon)$;
- (C4, label 6) $(\{x-1, 10-x\}, \eta(6,x) - \eta(1, x+1) - \epsilon)$;
- (C2) $(\{x, 11-x\}, \eta(1,x))$ and $(\{x-1, 10-x\}, \eta(j,x))$ for $2 \leq j \leq 6$.

In the next part, we show that such pattern can be solved by Positivstellensatz's.

5.1 Positivstellensatz's

We fix a linearly-ordered finite set X of variables and a finite set $\Gamma = \{g_1, \ldots, g_m\} \subseteq \mathfrak{R}[X]$ of polynomials. Let $[\![\Gamma]\!]$ be the set of all vectors $\boldsymbol{x} \in \mathbb{R}^{|X|}$ satisfying the propositional polynomial predicate $\bigwedge_{i=1}^{m} g_i \geq 0$. We first define pre-orderings and sums of squares as follows.

Definition 10 (Sums of Squares). *Define Θ to be the set of* sums-of-squares, *i.e.,*

$$\Theta := \left\{ \sum_{i=1}^{k} h_i^2 \mid k \in \mathbb{N} \text{ and } h_1, \ldots, h_k \in \mathfrak{R}[X] \right\}.$$

Definition 11 (Preordering). *The* preordering *generated by Γ is defined by:*

$$PO(\Gamma) := \left\{ \sum_{w\in\{0,1\}^m} h_w \cdot \prod_{i=1}^{m} g_i^{w_i} \mid \forall w.\ h_w \in \Theta \right\} .$$

Remark 4. It is well-known that a real-coefficient polynomial g of degree $2d$ is a sum of squares iff there exists a k-dimensional positive semi-definite real square matrix Q such that $g = \boldsymbol{y}^{\mathrm{T}} Q \boldsymbol{y}$, where k is the number of monomials of degree no greater than d and $\boldsymbol{y}$ is the column vector of all such monomials (cf. [27, Corollary 7.2.9]). This implies that the problem whether a given polynomial (with real coefficients) is a sum of squares can be solved by semi-definite programming [24].

Now we present the first Positivstellensatz, called Schmüdgen's Positivstellensatz.

Theorem 4 (Schmüdgen's Positivstellensatz [45]). *Let $g \in \mathfrak{R}[X]$. If the set $[\![\Gamma]\!]$ is compact and $g(\boldsymbol{x}) > 0$ for all $\boldsymbol{x} \in [\![\Gamma]\!]$, then $g \in PO(\Gamma)$.*

From Schmüdgen's Positivstellensatz, any polynomial g which is positive on $[\![\Gamma]\!]$ can be represented by

$$(\ddagger) : g = \sum_{w\in\{0,1\}^m} h_w \cdot g_w \ ,$$

where $g_w := \prod_{i=1}^{m} g_i^{w_i}$ and $h_w \in \Theta$ for each $w \in \{0,1\}^m$. To apply Schmüdgen's Positivstellensatz, the degrees of those h_w's are restricted to be no greater than a fixed natural number. Then from Remark 4 and by equating the coefficients of the same monomials between the two polynomials, Eq. (‡) results in a system of linear equalities that involves coefficients of g and variables (grouped as 2^m square matrices) under semi-definite constraints.

Example 6. Consider that $X = \{x\}$ and $\Gamma = \{1-x, 1+x\}$. Choose the maximal degree for sums of squares to be 2. Then from Remark 4, the form of Eq. (‡) can be written as:

$$g = \sum_{i=1}^{4} \left[\begin{pmatrix} 1 & x \end{pmatrix} \cdot \begin{pmatrix} a_{i,1,1} & a_{i,1,2} \\ a_{i,2,1} & a_{i,2,2} \end{pmatrix} \cdot \begin{pmatrix} 1 \\ x \end{pmatrix} \right] \cdot u_i$$

where $u_1 = 1$, $u_2 = 1 - x$, $u_3 = 1 + x$, $u_4 = 1 - x^2$ and each matrix $(a_{i,j,k})_{2\times 2}$ $(1 \le i \le 4)$ is a matrix of variables subject to be positive semi-definite.

Theorem 4 can be further refined by a weaker version of Putinar's Positivstellensatz.

Theorem 5 (Putinar's Positivstellensatz [41]). *Let $g \in \mathfrak{R}[X]$. If (i) there exists some $g_i \in \Gamma$ such that the set $\{\boldsymbol{x} \in \mathbb{R}^{|X|} \mid g_i(\boldsymbol{x}) \ge 0\}$ is compact and (ii) $g(\boldsymbol{x}) > 0$ for all $\boldsymbol{x} \in [\![\Gamma]\!]$, then*

$$(\S) \qquad g = h_0 + \sum_{i=1}^{m} h_i \cdot g_i$$

for some sums of squares $h_0, \ldots, h_m \in \Theta$.

Similar to Eqs. (‡) and (§) results in a system of linear equalities that involves variables for synthesis of a pRSM and matrices of variables under semi-definite constraints, provided that an upper bound on the degrees of sums of squares is enforced.

Example 7. Consider that $X = \{x\}$ and $\Gamma = \{1 - x^2, 0.5 - x\}$. Choose the maximal degree for sums of squares to be 2. Then the form of Eq. (§) can be written as:

$$g = \sum_{i=1}^{3} \left[\begin{pmatrix} 1 & x \end{pmatrix} \cdot \begin{pmatrix} a_{i,1,1} & a_{i,1,2} \\ a_{i,2,1} & a_{i,2,2} \end{pmatrix} \cdot \begin{pmatrix} 1 \\ x \end{pmatrix} \right] \cdot u_i$$

where $u_1 = 1$, $u_2 = 1 - x^2$, $u_3 = 0.5 - x$ and each matrix $(a_{i,j,k})_{2\times 2}$ $(1 \leq i \leq 3)$ is a matrix of variables subject to be positive semi-definite.

In the following, we introduce a Positivstellensatz entitled Handelman's Theorem when Γ consists of only linear (degree one) polynomials. For Handelman's Theorem, we assume that Γ consists of only linear (degree 1) polynomials and $[\![\Gamma]\!]$ is non-empty. (Note that whether a system of linear inequalities has a solution is decidable in PTIME [46].)

Definition 12 (Monoid). *The* monoid *of Γ is defined by:*

$$\mathit{Monoid}(\Gamma) := \left\{ \prod_{i=1}^{k} h_i \mid k \in \mathbb{N}_0 \text{ and } h_1, \ldots, h_k \in \Gamma \right\} .$$

Theorem 6 (Handelman's Theorem [25]). *Let $g \in \mathfrak{R}[X]$ be a polynomial such that $g(\boldsymbol{x}) > 0$ for all $\boldsymbol{x} \in [\![\Gamma]\!]$. If $[\![\Gamma]\!]$ is compact, then*

$$(\#) \qquad g = \sum_{i=1}^{d} a_i \cdot u_i$$

for some $d \in \mathbb{N}$, real numbers $a_1, \ldots, a_d \geq 0$ and $u_1, \ldots, u_d \in \mathit{Monoid}(\Gamma)$.

To apply Handelman's theorem, we consider a natural number which serves as a bound on the number of multiplicands allowed to form an element in $\mathit{Monoid}(\Gamma)$; then Eq. (#) results in a system of linear equalities involving $a_1, \ldots, a_d$. Unlike previous Positivstellensatz's, the form of Handelman's theorem allows us to construct a system of linear equalities free from semi-definite constraints.

Example 8. Consider that $X = \{x\}$ and $\Gamma = \{1 - x, 1 + x\}$. Fix the maximal number of multiplicands in an element of $\mathit{Monoid}(\Gamma)$ to be 2. Then the form of Eq. (#) can be rewritten as $g = \sum_{i=1}^{6} a_i \cdot u_i$ where $u_1 = 1$, $u_2 = 1 - x$, $u_3 = 1 + x$, $u_4 = 1 - x^2$, $u_5 = 1 - 2x + x^2$, $u_6 = 1 + 2x + x^2$ and each a_i $(1 \leq i \leq 6)$ is subject to be a non-negative real number.

5.2 The Algorithm for pRSM Synthesis

Based on the Positivstellensatz's introduced in the previous part, we present our algorithm for synthesis of pRSMs. Below, we fix an input probabilistic program P, an input polynomial invariant I and an input initial configuration $(\ell_0, \boldsymbol{x}_0)$ for P. Let $\mathcal{G} = (L, \bot, (X, R), \mapsto)$ be the associated CFG of P.

Description of the Algorithm PRSMSynth. We present a succinct description of the key ideas. The description of the key steps of the algorithm is as follows.

1. *Template η for a pRSM.* The algorithm fixes a natural number d as the maximal degree for a pRSM, constructs $\mathcal{M}_d$ as the set of all monomials over X of degree no greater than d, and set up a template d-pRSM η such that each $\eta(\ell, \cdot)$ is the polynomial $\sum_{h \in \mathcal{M}_d} a_{h,\ell} \cdot h$ where each $a_{h,\ell}$ is a (distinct) scalar variable (cf. C1).
2. *Bound for Sums of Squares and Monoid Multiplicands.* The algorithm fixes a natural number k as the maximal degree for a sum of squares (cf. Schmüdgen's and Putinar's Positivstellensatz) or as the maximal number of multiplicands in a monoid element (cf. Handelman's Theorem).
3. *RSM-Difference and Terminating-Negativity.* From Remark 3, the algorithm fixes ϵ to be 1 (cf. condition C3) and K to be -1 (cf. condition C4).
4. *Computation of pre-expectation* pre_η. With ϵ, K fixed to be resp. $1, -1$ in the previous step, the algorithm computes pre_η by Definition 7, whose all involved coefficients are linear combinations from $a_{h,\ell}$'s.
5. *Pattern Extraction.* The algorithm extracts instances conforming to pattern (†) from C2, C4 and formulae presented in Definition 9, and translates them into systems of linear equalities over variables among $a_{h,\ell}$'s, ϵ, K, and extra matrices of variables assumed to be positive semi-definite (cf. Schmüdgen's and Putinar's Positivstellensatz) or scalar variables assumed to be non-negative (cf. Handelman's Theorem) through Eqs. (‡), (§) and (#).
6. *Solution via Semidefinite or Linear Programming.* The algorithm calls semi-definite programming (for Schmüdgen's and Putinar's Positivstellensatz) or linear programming (for Handelman's Theorem) in order to check the feasibility or to optimize $\mathsf{UB}(P)$ (cf. Theorem 1 for upper bound of $\mathsf{ET}(P)$) over all variables among $a_{h,\ell}$'s and extra matrix/scalar variables from Eqs. (‡), (§) and (#). Note that the feasibility implies the existence of a (difference-bounded) d-pRSM; the existence of a d-pRSM in turn implies finite termination, and the existence of a difference-bounded d-pRSM in turn implies a concentration bound through Theorem 2.

The soundness of our algorithm is as follows.

Theorem 7 (Soundness). *Any function η synthesized through the algorithm* PRSMSynth *is a valid pRSM.*

Remark 5 (Efficiency). It is well-known that for semi-definite programs with a positive real number R to bound the Frobenius norm of any feasible solution, an approximate solution upto precision ϵ can be computed in polynomial

time in the size of the semi-definite program (with rational numbers encoded in binary), $\log R$ and $\log \epsilon^{-1}$ [24]. Thus, our sound approach presents an efficient method for analysis of many probabilistic programs. Moreover, when each propositional polynomial predicate in the probabilistic program involves only linear polynomials, then the sound form of Handelman's theorem can be applied, resulting in feasibility checking of systems of linear inequalities rather than semi-definite constraints. By polynomial-time algorithms for solving systems of linear inequalities [46], our approach is polynomial time (and thus efficient) over such programs.

Remark 6 (Semi-Completeness). Consider probabilistic programs of the following form: **while** ϕ **do if** $\star$ **then** P_1 **else** P_2 **od**, where P_1, P_2 are single assignments, $[\![\phi]\!]$ is compact, and invariants which assign to each label a propositional polynomial predicate is in DNF form that involves no strict inequality (i.e. no '$<$' or '$>$'). Upon such inputs, our approach is *semi-complete* in the sense that by raising the upper bounds for the degree of a sum of squares and the number of multiplicands in a monoid element, the algorithm PRSMSynth will eventually find a pRSM if it exists. This is because Theorems 4 to 6 are "semi-complete" when $[\![\Gamma]\!]$ is compact, as the terminal label can be separately handled by $\mathsf{PP}(\cdot)$ so that only compact Γ's for Positivstellensatz's may be formed, and the difference between strict and non-strict inequalities does not matter (cf. Remark 3).

6 Experimental Results

In this section, we present experimental results for our algorithm through the semi-definite programming tool SOSTOOLS [3] (that uses SeDuMi [1]) and the linear programming tool CPLEX [2]. Due to space constraints, the detailed description of the input probabilistic programs are in [11].

Experimental Setup. We consider six classical examples of probabilistic programs that exhibit distinct types non-linear behaviours. Our examples are, namely, *Logistic Map* adopted in [14] which was previously handled by Lagrangian relaxation and semi-definite programming whereas our approach uses linear programming, *Decay* that models a sequence of points converging stochastically to the origin, *Random Walk* that models a random walk within a bounded region defined through non-linear curves, *Gambler's Ruin* which is our running example (Example 1), *Gambler's Ruin Variant* which is a variant of Example 1, and *Nested Loop* which is a nested loop with stochastic increments. Except for *Gambler's Ruin Variant* and *Nested Loop*, our approach is semi-complete for all other examples (cf. Remark 6). In all the examples the invariants are straightforward and was manually integrated with the input. Since SOSTOOLS only produces numerical results, we modify "$\eta(\ell, \boldsymbol{x}) \geq 0$" in C2 to "$\eta(\ell, \boldsymbol{x}) \geq 1$" for Putinar's or Schmüdgen's Positivstellensatz and check whether the maximal numerical error of all equalities added to SOSTOOLS is sufficiently small over a bounded region. In our examples, the bounded region is $\{(x, y) \mid x^2 + y^2 \leq 2\}$ and the maximal numerical error should not exceed 1. Note that 1 is also our fixed ϵ in C4, and by Remark 3, the modification on C2 is

not restrictive. Instead, one may also pursue Sylvester's Criterion (cf. [27, Theorem 7.2.5]) to check membership of sums of squares through checking whether a square matrix is positive semi-definite or not.

Experimental Results. In Table 2, we present the experimental results, where 'Method' means that whether we use either Handelman's Theorem, Putinar's Positivstellensatz or Schmüdgen's Positivstellensatz to synthesize pRSMs, 'SOSTOOLS/CPLEX' means the running time for CPLEX/SOSTOOLS in seconds, 'error' is the maximal numerical error of equality constraints added into SOSTOOLS (when instantiated with the solutions), and $\eta(\ell_0, \cdot)$ is the polynomial for the initial label in the synthesized pRSM. The synthesized pRSMs (in the last column) refer to the variables of the program. All numbers except errors are rounded to 10^{-4}. For all the examples, our translation to the optimization problems are linear. We report the running times of the optimization tools and synthesized pRSMs. The experimental results were obtained on Intel Core i7-2600 machine with 3.4 GHz and 16 GB RAM.

Table 2. Experimental results

Example	Method	SOSTOOLS	error	$\eta(\ell_0, \cdot)$
Decay	Putinar	0.1248s	$\leq 10^{-9}$	$5282.3435x^2 + 5282.3435y^2 + 1$
Random Walk	Schmüdgen	0.7176s	$\leq 10^{-7}$	$-300x^2 - 300y^2 + 601$
Example	Method	CPLEX	-	$\eta(\ell_0, \cdot)$
Gambler's Ruin	Handelman	$\leq 10^{-2}$s	-	$33x - 3x^2$
Gambler's Ruin V	Handelman	$\leq 10^{-2}$s	-	$-21 + 100x - 70y - 100x^2 + 100xy$
Logistic Map	Handelman	$\leq 10^{-2}$s	-	$1000500.7496x$
Nested Loop	Handelman	$\leq 2 \cdot 10^{-2}$s	-	$48 + 160n + (m - x)(800n + 240)$

For all the examples we consider except Logistic Map, their almost-sure termination cannot be answered by previous approaches. For the Logistic-Map example, our reduction is to linear programming whereas existing approaches [14,47] reduce to semidefinite programming.

7 Conclusion and Future Work

In this paper, we extended linear ranking supermartingale (LRSM) for probabilistic programs proposed in [10,12] to polynomial ranking supermartingales (pRSM) for nondeterministic probabilistic programs. We developed the notion of (difference bounded) pRSM and proved that it is sound for almost-sure and finite termination, as well as for concentration bound (Theorems 1 and 2). Then we developed an efficient (sound but not complete) algorithm for synthesizing pRSMs through Positivstellensatz's (cf. Sect. 5.1), proved its soundness (Theorem 7) and argued its semi-completeness (Remark 6) over an important class of programs. Finally, our experiments demonstrate the effectiveness of our

synthesis approach over various classical probabilistic programs, where LRSMs do not exist (cf. Example 1 and Remark 2). Directions of future work are to explore (a) more elegant methods for numerical problems related to semi-definite programming, and (b) other forms of RSMs for more general class of probabilistic programs.

Acknowledgements. We thank anonymous referees for valuable comments. We also thank Hui Kong for his help on SOSTOOLS. The research was partly supported by Austrian Science Fund (FWF) NFN Grant No. S11407-N23 (RiSE/SHiNE), ERC Start grant (279307: Graph Games), ERC Advanced Grant (267989: QUAREM), and Natural Science Foundation of China (NSFC) under Grant No. 61532019.

References

1. SeDuMi 1.3 (2008). http://sedumi.ie.lehigh.edu/
2. IBM ILOG CPLEX Optimizer Interactive Optimizer Community Edition 12.6.3.0 (2010). http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/
3. SOSTOOLS v3.00 (2013). http://www.cds.caltech.edu/sostools/
4. Babic, D., Cook, B., Hu, A.J., Rakamaric, Z.: Proving termination of nonlinear command sequences. Form. Asp. Comput. **25**(3), 389–403 (2013)
5. Baier, C., Katoen, J.P.: Principles of Model Checking. MIT Press, Cambridge (2008)
6. Billingsley, P.: Probability and Measure, 3rd edn. Wiley, New York (1995)
7. Bournez, O., Garnier, F.: Proving positive almost-sure termination. In: Giesl, J. (ed.) RTA 2005. LNCS, vol. 3467, pp. 323–337. Springer, Heidelberg (2005)
8. Bradley, A.R., Manna, Z., Sipma, H.B.: Linear ranking with reachability. In: Etessami, K., Rajamani, S.K. (eds.) CAV 2005. LNCS, vol. 3576, pp. 491–504. Springer, Heidelberg (2005)
9. Bradley, A.R., Manna, Z., Sipma, H.B.: Termination of polynomial programs. In: Cousot [16], pp. 113–129
10. Chakarov, A., Sankaranarayanan, S.: Probabilistic program analysis with Martingales. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 511–526. Springer, Heidelberg (2013)
11. Chatterjee, K., Fu, H., Goharshady, A.K.: Termination analysis of probabilistic programs through positivstellensatz's (2016). arXiv CoRR: http://arxiv.org/abs/1604.07169
12. 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. ACM (2016)
13. Colón, M.A., Sipma, H.B.: Synthesis of linear ranking functions. In: Margaria, T., Yi, W. (eds.) TACAS 2001. LNCS, vol. 2031, pp. 67–81. Springer, Heidelberg (2001)
14. Cousot, P.: Proving program invariance and termination by parametric abstraction, Lagrangian relaxation and semidefinite programming. In: Cousot [16], pp. 1–24
15. Cousot, P., Cousot, R.: Abstract interpretation: a unified Lattice model for static analysis of programs by construction or approximation of fixpoints. In: POPL, pp. 238–252. ACM (1977)

16. Cousot, R. (ed.): VMCAI 2005. LNCS, vol. 3385. Springer, Heidelberg (2005)
17. Dubhashi, D., Panconesi, A.: Concentration of Measure for the Analysis of Randomized Algorithms, 1st edn. Cambridge University Press, New York (2009)
18. Durrett, R.: Probability: Theory and Examples, 2nd edn. Duxbury Press, Belmont (1996)
19. Esparza, J., Gaiser, A., Kiefer, S.: Proving termination of probabilistic programs using patterns. In: Madhusudan, P., Seshia, S.A. (eds.) CAV 2012. LNCS, vol. 7358, pp. 123–138. Springer, Heidelberg (2012)
20. Farkas, J.: A fourier-féle mechanikai elv alkalmazásai (Hungarian). Mathematikaiés Természettudományi Értesitö **12**, 457–472 (1894)
21. Fioriti, L.M.F., Hermanns, H.: Probabilistic termination: soundness, completeness, and compositionality. In: POPL, pp. 489–501. ACM (2015)
22. Floyd, R.W.: Assigning meanings to programs. Math. Asp. Comput. Sci. **19**, 19–33 (1967)
23. Foster, F.G.: On the stochastic matrices associated with certain queuing processes. Ann. Math. Stat. **24**(3), 355–360 (1953)
24. Grötschel, M., Lovasz, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization. Springer, Heidelberg (1993)
25. Handelman, D.: Representing polynomials by positive linear functions on compact convex polyhedra. Pacific J. Math. **132**, 35–62 (1988)
26. Hoeffding, W.: Probability inequalities for sums of bounded random variables. J. Am. Stat. Assoc. **58**(301), 13–30 (1963)
27. Horn, R.A., Johnson, C.R.: Matrix Analysis, 2nd edn. Cambridge University Press, Cambridge (2013)
28. Howard, H.: Dynamic Programming and Markov Processes. MIT Press, Cambridge (1960)
29. Hungerford, T.W.: Algebra. Springer, Heidelberg (1974)
30. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artif. intell. **101**(1), 99–134 (1998)
31. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: a survey. J. Artif. Intell. Res. **4**, 237–285 (1996)
32. Kemeny, J., Snell, J., Knapp, A.: Denumerable Markov Chains. D. Van Nostrand Company, Princeton (1966)
33. Kress-Gazit, H., Fainekos, G.E., Pappas, G.J.: Temporal-logic-based reactive mission and motion planning. IEEE Trans. Robot. **25**(6), 1370–1381 (2009)
34. 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)
35. McIver, A.K., Morgan, C.: Developing and reasoning about probabilistic programs in *pGCL*. In: Cavalcanti, A., Sampaio, A., Woodcock, J. (eds.) PSSE 2004. LNCS, vol. 3167, pp. 123–155. Springer, Heidelberg (2006)
36. McIver, A., Morgan, C.: Abstraction, Refinement and Proof for Probabilistic Systems. Monographs in Computer Science. Springer, New York (2005)
37. 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)
38. Motwani, R., Raghavan, P.: Randomized Algorithms. Cambridge University Press, New York (1995)
39. Paz, A.: Introduction to Probabilistic Automata. Computer Science and Applied Mathematics. Academic Press, New York (1971)

40. Podelski, A., Rybalchenko, A.: A complete method for the synthesis of linear ranking functions. In: Steffen, B., Levi, G. (eds.) VMCAI 2004. LNCS, vol. 2937, pp. 239–251. Springer, Heidelberg (2004)
41. Putinar, M.: Positive polynomials on compact semi-algebraic sets. Indiana Univ. Math. J. **42**, 969–984 (1993)
42. Rabin, M.: Probabilistic automata. Inf. Control **6**, 230–245 (1963)
43. 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)
44. Scheiderer, C.: Positivity and sums of squares: a guide to recent results. In: Putinar, M., Sullivant, S. (eds.) Emerging Applications of Algebraic Geometry. IMAVMA, vol. 149, pp. 271–324. Springer, New York (1996)
45. Schmüdgen, K.: The K-moment problem for compact semi-algebraic sets. Math. Ann. **289**, 203–206 (1991)
46. Schrijver, A.: Theory of Linear and Integer Programming. Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley, New York (1999)
47. Shen, L., Wu, M., Yang, Z., Zeng, Z.: Generating exact nonlinear ranking functions by symbolic-numeric hybrid method. J. Syst. Sci. Comput. **26**(2), 291–301 (2013)
48. Sohn, K., Gelder, A.V.: Termination detection in logic programs using argument sizes. In: PODS, pp. 216–226. ACM Press (1991)
49. Williams, D.: Probability with Martingales. Cambridge University Press, Cambridge (1991)
50. Yang, L., Zhou, C., Zhan, N., Xia, B.: Recent advances in program verification through computer algebra. Front. Comput. Sci. China **4**(1), 1–16 (2010)