Multi-Dimensional Probabilistic Ranking (Provisional Title)

ANONYMOUS AUTHOR(S)

ACM Reference format:

Anonymous Author(s). 2017. Multi-Dimensional Probabilistic Ranking (Provisional Title). 1, 1, Article 1 (May 2017), 11 pages.

https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 PRELIMINARIES

1.1 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.

XXXX-XXXX/2017/5-ART1 $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

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 $\mathcal{V}(\varphi) = \{x_1, \ldots, x_{n(\varphi)}\}$ 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 $[\![\varphi]\!]^{\lhd} = [\![\varphi]\!] \cap \mathbb{R}^{n(\varphi)}$.

1.2 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(**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 1. We refer to this class of affine probabilistic programs as Apps.

1.3 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* of measurable sets 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 [2, 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 [2]. 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 [4, 7, 8]. To define this process, we first define so called *probabilistic control flow graphs* [5].

Definition 1.1. A *probabilistic control flow graph (pCFG)* is a tuple $C = (L, V, \ell_{init}, \mathbf{x}_{init}, \mapsto, U, Pr, G)$, where

- L is a finite set of *locations* partitioned into four pairwise disjoint subsets L_{NB}, L_{PB}, L_D, and L_A of non-deterministic branching, probabilistic branching, deterministic, and assignment 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*;
- $\mapsto \subseteq L \times L$ is a transition relation'
- U is a function assigning to each transition outgoing from an assignment location a tuple (i, u), where $1 \leq i \leq |V|$ 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 $R \subseteq \mathbb{R}$.
- $Pr = \{Pr_\ell\}_{\ell \in L_{PB}}$ 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 $G(\tau_1) \vee \cdots \vee G(\tau_k) \equiv \mathit{true}$ and $G(\tau_i) \wedge G(\tau_j) \equiv \mathit{false}$ 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 $\mathit{supp}(d) \subseteq [\![\varphi_d]\!]^{\blacktriangleleft}$. Finally, we assume that each assignment location has at most (and thus exactly) one outgoing transition.

The translation from probabilistic programs to the corresponding pCFG is standard [6], and the details are presented in the supplementary material.

Configurations. A *configuration* of a pCFG C is a tuple $(\ell, \mathbf{x})$, where ℓ is a location of 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.

Executions and reachable configurations. We say that a configuration $(\ell', \mathbf{x}')$ is a *successor* of a configuration $(\ell, \mathbf{x})$ if there is a transition $\tau = (\ell, \ell')$ enabled in $(\ell, \mathbf{x})$ and $\mathbf{x}'$ satisfies the following:

- if ℓ is not an assignment location, then $\mathbf{x}' = \mathbf{x}$;

[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.

- if ℓ is an assignment location with $U(\tau) = (j, u)$, then $\mathbf{x}_{i+1} = \mathbf{x}_i(j \leftarrow a)$ where a satisfies one of the following depending on the type of u:
  - if u is an affine function, then $a = u(\mathbf{x})$;
  - if u is an integrable[2] distribution d, then $a \in \mathit{supp}(d)$;
  - if u is a set, then a is some element of u.

A *finite path* (or *execution fragment*) of length k in C is a finite sequence of configurations $(\ell_0, \mathbf{x}_0) \cdots (\ell_k, \mathbf{x}_k)$ such that for each $0 \leq i < k$ the configuration $(\ell_{i+1}, \mathbf{x}_{i+1})$ is a successor of $(\ell_i, \mathbf{x}_i)$. A *run* (or *execution*) in 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 $(\ell, \mathbf{x})$. We denote by $\mathit{Conf}_C, \mathit{Fpath}_C$ and Run_C the sets of all configurations, finite paths and runs in C, respectively, dropping the index C when known from the context.

Non-determinism and Schedulers. Due to the presence of non-determinism and probabilistic choices, a pCFG C may represent a multitude of possible behaviours. The probabilistic behaviour of 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 C needs to be resolved. This is achieved using the standard notion of a *scheduler*.

There are two sources of non-determinism in our programs: one in branching and one in assignments. We call a location ℓ non-deterministic if ℓ is a non-deterministic branching location or if ℓ is an assignment location with a non-deterministic assignment (i.e., the only transition τ outgoing from ℓ has $U(\tau) = (j, u)$ where u is a set). A configuration $(\ell, \mathbf{x})$ is non-deterministic if ℓ is non-deterministic.

Definition 1.2 (Schedulers). A scheduler in a pCFG C is a function σ assigning to every finite path that ends in a non-deterministic configuration $(\ell, \mathbf{x})$ a probability distribution on successor configurations of $(\ell, \mathbf{x})$.

Measurable schedulers. Note that schedulers can be viewed as partial functions from the set *Fpath* to the set of probability distributions over the set *Conf*. Since we deal with programs operating over real-valued variables, both *Fpath* and *Conf* can be uncountable sets. Hence, we impose an additional *measurability* condition on schedulers, so as to ensure that the semantics of probabilistic non-deterministic programs is defined in a mathematically sound way. First we need to clarify what are measurable sets of configurations and histories. We define a sigma-algebra $\mathcal{F}_{\mathit{Conf}}$ of measurable sets of configurations to be the sigma-algebra over *Conf* generated[3] by all sets of the form $\{\ell\} \times B$, where ℓ is a location of C and B is a Borel-measurable subset of $\mathbb{R}^{|V|}$. Next, the set of finite paths *Fpath* can be viewed as a subset of $\mathit{Conf} \cup \mathit{Conf} \times \mathit{Conf} \cup \mathit{Conf} \times \mathit{Conf} \times \mathit{Conf} \cup \ldots$. Hence, we define the sigma-algebra $\mathcal{H}$ of measurable sets of finite paths to be the sigma algebra generated by all sets of the form $Z_1 \times Z_2 \times \cdots \times Z_k \subseteq \mathit{Fpath}$ such that $k \in \mathbb{N}$ and $Z_i \in \mathcal{F}_{\mathit{Conf}}$ for all $1 \leq i \leq k$. Now we can define the measurability of schedulers. Recall that for each finite path π ending in a non-deterministic configuration we have that $\sigma(\pi)$ is a probability distribution on *Conf*. For each measurable set of configurations Z we denote by $\sigma(\pi)(Z)$ the probability that the random

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

[3] In general, it is known that for each set Ω and each collection of its subsets $F \subseteq 2^\Omega$ there exists at least one sigma-algebra $\mathcal{F}$ s.t. $F \subseteq \mathcal{F}$ and the intersection of all such sigma-algebras is again a sigma algebra – so called sigma-algebra generated by F [2].

draw from distribution $\sigma(\pi)$ selects an element of Z. We say that a scheduler σ is measurable if for each $Z \in \mathcal{F}_{Conf}$ and each $p \in [0,1]$ the set $\{\pi \in Fpath \mid \sigma(\pi)(Z) \leq p\}$ belongs to $\mathcal{H}$, i.e., it is a measurable set of paths.

While the definition of a measurable scheduler might seem somewhat technical, it is natural from measure-theoretic point of view and analogous definitions inevitably emerge in works dealing with systems that exhibit both probabilistic and non-deterministic behaviour over a continuous state space [9, 10]. In particular, if all the variables in a program range over a discrete set (such as the integers), then each scheduler in the associated pCFG is measurable.

Stochastic process. A pCFG C together with a scheduler σ define 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 $\pi_i = (\ell_0, \mathbf{x}_0)(\ell_1, \mathbf{x}_1)\cdots(\ell_i, \mathbf{x}_i)$ has already been produced. Then a successor configuration $(\ell_{i+1}, \mathbf{x}_{i+1})$ is chosen as follows:

- If ℓ_i is a non-deterministic location, then $(\ell_{i+1}, \mathbf{x}_{i+1})$ is sampled according to scheduler σ, i.e. from the distribution $\sigma(\pi_i)$.
- If ℓ_i is an assignment location (but not a non-deterministic one) with, there is exactly one transition $\tau = (\ell_i, \ell')$ outgoing from it and we put $\ell_{i+1} = \ell'$. Denoting $U(\tau) = (j, u)$, the vector $\mathbf{x}_{i+1}$ is then defined as $\mathbf{x}_{i+1} = \mathbf{x}_i(j \leftarrow a)$ where a is chosen depending on u:
  - 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.
- In all other cases we have $\mathbf{x}_{i+1} = \mathbf{x}_i$, and ℓ_{i+1} is determined as follows:
  - If ℓ_i is a probabilistic branching location, then a transition (ℓ_i, ℓ') is sampled from Pr_{ℓ_i} and we put $\ell_{i+1} = \ell'$;
  - If ℓ_i is deterministic, then there is exactly one transition (ℓ_i, ℓ') enabled in $(\ell_i, \mathbf{x}_i)$, in which case we put $\ell_{i+1} = \ell'$.

The above intuitive explanation can be formalized by showing that each pCFG 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 C, and a stochastic process $C^\sigma = \{\mathbf{C}_i^\sigma\}_{i=0}^\infty$ in this space such that for each run $\varrho \in \Omega$ we have that $\mathbf{C}_i^\sigma(\varrho)$ is the i-th configuration on 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 $i \geq 0$, are $\mathcal{R}$-measurable (i.e., for each $\mathbf{C}_i^\sigma$ and each measurable set of configurations $Z \in \mathcal{F}_{Conf}$ it holds $\{\omega \mid \mathbf{C}_i^\sigma(\omega) \in Z\} \in \mathcal{R}$). Equivalently, $\mathcal{R}$ can be defined as a sigma algebra generated by all set of runs of the form $F \times Conf^\infty$, where $F \in \mathcal{H}$ is a measurable set of finite paths. 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 $\mathbb{P}^\sigma$ proceeds via the standard *cylinder construction* [1, Theorem 2.7.2] and is somewhat technical, hence we omit it. We denote by $\mathbb{E}^\sigma$ the expectation operator in probability space $(\Omega, \mathcal{R}, \mathbb{P}^\sigma)$.

1.4 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 C_P. We say that a run ϱ of 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 C_P)

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

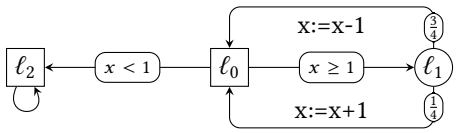

Fig. 1. An App modelling an asymmetric 1-D random walk and the associated pCFG. Probabilistic locations are depicted by circles, with probabilities given on outgoing transitions. Transitions are labelled by their effects. Location ℓ_0 is initial and ℓ_2 is terminal.

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 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:

(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 $\mathbb{P}^\sigma(\mathit{Term} < \infty) = 1$. 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 $\mathbb{P}^\sigma(\mathit{Term} < \infty) \geq b$).

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.

2 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.

2.1 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 2.1 (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 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 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 $\mathbf{x}_n \models I(\ell)$.

2.2 Supermartingales

(Super)martingales, are a standard tool of probability theory apt for analyzing probabilistic objects arising in computer science, from automata-based models [3] to general probabilistic programs [4**?** –7].

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 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 2.2 (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 2.3 (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 2.4 (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 *$\epsilon$-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 2.5 (Linear Expression Map). A *linear expression map (LEM)* for an App P is a function η assigning to each program location ℓ of 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 [4]. 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 2.6 (Pre-Expectation). Let P be an App such that $C_P = (L, V, \ell_{init}, \mathbf{x}_{init}, \mapsto, Pr, G)$ 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 2.7. (Linear Ranking Supermartingale) Let P be an App such that $C_P = (L, V, \ell_{init}, \mathbf{x}_{init}, \mapsto, Pr, G)$, let I be a linear predicate map and let $Z \subseteq L \times \mathbb{R}^{|V|}$ be some set of configurations. A linear ϵ-ranking supermartingale (ϵ-LRSM) for Z supported by I is a linear expression map η for P such that for all configurations $(\ell, \mathbf{x})$ of C_P with $(\ell, \mathbf{x}) \notin Z$ 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 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 2.8 ([6, 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 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*

```
x := 30, y := 20
while y ≥ 0 do
    x := x+sample(Uniform[−1/4, 1])
    y := y+sample(Uniform[−1, 1/4])
    while x ≤ 0 do skip od
od
```

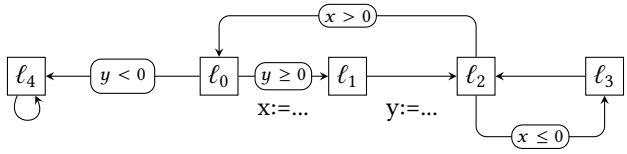

Fig. 2. A program with infinitely many reachable configurations which terminates with high probability, but not almost surely, together with a sketch of its pCFG.

(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 2.9. Consider the program in Figure 2. 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 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 2.9.

REFERENCES

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

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

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

[4] Aleksandar Chakarov and Sriram Sankaranarayanan. 2013. Probabilistic Program Analysis with Martingales. In *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings (Lecture Notes in Computer Science)*, Natasha Sharygina and Helmut Veith (Eds.), Vol. 8044. Springer, 511–526. https://doi.org/10.1007/978-3-642-39799-8_34

[5] Krishnendu Chatterjee, Hongfei Fu, and Amir Kafshdar Goharshady. 2016. Termination Analysis of Probabilistic Programs through Positivstellensatz's. *CoRR* abs/1604.07169 (2016). http://arxiv.org/abs/1604.07169

[6] Krishnendu Chatterjee, Hongfei Fu, Petr Novotný, and Rouzbeh Hasheminezhad. 2016. Algorithmic analysis of qualitative and quantitative termination problems for affine probabilistic programs. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 - 22, 2016*, Rastislav Bodík and Rupak Majumdar (Eds.). ACM, 327–342. https://doi.org/10.1145/2837614.2837639

[7] Luis María Ferrer Fioriti and Holger Hermanns. 2015. Probabilistic Termination: Soundness, Completeness, and Compositionality. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, January 15-17, 2015*, Sriram K. Rajamani and David Walker (Eds.). ACM, 489–501. https://doi.org/10.1145/2676726.2677001

[8] Dexter Kozen. 1981. Semantics of Probabilistic Programs. *J. Comput. System Sci.* 22, 3 (1981), 328–350. https://doi.org/10.1016/0022-0000(81)90036-2

[9] M. Neuhäußer, M. Stoelinga, and J.-P. Katoen. 2009. Delayed Nondeterminism in Continuous-Time Markov Decision Processes. In *Proceedings of FoSSaCS 2009*, Vol. 5504. 364–379.

[10] Martin R Neuhäußer and Joost-Pieter Katoen. 2007. Bisimulation and logical preservation for continuous-time Markov decision processes. In *International Conference on Concurrency Theory*. Springer, 412–427.