1-2017

Stochastic invariants for probabilistic termination

Krishnendu CHATTERJEE

Petr NOVOTNÝ

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

Stochastic Invariants for Probabilistic Termination

Krishnendu Chatterjee
IST Austria, Klosterneuburg, Austria
Krishnendu.Chatterjee@ist.ac.at

Petr Novotný
IST Austria, Klosterneuburg, Austria
petr.novotny@ist.ac.at

Đorđe Žikelić
University of Cambridge, UK
dz277@cam.ac.uk

Abstract

Termination is one of the basic liveness properties, and we study the termination problem for probabilistic programs with real-valued variables. Previous works focused on the qualitative problem that asks whether an input program terminates with probability 1 (almost-sure termination). A powerful approach for this qualitative problem is the notion of ranking supermartingales with respect to a given set of invariants. The quantitative problem (probabilistic termination) asks for bounds on the termination probability, and this problem has not been addressed yet. A fundamental and conceptual drawback of the existing approaches to address probabilistic termination is that even though the supermartingales consider the probabilistic behaviour of the programs, the invariants are obtained completely ignoring the probabilistic aspect (i.e., the invariants are obtained considering all behaviours with no information about the probability).

In this work we address the probabilistic termination problem for linear-arithmetic probabilistic programs with nondeterminism. We formally define the notion of *stochastic invariants*, which are constraints along with a probability bound that the constraints hold. We introduce a concept of *repulsing supermartingales*. First, we show that repulsing supermartingales can be used to obtain bounds on the probability of the stochastic invariants. Second, we show the effectiveness of repulsing supermartingales in the following three ways: (1) With a combination of ranking and repulsing supermartingales we can compute lower bounds on the probability of termination; (2) repulsing supermartingales provide witnesses for refutation of almost-sure termination; and (3) with a combination of ranking and repulsing supermartingales we can establish persistence properties of probabilistic programs.

Along with our conceptual contributions, we establish the following computational results: First, the synthesis of a stochastic invariant which supports some ranking supermartingale and at the same time admits a repulsing supermartingale can be achieved via reduction to the existential first-order theory of reals, which generalizes existing results from the non-probabilistic setting. Second, given a program with "strict invariants" (e.g., obtained via abstract interpretation) and a stochastic invariant, we can check in polynomial time whether there exists a linear repulsing supermartingale w.r.t. the stochastic invariant (via reduction to LP). We also present experimental evaluation of our approach on academic examples.

POPL'17, January 15–21, 2017, Paris, France

http://dx.doi.org/10.1145/3009837.3009873

Categories and Subject Descriptors F.3.1 [*Logics and Meanings of Programs*]: Specifying and Verifying and Reasoning about Programs

General Terms Verification

Keywords Probabilistic Programs, Termination, Martingales, Concentration

1. Introduction

Probabilistic programs. There is a huge recent interest in the formal analysis of probabilistic programs, since they provide a rich framework to model a wide variety of applications ranging from randomized algorithms [28, 58], to stochastic network protocols [5, 52], robot planning [44, 51], or modelling problems in machine learning [37], to name a few. The extension of the classical imperative programs with *random value generators*, that produce random values according to some desired probability distribution, gives rise to probabilistic programs. The formal analysis of such programs, and probabilistic systems in general, gives rise to a wealth of research questions, which have been studied across diverse fields, such as probability theory and statistics [29, 42, 48, 59, 62], formal methods [5, 52], artificial intelligence [43, 44], and programming languages [15, 19, 30, 34, 66].

Termination problem. Termination is one of the most basic liveness properties for programs. For non-probabilistic programs the proof for termination coincides with the construction of a *ranking function* [35], and many different approaches exist for construction of ranking functions for non-probabilistic programs [11, 22, 60, 68]. For probabilistic programs there are many natural extensions of the termination problem. The two most natural questions related to the probability of termination of an input program are the *qualitative* and *quantitative* problems which are as follows:

1. *Qualitative problem: almost-sure termination.* The basic qualitative question is the *almost-sure termination* problem that asks whether the program terminates with probability 1 [9, 34].
2. *Quantitative problem: probabilistic termination.* The natural generalization of the qualitative question is the quantitative question of *probabilistic termination* that asks for a lower bound on the probability of termination of the program.

The above questions are the basic and fundamental questions for the static analysis of probabilistic programs.

Nondeterminism in probabilistic programs. The role of *nondeterminism* is also quite fundamental in probabilistic programs. The nondeterminism is necessary in many cases such as for abstraction. For efficient static analysis of large programs, it is infeasible to track all the variables. Abstraction allows to ignore some variables, and for the sake of analysis the worst-case behaviour must be considered

for them, which is modelled as nondeterminism. Besides the modelling aspects, the presence of nondeterminism significantly changes the landscape of theoretical results, which we discuss below.

Previous results: almost-sure termination. Given the importance of the termination problem for probabilistic programs, the problem has been studied in great depth. However, much of the previous research focused on the qualitative problem. The details are as follows:

- *Discrete probabilistic choices.* First [55, 56] presented techniques for termination of probabilistic programs with nondeterminism, but restricted only to discrete probabilistic choices.
- *Infinite probabilistic choices without nondeterminism.* The approach of [55, 56] was extended in [15] to *ranking* martingales and supermartingales. The approach of [15] presents a sound (but not complete) approach for almost-sure termination of infinite-state probabilistic programs (without nondeterminism) with integer and real-valued random variables drawn from distributions including uniform, Gaussian, and Poison. The connection of termination of probabilistic programs without nondeterminism to *Lyapunov ranking functions* was established in [9]. For probabilistic programs with countable state space and without nondeterminism, the Lyapunov ranking functions provide a sound and complete method to prove termination in finite time, which implies almost-sure termination [9, 36]. Another sound approach [57] for almost-sure termination is to explore the exponential decrease of probabilities upon bounded-termination through abstract interpretation [26].
- *Infinite probabilistic choices with nondeterminism.* For probabilistic programs with nondeterminism the theoretical results change significantly. The Lyapunov ranking function method as well as the ranking martingale method are sound but not complete in the presence of nondeterminism [34]. Finally, for probabilistic programs with nondeterminism, a sound and complete (for a well-defined class of probabilistic programs) characterization for almost-sure termination is obtained in [34], by generalizing the ranking supermartingale approach of [15]. The question of algorithmic synthesis of ranking supermartingales has also been considered, for probabilistic programs with linear arithmetic, and special classes of ranking supermartingales (such as linear and polynomial ranking supermartingales [18, 19]).

In all the existing approaches above for infinite-state probabilistic programs with non-determinism, the key technique for almost-sure termination is the notion of a ranking supermartingale (RSM). Intuitively, a ranking supermartingale is a function assigning numbers to program configurations (where each configuration consists of the current control location and current valuation of program variables) with the following property: in each reachable configuration, the expected value of the RSM in the next execution step is strictly smaller than its current value. Thus, RSMs form a probabilistic counterpart of classical ranking functions.

RSMs with respect to invariants. Since precisely characterizing the set of reachable configurations is infeasible in practice, the previous works for almost-sure termination of infinite-state probabilistic programs consider the existence of ranking supermartingales (RSMs) with respect to invariants. An *invariant* is a set of constraints on the variables of the program, one for each program location, such that along *all* executions of the program, if a program location is visited, then the program variables must satisfy the constraints of the respective program location. Hence, each invariant represents an over-approximation of the set of reachable configurations. The computational problem for almost-sure termination is to decide the existence of a RSM for a probabilistic program w.r.t. an input invariant, i.e. a function assigning numbers to configurations such that for each configuration in the invariant, the expected value of the RSM in the next step is smaller than the current one.

```
x := 10
while x ≥ 0 do
    if x ≤ 100 then x := x+sample(Uniform[−2, 1])
    else x := x+sample(Uniform[−1, 2])
     fi
od
```

Figure 1. A probabilistic program modeling a generalization of an asymmetric one-dimensional random walk.

RSMs and probabilistic termination. In the *probabilistic termination* problem we are interested in computing termination probabilities when the program does not terminate almost-surely. While reasoning about termination probabilities of probabilistic programs was considered before (at least on a theoretical level, see also Related Work section), approaches based on RSMs were not yet considered for this purpose. A fundamental and conceptual problem here is that while RSMs take into account the probabilistic behaviour of the program, the invariants completely ignore the probabilistic aspect as they must hold along all executions (i.e., the invariants are obtained considering all behaviours without any information about the probability). Since all previous works on RSMs consider RSMs w.r.t. invariants, this implies a fundamental limitation of this tool to address probabilistic termination. We illustrate this with an example below.

Motivating example. Consider the probabilistic program shown in Figure 1, which is an asymmetric one-dimensional random walk. The random walk is denoted by value x. If x is smaller than 100, then its value is incremented by a number uniformly chosen between $[-2, 1]$, otherwise the increment is uniform in $[-1, 2]$. In this random walk, x can have any value above 0. But once the value reaches 100, with high probability the value drifts away, and the program does not terminate. In this example, there is no effective invariant, as x can have any value. However, the assertion $x \leq 100$ is violated only with very small probability, and as long as $x \leq 100$ holds, the value of x tends to decrease *on average*.

Our contributions. In this work we consider the probabilistic termination problem for linear-arithmetic probabilistic programs with nondeterminism. Our contributions are manifold, ranging from (a) definition of stochastic invariants for probabilistic termination; to (b) introduction of *repulsing supermartingales* (RepSMs) and their effectiveness; to (c) computational results; and (d) experimental results. We describe each of them in details below.

Stochastic invariants. We formally define the notion of *stochastic invariants* for the probabilistic termination problem. A stochastic invariant consists of a constraint on the program variables for each program location (as for invariants), and a threshold value p, such that the constraint is violated at the location with probability at most p. For example, in the probabilistic program of Figure 1 we can consider a stochastic invariant with constraint $x \leq 100$ at location corresponding to the **if** $x \leq 100$ test, with the threshold value being very small (less than 10^{-5}), since the probability that x exceeds 100 is very low due to asymmetry.

Repulsing supermartingales. We introduce a concept of repulsing supermartingales (RepSMs), which are in some sense dual to RSMs. A RepSM for a set of program configurations C has non-negative value inside C and decreases on average outside C. Intuitively, while RSMs show that a program execution cannot avoid some set C of

configurations indefinitely, RepSMs show that program executions that start outside of C tend to avoid C, and that they actually tend to "run away" from C in some well defined sense. The RepSMs are inspired by martingale methods used for analysing so-called one-counter MDPs [12, 13], but they are more general and apply to vastly larger class of systems. Our results for RepSMs are as follows:

1. *Stochastic invariants.* We show that RepSMs can be used to obtain bounds on the probability threshold of the stochastic invariants.
2. *Effectiveness.* We show the effectiveness of RepSMs in the following three ways:
   - First, with a combination of RSMs and RepSMs we show how to obtain lower bounds on the probability of termination (i.e., sound bounds for probabilistic termination). Hence for programs that do not terminate almost-surely, but with high probability, our method can obtain such bounds.
   - Second, in program analysis, refuting a property is as important as proving, as refutation is important in bug-hunting. We show that RepSMs can provide witnesses for refuting almost-sure termination. Moreover, even for programs that terminate almost-surely, but have infinite expected termination time, RepSMs can serve as witnesses for infinite expected termination time.
   - Finally, we show the effectiveness of RepSMs beyond the termination problem. For reactive systems that are non-terminating a very basic property is persistence, which requires that the execution eventually stays in a desired set of configurations. We show that a combination of RSMs and RepSMs can establish persistence properties of probabilistic programs.

Computational results. We present two computational results.

1. *Repulsing supermartingales w.r.t. stochastic invariants.* First, we consider the problem of efficient algorithms for deciding the existence of RepSMs w.r.t. to stochastic invariants. Since our goal is to obtain efficient algorithms, we consider the simplest class of RepSMs, namely, *linear repulsing supermartingales* (LRepSMs). We show that given a program with "strict" invariants" (e.g., obtained via abstract interpretation) and a stochastic invariant, the existence of a LRepSM w.r.t. the stochastic invariant can be decided in polynomial time (via reduction to LP) provided that the stochastic invariant uses only *polyhedral* constraints (i.e. conjunctions of inequalities).
2. *Synthesis.* Second, we consider the problem of synthesis of a stochastic invariant which supports some RSM and at the same time admits a RepSM. We show that the synthesis problem can be achieved via reduction to the existential first-order theory of reals. This result generalizes existing results from the non-probabilistic setting, and even in the non-probabilistic setting the best-known computational methods require the existential theory of reals.

Experimental results. We present a basic implementation of our approach, and present experimental results on academic examples. Our main contributions are conceptual and algorithmic, and the experiments serve as a validation of the new concepts.

Due to space constraints, some technical details are omitted. They can be found in the full version of this paper [20].

2. Preliminaries

2.1 Basic Notions, Linear Predicates, Valuations

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 assume basic knowledge of matrix calculus. We use boldface notation for vectors, e.g. $\mathbf{x}$, $\mathbf{y}$, etc., and we denote an i-th component of a vector $\mathbf{x}$ by $\mathbf{x}[i]$. For the purpose of matrix calculations we assume that (non-transposed) vectors are row vectors. If $\mathbf{v}, \mathbf{v}'$ are n and m dimensional vectors, respectively, then $(\mathbf{v}, \mathbf{v}')$ is an $(n+m)$-dimensional vector obtained by "concatenation" of $\mathbf{v}$ and $\mathbf{v}'$. We identify 1-dimensional vectors with numbers. For an n-dimensional vector $\mathbf{x}$, index $1 \leq i \leq n$, and number a we denote by $\mathbf{x}(i \leftarrow a)$ a vector $\mathbf{y}$ such that $\mathbf{y}[i] = a$ and $\mathbf{y}[j] = \mathbf{x}[j]$ for all $1 \leq j \leq n$, $j \neq i$. For comparison of vectors (e.g. as in $\mathbf{x} \leq \mathbf{y}$), we consider componentwise comparison. For comparing functions f, g with the same domains, we write $f \leq g$ if $f(x) \leq g(x)$ for all x in the domain.

Variables and valuations. Throughout the paper we fix a countable set of variables $\mathcal{V}$. We consider some arbitrary but fixed linear order on the set of all variables. Hence, given some set of variables V we can enumerate its members in ascending order (w.r.t. the fixed ordering) and write $V = \{x_1, x_2, x_3, \dots\}$.

Affine expressions. An *affine expression* over the set of variables $\{x_1, \dots, x_n\}$ is an expression of the form $d + \sum_{i=1}^{n} a_i x_i$, where $d, a_1, \dots, a_n$ are real-valued constants. Each affine expression E over $\{x_1, \dots, x_n\}$ determines a function which for each m-dimensional vector $\mathbf{x}$, where $m \geq n$, returns a number resulting from substituting each x_i in E by $\mathbf{x}[i]$. Slightly abusing our notation, we denote this function also by E and the value of this function on argument $\mathbf{x}$ by $E(\mathbf{x})$. A function of the form $E(\mathbf{x})$ for some affine expression E is called affine.

Linear constraint, assertion, predicates. We use the following nomenclature:

- *Linear Constraint.* A *linear constraint* is a formula of the form ψ or $\neg\psi$, where ψ is a non-strict inequality between affine expressions.
- *Linear Assertion.* A *linear assertion* is a finite conjunction of linear constraints.
- *Propositionally Linear Predicate.* A *propositionally linear predicate* (PLP) is a finite disjunction of linear assertions.

Arity and satisfaction of PLP. For a PLP φ we denote by $\mathcal{V}(\varphi)$ the set of all variables that appear in φ. As noted above, we stipulate that $\mathcal{V}(\varphi) = \{x_1, \dots, x_{n(\varphi)}\}$ for some $n(\varphi) \in \mathbb{N}$. A vector $\mathbf{x}$ of dimension $m \geq n(\varphi)$ *satisfies* φ, we write $\mathbf{x} \models \varphi$, if the arithmetic formula obtained by substituting each occurrence of a variable x_i in φ by $\mathbf{x}[i]$ is valid. We denote $[\![\varphi]\!] = \{\mathbf{x} \in \mathbb{R}^m \mid m \geq n(\varphi) \wedge \mathbf{x} \models \varphi\}$ and $[\![\varphi]\!]^{\triangleleft} = [\![\varphi]\!] \cap \mathbb{R}^{n(\varphi)}$.

2.2 Syntax of Affine Probabilistic Programs (APPs)

The Syntax. We consider the standard syntax for affine probabilistic programs, which encompasses basic programming mechanisms such as assignment statement (indicated by ':='), while-loop, if-branch. We also consider basic probabilistic mechanisms such as probabilistic branch (indicated by 'prob') and random sampling (e.g. $x :=$ **sample**$(\mathrm{Uniform}[-2, 1])$ assigns to x a random number uniformly sampled from interval $[-2, 1]$). We also allow constructs for (demonic) non-determinism, in particular non-deterministic

branching indicated by '**if** $\star$ **then...**' construct and non-deterministic assignment. Variables (or identifiers) of a probabilistic program are of *real* type, i.e., values of the variables are real numbers. We allow only affine expressions in test statements and in the right-hand sides of assignments. We also assume that assume that each APP $\mathcal{P}$ is preceded by an initialization preamble in which each variable appearing in $\mathcal{P}$ is assigned some concrete number. Due to space restrictions, details (such as grammar) are relegated to the Appendix. For an example see Figure 2. We refer to this class of affine probabilistic programs as APPs.

2.3 Semantics of Affine Probabilistic Programs

We now formally define the semantics of APP's. In order to do this, we first recall some fundamental concepts from probability theory.

Basics of Probability Theory. The crucial notion is the one of a 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 *sigma-algebra* over Ω, i.e. a collection of subsets of Ω that contains the empty set $\emptyset$, and that is closed under complementation and countable unions, and $\mathbb{P}$ is a *probability measure* on $\mathcal{F}$, i.e., a function $\mathbb{P}\colon \mathcal{F} \to [0,1]$ such that

- $\mathbb{P}(\emptyset) = 0$,
- for all $A \in \mathcal{F}$ it holds $\mathbb{P}(\Omega \setminus A) = 1 - \mathbb{P}(A)$, and
- for all pairwise disjoint countable set sequences $A_1, A_2, \cdots \in \mathcal{F}$ (i.e., $A_i \cap A_j = \emptyset$ for all $i \neq j$) we have $\sum_{i=1}^{\infty} \mathbb{P}(A_i) = \mathbb{P}(\bigcup_{i=1}^{\infty} A_i)$.

Random variables and filtrations. A *random variable* in a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ is an $\mathcal{F}$-measurable function $R\colon \Omega \to \mathbb{R} \cup \{\infty\}$, i.e., a function such that for every $a \in \mathbb{R} \cup \{\infty\}$ the set $\{\omega \in \Omega \mid R(\omega) \leq a\}$ belongs to $\mathcal{F}$. We denote by $\mathbb{E}[R]$ the *expected value* of a random variable X (see [8, Chapter 5] for a formal definition). A *random vector* in $(\Omega, \mathcal{F}, \mathbb{P})$ is a vector whose every component is a random variable in this probability space. A *stochastic process* in a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ is an infinite sequence of random vectors in this space. We will also use random variables of the form $R\colon \Omega \to S$ for some finite set S, which is easily translated to the variables above. A *filtration* of a sigma-algebra $\mathcal{F}$ is a sequence $\{\mathcal{F}_i\}_{i=0}^{\infty}$ of σ-algebras such that $\mathcal{F}_0 \subseteq \mathcal{F}_1 \subseteq \cdots \subseteq \mathcal{F}_n \subseteq \cdots \subseteq \mathcal{F}$.

Distributions. We assume the standard definition of a probability distribution specified by a cumulative distribution function [8]. We denote by $\mathcal{D}$ be a set of probability distributions on real numbers, both discrete and continuous.

Probabilistic Control Flow Graphs. The semantics can be defined as the semantics of an uncountable state-space Markov decision process (MDP) (uncountable due to real-valued variables). We take an operational approach to define the semantics, and associate to each program a certain stochastic process [15, 34, 49]. To define this process, we first define so called *probabilistic control flow graphs* [18].

Definition 1. *A* probabilistic control flow graph (pCFG) *is a tuple* $\mathcal{C} = (L, V, \ell_{init}, \mathbf{x}_{init}, \mapsto, Pr, G)$*, where*

- L *is a finite set of* locations *partitioned into three pairwise disjoint subsets* L_N*,* L_P*, and* L_D *of non-deterministic, probabilistic, and deterministic locations;*
- $V = \{x_1, \ldots, x_{|V|}\}$ *is a finite set of* program variables *(note that* $V \subseteq \mathcal{V}$*) ;*
- ℓ_{init} *is an initial location and* $\mathbf{x}_{init}$ *is an initial* assignment vector*;*
- $\mapsto$ *is a transition relation, whose members are tuples of the form* (ℓ, i, u, ℓ')*, where* ℓ *and* ℓ' *are source and target program locations, respectively,* $1 \leq i \leq |V|$ *is a* target variable index*, and* u *is an* update element*, which can be one of the following mathematical objects: (a) an affine function* $u\colon \mathbb{R}^{|V|} \to \mathbb{R}$*; (b) a distribution* $d \in \mathcal{D}$*; or (c) a set* $R \subseteq \mathbb{R}$*.*
- $Pr = \{Pr_\ell\}_{\ell \in L_P}$ *is a collection of probability distributions, where each* Pr_ℓ *is a discrete probability distribution on the set of all transitions outgoing from* ℓ*.*
- G *is a function assigning a propositionally linear predicate (a* guard*) over* V *to each transition outgoing from a deterministic location.*

We assume that each location has at least one outgoing transition. Also, for every deterministic location ℓ *we assume the following: if* $\tau_1, \ldots, \tau_k$ *are all transitions outgoing from* ℓ*, then* $G(\tau_1) \vee \cdots \vee G(\tau_k) \equiv$ *true and* $G(\tau_i) \wedge G(\tau_j) \equiv$ *false for each* $1 \leq i < j \leq k$*. Moreover, for each distribution* d *appearing in the pCFG we assume the following features are known: expected value* $\mathbb{E}[d]$ *of* d *and a single-variable PLP* φ_d *such that the* support *of* d *(i.e. the smallest closed set of real numbers whose complement has probability zero under* d*)*[1] *satisfies* $supp(d) \subseteq [\![\varphi_d]\!]^{\lhd}$*. Finally, we assume that for each transition* (ℓ, j, u, ℓ') *such that* u *is a set the location* ℓ *is deterministic. This is just a technical assumption yielding no loss of generality, and it somewhat simplifies notation.*

Configurations. A *configuration* of a pCFG $\mathcal{C}$ is a tuple $(\ell, \mathbf{x})$, where ℓ is a location of $\mathcal{C}$ and $\mathbf{x}$ is an $|V|$-dimensional vector. We say that a transition τ is *enabled* in a configuration $(\ell, \mathbf{x})$ if ℓ is the source location of τ and in addition, $\mathbf{x} \models G(\tau)$ provided that ℓ is deterministic. A configuration $(\ell, \mathbf{x})$ is non-deterministic/probabilistic/deterministic if ℓ is non-deterministic/probabilistic/deterministic, respectively.

Executions and reachable configurations. A *finite path* (or *execution fragment*) in $\mathcal{C}$ is a finite sequence of configurations $(\ell_0, \mathbf{x}_0) \cdots (\ell_k, \mathbf{x}_k)$ such that for each $0 \leq i < k$ there is a transition $(\ell_i, j, u, \ell_{i+1})$ enabled in $(\ell_i, \mathbf{x}_i)$ such that $\mathbf{x}_{i+1} = \mathbf{x}_i(j \leftarrow a)$ where a satisfies one of the following:

- u is a function $f\colon \mathbb{R}^{|X|} \to \mathbb{R}$ and $a = f(\mathbf{x}_i)$;
- u is an integrable[2] distribution d and $a \in supp(d)$; or
- u is a set and $a \in u$.

A *run* (or *execution*) in $\mathcal{C}$ is an infinite sequence of configurations whose every finite prefix is a finite path. A configuration $(\ell, \mathbf{x})$ is *reachable* from the initial configuration $(\ell_{init}, \mathbf{x}_{init})$ if there is a finite path starting in $(\ell_{init}, \mathbf{x}_{init})$ that ends in $(\ell, \mathbf{x})$.

Schedulers. Due to the presence of non-determinism and probabilistic choices, a pCFG $\mathcal{C}$ may represent a multitude of possible behaviours. The probabilistic behaviour of $\mathcal{C}$ can be captured by constructing a suitable probability measure over the set of all its runs. Before this can be done, non-determinism in $\mathcal{C}$ needs to be resolved. This is done using the standard notion of a *scheduler*.

Definition 2 (Schedulers)**.** *A scheduler in an pCFG* $\mathcal{C}$ *is a tuple* $\sigma = (\sigma_t, \sigma_a)$*, where*

- σ_t *(here 't' stands for 'transition') is a function assigning to every finite path that ends in a non-deterministic configuration*

[1] In particular, a support of a *discrete* probability distribution d is simply the at most countable set of all points on a real line that have positive probability under d.

[2] A distribution on some numerical domain is integrable if its expected value exists and is finite. In particular, each Dirac distribution is integrable.

$(\ell, \mathbf{x})$ *a probability distribution on transitions outgoing from* ℓ*; and*

- σ_a *(here 'a' stands for 'assignment') is a function which takes as an argument a finite path ending in a deterministic configuration in which some transition* (ℓ, j, u, ℓ') *with* u *being a set is enabled, and for such a path it returns a probability distribution on* u*.*

Stochastic process. A pCFG $\mathcal{C}$ together with a scheduler σ can be seen as a stochastic process which produces a random run $(\ell_0, \mathbf{x}_0)(\ell_1, \mathbf{x}_1)(\ell_2, \mathbf{x}_2)\cdots$. The evolution of this process can be informally described as follows: we start in the initial configuration, i.e. $(\ell_0, \mathbf{x}_0) = (\ell_{init}, \mathbf{x}_{init})$. Now assume that i steps have elapsed, i.e. a finite path $(\ell_0, \mathbf{x}_0)(\ell_1, \mathbf{x}_1)\cdots(\ell_i, \mathbf{x}_i)$ has already been produced. Then

- A transition $\tau = (\ell, j, u, \ell')$ enabled in $(\ell_i, \mathbf{x}_i)$ is chosen as follows:
  - If ℓ_i is non-deterministic then τ is chosen randomly according to the distribution specified by scheduler σ, i.e. according to the distribution $\sigma_t((\ell_0, \mathbf{x}_0)(\ell_1, \mathbf{x}_1)\cdots(\ell_i, \mathbf{x}_i))$.
  - If ℓ_i is probabilistic, then τ is chosen randomly according to the distribution Pr_{ℓ_i}.
  - If ℓ_i is deterministic, then by the definition of a pCFG there is exactly one enabled transition outgoing from ℓ_i, and this transition is chosen as τ.
- Once τ is chosen as above, we put $\ell_{i+1} = \ell'$. Next, we put $\mathbf{x}_{i+1} = \mathbf{x}_i(j \leftarrow a)$, where a chosen as follows:
  - If u is a function $u\colon \mathbb{R}^{|V|} \to \mathbb{R}$, then $a = f(\mathbf{x}_i)$.
  - If u is a distribution d, then a is sampled from d.
  - If u is a set, then a is sampled from a distribution $\sigma_a((\ell_0, \mathbf{x}_0)(\ell_1, \mathbf{x}_1)\cdots(\ell_i, \mathbf{x}_i))$.

The above intuitive explanation can be formalized by showing that each pCFG $\mathcal{C}$ together with a scheduler σ uniquely determines a certain probabilistic space $(\Omega, \mathcal{R}, \mathbb{P}^\sigma)$ in which Ω is a set of all runs in $\mathcal{C}$, and a stochastic process $\mathcal{C}^\sigma = \{\mathbf{C}_i^\sigma\}_{i=0}^\infty$ in this space such that for each $\varrho \in \Omega$ we have that $\mathbf{C}_i^\sigma(\varrho)$ is the i-th configuration on run ϱ (i.e., $\mathbf{C}_i^\sigma$ is a random vector $(\ell_i^\sigma, \mathbf{x}_i^\sigma)$ with ℓ_i^σ taking values in L and $\mathbf{x}_i^\sigma$ being a random vector of dimension $|V|$ consisting of real-valued random variables). The sigma-algebra $\mathcal{R}$ is the smallest (w.r.t. inclusion) sigma algebra under which all the functions $\mathbf{C}_i^\sigma$, for all $i \geq 0$ and all schedulers σ, are $\mathcal{R}$-measurable (a function f returning vectors is $\mathcal{R}$-measurable if for all real-valued vectors $\mathbf{y}$ of appropriate dimension the set $\{\omega \in \Omega \mid f(\omega) \leq \mathbf{y}\}$ belongs to $\mathcal{R}$). The probability measure $\mathbb{P}^\sigma$ is such that for each i, the distribution of $\mathbf{C}_i^\sigma$ reflects the aforementioned way in which runs are randomly generated. The formal construction of $\mathcal{R}$ and $\mathbb{P}^\sigma$ is standard [8] and somewhat technical, hence we omit it. We denote by $\mathbb{E}^\sigma$ the expectation operator in probability space $(\Omega, \mathcal{R}, \mathbb{P}^\sigma)$. The translation from probabilistic programs to the corresponding pCFG is standard [19], and the details are presented in [20]. We point out that the construction produces pCFGs with a property that only transitions outgoing from a deterministic state can update program variables. All other transitions are assumed to be of the form $(\ell, 1, id_1, \ell')$ for some locations ℓ, ℓ', where $id_1(\mathbf{x}) = \mathbf{x}[1]$ for all $\mathbf{x}$. We use this to simplify notation. An illustration of a pCFG is given in Figure 2.

2.4 Almost-Sure and Probabilistic Termination

We consider computational problems related to the basic liveness properties of APPs, namely *termination* and its generalization, *reachability*.

Termination, reachability, and termination time. In the following, consider an APP P and its associated pCFG $\mathcal{C}_P$. We say that a run ϱ of $\mathcal{C}_P$ *reaches* a set of configurations C if it contains a configuration from C. A run *terminates* if it reaches a configuration whose first component (i.e. a location of $\mathcal{C}_P$) is the location ℓ_P^{out} corresponding to the value of the program counter after executing P. To each set of configurations C we can assign a random variable T^C such that for each run ϱ the value $T^C(\varrho)$ represents the first point in time when the current configuration on ϱ is in C. If a run ϱ does *not* reach a set C, then $T^C(\varrho) = \infty$. We call T^C the *reachability time* of C. In particular, if C is the set of all configurations $(\ell, \mathbf{x})$ such that $\ell = \ell_P^{out}$ (the terminal location of $\mathcal{C}_P$), then T^C is called a *termination time*, as it returns the number of steps after which ϱ terminates. Since termination time is an important concept on its own, we use a special notation $Term$ for it. Since a probabilistic program may exhibit more than one run, we are interested in probabilities of runs that terminate or reach some set of configurations. This gives rise to the following fundamental computational problems regarding termination:

```
x := 10
while x ≥ 1 do
    if prob(0.75) then x := x − 1 else x := x + 1
    fi
od
```

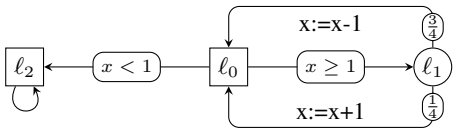

Figure 2. An APP modelling an asymmetric 1-D random walk and the associated pCFG. Probabilistic locations are depicted by circles, with probabilities given on outgoing transitions. Transitions are labelled by their effects. Location ℓ_0 is initial and ℓ_2 is terminal.

1. *Almost-sure termination:* A probabilistic program P is almost-surely (a.s.) terminating if under each scheduler σ it holds that $\mathbb{P}^\sigma(\{\varrho \mid \varrho \text{ terminates}\}) = 1$, or equivalently, if for each σ it holds $\mathbb{P}^\sigma(Term < \infty) = 1$. In almost-sure termination question for P we aim to prove that P is almost-surely terminating.
2. *Probabilistic termination:* In probabilistic termination question for P we aim to compute a *lower bound on the probability of termination*, i.e. a bound $b \in [0, 1]$ such that for each scheduler σ it holds $\mathbb{P}^\sigma(\{\varrho \mid \varrho \text{ terminates}\}) \geq b$ (or equivalently $\mathbb{P}^\sigma(Term < \infty) \geq b$).

We also define corresponding questions for the more general reachability concept.

1. *Almost-sure reachability:* For a set C of configurations of a probabilistic program P, prove (if possible) that under each scheduler σ it holds that $\mathbb{P}^\sigma(T^C < \infty) = 1$.
2. *Probabilistic reachability:* For a set C of configurations of a probabilistic program P, compute a bound $b \in [0, 1]$ such that for each scheduler σ it holds $\mathbb{P}^\sigma(T^C < \infty) \geq b$.

Since termination is a special case of reachability, each solution to the almost-sure or probabilistic reachability questions provides solution for the corresponding termination questions.

3. Invariants and Ranking Supermartingales

In this section we recall known methods and constructs for solving the qualitative termination and reachability questions for APPs, namely linear invariants and ranking supermartingales. We also demonstrate that these methods are not sufficient to address the

quantitative variants of these questions (i.e., probabilistic termination). In order to discuss the necessary concepts, we recall the basics of martingales, which is relevant for both this and subsequent sections.

3.1 Pure Invariants

Invariants are a vital element of many program analysis techniques. Intuitively, invariants are maps assigning to each program location ℓ of some pCFG a predicate which is guaranteed to hold whenever ℓ is entered. To avoid confusion with stochastic invariants, that we introduce later, we call these standard invariants *pure invariants*.

Definition 3 (Linear Predicate Map (LPM) and Pure Invariant)**.** *We define the following:*

1. *A* linear predicate map (LPM) *for an* APP P *is a function I assigning to each location ℓ of the pCFG $\mathcal{C}_P$ a propositionally linear predicate $I(\ell)$ over the set of program variables of P.*
2. *A* pure linear invariant *(or just a pure invariant) for an* APP P *is a linear predicate map I for P with the following property: for each location ℓ of $\mathcal{C}_P$ and each finite path $(\ell_0, \mathbf{x}_0), \cdots, (\ell_n, \mathbf{x}_n)$ such that $(\ell_0, \mathbf{x}_0) = (\ell_{init}, \mathbf{x}_{init})$ and $\ell_n = \ell$ it holds $\mathbf{x}_n \models I(\ell)$.*

3.2 Supermartingales

(Super)martingales, are a standard tool of probability theory apt for analyzing probabilistic objects arising in computer science, from automata-based models [14] to general probabilistic programs [6, 15, 18, 19, 34].

Let us first recall basic definitions and results related to supermartingales, which we need in our analysis.

Conditional Expectation. Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, $X \colon \Omega \to \mathbb{R}$ an $\mathcal{F}$-measurable function, and $\mathcal{F}' \subseteq \mathcal{F}$ sub-sigma-algebra of $\mathcal{F}$. The *conditional expectation* of X given $\mathcal{F}'$ is an $\mathcal{F}'$-measurable random variable denoted by $\mathbb{E}[X|\mathcal{F}']$ which satisfies, for each set $A \in \mathcal{F}'$, the following:

$$\mathbb{E}[X \cdot 1_A] = \mathbb{E}[\mathbb{E}[X|\mathcal{F}] \cdot 1_A], \tag{1}$$

where $1_A \colon \Omega \to \{0, 1\}$ is an *indicator function* of A, i.e. function returning 1 for each $\omega \in A$ and 0 for each $\omega \in \Omega \setminus A$. Note that the left hand-side of (1) intuitively represents the expected value of $X(\omega)$ with domain restricted to A.

Recall that in context of probabilistic programs we work with probability spaces of the form $(\Omega, \mathcal{R}, \mathbb{P}^\sigma)$, where Ω is a set of runs in some $\mathcal{C}$ and $\mathcal{F}$ is (the smallest) sigma-algebra such that all the functions $\mathbf{C}_i^\sigma$, where $i \in \mathbb{N}_0$ and σ is a scheduler, are $\mathcal{R}$-measurable. In such a setting we can also consider sub-sigma-algebras $\mathcal{R}_i$, $i \in \mathbb{N}_0$, of $\mathcal{R}$, where $\mathcal{R}_i$ is the smallest sub-sigma-algebra of $\mathcal{R}$ such that all the functions $\mathbf{C}_j^\sigma$, $0 \leq j \leq i$, are $\mathcal{R}_i$-measurable. Intuitively, each set A belonging to such an $\mathcal{R}_i$ consists of runs whose first i steps satisfy some property, and the probability space $(\Omega, \mathcal{R}_i, \mathbb{P}^\sigma)$ allows us to reason about probabilities of certain events happening in the first i steps of program execution. Then, for each $A \in \mathcal{R}_i$, the value $\mathbb{E}[\mathbb{E}[X|\mathcal{R}_i] \cdot 1_A]$ represents the expected value of $X(\varrho)$ for the randomly generated run ϱ provided that we restrict to runs whose prefix of length i satisfies the property given by A. Note that the sequence $\mathcal{R}_0, \mathcal{R}_1, \mathcal{R}_2, \ldots$ forms a filtration of $\mathcal{R}$, which we call a *canonical filtration*.

Definition 4 (Supermartingale)**.** *Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and $\{\mathcal{F}_i\}_{i=0}^\infty$ a filtration of $\mathcal{F}$. A sequence of random variables $\{X_i\}_{i=0}^\infty$ is a* supermartingale *w.r.t. filtration $\{\mathcal{F}_i\}_{i=0}^\infty$ if it satisfies these conditions:*

1. *The process $\{X_i\}_{i=0}^\infty$ is* adapted *to $\{\mathcal{F}_i\}_{i=0}^\infty$, i.e. for all $i \in \mathbb{N}_0$ it holds that X_i is $\mathcal{F}_i$-measurable.*
2. *For all $i \in \mathbb{N}_0$ it holds $\mathbb{E}[|X_i|] < \infty$.*
3. *For all $i \in \mathbb{N}_0$ it holds*

$$\mathbb{E}[X_{i+1}|\mathcal{F}_i] \leq X_i. \tag{2}$$

A supermartingale $\{X_i\}_{i=0}^\infty$ has c-bounded differences, where $c \geq 0$, if $|X_{i+1} - X_i| < c$ for all $i \in \mathbb{N}_0$

Intuitively, a supermartingale is a stochastic process whose average value is guaranteed not to rise as time evolves, even if some information on the past evolution of the process is revealed. We often need to work with supermartingales whose value is guaranteed to decrease on average, until a certain condition is satisfied. The point in time in which such a condition is satisfied is called a *stopping time*.

Definition 5 (Stopping time)**.** *Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and $\{\mathcal{F}_i\}_{i=0}^\infty$ a filtration. A random variable $T \colon \Omega \to \mathbb{N}_0$ is called a* stopping time *w.r.t. $\{\mathcal{F}_i\}_{i=0}^\infty$ if for all $j \in \mathbb{N}_0$ the set $\{\omega \in \Omega \mid T(\omega) \leq j\}$ belongs to $\mathcal{F}_j$.*

In particular, for each set of configurations C the reachability time T^C of C is a stopping time w.r.t. the canonical filtration, since at each time j we can decide whether $T^C > j$ or not by looking at the prefix of a run of length j. Finally, we recall the fundamental notion of a ranking supermartingale.

Definition 6 (Ranking supermartingale)**.** *Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, $\{\mathcal{F}_i\}_{i=0}^\infty$ a filtration of $\mathcal{F}$, T a stopping time w.r.t. that filtration, and $\epsilon \geq 0$. A supermartingale $\{X_i\}_{i=0}^\infty$ (w.r.t. $\{\mathcal{F}_i\}_{i=0}^\infty$) is* ϵ-decreasing *until T if it satisfies the following additional condition: for all $i \in \mathbb{N}_0$ it holds*

$$\mathbb{E}[X_{i+1}|\mathcal{F}_i] \leq X_i - \epsilon \cdot \mathbf{1}_{T>i}. \tag{3}$$

Further, $\{X_i\}_{i=0}^\infty$ is an ϵ-ranking *supermartingale (ϵ-RSM) for T if it is ϵ-decreasing until T and for each $\omega \in \Omega, j \in \mathbb{N}_0$ it holds $T(\omega) > j \Rightarrow X_j(\omega) \geq 0$.*

Intuitively, if T is the reachability time T^C of some set C, then the previous definition requires that an ϵ-ranking supermartingale must decrease by at least ϵ on average up to the point when C is reached for a first time. After that, it must not increase (on average). The above definition is a bit more general than the standard one in the literature as we also consider reachability as opposed to only termination.

Martingales in Program Analysis. In the context of APP analysis, we consider a special type of supermartingales given as functions of the current values of program variables. In this paper we focus on the case when these functions are *linear*.

Definition 7 (Linear Expression Map)**.** *A* linear expression map (LEM) *for an* APP P *is a function η assigning to each program location ℓ of $\mathcal{C}_P$ an affine expression $\eta(\ell)$ over the program variables of P.*

Each LEM η and location ℓ determines an affine function $\eta(\ell)$ which takes as an argument an n-dimensional vector, where n is the number of distinct variables in P. We use $\eta(\ell, \mathbf{x})$ as a shorthand notation for $\eta(\ell)(\mathbf{x})$. Martingales for APP analysis are defined via a standard notion of pre-expectation [15]. Intuitively, a pre-expectation of η is a function which for each configuration $(\ell, \mathbf{x})$ returns the maximal expected value of η after one step is made from this configuration, where the maximum is taken over all possible non-deterministic choices.

Definition 8 (Pre-Expectation)**.** *Let P be an* APP *such that $\mathcal{C}_P = (L, V, \ell_{init}, \mathbf{x}_{init}, \mapsto, Pr, G)$ and let η a linear expression map*

for P. The pre-expectation of η is a function $pre_\eta \colon L \times \mathbb{R}^{|V|} \to \mathbb{R}$ defined as follows:

- *if ℓ is a probabilistic location, then*

$$pre_\eta(\ell, \mathbf{x}) := \sum_{(\ell,1,id_1,\ell') \in \mapsto} Pr_\ell\left((\ell, 1, id_1, \ell')\right) \cdot \eta(\ell', \mathbf{x});$$

- *if ℓ is a non-deterministic location, then*

$$pre_\eta(\ell, \mathbf{x}) := \max_{(\ell,1,id_1,\ell') \in \mapsto} \eta(\ell', \mathbf{x});$$

- *if ℓ is a deterministic location, then for each $\mathbf{x}$ the value $pre_\eta(\ell, \mathbf{x})$ is determined as follows: there is exactly one transition $\tau = (\ell, j, u, \ell')$ such that $\mathbf{x} \models G(\tau)$. We distinguish three cases:*
  - *If $u \colon \mathbb{R}^{|V|} \to \mathbb{R}$ is a function, then*

$$pre_\eta(\ell, \mathbf{x}) := \eta(\ell', \mathbf{x}(j \leftarrow u(\mathbf{x}))).$$

  - *If u is a distribution d, then*

$$pre_\eta(\ell, \mathbf{x}) := \eta(\ell', \mathbf{x}(j \leftarrow \mathbb{E}[d])),$$

    where $\mathbb{E}[d]$ is the expected value of the distribution d.
  - *If u is a set, then*

$$pre_\eta(\ell, \mathbf{x}) := \max_{a \in u} \eta(\ell', \mathbf{x}(j \leftarrow a)).$$

Definition 9. *(Linear Ranking Supermartingale) Let P be an* APP *such that $\mathcal{C}_P = (L, V, \ell_{init}, \mathbf{x}_{init}, \mapsto, Pr, G)$, let I be a linear predicate map and let $C \subseteq L \times \mathbb{R}^{|V|}$ be some set of configurations. A linear ϵ-ranking supermartingale (ϵ-LRSM) for C supported by I is a linear expression map η for P such that for all configurations $(\ell, \mathbf{x})$ of $\mathcal{C}_P$ with $(\ell, \mathbf{x}) \notin C$ and $\mathbf{x} \models I(\ell)$ the following two conditions hold:*

- $\eta(\ell, \mathbf{x}) \geq 0$
- $pre_\eta(\ell, \mathbf{x}) \leq \eta(\ell, \mathbf{x}) - \epsilon$

A linear ϵ-ranking supermartingale supported by I has c-bounded differences if for each $(\ell, \mathbf{x})$ such that $\mathbf{x} \models I(\ell)$ and each configuration $(\ell', \mathbf{x}')$ such that $(\ell, \mathbf{x})(\ell', \mathbf{x}')$ is a path in $\mathcal{C}_P$ it holds $|\eta(\ell, \mathbf{x}) - \eta(\ell', \mathbf{x}')| \leq c$.

The relationship between ϵ-LRSM in APPs, (pure) invariants, and almost-sure termination is summarized in the following theorem.

Theorem 1 ([19, Theorem 1])**.** *Let P be an* APP*, σ a scheduler, and $(\Omega, \mathcal{R}, \mathbb{P}^\sigma)$ the corresponding probability space. Further, let C be the set of terminating configurations of $\mathcal{C}_P$ (i.e., the termination location is reached), such that there exist an $\epsilon > 0$ and an ϵ-linear ranking supermartingale η supported by a pure invariant I. Then*

1. *$\mathbb{P}^\sigma(Term < \infty) = 1$, i.e. termination is ensured almost-surely.*
2. *$\mathbb{E}^\sigma[Term] < \eta(\ell_{init}, \mathbf{x}_{init})/\epsilon$.*

The previous result shows that if there exists an ϵ-LRSM supported by a pure invariant I, for $\epsilon > 0$, then under each scheduler termination is ensured almost-surely. We now demonstrate that pure invariants, though effective for almost-sure termination, are ineffective to answer probabilistic termination questions.

Example 1. *Consider the program in Figure 3. In each iteration of the outer loop each of the variables is randomly modified by adding a number drawn from some uniform distribution. Average increase of x in each iteration is $\frac{3}{8}$, while average decrease of y is $-\frac{3}{8}$. It is easy to see that a program does not terminate almost-surely: there is for instance a tiny but non-zero probability of x being decremented by at least $\frac{1}{8}$ in each of the first 240 loop iterations, after which we are stuck in the infinite inner loop. On the other hand, the expectations above show that there is a "trend" of y decreasing and x increasing, and executions that follow this trend eventually*

$x := 30, y := 20$
while $y \geq 0$ **do**
  $x := x +$ **sample**(Uniform$[-\frac{1}{4}, 1]$)
  $y := y +$ **sample**(Uniform$[-1, \frac{1}{4}]$)
  while $x \leq 0$ **do skip od**
od

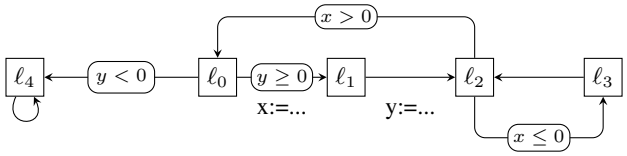

Figure 3. A program with infinitely many reachable configurations which terminates with high probability, but not almost surely, together with a sketch of its pCFG.

decrement y below 0 without entering the inner loop. Hence, the probabilistic intuition tells us that the program terminates with a high probability. However, the techniques of this section cannot prove this high-probability termination, since existence of an ϵ-LRSM (with $\epsilon > 0$) supported by a pure invariant already implies a.s. termination, and so no such ϵ-LRSM can exist for the program.

In the next section we generalize the notion of pure invariants to stochastic invariants for probabilistic termination to resolve issues like Example 1.

4. Stochastic Invariants and Probabilistic Termination

In this section we introduce *stochastic invariants*. Intuitively, stochastic invariants are linear predicate maps extended with an upper bound on the probability of their violation.

Definition 10 (Stochastic Linear Predicate Maps and Invariants)**.** *Stochastic linear predicate maps and stochastic invariants are defined as follows:*

- *A stochastic linear predicate map (SLPM) for an* APP *P is a pair (PI, p) where PI is a linear predicate map and $p \in [0, 1]$ is a probability.*
- *A stochastic linear invariant (or just a stochastic invariant) for an* APP *P is an SLPM (PI, p) for P with the following property: if we denote by $Fail(PI)$ the set of all runs initiated in $(\ell_{init}, \mathbf{x}_{init})$ that reach a configuration of the form $(\ell, \mathbf{x})$ with $\mathbf{x} \not\models PI(\ell)$, then for all schedulers σ it holds $\mathbb{P}^\sigma(Fail(PI)) \leq p$.*

Example 2. *Consider the* APP *consisting of a single statement $x :=$* ***sample***(Uniform$[0, 2]$)*. Denoting ℓ^{in}, ℓ^{out} the initial and terminal location of this program, respectively, the stochastic LPM $(PI, \frac{1}{2})$, where PI is such that $PI(\ell^{out}) \equiv x \geq 1$ and $PI \equiv true$ is a stochastic invariant for the program.*

Example 3. *Consider the example in Figure 3 and a stochastic LPM (PI, p) for the program such that $PI(\ell_2) \equiv x \geq 1$, $PI(\ell) \equiv true$ for all the other locations, and $p = 10^{-5}$. Then it is possible to prove that (PI, p) is a stochastic invariant for the program.*

Before presenting our result related to stochastic invariants, we first present a technical result. Intuitively, the result states that if we have an ϵ-LRSM for some set of configurations C supported by some linear predicate map I, then we can use it to obtain a supermartingale which decreases by at least ϵ (on average) in each step until we reach either a configuration in C or a configuration that does not satisfy I. In particular, if I is a pure invariant, the resulting supermartingale

decreases until we reach C. This is a result about pure invariants, which we will extend to stochastic invariants.

Lemma 1. *Let P be an* APP *and η a linear ϵ-ranking supermartingale for some set C of configurations of $\mathcal{C}_P$ supported by I. Let $\neg I$ be the set of all configurations $(\ell, \mathbf{x})$ such that $\mathbf{x} \not\models I(\ell)$. Finally, let $\{X_i\}_{i=0}^{\infty}$ be a stochastic process defined by*

$$X_i(\varrho) = \begin{cases} \eta(\mathbf{C}_i^{\sigma}(\varrho)) & \text{if } T^{C \cup \neg I} \geq i \\ X_{i-1}(\varrho) & \text{otherwise.} \end{cases}$$

Then under each scheduler σ the stochastic process $\{X_i\}_{i=0}^{\infty}$ is an ϵ-ranking supermartingale for $T^{C \cup \neg I}$. Moreover, if η has c-bounded differences, then so has $\{X_i\}_{i=0}^{\infty}$. In particular, if I is a pure invariant of P, then $\{X_i\}_{i=0}^{\infty}$ is an ϵ-ranking supermartingale for T^C.

We now establish a crucial connection between stochastic invariants, linear ranking supermartingales, and quantitative reachability (and thus quantitative termination).

Theorem 2. *Let $\{(PI_1, p_1), \ldots, (PI_n, p_n)\}$ be a set of stochastic linear invariants for* APP *P, and let I be a linear predicate map for P such that for each location ℓ of $\mathcal{C}_P$ the formula $I(\ell)$ is entailed by the formula $PI_1(\ell) \wedge \cdots \wedge PI_n(\ell)$. If there exists a linear ϵ-ranking supermartingale η for a set of configurations C such that η is supported by I, then under each scheduler σ it holds $\mathbb{P}^{\sigma}(T^C < \infty) \geq 1 - \sum_{j=1}^{n} p_j$.*

Proof. From Lemma 1 it follows that there is an ϵ-ranking supermartingale $\{X_i\}_{i=0}^{\infty}$ for $T^{C \cup \neg I}$. We can prove a generalization of Theorem 1 for other stopping times apart from $Term$, which gives us that under each scheduler σ the set of configurations $C \cup \neg I$ is reached with probability 1, where $\neg I$ is the set of all $(\ell, \mathbf{x})$ such that $\mathbf{x} \not\models I(\ell)$. But since each (PI_j, p_j) is a stochastic invariant, the probability that $\neg PI_j$ is reached is at most p_j under each scheduler. Using union bound the probability of reaching $\bigcup_{j=1}^{n} \neg PI_j$ is at most $\sum_{j=1}^{n} p_j$, from which the result follows. □

Example 4. *Let $(PI, 10^{-5})$ be the stochastic invariant from Example 3 (concerning Figure 3). For the corresponding program we can easily infer a pure invariant I' such that $I'(\ell_1) \equiv y \geq 0$, $I'(\ell_2) = I'(\ell_0) \equiv y \geq -1$ and $I'(\ell_4) \equiv I'(\ell_3) \equiv$ true (actually, standard methods would likely infer stronger pure invariants, but I' is sufficient for the sake of example). Consider a LEM η defined as follows $\eta(\ell_0) = 8y + 9$, $\eta(\ell_1) = 8y + 8$, $\eta(\ell_2) = 8y + 10$, $\eta(\ell_3) = 8y + 11$ and $\eta(\ell_4) = -1$. Then η is a 1-LRSM for the set of terminal configurations supported by LPM $I = I' \wedge PI$ (where the conjunction is locationwise). Now consider a set $\{(I', 0), (PI, 10^{-5})\}$. From Example 3 and from the fact that I' is a pure invariant it follows that both members of the set are stochastic invariants, and clearly $I' \wedge PI$ entails I. From Theorem 2 it follows that the program terminates with probability at least* 0.99999.

Theorem 2 shows a way in which probabilistic reachability and termination properties of APPs can be proved by use of ranking supermartingales and stochastic invariants. As highlighted in Example 3, the crucial question now is proving the existence of suitable stochastic invariants for APP. While there are various methods of obtaining pure linear invariants [23], e.g. those based on abstract interpretation [26], constraint solving [11] etc., these methods do not support reasoning about probabilities of a given assertion being satisfied, and thus they are not sufficient for obtaining stochastic invariants. In the next section we propose a framework for reasoning about stochastic invariants using repulsing supermartingales.

5. Proving Stochastic Invariance with Repulsing Supermartingales

Consider that we want to use stochastic invariants to prove that some APP P terminates with a high probability, by using Theorem 2. We need to achieve two things:

a. obtain a linear predicate map PI which supports some linear ranking supermartingale for the termination time $Term$ of P; and
b. obtain an upper bound p on the probability that PI is violated.

The part a. is not in any way related to the probability of p being satisfied, and hence we can aspire to adapt some of the techniques for generation of pure invariants. The part b. is substantially trickier, since it requires quantitative reasoning about the highly complex stochastic process $\{\mathbf{C}_i^{\sigma}\}_{i=0}^{\infty}$. To achieve this task, we introduce a notion of ϵ-*repulsing* supermartingale.

Intuitive idea of repulsing supermartingales. Intuitively, ϵ-repulsing supermartingales are again required to decrease by at least ϵ on average in every step until some stopping time, e.g. until reaching some set C of configurations. But now, instead of requiring the value of the process to be non-negative until C reached, we require it to be non-negative *upon* reaching C. This is because we typically work with repulsing supermartingales whose initial value is non-positive. Then, intuitively an ϵ-repulsing supermartingale is driven *away* from non-negative values by at least ϵ per step, which provides a probabilistic argument for showing that the C is reached with small probability.

Definition 11 (Repulsing supermartingale)**.** *Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, $\{\mathcal{F}_i\}_{i=0}^{\infty}$ a filtration of $\mathcal{F}$, T a stopping time w.r.t. that filtration, and $\epsilon \geq 0$. A supermartingale $\{X_i\}_{i=0}^{\infty}$ (w.r.t. $\{\mathcal{F}_i\}_{i=0}^{\infty}$) is* ϵ-repulsing *for T if it is ϵ-decreasing until T and for each $\omega \in \Omega, j \in \mathbb{N}_0$ it holds $T(\omega) = j \Rightarrow X_j(\omega) \geq 0$.*

To apply repulsing supermartingales to concrete programs, we again define the important special case of *linear repulsing supermartingales*.

Definition 12 (Linear repulsing supermartingale)**.** *Let P be an* APP *such that $\mathcal{C}_P = (L, V, \ell_{init}, \mathbf{x}_{init}, \mapsto, Pr, G)$, let I be a linear predicate map and let $C \subseteq L \times \mathbb{R}^{|V|}$ be some set of configurations. A linear ϵ-repulsing supermartingale (ϵ-LRepSM) for a set C supported by I is an LEM η for P such that for all configurations $(\ell, \mathbf{x})$ of $\mathcal{C}_P$ such that $\mathbf{x} \models I(\ell)$ the following holds*

- *if $(\ell, \mathbf{x}) \in C$, then $\eta(\ell, \mathbf{x}) \geq 0$*
- *if $(\ell, \mathbf{x}) \notin C$ and ℓ is not a terminal location, then $pre_{\eta}(\ell, \mathbf{x}) \leq \eta(\ell, \mathbf{x}) - \epsilon$,*

An ϵ-LRepSM supported by I has c-bounded differences if for each pair of locations ℓ, ℓ', each transition τ from ℓ, ℓ', and each pair of configurations $(\ell, \mathbf{x})$, $(\ell', \mathbf{x}')$ such that $\mathbf{x} \models I(\ell) \wedge G(\tau)$ and $(\ell', \mathbf{x}')$ can be produced by performing τ in $(\ell, \mathbf{x})$ it holds $|\eta(\ell, \mathbf{x}) - \eta(\ell', \mathbf{x}')| \leq c$.

Example 5 (Illustration of LRepSM)**.** *Consider the program shown in Figure 4, with initial value $x := 10$. Consider a linear predicate map PI such that $PI(\ell_0) \equiv x \leq 500$ and $PI(\ell_1) \equiv PI(\ell_2) \equiv$ true. Consider an LEM η that assigns to each pair (ℓ_i, x) a value $7 \cdot x + d_i$, where d_i is the i-th component of the ordered tuple $(-3499, -3500, -3500)$. It is straightforward to verify that η is a 1-LRepSM for $\neg PI$ supported by a trivial pure invariant assigning true to each location.*

The connection between ϵ-LRepSMs and general ϵ-repulsing supermartingales is similar as for their ranking variants (Lemma 1).

```
x := 10
while x ≥ 0 do
    if prob(0.5) then x := x + 1
       else x := x − 2
    fi
od
```

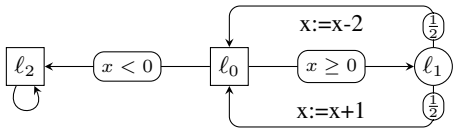

Figure 4. A probabilistic program example, with the accompanying pCFG.

That is, from ϵ-LRepSMs we can obtain a stochastic process which is a supermartingale w.r.t. the canonical filtration, which decreases at least by ϵ on average until the some set C is reached, and upon reaching C its value is non-negative.

Lemma 2. *Let P be an* APP *and η an ϵ-LRepSM for some set C of configurations of $\mathcal{C}_P$ supported by some linear predicate map I. Let $\neg I$ be the set of all configurations $(\ell, \mathbf{x})$ such that $\nu_{\mathbf{x}} \not\models I(\ell)$. Finally, let $\{X_i\}_{i=0}^{\infty}$ be a stochastic process defined by*

$$X_i(\varrho) = \begin{cases} \eta(\mathbf{C}_i^{\sigma}(\varrho)) & \text{if } T^{C \cup \neg I} \geq i \\ X_{i-1}(\varrho) & \text{otherwise.} \end{cases}$$

Then under each scheduler σ the stochastic process $\{X_i\}_{i=0}^{\infty}$ is an ϵ-repulsing supermartingale for $T^{C \cup \neg I}$. Moreover, if η has c-bounded differences, then so has $\{X_i\}_{i=0}^{\infty}$. In particular, if I is a pure invariant of P, then $\{X_i\}_{i=0}^{\infty}$ is an ϵ-repulsing supermartingale for T^C.

We now show how to obtain an upper bound on the probability of an invariant failure via repulsing supermartingales. Techniques used within the proof of Theorem 1 (which are similar to the proof of Lemma 5.5 in [34]) are not applicable, as they crucially rely on the fact that the supermartingale is non-negative before reaching C. Instead, we use a powerful tool of Martingale theory called Azuma's inequality.

Theorem 3 (Azuma's inequality [4])**.** *Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and $\{X_i\}_{i=0}^{\infty}$ a supermartingale w.r.t. $\mathcal{F}$ with c-bounded differences. Then for each $n \in \mathbb{N}_0$ and each $\lambda > 0$ it holds*

$$\mathbb{P}(X_n - X_0 \geq \lambda) \leq e^{-\frac{\lambda^2}{2nc^2}}.$$

Intuitively, Azuma's inequality provides exponentially decreasing tail bound on the probability that a supermartingale exhibits a large deviation from its expected value. In the following lemma (inspired by martingale use in [12]) the Azuma's inequality is used to obtain exponentially decreasing bound on probability that the set C is reached in exactly n steps.

Lemma 3. *Let C be a set of configurations of an* APP *P. Denote by F_n the set of all runs ϱ such that $T^C(\varrho) = n$. Suppose that there exist $\epsilon > 0$, $c > 0$ and a linear ϵ-repulsing supermartingale η for C supported by some pure invariant I such that η has c-bounded differences and $\eta(\ell_{init}, \mathbf{x}_{init}) < 0$. Then under each scheduler σ it holds*

$$\mathbb{P}^{\sigma}(F_n) \leq \alpha \cdot \gamma^n,$$

where $\gamma = e^{-\frac{\epsilon^2}{2(c+\epsilon)^2}}$, $\alpha = e^{\frac{\epsilon \cdot m_0}{(c+\epsilon)^2}}$ and $m_0 = \eta(\ell_{init}, \mathbf{x}_{init})$.

Key proof idea. We use η to obtain a supermartingale $\{\tilde{X}_i\}_{i=0}^{\infty}$ with c-bounded differences such that for for each run $\varrho \in F_n$ it holds $\tilde{X}_n(\varrho) - \tilde{X}_0(\varrho) \geq n \cdot \epsilon - m_0$. We then apply the Azuma's inequality on $\{\tilde{X}_i\}_{i=0}^{\infty}$ to get the desired bound on the probability of $\tilde{X}_n(\varrho) - \tilde{X}_0(\varrho) \geq n \cdot \epsilon - m_0$ and thus also on the probability of F_n. □

Proof. Using Lemma 2 we get from η a stochastic process $\{X_i\}_{i=0}^{\infty}$ which is, for each scheduler σ, an ϵ-repulsing supermartingale for the stopping time T^C with c-bounded differences. Now we define a stochastic process $\{\tilde{X}_i\}_{i=0}^{\infty}$ by putting

$$\tilde{X}_i(\varrho) = \begin{cases} X_i(\varrho) + i \cdot \epsilon & \text{if } T^C(\varrho) \geq i \\ \tilde{X}_{i-1}(\varrho) & \text{otherwise.} \end{cases}$$

Since $\{X_i\}_{i=0}^{\infty}$ is ϵ-decreasing until T^C, the process $\{\tilde{X}_i\}_{i=0}^{\infty}$ is a supermartingale. Moreover, it is easy to check that $\{\tilde{X}_i\}_{i=0}^{\infty}$ has $(c+\epsilon)$-bounded differences. Now for each ϱ we have $\tilde{X}_0(\varrho) = \eta(\ell_{init}, \mathbf{x}_{init}) < 0$. Moreover, from the definitions of $\{X_i\}_{i=0}^{\infty}$ and $\{\tilde{X}_i\}_{i=0}^{\infty}$ we get that $\varrho \in F_n$ implies $\tilde{X}_n(\varrho) = X_n(\varrho) + n \cdot \epsilon = \eta(\mathbf{C}_n^{\sigma}(\varrho)) + n \cdot \epsilon \geq n \cdot \epsilon$ (since η assigns non-negative value to configurations in C and $\varrho \in F_n$ is within C in step n), and adding $-\tilde{X}_0(\varrho)$ to both sides yields $\varrho \in F_n \Rightarrow \tilde{X}_n(\varrho) - \tilde{X}_0(\varrho) \geq n \cdot \epsilon - m_0$; recall $m_0 = \eta(\ell_{init}, \mathbf{x}_{init}) = \tilde{X}_0(\varrho)$. Hence, for each scheduler σ we have

$$\mathbb{P}^{\sigma}(F_n) \leq \mathbb{P}^{\sigma}(\tilde{X}_n - \tilde{X}_0 \geq n \cdot \epsilon - m_0). \tag{4}$$

Applying the Azuma's inequality for $\{\tilde{X}_i\}_{i=0}^{n}$ on (4) we get

$$\begin{aligned} \mathbb{P}^{\sigma}(F_n) &\leq \mathbb{P}(\tilde{X}_n - \tilde{X}_0 \geq n \cdot \epsilon - m_0) \\ &\leq \alpha \cdot e^{-\frac{n^2 \cdot \epsilon^2}{2n(c+\epsilon)^2}} = \alpha \cdot \gamma^n, \end{aligned}$$

where $\alpha = e^{\frac{\epsilon \cdot m_0}{(c+\epsilon)^2}}$. □

Using the above lemma we can bound the probability of reaching C by a geometric series which can be easily evaluated.

Theorem 4. *Let C be a set of configurations of an* APP *P. Suppose that there exist $\epsilon > 0$, $c > 0$ and a linear ϵ-repulsing supermartingale η for C supported by some pure invariant I such that η has c-bounded differences and $\eta(\ell_{init}, \mathbf{x}_{init}) < 0$. Then under each scheduler σ it holds*

$$\mathbb{P}^{\sigma}(T^C < \infty) \leq \alpha \cdot \frac{\gamma^{\lceil |\eta(\ell_{init}, \mathbf{x}_{init})| / c \rceil}}{1 - \gamma}, \tag{5}$$

where $\gamma = e^{-\frac{\epsilon^2}{2(c+\epsilon)^2}}$ and $\alpha = e^{\frac{\epsilon \cdot m_0}{(c+\epsilon)^2}}$ and $m_0 = \eta(\ell_{init}, \mathbf{x}_{init})$ is the initial value.

Proof. For each n let F_n be as in Lemma 3. Denote by A the number $\lceil |\eta(\ell_{init}, \mathbf{x}_{init})| / c \rceil$. Observe that $F_n = \emptyset$ for each $n < A$. Indeed, we need at least A steps to reach C from the initial configuration, because $\eta(\ell_{init}, \mathbf{x}_{init}) \leq 0$ (by the definition of a linear ranking supermartingale), the value of η can increase by at most c in each step, and reaching C entails that the value of η becomes non-negative. Hence, for each scheduler σ we get

$$\begin{aligned} \mathbb{P}^{\sigma}(T^C < \infty) &= \sum_{n=0}^{\infty} \mathbb{P}^{\sigma}(F_n) = \sum_{n=A}^{\infty} \mathbb{P}^{\sigma}(F_n) \leq \sum_{n=A}^{\infty} \alpha \cdot \gamma^n \\ &= \alpha \cdot \frac{\gamma^A}{1 - \gamma}, \end{aligned}$$

as required (the inequality at the end of the first line comes from Lemma 3). □

Example 6 (Illustration of Theorem 4)**.** *Looking back at Example 5, the absolute value of the change in η at each step is bounded from above by 12. Since the initial value of η is -3429, we use Azuma's inequality and Theorem 4 to get the probability bound $5.06 \cdot 10^{-6}$ on the violation of PI.*

We now present the corollary that establishes the effectiveness of LRepSM for stochastic invariants.

Corollary 1. *Let PI be a linear predicate map. Denote by $\neg PI$ the set of all configurations $(\ell, \mathbf{x})$ such that $\mathbf{x} \not\models PI(\ell)$. Assume that there exist $\epsilon > 0$, $c > 0$, and an ϵ-LRepSM η for $\neg PI$ with c-bounded differences such that $\eta(\ell_{init}, \mathbf{x}_{init}) < 0$. Then (PI, p) with $p = e^{\frac{\epsilon \cdot m_0}{(c+\epsilon)^2}} \cdot \frac{\gamma^{\lceil |\eta(\ell_{init}, \mathbf{x}_{init})| / c \rceil}}{1-\gamma}$ (γ and m_0 are as in Theorem 4) is a stochastic invariant.*

We note that the bound obtained from (5) is sound, but not necessarily tight. The magnitude of this bound crucially depends on η and on initial valuation of variables. In Section 8 we discuss how to find a LRepSM η providing good bounds in practice (as also illustrated in Example 6).

6. Effectiveness of Repulsing Supermartingales

In this section we discuss the effectiveness of repulsing supermartingales in several problems in analysis of probabilistic programs.

6.1 Probabilistic Termination

In Section 5 we establish the effectiveness of repulsing supermartingales for stochastic invariants. Theorem 2 shows that stochastic invariants along with ranking supermartingales can obtain bounds for the probabilistic termination problem. Hence the combination of repulsing and ranking supermartingales can answer the probabilistic termination problem.

6.2 Refuting Almost-Sure and Finite Termination

While a significant effort in analysis of non-probabilistic programs is devoted to proving termination, for bug-hunting purposes the analysis is often complemented by methods that aim to prove that a given program does *not* terminate [3, 21, 39, 53, 69]. Similarly for probabilistic programs we can ask for refutation of almost-sure termination of a given program. We show how RepSMs can be used to this end.

If we have an ϵ-LRepSM η for the set of terminal configurations and the bound obtained from Theorem 4 is smaller than 1, then η in particular proves that the program does not terminate almost surely (from the given initial configuration). However repulsing supermartingales can refute a.s. termination even for programs where the bound obtained by using Theorem 4 is ≥ 1. To show this we use another powerful tool of martingale theory: the *optional stopping theorem*.

Theorem 5 (Optional Stopping, [70, Theorem 10.10])**.** *Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, $\{X_i\}_{i=0}^{\infty}$ a supermartingale w.r.t. some filtration $\{\mathcal{F}_i\}_{i=0}^{\infty}$, and T a stopping time w.r.t the same filtration. Assume that $\mathbb{E}[T] < \infty$ and $\{X_i\}_{i=0}^{\infty}$ has c-bounded differences for some $c \in \mathbb{R}$. Then*

$$\mathbb{E}[X_0] \geq \mathbb{E}[X_T].$$

The optional stopping theorem guarantees that under given assumptions, the expected value of the supermartingale at the time of stopping (which can be, e.g. the time of program termination) is bounded from above by the expected initial value of the supermartingale. We can use the theorem to obtain the following.

Theorem 6. *Let C be a set of configurations of an* APP *P. Suppose that there exist $\epsilon > 0$, $c > 0$ and a linear ϵ-repulsing supermartingale η for C supported by some pure invariant I such that η has c-bounded differences. If $\eta(\ell_{init}, \mathbf{x}_{init}) < 0$, then under each scheduler σ it holds*

$$\mathbb{P}^{\sigma}(T^C < \infty) < 1.$$

Key proof idea. Theorem 4 shows that the existence of η implies the following: if a program execution reaches with positive probability a configuration $(\ell, \mathbf{x})$ such that $\eta(\ell, \mathbf{x})$ is below some sufficiently small negative number A (whose magnitude depends only on η and c), then the program does not terminate almost-surely. It thus suffices to prove that the program reaches such a configuration with positive probability under each scheduler σ. We define a stopping time T that returns a first point in time in which we reach either C or a configuration $(\ell, \mathbf{x})$ with $\eta(\ell, \mathbf{x}) \leq 2A$ and apply the optional stopping theorem on the ϵ-RepSM obtained from η. It can be proved that expectation of T is finite, so optional stopping theorem applies to T. Now to get a contradiction we assume that a configuration with η-value smaller than $2A$ is reached with probability 0. Then at time T the current configuration is almost-surely in C so the expected value of the supermartingale at time T is non-negative. But the optional stopping theorem forces this expectation to be bounded from above by the initial value of the RepSM, i.e. by $\eta(\ell_{init}, \mathbf{x}_{init})$, which is negative, a contradiction. □

Another important concept in the analysis of probabilistic programs is *finite termination* [19], sometimes also called *positive termination* [34]. A program is said to terminate finitely if its expected termination time is finite. Of course, when a program terminates with probability less than 1 it is not finitely terminating. However, there are programs that terminate almost-surely but the expected termination time is infinite. Indeed, consider a program modelling a symmetric 1-dimensional random walk with a boundary:

while $x \geq 0$ **do if prob**(0.5) **then** $x := x+1$ **else** $x := x-1$ **fi od**

From the theory of random walks it follows that for each positive initial value of x the program terminates almost-surely but its expected termination time is infinite. Even for such programs the positive termination can be refuted, this time by using 0-repulsing supermartingales.

Theorem 7. *Let C be a set of configurations of an* APP *P. Suppose that there exist $\epsilon \geq 0$, $c > 0$ and a linear ϵ-repulsing supermartingale η for C supported by some pure invariant I such that η has c-bounded differences. If $\eta(\ell_{init}, \mathbf{x}_{init}) < 0$, then under each scheduler σ it holds*

$$\mathbb{E}^{\sigma}(T^C) = \infty. \tag{6}$$

Proof. Let $\{X_i\}_{i=0}^{\infty}$ be the ϵ-repulsing supermartingale obtained from η using Lemma 2. Assume, for the sake of contradiction, that there exists a scheduler σ such that $\mathbb{E}^{\sigma}(T^C) < \infty$. Since $\{X_i\}_{i=0}^{\infty}$ has c-bounded differences, using the optional stopping theorem we get $\mathbb{E}^{\sigma}[X_0] \geq \mathbb{E}^{\sigma}[X_{T^C}]$. But from the definition of η we get $\mathbb{E}^{\sigma}[X_0] = \eta(\ell_{init}, \mathbf{x}_{init}) < 0$ and $\mathbb{E}^{\sigma}[X_{T^C}] \geq 0$, since η attains non-negative values inside C. Hence, we derived a contradiction $0 > 0$. □

Example 7. *For the 1-dimensional symmetric RW program pictured above (counter-example for finite-termination) it is easy to find a 0-LRepSM η for the set of terminal configurations supported by a pure invariant : say that η is equal to $-x-1$ in all locations but*

the terminal one, where it is equal to 0*. The supporting invariant is, e.g.* $x \geq -1$ *for all locations.*

6.3 Proving Almost-Sure Persistence

The applicability of repulsing submartingales extends beyond reachability properties. In some applications of probabilistic programs, such as modelling of complex reactive systems [17], it is customary to consider programs that are *not* terminating but continue to execute forever, e.g. because they model a system which should continuously respond to inputs from the environment (e.g. a thermostat [2]). One of the basic properties of such programs is *persistence* [17]. A set of configurations C is said to be *almost-surely persistent* if under each scheduler σ it holds with probability 1 that all but finitely many configurations along a run belong to C (or in other words, that we will eventually see only configurations from C). In [17] they presented a method of proving almost-sure persistence via so called geometric supermartingales. We present an alternative proof technique based on combination of ranking and repulsing supermartingales.

Theorem 8. *Let C be be a set of configurations of some* APP P*. Denote by $\neg C$ the set of all configurations of C that do not belong to C. Assume that there exist the following:*

1. *An $\epsilon > 0$, $c > 0$, and an ϵ-LRepSM η with c-bounded differences for the set $\neg C$ supported by some pure invariant I.*
2. *An $\epsilon > 0$, $K < 0$, and an ϵ-LRSM for the set $D = \{(\ell, \mathbf{x}) \mid \eta(\ell, \mathbf{x}) \leq K$ and $\mathbf{x} \models I(\ell)\}$.*

Then the set C is almost-surely persistent.

Key proof idea. Item 2. ensures that from any reachable configuration we eventually reach the set D with probability 1 (Theorem 1). Item 1. ensures, that each time we enter D the probability that we never return back to $\neg C$ is positive (Theorem 6). As a matter of fact, it can be shown that this probability is bounded away from zero by a number $p > 0$ which depends only on c, K and η, but not on a concrete configuration in which we enter D. Hence, the probability that we enter D and after that reach $\neg C$ again at least n times is at most p^n. For n going to ∞ this converges to 0, showing that the probability of infinitely often seeing a configuration from $\neg C$ is 0. □

Example 8. *As a simple example, consider the program*

$$\textbf{\textit{while}}\ true\ \textbf{\textit{do}}\ x := \textbf{\textit{sample}}(\mathrm{Uniform}(-2, 1))\ \textbf{\textit{od}}.$$

For any $n \in \mathbb{Z}$ let C_n be the set of configurations in which the value of x is at most n. For each such n we have, inside the loop, a $\frac{1}{4}$-repulsing supermartingale $x - n$ for $\neg C_n$, and we also have a $\frac{1}{4}$-ranking supermartingale $x - n + 1$ for the set $\{(\ell, \mathbf{x}) \mid x - n \leq -1\}$ (both supported by invariants that are true everywhere). Since both supermartingales have bounded differences, we get that each set C_n is persistent.

7. Computational Results

In this section we discuss computational aspects of our framework. Since synthesis of ϵ-ranking supermartingales supported by a linear predicate map was already addressed in the previous work [15, 19], we focus on algorithms related to those aspects of probabilistic reachability which are new, i.e. those related to stochastic invariants and repulsing supermartingales. Since our techniques are extensions of already known techniques for ranking supermartingales and invariant synthesis, we present only a high-level description.

The two main algorithmic problems that we consider are the following:

1. For a given APP P with a given pure invariant I and a linear predicate map PI compute a number p such that (PI, p) is a stochastic invariant.
2. For a given APP P and a set C of configurations, compute a linear predicate map PI and a number p such that (PI, p) is a stochastic invariant supporting some ϵ-LRSM for the set C.

We assume that the set C in item 2 above is given by some linear predicate map IC, i.e. it is a set of all $(\ell, \mathbf{x})$ such that $\mathbf{x} \models IC(\ell)$. This ensures that all the objects we work with are linear, which allows for a more efficient solution. In particular, the set of all terminal configurations can be easily given in this way, so point 2 also concerns obtaining stochastic invariants for proving high-probability termination.

We start with presenting an algorithm for item 1 above, and then show how an algorithm for item 2 can be obtained as a straightforward generalization of 1.

We aim to compute the bound p using Corollary 1, i.e. we want to compute an ϵ-LRepSM for the set $\neg PI$ with c-bounded-differences supported by I. Note that $\neg PI$ can also be expressed by a linear predicate map, and this LPM can be computed in polynomial time provided that PI is *polyhedral*, i.e. that each $PI(\ell)$ is a linear assertion (a conjunction of linear inequalities). We call a set of configurations polyhedral if it can be defined by a polyhedral LPM.

We adapt a well known constrained-based method for generating linear ranking functions and (non-stochastic) invariants in non-probabilistic programs [22, 24, 60], which was adapted for synthesizing ϵ-LRSMs in probabilistic programs [15, 19]. We briefly recall this approach and explain its adaptation. So suppose that we are given a program P, a polyhedral set C, and an LPM I, and we want to compute numbers $\epsilon > 0$, $c > 0$, and, for each location ℓ of $\mathcal{C}_P$, coefficients $b^\ell, a^\ell_1, \ldots, a^\ell_{|V|}$ such that the LEM η given by $\eta(\ell) = b^\ell + \sum_{i=1}^{|V|} a^\ell_i \cdot x_i$ is an ϵ-LRSM for C with c-bounded differences supported by I. Since LRSMs can be re-scaled by an arbitrary positive constant, we can assume that $\epsilon \geq 1$ and $c \geq 1$ (it always holds that $c \geq \epsilon$). We denote by U the set $\{b^\ell, a^\ell_1, \ldots, a^\ell_{|V|} \mid \ell \in L\} \cup \{c, \epsilon\}$. The algorithm of [19] constructs a system of linear inequalities $\mathbf{y} \cdot Z \geq \mathbf{d}$ (here Z is a matrix) that is *adjusted to P, C, and I*, which means that each term in each inequality of the system has one of the following forms:

- It is a variable with a name corresponding to some element of U.
- It is a term of the form $y \cdot z$, where y is a variable and z is a coefficient (i.e. a number) appearing in I (i.e. some inequality of I contains a term of the form $z \cdot x_i$ for some i).
- It is a term of the form $y \cdot z$, where y is a variable and z is a coefficient (i.e. a number) appearing in the LPM describing C.

Moreover, any solution of the system yields an ϵ-LRSM for C with c-bounded differences (by substituting the solution values for variables in U). If the system is unsolvable, then no such LRSM exists.

Intuitively, the construction of $\mathbf{y} \cdot Z \geq \mathbf{d}$ proceeds as follows: the algorithm first translates the conditions in Definition 9 into a conjunction of formulas of the form

$$\exists u_1 \ldots \exists u_m \forall x_1 \ldots \forall x_{|V|}\, \varphi \Rightarrow \psi, \tag{7}$$

where $u_1, \ldots, u_m$ are all the elements of U, φ is a linear assertion over variables $\{x_1, \ldots, x_{|V|}\}$ whose coefficients are numbers that appear as coefficients in I or in description of C, and

ψ is an arithmetic expression involving numbers and elements of $U \cup \{x_1, \ldots, x_{|V|}\}$ which is linear if the elements of the set $\{c, \epsilon, x_1, \ldots, x_{|V|}\}$ are taken as variables (in particular, its coefficients are independent of I and C). The algorithm then utilizes Farkas's lemma [31] to convert each such formula into an equivalent existentially quantified linear assertion, i.e. a system of linear inequalities adjusted to P, C, and I.

To obtain the required ϵ, c and LEM η it thus suffices to solve the linear system $\mathbf{y} \cdot Z \geq \mathbf{d}$. The construction of the system can be done in polynomial time provided that C is polyhedral. Note that in particular, the set of terminal configurations is polyhedral.

Now assume that instead of synthesizing an ϵ-LRSM for some set C we want to synthesize an ϵ-LRepSM for $C = \neg PI$. The point is that $\neg PI$ is again expressed by a linear predicate map and all the formulas arising from conditions in the definition of an ϵ-LRepSM again have the form (7). This is easy to see as almost all conditions in the definition of a LRepSM are the same as for LRSM. The only difference is in the non-negativity condition, where in LRSMs we require non-negativity outside C, while in LRepSMs inside C. But for all locations ℓ both these constraints are of the form "for all $\mathbf{x}$ satisfying a given linear predicate, $b^\ell + \sum_{i=1}^{|V|} a_i^\ell \cdot x_i \geq 0$", and thus can be transformed into a conjunction of formulae of the form (7). Hence, we can again reduce computing an ϵ-LRepSM for $\neg PI$ with c-bounded differences to solving a system of linear constraints, and the resulting system of linear constraints is again adjusted to P, C, and I. The method is complete in the sense that an ϵ-LRepSM for C with c-bounded differences exists if and only if the system of linear inequalities $\mathbf{y} \cdot Z \geq \mathbf{d}$ has a solution. This is proved in the same way as for LRSMs in [19].

As shown in Section 5, different LRepSMs can produce different upper bounds on the probability of reaching $\neg PI$. Theorem 4 shows that in order to get good bounds, it is vital that the computed LRepSM maximizes $|\eta(\ell_{init}, \mathbf{x}_{init})|/c$. Since this function is not linear in c and coefficients of η, we do not look for optimal η and c directly but instead we compute optimal LRepSMs η for multiple heuristically chosen values of c and then pick the one giving the best result.

The algorithm can be summarized as follows:

1. We fix $\epsilon = 1$ (this is w.l.o.g. as LRepSMs can be rescaled arbitrarily).
2. Using the constrained based-approach described above, we compute the minimal c such that there exists a 1-LRepSM for $\neg PI$ with c-bounded differences. We do this by constructing the system of linear inequalities $\mathbf{y} \cdot Z \geq \mathbf{d}$ and use linear programming to minimize c under the constraints given by the system. Denote by $c_{\min}$ the optimal c.
3. For some fixed number of iterations N we do the following: for each $0 \leq j \leq N$ we:
   - Compute a 1-LRepSM η_j for $\neg PI$ such that η_j has $(c_{\min} + j)$-bounded differences and minimizes $\eta_j(\ell_{init}, \mathbf{x}_{init})$ (i.e. maximizes $|\eta_j(\ell_{init}, \mathbf{x}_{init})|$). We do this again by constructing the system $\mathbf{y} \cdot Z \geq \mathbf{d}$ (we need to change just the terms referring to difference bound) and minimizing the objective function $\eta_j(\ell_{init}, \mathbf{x}_{init})$ using LP subject to the constraints of the system (since $(\ell_{init}, \mathbf{x}_{init})$ is given, the objective function is linear).
   - Apply Theorem 4 on η_j to get a bound p_j on reaching $\neg PI$.
4. We put $p = \min_{c_{\min} \leq j \leq c_{\min}+N} p_j$ and output (PI, p) as a stochastic invariant.

In our experiments we used $N = 1000$.

Now we turn to problem 2, i.e. computation of a stochastic invariant (PI, p) such that PI supports a linear ranking supermartingale for some set C. Since PI might have in principle unbounded size, we first have to fix a template for it, i.e. specify how many conjuncts and disjuncts can each $PI(\ell)$ consist of. This amounts to specifying a *symbolic* linear predicate map SI, where coefficients in each linear inequality of SI are not concrete numbers but abstract symbols. Note that symbolic LPMs can be also used to describe unknown sets of configurations.

Now take a look back on the above algorithms for computing ϵ-LRSM or ϵ-LRepSM for a given set C supported by a given LPM I. Previously, we used LPMs with concrete coefficients to encode both C and I on input, but we can supplant these with symbolic LPMs, effectively parametrizing the inputs C and I. Since the original algorithms produce systems adjusted to P, C and I, when the algorithms are run with a symbolic LPM instead of a concrete LPM on input, they produce a system of *quadratic* inequalities (as the coefficients in C and I are now unknown). It can be easily shown that there is a one-to-one correspondence between solutions of such a quadratic system and tuples $(\eta, C, I, \epsilon, c)$, where η is an ϵ-LRSM (or ϵ-LRepSM, depending on which of the two algorithms we use) with c-bounded differences for a set C supported by an LPM I such that C and I can be formed by instantiating the unknown parameters with concrete numbers.

We now construct two quadratic systems of inequalities: system $\mathcal{S}_1$, produced by the LRSM algorithm on input P, C (here C is a given set of configurations, i.e. a concrete set), and SI (which is a symbolic LPM encoding a template for the stochastic invariant we seek), and system $\mathcal{S}_2$, produced by the LRepSM algorithm on input P, $\neg SI$ (a set encoded by a symbolic LPM; a locationwise negation of the aforementioned template), and $True$, where $True$ is a (concrete) trivial invariant true in every location. Note that in the first system we treat SI as a symbolic representation of an LPM, while in the second one we treat it as a symbolic representation of a set of configurations to avoid. We then identify the variables in S_1 and S_2 referring to the same unknown coefficients in SI. Simultaneously solving both systems yields triples $(\eta, C, PI, \epsilon, c)$ and $(\eta', \neg PI, True, \epsilon', c')$, where η is an ϵ-LRSM with c-bounded differences for C supported by PI, and η' is an ϵ'-LRepSM with c'-bounded references for $\neg PI$ that can be used to bound the probability of violating PI. We note that checking the solvability of a quadratic systems of inequalities can be done in PSPACE by reduction to existential first-order theory of reals. Also note that instead of $True$ we can use any other pure invariant.

Theorem 9. *Existence of a LRepSM for a given set C can be decided in polynomial time provided that C is polyhedral. Existence of an LPM PI such that PI supports some LRSM for a given polyhedral set and at the same time $\neg PI$ admits an LRepSM can be reduced to existential first-order theory of reals and thus decided in PSPACE.*

8. Experimental Results

In this section we present some basic experimental results for our methods. The experimental results are basic and to verify that the new concepts we introduce are relevant. We consider three simple academic examples described below. In the corresponding pseudocode, we present invariants in square brackets.

1. *Example 1:* The first example is a one-dimensional random walk which initially moves with higher probability to the left as compared to the right. However, if x is incremented above 1000, the process starts drifting away from zero, so the program does

	Initial Configuration	PI Violation Probability Bound
Example 1	(i) $x := 10$, (ii) $x := 50$, (iii) $x := 100$	(i) $5.1 \cdot 10^{-5}$, (ii) $1.0 \cdot 10^{-4}$, (iii) $2.5 \cdot 10^{-4}$
Example 2	(i) $x, y := 1000, 10$, (ii) $x, y := 500, 40$, (iii) $x, y := 400, 50$	(i) $2.4 \cdot 10^{-11}$, (ii) $5.5 \cdot 10^{-4}$, (iii) $1.9 \cdot 10^{-2}$
Example 3	(i) $x, y, z := 100, 100, 100$, (ii) $x, y, z := 100, 150, 200$, (iii) $x, y, z := 300, 100, 150$	(i) $4.4 \cdot 10^{-17}$, (ii) $2.9 \cdot 10^{-9}$, (iii) $1.3 \cdot 10^{-7}$

Table 1. Experimental results

```
while x ≥ 0 do                    [x ≥ −2]
   if x ≤ 1000 then               [x ≥ 0]
      if prob(0.5) then           [x ≥ 0 and x ≤ 1000]
         x := x − 2               [x ≥ 0 and x ≤ 998]
      else
         x := x + 1               [x ≥ 0 and x ≤ 1001]
      fi
   else
      if prob(0.5) then           [x ≥ 1001]
         x := x − 1               [x ≥ 1000]
      else
         x := x + 2               [x ≥ 1002]
      fi
   fi
od

PI: [x ≤ 1000] at location 2
(first 'if'−branching),
'true' elsewhere.
```

Figure 5. Example 1.

```
while 1 ≤ y do                    [y ≥ 0]
   if prob(0.5) then              [y ≥ 1]
      if prob(0.75) then          [y ≥ 1]
         x := x + 1               [y ≥ 1]
      else
         x := x − 1               [y ≥ 1]
      fi
   else
      if prob(0.75) then          [y ≥ 1]
         y := y − 1               [y ≥ 0]
      else
         y := y + 1               [y ≥ 2]
      fi
   fi                             [y ≥ 0]
   while x ≤ 0 do
      x := 0                      [x ≤ 0 and y ≥ 0]
   od
od

PI: [x ≥ 1] at location 9
(entry of inner while−loop),
'true' elsewhere.
```

Figure 6. Example 2.

not terminate almost-surely. The details of the example along with invariants is given in Figure 5.

2. *Example 2:* In the second example, we have two variables x and y, and the program models a generalized 2-dimensional random walk. Variable x tends to drift away from zero while y tends to drift towards zero and thus towards satisfaction of the termination condition. However, if x hits zero, the program gets stuck in an infinite loop, so we want to show that the probability of this happening is small. The details of the example along with invariants is given in Figure 6.
3. *Example 3:* In the third example (Figure 7), we have three variables x, y, z. For various combinations, with high probability we either decrease both x and y, or z, and with low probability we either increase both x and y, or z. But the increments and decrements are not proportional, and this is indeed a 3-dimensional example. We note that the program in this example terminates almost-surely, but this does not simplify the computation of the probability bound for the given LPM PI.

We consider various initial configurations of the examples. For each example we obtain a probability threshold for a given LPM PI (and thus obtain a stochastic invariant). Our experimental results are shown in Table 1. In all the cases, our method creates a linear program which can be efficiently solved using any standard solver (such as lpsolve [7], CPlex [1]).

9. Related Work

Probabilistic programs. In the 70's and 80's, several semantic approaches for reasoning about probabilistic programs (including termination probabilities) were considered, most of them being based on probabilistic extensions of *dynamic logic* [61]. In [63], one such extension, PROB-DL, is applied to a restricted class of programs where there are no if-then-else branchings and no variable tests in loop guards (instead, loops are terminated according to a geometric distribution). A powerful probabilistic logic called **Pr**Dl was introduced in [33], allowing for first-order reasoning about events in the domain of computation and their effects on probabilities of assertions. The authors present an axiom system for *Pr*Dl that is complete relatively to the underlying domain-specific logic (which might be undecidable in general), which allows one to check the validity of program properties "directly, (though,. . . , in general, not effectively)" [33]. Decidable propositional fragments of probabilistic dynamic logic were studied in [32, 50], although as noted in [50], for practical verification purposes these would need to be extended with logic for reasoning about the computation domain. Moreover, none of the above approaches consider programs with non-determinism.

In the realm of probabilistic programs with non-determinism, the termination problems for probabilistic programs with discrete choices have been considered in [55, 56], but for probabilistic programs with infinite-state space and choices, only the qualitative problem has been studied. The qualitative problem of almost-sure termination has been considered in several works such as [9, 15, 18, 19, 34]. The termination for concurrent probabilistic programs under fairness was considered in [67]. A sound and complete characterization of almost-sure termination for countable state space was given in [40]. A sound and complete method for proving termination of finite-state programs was given in [30]. All previous works either consider discrete probabilistic choices or finite-state space, or for general probabilistic programs consider the qualitative problem of almost-sure termination. All works for almost-sure termination consider RSMs w.r.t. a given invariant. In contrast, in this work we consider stochastic invariants and the probabilistic termination problem, and

```
while x ≥ 0 and y ≥ 0 and z ≥ 0 do
          [x ≥ −1 and y ≥ −1 and z ≥ −1]
   if prob(0.9) then
          [x ≥ 0 and y ≥ 0 and z ≥ 0]
      if prob(0.5) then
          [x ≥ 0 and y ≥ 0 and z ≥ 0]
         x, y := x − 1, y − 1
          [x ≥ −1 and y ≥ −1 and z ≥ 0]
      else z := z − 1
          [x ≥ 0 and y ≥ 0 and z ≥ −1]
      fi
   else
      if prob(0.5) then
          [x ≥ 0 and y ≥ 0 and z ≥ 0]
         x, y := x + 0.1, y + 0.2
          [x ≥ 0 and y ≥ 0 and z ≥ 0]
      else z := z + 0.1
          [x ≥ 0 and y ≥ 0 and z ≥ 0]
      fi
   fi
od

PI: [x + y + z ≤ 1000] at location 1
(entry of outer loop),
'true' elsewhere.
```

Figure 7. Example 3.

our results are applicable to probabilistic programs with infinite-state space.

The use of martingales in probabilistic program analysis extends beyond termination properties. In [27] martingales are used to derive bounds on expected termination time of randomized algorithms. In [16] they introduce *expectation invariants* for single-loop probabilistic programs, which are statements about *expected value* of program expressions whose validity is invariant during the program execution. In contrast, our stochastic invariant approach reasons about the *probability* of a given assertion's validity.

There is also work on establishing a probability that a certain assertion holds. In [64] they consider approximating the probability of assertions using optimized simulation, under semantics assuming terminating while loops. A method for approximating a probability of assertion based on symbolic execution was given in [65], where they also assume almost-surely terminating programs. Several works considered approximating the behaviour of probabilistic programs by abstracting them into finite Markov chains or MDPs [41, 47]. On the other hand, our repulsing supermartingales do not need any abstraction or simulation techniques to work, although we conjecture that they could be fruitfully combined with abstraction approaches to "cut away" configurations that are unlikely to be reached and thus reduce the size of the abstractions. In [46], a Hoare-style calculus based on *weakest pre-expectations* is used to reason about probabilistic effects of terminating programs, with a practical application presented in [38]. The weakest pre-condition style of reasoning was also adapted for reasoning about expected running times of probabilistic programs [45].

Non-probabilistic programs. Termination analysis of non-probabilistic programs has received a lot of attention over the last decade, such as [10, 11, 22, 25, 54, 60, 68]. Most of these works consider various notions of ranking functions for termination. RSMs are a generalization of ranking functions, which has been studied for almost-sure termination. The extension of almost-sure termination to probabilistic termination needs new conceptual ideas and methods which we present in this work.

10. Conclusion and Future Work

We considered the basic quantitative question of probabilistic termination for probabilistic programs. We introduced stochastic invariants for probabilistic termination, and repulsing supermartingales as the new concept that allows us to analyse the problem of probabilistic termination. There are several directions for future work. The first one is to consider special cases of non-linear repulsing supermartingales (such as polynomial repulsing supermartingales), and study whether efficient algorithmic approaches can be developed for them as well. The second interesting direction is to consider practical approaches for the synthesis of stochastic invariants, as the theoretical results use the existential first order theory of the reals.

Acknowledgements

This research was partially supported by Austrian Science Fund (FWF) NFN Grant No S11407-N23 (RiSE/SHiNE), ERC Start grant (279307: Graph Games), and Vienna Science and Technology Fund (WWTF) through project ICT15-003. The research leading to these results has received funding from the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7/2007-2013) under REA grant agreement no [291734]. Đorđe Žikelić participated in this research during a research visit at IST Austria, funded by the OeAD Sonderstipendien, IST AUSTRIA programme awarded by the Austrian Agency for International Cooperation in Education and Research.

References

[1] IBM ILOG CPLEX Optimizer. http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/, 2010.

[2] A. Abate, J.-P. Katoen, J. Lygeros, and M. Prandini. Approximate Model Checking of Stochastic Hybrid Systems. *European Journal of Control*, 16(6):624–641, 2010. ISSN 0947-3580.

[3] M. F. Atig, A. Bouajjani, M. Emmi, and A. Lal. *Detecting Fair Nontermination in Multithreaded Programs*, pages 210–226. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-31424-7.

[4] K. Azuma. Weighted sums of certain dependent random variables. *Tohoku Mathematical Journal, Second Series*, 19(3):357–367, 1967.

[5] C. Baier and J.-P. Katoen. *Principles of model checking*. MIT Press, 2008. ISBN 978-0-262-02649-9.

[6] G. Barthe, T. Espitau, L. M. F. Fioriti, and J. Hsu. Synthesizing Probabilistic Invariants via Doob's Decomposition. In *Computer Aided Verification - 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part I*, pages 43–61, 2016.

[7] M. Berkelaar, K. Eikland, and P. Notebaert. lp_solve: Open source (Mixed-Integer) Linear Programming system. URL `http://sourceforge.net/projects/lpsolve/`.

[8] P. Billingsley. *Probability and Measure*. 1995.

[9] O. Bournez and F. Garnier. Proving Positive Almost-Sure Termination. In *RTA*, pages 323–337, 2005.

[10] A. R. Bradley, Z. Manna, and H. B. Sipma. The Polyranking Principle. In *ICALP*, pages 1349–1361, 2005.

[11] A. R. Bradley, Z. Manna, and H. B. Sipma. Linear Ranking with Reachability. In K. Etessami and S. K. Rajamani, editors, *Computer Aided Verification, 17th International Conference, CAV 2005, Edinburgh, Scotland, UK, July 6-10, 2005, Proceedings*, volume 3576 of

Lecture Notes in Computer Science, pages 491–504. Springer, 2005. ISBN 3-540-27231-3.

[12] T. Brázdil, V. Brožek, K. Etessami, and A. Kučera. Approximating the termination value of one-counter MDPs and stochastic games. *Inf. Comput.*, 222:121–138, 2013.

[13] T. Brázdil, S. Kiefer, and A. Kučera. Efficient Analysis of Probabilistic Programs with an Unbounded Counter. *J. ACM*, 61(6):41:1–41:35, Dec. 2014. ISSN 0004-5411.

[14] T. Brázdil, S. Kiefer, A. Kučera, P. Novotný, and J.-P. Katoen. Zero-Reachability in Probabilistic Multi-Counter Automata. In *Proceedings of LICS 2014*, 2014.

[15] A. Chakarov and S. Sankaranarayanan. Probabilistic Program Analysis with Martingales. In N. Sharygina and H. Veith, editors, *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, volume 8044 of *Lecture Notes in Computer Science*, pages 511–526. Springer, 2013. ISBN 978-3-642-39798-1.

[16] A. Chakarov and S. Sankaranarayanan. *Expectation Invariants for Probabilistic Program Loops as Fixed Points*, pages 85–100. Springer International Publishing, 2014. ISBN 978-3-319-10936-7.

[17] A. Chakarov, Y.-L. Voronin, and S. Sankaranarayanan. *Deductive Proofs of Almost Sure Persistence and Recurrence Properties*, pages 260–279. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016. ISBN 978-3-662-49674-9.

[18] K. Chatterjee, H. Fu, and A. K. Goharshady. Termination Analysis of Probabilistic Programs through Positivstellensatz's. *CoRR*, abs/1604.07169, 2016.

[19] K. Chatterjee, H. Fu, P. Novotný, and R. Hasheminezhad. Algorithmic analysis of qualitative and quantitative termination problems for affine probabilistic programs. In R. Bodík and R. Majumdar, editors, *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 - 22, 2016*, pages 327–342. ACM, 2016. ISBN 978-1-4503-3549-2.

[20] K. Chatterjee, P. Novotný, and Đ. Žikelić. Stochastic Invariants for Probabilistic Termination. *CoRR*, abs/1611.01063, 2016.

[21] H.-Y. Chen, B. Cook, C. Fuhs, K. Nimkar, and P. O'Hearn. *Proving Nontermination via Safety*, pages 156–171. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. ISBN 978-3-642-54862-8.

[22] M. Colón and H. Sipma. Synthesis of Linear Ranking Functions. In T. Margaria and W. Yi, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 7th International Conference, TACAS 2001 Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2001 Genova, Italy, April 2-6, 2001, Proceedings*, volume 2031 of *Lecture Notes in Computer Science*, pages 67–81. Springer, 2001. ISBN 3-540-41865-2.

[23] M. A. Colón and H. B. Sipma. *Practical Methods for Proving Program Termination*, pages 442–454. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002. ISBN 978-3-540-45657-5.

[24] M. A. Colón, S. Sankaranarayanan, and H. B. Sipma. Linear invariant generation using non-linear constraint solving. In *International Conference on Computer Aided Verification*, pages 420–432. Springer, 2003.

[25] B. Cook, A. See, and F. Zuleger. Ramsey vs. Lexicographic Termination Proving. In *TACAS*, pages 47–61, 2013.

[26] P. Cousot and R. Cousot. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In R. M. Graham, M. A. Harrison, and R. Sethi, editors, *Conference Record of the Fourth ACM Symposium on Principles of Programming Languages, Los Angeles, California, USA, January 1977*, pages 238–252. ACM, 1977.

[27] D. Dubhashi and A. Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. 2009.

[28] D. Dubhashi and A. Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, New York, NY, USA, 1st edition, 2009. ISBN 0521884276, 9780521884273.

[29] R. Durrett. *Probability: Theory and Examples (Second Edition)*. Duxbury Press, 1996.

[30] J. Esparza, A. Gaiser, and S. Kiefer. Proving Termination of Probabilistic Programs Using Patterns. In *CAV*, pages 123–138, 2012.

[31] J. Farkas. A Fourier-féle mechanikai elv alkalmazásai (Hungarian). *Mathematikaiés Természettudományi Értesitö*, 12:457–472, 1894.

[32] Y. A. Feldman. A decidable propositional dynamic logic with explicit probabilities. *Information and Control*, 63(1):11–38, 1984. ISSN 0019-9958.

[33] Y. A. Feldman and D. Harel. A probabilistic dynamic logic. In *Proceedings of the fourteenth annual ACM Symposium on Theory of computing*, pages 181–195. ACM, 1982.

[34] L. M. F. Fioriti and H. Hermanns. Probabilistic Termination: Soundness, Completeness, and Compositionality. In S. K. Rajamani and D. Walker, editors, *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, January 15-17, 2015*, pages 489–501. ACM, 2015. ISBN 978-1-4503-3300-9.

[35] R. W. Floyd. Assigning meanings to programs. *Mathematical Aspects of Computer Science*, 19:19–33, 1967.

[36] F. G. Foster. On the Stochastic Matrices Associated with Certain Queuing Processes. *The Annals of Mathematical Statistics*, 24(3):pp. 355–360, 1953.

[37] A. D. Gordon, T. A. Henzinger, A. V. Nori, and S. K. Rajamani. Probabilistic programming. In *Proceedings of the on Future of Software Engineering*, pages 167–181. ACM, 2014.

[38] F. Gretz, J.-P. Katoen, and A. McIver. Prinsys - On a Quest for Probabilistic Loop Invariants. In *Quantitative Evaluation of Systems - 10th International Conference, QEST 2013, Buenos Aires, Argentina, August 27-30, 2013. Proceedings*, pages 193–208, 2013.

[39] A. Gupta, T. A. Henzinger, R. Majumdar, A. Rybalchenko, and R.-G. Xu. Proving Non-termination. *SIGPLAN Not.*, 43(1):147–158, Jan. 2008. ISSN 0362-1340.

[40] S. Hart and M. Sharir. Concurrent Probabilistic Programs, Or: How to Schedule if You Must. *SIAM J. Comput.*, 14(4):991–1012, 1985.

[41] H. Hermanns, B. Wachter, and L. Zhang. Probabilistic CEGAR. In *CAV*, LNCS 5123, pages 162–175. Springer, 2008.

[42] H. Howard. *Dynamic Programming and Markov Processes*. MIT Press, 1960.

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

[44] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1):99–134, 1998.

[45] B. L. Kaminski, J.-P. Katoen, C. Matheja, and F. Olmedo. *Weakest Precondition Reasoning for Expected Run–Times of Probabilistic Programs*, pages 364–389. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016. ISBN 978-3-662-49498-1.

[46] J.-P. Katoen, A. McIver, L. Meinicke, and C. C. Morgan. Linear-Invariant Generation for Probabilistic Programs: - Automated Support for Proof-Based Methods. In *SAS*, volume LNCS 6337, Springer, pages 390–406, 2010.

[47] M. Kattenbelt, M. Kwiatkowska, G. Norman, and D. Parker. Abstraction refinement for probabilistic software. In *International Workshop on Verification, Model Checking, and Abstract Interpretation*, pages 182–197. Springer, 2009.

[48] J. Kemeny, J. Snell, and A. Knapp. *Denumerable Markov Chains*. D. Van Nostrand Company, 1966.

[49] D. Kozen. Semantics of Probabilistic Programs. *Journal of Computer and System Sciences*, 22(3):328–350, 1981. ISSN 0022-0000.

[50] D. Kozen. A Probabilistic PDL. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, STOC '83, pages 291–297, New York, NY, USA, 1983. ACM. ISBN 0-89791-099-0.

[51] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas. Temporal-Logic-Based Reactive Mission and Motion Planning. *IEEE Transactions on Robotics*, 25(6):1370–1381, 2009.

[52] M. Z. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of Probabilistic Real-Time Systems. In *CAV*, LNCS 6806, pages 585–591, 2011.

[53] D. Larraz, K. Nimkar, A. Oliveras, E. Rodríguez-Carbonell, and A. Rubio. *Proving Non-termination Using Max-SMT*, pages 779–796. Springer International Publishing, Cham, 2014. ISBN 978-3-319-08867-9.

[54] C. S. Lee, N. D. Jones, and A. M. Ben-Amram. The size-change principle for program termination. In *POPL*, pages 81–92, 2001.

[55] A. McIver and C. Morgan. Developing and Reasoning About Probabilistic Programs in *pGCL*. In *PSSE*, pages 123–155, 2004.

[56] A. McIver and C. Morgan. *Abstraction, Refinement and Proof for Probabilistic Systems*. Monographs in Computer Science. Springer, 2005.

[57] D. Monniaux. An Abstract Analysis of the Probabilistic Termination of Programs. In P. Cousot, editor, *Static Analysis, 8th International Symposium, SAS 2001, Paris, France, July 16-18, 2001, Proceedings*, volume 2126 of *Lecture Notes in Computer Science*, pages 111–126. Springer, 2001. ISBN 3-540-42314-1.

[58] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, New York, NY, USA, 1995. ISBN 0-521-47465-5, 9780521474658.

[59] A. Paz. *Introduction to probabilistic automata (Computer science and applied mathematics)*. Academic Press, 1971.

[60] A. Podelski and A. Rybalchenko. A Complete Method for the Synthesis of Linear Ranking Functions. In B. Steffen and G. Levi, editors, *Verification, Model Checking, and Abstract Interpretation, 5th International Conference, VMCAI 2004, Venice, January 11-13, 2004, Proceedings*, volume 2937 of *Lecture Notes in Computer Science*, pages 239–251. Springer, 2004. ISBN 3-540-20803-8.

[61] V. R. Pratt. Semantical consideration on floyo-hoare logic. In *Foundations of Computer Science, 1976., 17th Annual Symposium on*, pages 109–121, Oct 1976.

[62] M. Rabin. Probabilistic automata. *Information and Control*, 6:230–245, 1963.

[63] J. H. Reif. Logics for Probabilistic Programming (Extended Abstract). In *Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing*, STOC '80, pages 8–13, New York, NY, USA, 1980. ACM. ISBN 0-89791-017-6.

[64] A. Sampson, P. Panchekha, T. Mytkowicz, K. S. McKinley, D. Grossman, and L. Ceze. Expressing and verifying probabilistic assertions. In M. F. P. O'Boyle and K. Pingali, editors, *ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '14, Edinburgh, United Kingdom - June 09 - 11, 2014*, page 14. ACM, 2014. ISBN 978-1-4503-2784-8.

[65] S. Sankaranarayanan, A. Chakarov, and S. Gulwani. Static Analysis for Probabilistic Programs: Inferring Whole Program Properties from Finitely Many Paths. In *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '13, pages 447–458, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2014-6.

[66] S. Sankaranarayanan, A. Chakarov, and S. Gulwani. Static analysis for probabilistic programs: inferring whole program properties from finitely many paths. In *PLDI*, pages 447–458, 2013.

[67] M. Sharir, A. Pnueli, and S. Hart. Verification of Probabilistic Programs. *SIAM J. Comput.*, 13(2):292–314, 1984.

[68] K. Sohn and A. V. Gelder. Termination Detection in Logic Programs using Argument Sizes. In D. J. Rosenkrantz, editor, *Proceedings of the Tenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 29-31, 1991, Denver, Colorado, USA*, pages 216–226. ACM Press, 1991. ISBN 0-89791-430-9.

[69] H. Velroyen and P. Rümmer. *Non-termination Checking for Imperative Programs*, pages 154–170. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-79124-9.

[70] D. Williams. *Probability with Martingales*. 1991.