Towards Algebraic Analysis of Probabilistic Programs

Luisa Collodi, Michele Boreale and Alessandro Pompa Di Gregorio

Università di Firenze, Dipartimento di Statistica, Informatica, Applicazioni "G. Parenti", V.le Morgagni 65, 50134, Italy.

Abstract

We report on ongoing research efforts aimed at algebraic symbolic analysis of running time in probabilistic programs. We represent (sub-)probability distributions via generating functions and interpret loops as fixed-points in metric spaces.

Keywords

Probabilistic programming, generating functions, metric spaces, Banach theorem

1. Introduction

Probabilistic Programming (PP) [24, 4] is concerned with formal tools to define and analyze probabilistic models programmatically. Applications of PP include the modeling of randomized algorithms, probabilistic graphical models such as Bayesian networks, cognitive and decision-making models, and security protocols [26, 30, 35, 1, 32, 27]. Analyzing probabilistic programs is intrinsically challenging: even in the restricted setting where only a coin-flipping primitive is available, computing the expected value of a variable at program termination is as hard as solving the universal halting problem, while computing higher moments (such as variances) is provably harder; see [28] and references therein.

In this communication, we report on ongoing research efforts aimed at symbolic or exact, as opposed to simulation-based [37, 16], analysis of probabilistic programs, in the spirit of works such as [4, 28, 6, 38, 29, 5]. Our focus is on termination and running time analysis [28, 6, 38]. We adopt a denotational perspective in the spirit of Kozen [31]: we interpret programs as transformers of *functionals* — maps from initial states to (sub)-probability distributions on running time, represented via generating functions [8, 28, 29] (Section 2.1).

Our approach departs from previous work in a few key respects. In particular, we work in a metric-space setting (over generating functions and functionals), where fixed points needed to interpret loops are shown to exist and be unique leveraging the Banach fixed-point theorem [2] (Section 2.2). Compared to a traditional order-theoretic framework, based on the Knaster–Tarski or Kleene fixed-point theorems [40], our approach appears to be more streamlined. In particular, the uniqueness and convergence rate guaranteed by Banach theorem suggest novel promising approximation and algebraic techniques for symbolic analysis, which we will illustrate through a simple example (Section 3).

Related and Further Work The semantics of probabilistic programs, starting with the seminal work of Kozen [31] and the monograph by McIver and Morgan [33], is by now a well-established topic. Other semantics for discrete probabilistic while-programs are discussed in the aforementioned references; see also [33, 21, 25, 7, 5, 39, 22, 36]. A generating function approach has been considered in [28], and analysis techniques have been put forward for a class of programs with rational generating functions. Here we target a broader class of programs, with possibly non-rational (algebraic) generating functions. A different approach based on invariants and theorem proving is discussed in [20, 6]. The investigations discussed here are part of a broader research agenda, aimed at developing flexible, compositional formal methods applicable across diverse, probability-related domains, including dynamical systems with safety-related aspects [10, 9, 11, 12, 13, 15], information leakage and security [19, 18], distributed systems with notions of failure and recovery [14], randomized model counting and testing [17, 34].

ICTCS 2025: Italian Conference on Theoretical Computer Science, September 10–12, 2025, Pescara, Italy

luisa.collodi@unifi.it (L. Collodi); michele.boreale@unifi.it (M. Boreale); alessandro.pompa@edu.unifi.it (A. Pompa Di Gregorio)

2. Semantics of Probabilistic Programs

2.1. Preliminaries

Let sD denote the set of *sub-probability generating functions (gf)* in the variable z. These are functions of the complex variable z of the form $f(z) = \sum_{n=0}^{\infty} a_n z^n$ such that $a_n \in [0,1]$ and $f(1) = \sum_{n=0}^{\infty} a_n \leq 1$. If equality holds, we say f is a probability generating function, i.e. a distribution on $\mathbb{N}$. For $n \geq 0$, we let $[z^n]f(z) := a_n$, the n-th coefficient of f. Note that $f(1) = \sum_{j\geq 0} a_j$ is the total probability mass of the (sub-)probability distribution; $f'(1) = \frac{\mathrm{d}}{\mathrm{d}z} f(z)_{|z=1} = \sum_{j\geq 1} j \cdot a_j$ is the expected value of the (sub-)distribution. Higher-order moments can be computed similarly; see [41]. sD can be made into a complete metric space by introducing the following (ultra-metric) distance. Given $f(z) = \sum_{n=0}^{\infty} a_n z^n$ and $g(z) = \sum_{n=0}^{\infty} b_n z^n$, define the distance:

$$d_{\mathsf{sD}}(f,g) := 2^{-\min\{i\geq 0 : a_i \neq b_i\}} \tag{1}$$

with the convention that $\min \emptyset = +\infty$ and $2^{-\infty} = 0$. Note that, for any integer $k \geq 0$, $d_{\mathsf{sD}}(f,g) \leq 2^{-k}$ iff $f(z) - g(z) = O(z^k)$, convening that $O(z^\infty) = 0$. It is a standard fact that $(\mathsf{sD}, d_{\mathsf{sD}})$ forms a *complete*[1] metric space. Fix an integer $m \geq 1$ — representing the number of program variables. We let $\mathsf{F}_m := \{G : \mathbb{Z}^m \to \mathsf{sD}\}$ be the set of *functionals* from $\mathbb{Z}^m$ to sD. In what follows, we will let $x = (x_1, ..., x_m)$ range over $\mathbb{Z}^m$. We lift the complete metric space structure of sD to F_m by defining the following distance for $G_1, G_2 \in \mathsf{F}$:

$$d_{\mathsf{F}}(G_1, G_2) := \sup_{x\in\mathbb{Z}^m} d_{\mathsf{sD}}(G_1(x), G_2(x)). \tag{2}$$

With this notion of distance, $(\mathsf{F}_m, d_{\mathsf{F}})$ is in turn a complete metric space. We shall denote by $\mathbf{0}$ (resp. $\mathbf{1}$) the functional that assigns the constant gf $f(z) = 0$ (resp. constant $f(z) = 1$) to any $x \in \mathbb{Z}^m$. The convex combination of two or more functionals, say (for $p \in [0,1]$), $pG_1 + (1-p)G_2 := \lambda x.\,(\,pG_1(x) + (1-p)G_2(x)\,)$, is still a functional in F_m. Likewise, for any predicate $\phi : \mathbb{Z}^m \to \{0,1\}$, the linear combination, $(\neg\phi)\cdot G_1 + \phi \cdot G_2 := \lambda x.\,(\,(1-\phi(x))\cdot G_1(x) + \phi(x)\cdot G_2(x)\,)$, is still in F_m. In what follows, we shall omit the index m and write just F whenever m is understood from the context.

2.2. Syntax and Fixed Point Semantics

Probabilistic programs [4, 24] extend ordinary programs in the sense that they allow probabilistic choice between blocks of instructions. We consider here the probabilistic Guarded Command Language pGCL of [33], which focuses on discrete probability distributions, represented by guarded choices. We do not consider nondeterminism at this stage. We also do not consider observe() statements, which can be encoded using fail variables like in [38]. Programs are built from integer variables $x_1, ..., x_m$, for a fixed $m \geq 1$. We let ϕ range over *predicates*, that is functions $\mathbb{Z}^m \to \{0,1\}$, and h over functions $\mathbb{Z}^m \to \mathbb{Z}$. We let P range over the set of programs.

$$P ::= \texttt{skip} \mid x_i :\!= h \mid p_1 : P_1 \,\square\cdots\square\, p_n : P_n \mid \texttt{if}\ \phi\ P_1\ \texttt{else}\ P_2 \mid P_1; P_2 \mid \texttt{while}\ \phi\ P$$

Here, $p_1, \ldots, p_n$ represents a probability distribution, hence we require $\sum_{i=1}^{n} p_i = 1$ and $p_i \geq 0\ \forall i \in \{1, \ldots, n\}$. In the sequel, we will let $x = (x_1, \ldots, x_m)$ denote the entire variables tuple and sometimes, slightly abusing notation, also a tuple in $\mathbb{Z}^m$.

Our intent can be summarized as: given a probabilistic while-loop P, gather information about its runtime through iteration counting. Intuitively, we can think of a hidden counter in our program that gets incremented each time an iteration of a while-loop is executed. We are interested in the *final value* of this counter, say C. Since P is probabilistic, and may or may not terminate, for each given input, C represents a *sub*-probability distribution on the set of naturals 0, 1, 2,... Natural questions about C are: What is the probability of termination? What are the expected value of C and its standard deviation,

[1] Every Cauchy sequence in the space has a limit.

given termination? Or even: what is the probability that C deviates from its expected value by more than a given amount?

In what follows, we define a semantics of programs, in which $[\![P]\!]$ is represented by a functional transformer. Formally, for each probabilistic program P, $[\![P]\!]$ takes each $G \in \mathsf{F}$ into a $G' \in \mathsf{F}$, that is $[\![P]\!] : \mathsf{F} \to \mathsf{F}$. In what follows, we shall make use of the operations over F introduced in the previous section, as well as of the following abbreviations, where $h : \mathbb{Z}^m \to \mathbb{Z}$.

$$G[h(x)/x_i] := \lambda x.\, G(x_1, ..., x_{i-1}, h(x), x_{i+1}, ...x_m) \qquad\qquad z \cdot G := \lambda x.\, z \cdot (G(x)).$$

In the semantic clauses one should think of G as the 'continuation' of P — what is executed after P terminates. In the clause for the while loop, the formal variable z 'marks', in the parlance of generating functions, the number of loop iterations; $\mathrm{fix}(\cdot)$ denotes the fixed-point operator.

- **Skip:** $[\![\texttt{skip}]\!](G) = G$.
- **Assignment:** $[\![x_i := h]\!](G) = G[h(x)/x_i]$.
- **Probabilistic Choice:** $[\![p_1 : P_1 \,\square \cdots \square\, p_n : P_n]\!](G) = \sum_{i=1}^{n} p_i \cdot [\![P_i]\!](G)$.
- **Conditional:** $[\![\texttt{if}\ \phi\ P_1\ \texttt{else}\ P_2]\!](G) = \phi \cdot [\![P_1]\!](G) + (\neg\phi) \cdot [\![P_2]\!](G)$.
- **Composition:** $[\![P_1; P_2]\!](G) = [\![P_1]\!]([\![P_2]\!](G))$.
- **While Loop:** $[\![\texttt{while}\,\phi\,P]\!](G) = \mathrm{fix}(\Psi_{\phi,P,G})$, where $\Psi_{\phi,P,G} : \mathsf{F} \to \mathsf{F}$ is the transformer defined as follows: for each $F \in \mathsf{F}$

$$\Psi_{\phi,P,G}(F) := (\neg\phi) \cdot G + \phi \cdot z \cdot [\![P]\!](F)\,. \tag{3}$$

The last clause of the above definition is justified by the following theorem. Recall that, given a metric space (Y, d), a function $\gamma : Y \to Y$ is a *contraction* on Y if there is a constant $c \in [0, 1)$ s.t. for all $y_1, y_2 \in Y$ we have $d(\gamma(y_1), \gamma(y_2)) \leq c \cdot d(y_1, y_2)$. The Banach fixed point theorem states that if (Y, d) is a nonempty, complete metric space and γ is a contraction on Y, then there is a *unique* $y^* \in Y$ s.t. $\gamma(y^*) = y^*$. The unique y^* is denoted by $\mathrm{fix}(\gamma)$.

Theorem 1 (fixed-points)**.** *For each program P, predicate ϕ and $G \in \mathsf{F}$, the functional $\Psi_{\phi,P,G}$ defined in* (3) *is a contraction in the complete metric space $(\mathsf{F}, d_{\mathsf{F}})$ with constant $c \leq \frac{1}{2}$. As a consequence of the Banach fixed point theorem, $\Psi_{\phi,P,G}$ has a unique fixed point $\mathrm{fix}(\Psi_{\phi,P,G}) \in \mathsf{F}$.*

The *semantics* of P is a functional $\{\![P]\!\} \in \mathsf{F}$, defined as

$$\{\![P]\!\} := [\![P]\!](\mathbf{1})\,. \tag{4}$$

For each $x \in \mathbb{Z}^m$, $\{\![P]\!\}(x)$ is a sub-probability gf which, for the sake of compact notation, we will denote by $\{\![P]\!\}_x$; wanting to make the argument z of $\{\![P]\!\}_x$ explicit, we will write this as $\{\![P]\!\}_x(z)$. So $[z^n]\{\![P]\!\}_x(z)$ is the probability that P with input x terminates in precisely n iterations; $\{\![P]\!\}_x(1)$ is the global probability of termination; and $\frac{\mathrm{d}}{\mathrm{d}z}\{\![P]\!\}_x(z)_{|z=1} / \{\![P]\!\}_x(1)$ is the expected number of steps to termination (running time), given that termination occurs, assuming $\{\![P]\!\}_x(1) > 0$. Higher order moments of $\{\![P]\!\}_x(z)$, involved in e.g. variance, can be computed similarly. Most important, assuming $\{\![P]\!\}_x(z)$ has a radius of convergence $R > 1$, and letting C denote the induced random variable (if $\{\![P]\!\}_x(z)$ is a proper distribution), exponential tail bounds of the form $\Pr(C > k) \leq D \cdot (1/R)^k$, for D a suitable constant, can be easily deduced; we omit the details. How to effectively extract this information from $\{\![P]\!\}$ is the subject of our current research.

Example 1. Fix $m = 1$ and write $x_1 = d$. Consider the program P

$$\texttt{while}\ (d \geq 0)\ \{1/3\colon\ d := d - 1\ \square\ 2/3\colon\ d := d + 1\ \}$$

The fixed-point $\{\![P]\!\}$ of (3) with $G = \mathbf{1}$ satisfies the following equation

$$\{\![P]\!\} = \Psi_{d\geq 0, Q, \mathbf{1}}(\{\![P]\!\}) = \lambda x.\, \left([x < 0]1 + [x \geq 0]\big(z\frac{1}{3}\{\![P]\!\}_{x-1} + z\frac{2}{3}\{\![P]\!\}_{x+1}\big) \right). \tag{5}$$

For specific values of x, this equation can be applied repeatedly for computing any number of coefficients of $\{\![P]\!\}_x(z)$. This will be elaborated in the next section.

3. Towards an Algebraic Theory of Probabilistic Programs

The fixed point semantics described in the previous section provides a firm starting point for the development of new analysis techniques for PP. We outline below two promising approaches that are the subject of our current investigation.

Approximations Based on Banach Fixed Point Theorem As mentioned in the Introduction, one appealing feature of Banach fixed-point theorem, compared to order-theoretic ones (Knaster-Tarski, Kleene,...), is that it immediately implies uniqueness. Moreover, it makes easy it to assess the quality of approximations given by finite iterations of Ψ. We elaborate on these points below.

Firstly, let us endow sD and F with order structures. On sD, let us consider the following partial order $\sqsubseteq$: for each $f_1, f_2 \in \mathsf{sD}$, say $f_1 = \sum_{j\geq 0} a_j z^j$ and $f_2 = \sum_{j\geq 0} b_j z^j$, we let $f_1 \sqsubseteq f_2$ if and only $a_j \leq b_j$ for each $j \geq 0$. The bottom element of this partial order is 0, the constant zero gf. The partial order $(\mathsf{sD}, \sqsubseteq)$ is lifted to a partial order $(\mathsf{F}, \sqsubseteq)$ on F point-wise: $G_1 \sqsubseteq G_2$ if and only if for each $x \in \mathbb{Z}^m$, we have $G_1(x) \sqsubseteq G_2(x)$ in sD. Letting again 0 denote the constant zero gf, the least element of this partial order is $\mathbf{0} := \lambda x.0$. The following result highlights an important property of our semantics.

Theorem 2 (monotonicity). *For each ϕ, P, G, $\Psi_{\phi,P,G} : \mathsf{F} \to \mathsf{F}$ is monotonic w.r.t. $(\mathsf{F}, \sqsubseteq)$.*

Let us now fix generic program $P = \texttt{while}\,\phi\,Q$, and consider the corresponding transformer $\Psi_{\phi,Q,G}$. We focus on $G = \mathbf{1}$, as used in our semantics (4), and abbreviate $\Psi := \Psi_{\phi,Q,\mathbf{1}}$. A corollary of the Banach's theorem is that the unique fixed point is the limit of the iterates $\Psi^{(k)}(I)$, starting from *any* $I \in \mathsf{F}$. That is, defining $\Psi^{(0)}(I) = I$ and $\Psi^{(k+1)}(I) := \Psi(\Psi^{(k)}(I))$, we have:

$$\{\!|P|\!\} = \mathrm{fix}(\Psi) = \lim_{k\to+\infty} \Psi^{(k)}(I)$$

where the $\lim$ is taken in the metric space $(\mathsf{F}, d_{\mathsf{F}})$. Starting in particular from $I = \mathbf{0}$, which is the least element in F, and applying the monotonicity of Ψ (Th. 2), we have[2]

$$\mathbf{0} \sqsubseteq \Psi(\mathbf{0}) \sqsubseteq \Psi(\Psi(\mathbf{0})) \sqsubseteq \cdots \sqsubseteq \Psi^{(k)}(\mathbf{0}) \sqsubseteq \cdots \sqsubseteq \mathrm{fix}(\Psi) = \{\!|P|\!\}\,.$$

$\Psi^{(k)}(\mathbf{0})$ are successive under-approximations of $\{\!|P|\!\}$: what is the nature and quality of such approximations? Another corollary of Banach's theorem is that the distance between the fixed point and the iterates decreases exponentially. More precisely, for every $k > 0$:

$$d_{\mathsf{F}}\left(\Psi^{(k)}(\mathbf{0}), \mathrm{fix}(\Psi)\right) \leq \frac{c^k}{1-c} \cdot d_{\mathsf{F}}(\mathbf{0}, \Psi(\mathbf{0}))$$

where c is the contraction constant. In our case $c \leq \frac{1}{2}$, hence $\frac{c^k}{1-c} \leq 2^{-(k-1)}$. Assuming $d_{\mathsf{F}}(\mathbf{0}, \Psi(\mathbf{0})) \leq 2^{-1}$, in the light of the definition of d_{F}, cf. (1) and (2), this means for each $x \in \mathbb{Z}^m$: $(\Psi^{(k)}(\mathbf{0}))(x) - (\mathrm{fix}(\Psi))(x) = O(z^k)$. So every application of Ψ gains us *at least one* coefficient in the expansions of *all* the gf's encoded by $\{\!|P|\!\} = \mathrm{fix}(\Psi)$, one for each initial state $x \in \mathbb{Z}^m$.

Example 2. *Consider the program in Example 1. After $\Psi^{(0)}(\mathbf{0}) = \mathbf{0}$, the first few iterates of $\Psi^{(k)}(\mathbf{0})$ are the following:*

$$\Psi^{(1)}(\mathbf{0}) = \lambda x.\begin{cases} 1 & \text{if } x < 0 \\ 0 & \text{otherwise.} \end{cases} \qquad \Psi^{(2)}(\mathbf{0}) = \lambda x.\begin{cases} 1 & \text{if } x < 0 \\ \frac{z}{3} & \text{if } x = 0 \\ 0 & \text{otherwise.} \end{cases} \qquad \Psi^{(3)}(\mathbf{0}) = \lambda x.\begin{cases} 1 & \text{if } x < 0 \\ \frac{z}{3} & \text{if } x = 0 \\ \frac{z^2}{9} & \text{if } x = 1 \\ 0 & \text{otherwise.} \end{cases}$$

Continuing this way, one can see that, say for $x = 0$:

$$\{\!|P|\!\}_0(z) = \frac{1}{3}z + \frac{2}{27}z^3 + \frac{8}{243}z^5 + \frac{40}{2187}z^7 + \frac{224}{19683}z^9 + O(z^{11})\,. \tag{6}$$

[2] In the limit, we also use the fact that for each $x \in \mathbb{Z}^m$, $k \geq 0$: $(\Psi^{(k)}(\mathbf{0}))(x) - (\mathrm{fix}(\Psi))(x) = O(z^k)$. In passing, this is not directly related to the rate of convergence of $\{\!|P|\!\}_x(z)$ and does not entail a decision procedure for a.s. termination of P.

Concerning approximation from above, a similar reasoning shows that, whenever we start from any I such that $I \sqsupseteq \mathrm{fix}(\Psi)$, or even s.t. $\Psi(I) \sqsubseteq I$, then the iterates form a descending chain[3]:

$$I \sqsupseteq \Psi(I) \sqsupseteq \cdots \Psi^{(k)}(I) \sqsupseteq \cdots \sqsupseteq \mathrm{fix}(\Psi) = \{\!|P|\!\}.$$

Ideally, I and the subsequent iterates, should provide tighter and tighter upper bounds on such quantities as: $\{\!|P|\!\}_x(1)$ (termination probability) and the (reciprocal of the) radius of convergence R of $\{\!|P|\!\}_x$ (exponential rate of decrease), for each $x \in \mathbb{Z}^m$. In practice, finding such an I in specific cases has proven a major difficulty. One could consider adding a top element $\mathbf{T}$ to F, like $\mathbf{T} = \lambda x.\ \sum_{j \geq 0} z^j = \lambda x.\ \frac{1}{1-z}$; but, while determining a descending chain of iterates, choosing $I = \mathbf{T}$ yields no useful information on $\{\!|P|\!\}$ in the above sense. How to effectively find an informative I is a subject of our ongoing research; techniques based on templates [6] seem promising.

A Connection With Directed Lattice Paths A rather different approach to the analysis of probabilistic programs is expressing the behaviour of a loop, whenever possible, in terms of *directed lattice paths* [3]. Basically, these are random walks in the $\mathbb{Z}$ line, starting from an arbitrary position $x \in \mathbb{Z}$, with left and right jumps of fixed magnitude and probability. In the cases that interest us, a walk terminates as soon as it reaches a negative value. Indeed, given a loop program $P = \texttt{while}\, d \geq 0\, Q$, a sequence of values taken on by the iteration variable d can be seen as such a path. For our running example, a terminating execution corresponds to either a zero length path from $x < 0$, or to path starting from $d = x \geq 0$, ending at $d = 0$ without ever going below 0 — this is called an *excursion* in [3] — followed by a *final* step, corresponding to an execution of the body's loop where the branch $x := x - 1$ is taken, with probability 1/3. More precisely, let $E_x(z)$ denote the generating function of excursions starting at x, with jumps in $\{-1, +1\}$ and associated weights $\{1/3, 2/3\}$ (note that $E_x(z) = 0$ for $x < 0$). Then for all $x \in \mathbb{Z}$:

$$\{\!|P|\!\}_x(z) = [x < 0]1 + [x \geq 0]\left(\frac{1}{3}z \cdot E_x(z)\right) \tag{7}$$

where the factor $\frac{1}{3}z$ accounts for the final step (iteration), in which the first branch of the probabilistic choice is taken, causing the program to terminate. Importantly, in the actual proof of (7), one makes use of the uniqueness of the fixed point granted by Banach theorem: in fact, for (7) to hold it is sufficient that the functional G defined by the RHS of (7) satisfies $G = \Psi(G)$: this can be shown by combinatorial arguments.

There exist consolidated techniques to work out algebraic characterizations of generating functions for directed lattice paths. For the case in question, applying the techniques of [3], one easily obtains, for excursions with $x \geq 0$

$$E_x(z) = \frac{3}{z}\left(\frac{1}{4} \cdot \frac{3 - \sqrt{9 - 8z^2}}{z}\right)^{x+1} \quad \text{which leads to:} \quad \{\!|P|\!\}_x(z) = \left(\frac{1}{4} \cdot \frac{3 - \sqrt{9 - 8z^2}}{z}\right)^{x+1}. \tag{8}$$

For instance, for $x \geq 0$, we have $\{\!|P|\!\}_x(1) = 2^{-x-1}$ (probability of termination), while $\frac{\mathrm{d}}{\mathrm{d}z}\{\!|P|\!\}_x(z)_{|z=1} / \{\!|P|\!\}_x(1) = 3(x+1)$ (expected running time, given termination). Moreover, the radius of convergence of $\{\!|P|\!\}_x(z)$ coincides with its singularity nearest to the origin on the positive semiaxis, $R = \frac{3}{4}\sqrt{2} \approx 1.060660\ldots$. For $x = 0$, the Taylor series from $z = 0$ of $\{\!|P|\!\}_x(z)$ in (8) coincides with (6).

While the directed lattice approach promises very precise results, it is certainly less general than the approximation approach discussed in the previous paragraph. In fact, we conjecture that the directed lattice approach applies to a class of programs that generalizes the *Constant Probability* programs considered in [23].

Acknowledgments Work partially supported by the project SERICS (PE00000014), EU funded NRRP MUR program - NextGenerationEU.

[3]The analogue principle in a framework based on Kleene fixed-point theorem would be *Park's induction* [6].

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] Christel Baier and Joost-Pieter Katoen. *Principles of model checking.* MIT Press, 2008.

[2] Stefan Banach. Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fundamenta Mathematicae* 3(1): 133-181, 1922.

[3] Cyril Banderier and Philippe Flajolet. Basic analytic combinatorics of directed lattice paths. *Theor. Comput. Sci.*, 281(1-2): 37-80, 2002. 10.1016/S0304-3975(02)00007-5

[4] Gilles Barthe, Joost-Pieter Katoen, and Alexandra Silva. *Foundations of Probabilistic Programming.* Cambridge University Press, 2020. 10.1017/9781108770750

[5] Ezio Bartocci, Laura Kovács and Miroslav Stankovic. Automatic Generation of Moment-Based Invariants for Prob-Solvable Loops. *ATVA 2019*, LNCS, 11781: 255–276, Springer, 2019. 10.1007/978-3-030-31784-3_15

[6] Kevin Batz, Mingshuai Chen, Sebastian Junges, Benjamin Lucien Kaminski, Joost-Pieter Katoen and Christoph Matheja. Probabilistic Program Verification via Inductive Synthesis of Inductive Invariants. *TACAS 2023*, LNCS, 13994: 410-429, Springer, 2023. 10.1007/978-3-031-30820-8_25

[7] Benjamin Bichsel, Timon Gehr and Martin T. Vechev. Fine-Grained Semantics for Probabilistic Programs. *ESOP 2018*, LNCS, 10801: 145-185, Springer, 2018. 10.1007/978-3-319-89884-1_6

[8] Michele Boreale. Analysis of Probabilistic Systems via Generating Functions and Padé Approximation. *ICALP 2015*, LNCS, 9135: 82–94, Springer, 2015. 10.1007/978-3-662-47666-6_7

[9] Michele Boreale. Complete Algorithms for Algebraic Strongest Postconditions and Weakest Preconditions in Polynomial ODE'S. *SOFSEM 2018*, LNCS 10706: 442-455, Springer, 2018. 10.1007/978-3-319-73117-9_31

[10] Michele Boreale. Algorithms for exact and approximate linear abstractions of polynomial continuous systems. *HSCC 2018*: 207-216, ACM, 2018. 10.1145/3178126.3178137

[11] Michele Boreale. Algebra, coalgebra, and minimization in polynomial differential equations. *Log. Methods Comput. Sci.* 15(1), 2019. 10.23638/LMCS-15(1:14)2019

[12] Michele Boreale. On the Coalgebra of Partial Differential Equations. *MFCS 2019*: LIPIcs 138: 24:1-24:13, 2019. 10.4230/LIPICS.MFCS.2019.24

[13] Michele Boreale. Automatic pre- and postconditions for partial differential equations. *Inf. Comput.* 285: 104860, 2022. 10.1016/J.IC.2021.104860

[14] Michele Boreale, Roberto Bruni, Rocco De Nicola, Michele Loreti. CaSPiS: a calculus of sessions, pipelines and services. *Math. Struct. Comput. Sci.* 25(3): 666-709, 2015. 10.1017/S0960129512000953

[15] Michele Boreale, Luisa Collodi. Bayesian Parameter Estimation with Guarantees via Interval Analysis and Simulation. *VMCAI 2023*, LNCS, 13881: 106–128, Springer, 2023. 10.1007/978-3-031-50521-8_7

[16] Michele Boreale, Luisa Collodi. Guaranteed Inference for Probabilistic Programs: A Parallelisable, Small-Step Operational Approach. *VMCAI 2024*, LNCS, 14500: 141–162, Springer, 2024. 10.1007/978-3-031-50521-8_7

[17] Michele Boreale, Daniele Gorla. Approximate Model Counting, Sparse XOR Constraints and Minimum Distance. *The Art of Modelling Computational Systems*, LNCS 11760: 363-378, 2019. 10.1007/978-3-030-31175-9_21

[18] Michele Boreale, Francesca Pampaloni. Quantitative Information Flow under Generic Leakage Functions and Adaptive Adversaries. *FORTE 2014*, LNCS 8461: 166-181, Springer, 2014. 10.1007/978-3-662-43613-4_11

[19] Michele Boreale, Michela Paolini. On Formally Bounding Information Leakage by Statistical Estimation. *ISC 2014*, LNCS 8783: 216-236, Springer, 2014. 10.1007/978-3-319-13257-0_13

[20] Mingshuai Chen, Joost-Pieter Katoen, Lutz Klinkenberg and Tobias Winkler. Does a Program

Yield the Right Distribution? - Verifying Probabilistic Programs via Generating Functions. *CAV 2022*, LNCS, 13371: 79-101, Springer, 2022. 10.1007/978-3-031-13185-1_5

[21] Alessandra Di Pierro and Herbert Wiklicky. Semantics of Probabilistic Programs: A Weak Limit Approach. *APLAS 2013*, LNCS, 8301: 241-256, Springer, 2013. 10.1007/978-3-319-03542-0_18

[22] Timon Gehr, Sasa Misailovic and Martin T. Vechev. Psi: Exact symbolic inference for probabilistic programs. *CAV 2016*, LNCS, 9779: 62–83, Springer, 2016 10.1007/978-3-319-41528-4_4

[23] Jürgen Giesl, Peter Giesl and Marcel Hark. Computing Expected Runtimes for Constant Probability Programs. *CADE 2019*, LNCS, 11716: 296–286, Springer, 2019. 10.1007/978-3-030-29436-6_16

[24] Andrew D. Gordon, Thomas A. Henzinger, Aditya V. Nori, and Sriram K. Rajamani. Probabilistic programming. *Proceedings of Future of Software Engineering (FOSE2014)*, pages 167–181, 2014. 10.1145/2593882.2593900

[25] Friedrich Gretz, Joost-Pieter Katoen and Annabelle McIver. Operational versus weakest pre-expectation semantics for the probabilistic guarded command language. *Perform. Evaluation*, 73: 110–132, 2014. 10.1016/J.PEVA.2013.11.004

[26] Ralf Herbrich, Tom Minka and Thore Graepel. TrueSkill$^{\text{TM}}$: A Bayesian Skill Rating System. *NeurIPS 2006*, 569-576, MIT Press, 2006.

[27] Sebastian Junges, Nils Jansen, Joost-Pieter Katoen and Ufuk Topcu. Probabilistic Verification for Cognitive Models. *AAAI Fall Symposia*, AAAI Press, 2016.

[28] Lutz Klinkenberg, Kevin Batz, Benjamin Lucien Kaminski, Joost-Pieter Katoen, Joshua Moerman and Tobias Winkler. Generating Functions For Probabilistic Programs. Proceedings of the 30th International Symposium, LOPSTR 2020, Bologna, 12561:231–248. 10.1007/978-3-030-68446-4_12

[29] Lutz Klinkenberg, Christian Blumenthal, Mingshuai Chen, Darion Haase and Joost-Pieter Katoen. Exact Bayesian Inference for Loopy Probabilistic Programs using Generating Functions. *Proc. ACM Program. Lang.*, 8(OOPSLA1): 923-953, 2024. 10.1145/3649844

[30] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models - Principles and Techniques.* MIT Press, 2009.

[31] Dexter Kozen. Semantics of probabilistic programs. *Journal of Computer and System Sciences*, 22(3):328-350, 1981. 10.1016/0022-0000(81)90036-2

[32] Marta Z. Kwiatkowska, Gethin Norman and David Parker. PRISM 4.0: Verification of Probabilistic Real-Time Systems. *CAV 2011*, LNCS, 6806: 585-591, Springer, 2011. 10.1007/978-3-642-22110-1_47

[33] Annabelle McIver, Carroll Morgan. *Abstraction, Refinement and Proof for Probabilistic Systems.* Monographs in Computer Science, Springer, 2005. 10.1007/B138392

[34] Héctor D. Menéndez, Michele Boreale, Daniele Gorla, David Clark. Output Sampling for Output Diversity in Automatic Unit Test Generation. *IEEE Trans. Software Eng.* 48(2): 295-308, 2022. 10.1109/TSE.2020.2987377

[35] Rajeev Motwani and Prabhakar Raghavan *Randomized Algorithms.* Cambridge University Press, 1995. 10.1017/CBO9780511814075

[36] Van Chan Ngo, Quentin Carbonneaux and Jan Hoffmann. Bounded expectations: resource analysis for probabilistic programs. *PLDI 2018*, 496-512, ACM, 2018. 10.1145/3192366.3192394

[37] Aditya V. Nori, Chung-Kil Hur, Sriram K. Rajamani and Selva Samuel. R2: An Efficient MCMC Sampler for Probabilistic Programs. *AAAI 2014*, 2476-2482, AAAI Press, 2014. 10.1609/AAAI.V28I1.9060

[38] Federico Olmedo, Friedrich Gretz, Nils Jansen, Benjamin Lucien Kaminski, Joost-Pieter Katoen and Annabelle McIver. Conditioning in Probabilistic Programming. *ACM Trans. Program. Lang. Syst.*, 40(1): 4:1–4:50, 2018. 10.1145/3156018

[39] Francesca Randone, Luca Bortolussi, Emilio Incerto and Mirco Tribastone. Inference of Probabilistic Programs with Moment-Matching Gaussian Mixtures. *Proc. ACM Program. Lang.*, 8(POPL): 1882-1912, 2024. 10.1145/3632905

[40] Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific J. Math.*, 5(2): 285-309, 1955.

[41] Herbert S. Wilf. *Generatingfunctionology.* A. K. Peters Ltd, USA, 2006.