Towards algebraic asymptotics for probabilistic loops

Michele Boreale, Luisa Collodi and Alessandro Pompa Di Gregorio

Università di Firenze, Dipartimento di Statistica, Informatica, Applicazioni "G. Parenti", Italy.

Abstract

We report on ongoing research efforts aimed at exact asymptotic analysis of running time in probabilistic while loops. We focus on guarded-counter programs, translate them into probabilistic pushdown automata with global resets, and derive polynomial systems for their probability generating functions. After extracting dominant singularities, transfer theorems from analytic combinatorics yield coefficient asymptotics for runtime distributions.

Keywords

Probabilistic programming, generating functions, polynomial systems, asymptotic analysis

1. Introduction

Probabilistic Programming (PP) [1, 2] provides formal specification and verification tools for analyzing probabilistic models programmatically. Applications range from randomized algorithms and probabilistic graphical models to cognitive models, security protocols, and model checking [3, 4, 5, 6, 7]. The analysis of probabilistic programs is notoriously difficult: even for simple discrete languages, exact quantitative questions about termination, expected runtime, higher moments or distribution tails quickly lead to undecidability [8] This motivates both simulation-based approaches [9, 10] and the search for interesting tractable fragments [11, 12, 13, 14, 15].

In this communication, we report on ongoing work on an algebraic method for probabilistic while loops with one guarded counter, *generalized constant probability programs* (GCPs). The loop guard is $d > 0$, and each iteration performs a state update, and either a relative $d := d + k$ or an absolute $d := h$ counter update (Fig. 1 left). We encode such programs as a type of probabilistic pushdown automata (pPDAs) with resets, over a unary input alphabet, where the stack height represents the counter value and each consumed letter marks one loop iteration (Fig. 1 right). The acceptance generating function $A(z) = \sum_{n\geq 0} a_n z^n$ induced by such pPDA is the probability (sub-)distribution of the program's runtime, counted as number of loop iterations. We provide a method to build a system of polynomial equations $\mathcal{P}$ of which $A(z)$ is the unique analytic solution, akin to the classical PDA-to-CFG construction [16]. For the pPDAs arising from our program translation, the classic analytic Implicit Function Theorem (IFT) is sufficient to identify $A(z)$ as (a component of) the unique solution of $\mathcal{P}$ at the origin. Gröbner elimination [17] is then used to produce a single polynomial equation $P(z, A) = 0$ satisfied by $A(z)$. Singularity analysis via Puiseux series from analytic combinatorics [18] leads finally to exact asymptotics for the coefficients a_n.

Related and further work. The formal semantics of probabilistic programs goes back to Kozen [19] and to the monograph of McIver and Morgan [20]; further semantic accounts for discrete probabilistic programs include [21, 22, 23, 24, 25, 26, 27]. Generating-function techniques for probabilistic programs have been developed in [11, 12, 13] and used for exact inference and verification [14]. Differently from these works, we focus on runtime and coefficient asymptotics, leveraging algebraic generating functions and pPDAs. This is also related to the line of work on pPDAs by Brazdil et al. [28, 29], which focus on moments and on tail bounds, rather than exact asymptotics. Our variables R_{pXq} are the weighted analogue of the classical PDA-to-CFG triple construction [16]. Our final asymptotic step relies on the

ICTCS 2026.

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

© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-09-19

CEUR-WS.org/Vol-4269/paper68.pdf

```
while (d > 0){
  if (q = 1) {1/2 : d := d + 1; q := 2 [] 1/2 : d := d − 1; q := 1}
  else {2/3 : d := d + 1; q := 2 [] 1/6 : d := d − 2; q := 1 [] 1/6 : d := 0; q := 2}}
```

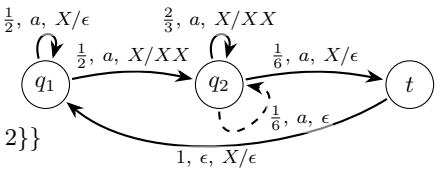

Figure 1: Left: a GCP program. **Right**: a pPDA with global reset, obtained from translation of the program on the left. Solid arrows are pushdown transitions, the dashed arrow is a reset transition.

analytic-combinatorics viewpoint on algebraic functions [18, 30]. Current work includes extensions beyond GCP programs, as well as alternative methods avoiding the expensive Gröbner elimination step [31, 32]. The present investigation is part of a broader research agenda, aimed at developing flexible formal methods applicable across diverse, probability-related domains, including dynamical systems with safety-related aspects [33, 34, 35, 36, 37, 38, 39], information leakage and security [40, 41, 42, 43], distributed systems with notions of failure and recovery [44, 45], randomized testing [46, 47].

2. Probabilistic While-loops and pPDAs

GCP programs We introduce GCPs, a subclass of pGCL [20], also considered in [32]. A GCP has one integer counter d taking values in $\mathbb{Z}$, and one finite-valued control variable q taking values in $\{1, \ldots, N\}$, for a fixed integer $N \geq 1$. The loop guard is $d > 0$, thus a computation terminates as soon as $d \leq 0$. The program allows two types of counter updates: relative counter update $d := d + k$ for $k \in \mathbb{Z}$, and absolute counter update $d := h$ for $h \in \mathbb{N}$. In any case, this is followed by an absolute update of q. The case $h = 0$ is immediate termination, since the next guard test fails.

Definition 1 (GCP programs)**.** Fix $N \geq 1$. A *N-state GCP specification* is a family $\mathcal{T} = \big((\mathcal{S}_i, \mathcal{C}_i)\big)_{1 \leq i \leq N}$, where $\mathcal{S}_i \subseteq (0, 1] \times \mathbb{Z} \times \{1, \ldots, N\}$ (relative updates from state i), and $\mathcal{C}_i \subseteq (0, 1] \times \mathbb{N} \times \{1, \ldots, N\}$ (absolute updates from state i) are such that $\sum_{(\lambda,k,j)\in\mathcal{S}_i} \lambda + \sum_{(\lambda,h,j)\in\mathcal{C}_i} \lambda = 1$.

In traditional syntax, the program associated with $\mathcal{T}$ can be written as follows. We write $[\!]_{i\in I}\, \lambda_i : P_i$ for probabilistic choice among commands P_i, where $\lambda_i > 0$. Empty probabilistic choices are omitted.

$$
\begin{aligned}
&\texttt{while } (d > 0)\ \{\\
&\quad \texttt{if } q = 1\ \{\, [\!]_{(\lambda,k,j)\in\mathcal{S}_1} \lambda : d := d + k;\ q := j\ [\!]\ [\!]_{(\lambda,h,j)\in\mathcal{C}_1} \lambda : d := h;\ q := j\,\}\\
&\quad \texttt{else if } q = 2\ \{\, [\!]_{(\lambda,k,j)\in\mathcal{S}_2} \lambda : d := d + k;\ q := j\ [\!]\ [\!]_{(\lambda,h,j)\in\mathcal{C}_2} \lambda : d := h;\ q := j\,\}\\
&\quad \ldots\\
&\quad \texttt{else if } q = N\ \{\, [\!]_{(\lambda,k,j)\in\mathcal{S}_N} \lambda : d := d + k;\ q := j\ [\!]\ [\!]_{(\lambda,h,j)\in\mathcal{C}_N} \lambda : d := h;\ q := j\,\}\,\}.
\end{aligned}
$$

Probabilistic PDAs with resets A finite *probabilistic pushdown automaton with resets* over the unary input alphabet $\Sigma = \{a\}$ is a triple $A = (Q, \mathcal{Z}, \Delta)$, where Q is the finite set of *control states*, $\mathcal{Z}$ a finite *stack alphabet*, and $\Delta = (\Delta_{\mathrm{loc}}, \Delta_{\mathrm{rst}})$ is a pair of finite sets of *local* and *reset* transitions, specified as follows. We have $\Delta_{\mathrm{i}} \subseteq Q \times \mathcal{Z} \times [0, 1] \times \{a, \epsilon\} \times Q \times \mathcal{Z}^*$ for $i \in \{\mathrm{loc}, \mathrm{rst}\}$. A *local transition* $e \in \Delta_{\mathrm{loc}}$ is written as $qX \xrightarrow{\pi,\sigma} r\beta$, where $q, r \in Q, X \in \mathcal{Z}, \beta \in \mathcal{Z}^*, \pi \in [0, 1], \sigma \in \{a, \epsilon\}$. The transition pops X from the stack, pushes β onto it, moves to state r, and consumes σ. We define the weight function: $w(a) = z, w(\epsilon) = 1$, where z is a formal variable that tracks the length of a computation, with each z-labeled transition contributing one unit. A *reset transition* $e \in \Delta_{\mathrm{rst}}$ is written as $qX \xrightarrow{\pi,a}_{\mathrm{rst}} r\gamma$ with $\gamma \in \mathcal{Z}^*$. Such a transition reads one input letter, discards the entire current stack, and restarts in state r with stack γ. Thus, unlike a local transition, it does not expose the stack below X. The case $\gamma = \epsilon$ is immediate empty-stack termination. The automaton A is assumed to be *sub-stochastic*, i.e. for each $q \in Q$, the sum of the probabilities π of all (local or reset) transitions from q is ≤ 1. Stacks $\gamma \in \mathcal{Z}^*$ are written with their top symbol on the left. A *configuration* is a pair $(q, \gamma) \in Q \times \mathcal{Z}^*$, written $q\gamma$. A local transition $e : qX \xrightarrow{\pi,\sigma} r\beta$ is *enabled* in every configuration of the form $(q, X\eta)$, and induces the move $(q, X\eta) \xrightarrow{e} (r, \beta\eta)$. Its input contribution is $|e|_a = 1$ if $\sigma = a$, and $|e|_a = 0$ if $\sigma = \epsilon$. A reset transition

$e: qX \xrightarrow{\pi,a}_{\mathrm{rst}} r\gamma$ is enabled in $(q, X\eta)$ and induces $(q, X\eta) \xrightarrow{e} (r, \gamma)$; thus the lower stack context η is discarded. Its input contribution is $|e|_a = 1$. Configurations with empty stack are *terminal*. A *run* is a finite sequence $\rho = (C_0, e_1, C_1, e_2, \ldots, e_m, C_m)$ $(m \geq 0)$ of alternating (local or reset) transitions e_i and configurations C_i such that $C_{i-1} \xrightarrow{e_i} C_i$ for every $1 \leq i \leq m$. We write $C_0 \xRightarrow{\rho} C_m$ for such a run. Its probability weight is $\mathrm{wt}(\rho) := \prod_{i=1}^{m} \pi(e_i)$, and its input length is $|\rho| := \sum_{i=1}^{m} |e_i|_a$. The empty run $\rho = \epsilon$ is allowed, with weight 1, input length 0, and satisfies $C \xRightarrow{\epsilon} C$. A run is *accepting* if its final configuration C_m is a terminal.

From GCPs programs to pPDAs Fix $\mathcal{T} = \big((\mathcal{S}_i, \mathcal{C}_i)\big)_{1 \leq i \leq N}$, an N-state GCP program specification. We translate $\mathcal{T}$ into a pPDA with resets $A_{\mathcal{T}} = (Q, \mathcal{Z}, \Delta)$, with $Q = \{q_1, ..., q_N\} \cup Q_{\mathrm{aux}}$ for Q_{aux} a finite set of auxiliary states as specified below, $\mathcal{Z} = \{X\}$ and Δ built as follows. The driving idea is that a counter value $d \geq 0$ is represented by the unary stack X^d. Each transition (λ, k, j) in $\mathcal{T}$ is translated into one local or reset transition, possibly followed by some ϵ-transitions. For every $i = 1, ..., N$, consider the following. In the second clause, we have $t_\ell^{i,j,m} \in Q_{\mathrm{aux}}$ for $1 \leq \ell < m$, while $t_m^{i,j,m} := q_j$.

1. For each $(\lambda, k, j) \in \mathcal{S}_i$ with $k \geq 0$, we add to Δ_{loc}: $q_i X \xrightarrow{\lambda,a} q_j X^{k+1}$.
2. For each $(\lambda, k, j) \in \mathcal{S}_i$ with $k = -m < 0$, we add to Δ_{loc}: $q_i X \xrightarrow{\lambda,a} t_1^{i,j,m}\epsilon$ and $t_\ell^{i,j,m} X \xrightarrow{1,\epsilon} t_{\ell+1}^{i,j,m}\epsilon$ $(1 \leq \ell \leq m-1)$.
3. For each $(\lambda, h, j) \in \mathcal{C}_i$, we add to Δ_{rst}: $q_i X \xrightarrow{\lambda,a}_{\mathrm{rst}} q_j X^h$.

In the second clause, if the stack becomes empty at any point during the pop chain, the loop has terminated: this matches the intended GCP semantics, since $d + k \leq 0$ implies that the next test of the guard $d > 0$ fails; the final control state is then irrelevant. Overall, the only ϵ-transitions introduced by the translation are the auxiliary pop transitions for updates $k < -1$. They form finite acyclic chains. Global reset transitions consume one input letter, and hence introduce no ϵ-cycles. The third case includes $h = 0$ (termination).

Example 1. Consider the GCP program in Fig. 1, left. $A_{\mathcal{T}}$ has five local and one reset transitions. See Fig. 1, right.

$$q_1X \xrightarrow{1/2,a} q_2XX, \quad q_1X \xrightarrow{1/2,a} q_1\epsilon, \quad q_2X \xrightarrow{2/3,a} q_2XX, \quad q_2X \xrightarrow{1/6,a} t\epsilon, \quad tX \xrightarrow{1,\epsilon} q_1\epsilon, \quad q_2X \xrightarrow{1/6,a}_{\mathrm{rst}} q_2\epsilon.$$

3. Summary series for pPDAs

Definition 2. For $n \geq 0, p \in Q$ and $\beta \in \mathcal{Z}^*$, let $\mathcal{A}_{p\beta}^{(n)} := \{\rho : |\rho| = n \text{ and } \exists q : p\beta \xRightarrow{\rho} q\epsilon\}$. Set $a_{p\beta}^{(n)} := \sum_{\rho \in \mathcal{A}_{p\beta}^{(n)}} \mathrm{wt}(\rho)$. The *acceptance* generating function for $p\beta$ is: $A_{p\beta}(z) := \sum_{n \geq 0} a_{p\beta}^{(n)} z^n$.

So the n-th coefficient of $A_{p\beta}(z)$ is the probability of termination in exactly n steps; $A_{p\beta}(1)$ is the termination probability; $A'_{p\beta}(1)$ is the expected runtime, and so on. $A_{p\beta}(z)$ is therefore the mathematical object we are interested in. For the purpose of its analysis, it is convenient to decompose $A_{p\beta}(z)$ in terms of two other types of series, which we introduce next. For $p, q \in Q$ and $\beta \in \mathcal{Z}^*$, define $\mathcal{R}_{p\beta q}^{(n)} := \{\rho : |\rho| = n,\ p\beta \xRightarrow{\rho} q\epsilon \text{ and no reset transitions occur in } \rho\}$, and set $r_{p\beta q}^{(n)} := \sum_{\rho \in \mathcal{R}_{p\beta q}^{(n)}} \mathrm{wt}(\rho)$. The *local-return* generating function is $R_{p\beta q}(z) := \sum_{n \geq 0} r_{p\beta q}^{(n)} z^n$. This is the series of runs starting in p, consuming β and terminating in q with no reset transition. Next, define $\mathcal{E}_{p,\beta}^{(n)} := \{\rho : |\rho| = n \text{ and } \exists q : p\beta \xRightarrow{\rho} q\epsilon \text{ and } \rho \text{ contains at least one reset transition}\}$, and set $s_{p\beta}^{(n)} := \sum_{\rho \in \mathcal{E}_{p\beta}^{(n)}} \mathrm{wt}(\rho)$. The *escape* generating function is $E_{p\beta}(z) = \sum_{n \geq 0} s_{p\beta}^{(n)} z^n$. The fundamental relation linking $A(z), R(z)$ and $E(z)$ is:

$$A_{p\beta}(z) = E_{p\beta}(z) + \sum_{q \in Q} R_{p\beta q}(z). \tag{1}$$

This will be exploited in the next section. It is easy to prove that $A_{p\beta}(z)$, $R_{p\beta q}(z)$ and $E_{p\beta}(z)$ are sub-probability generating functions. Therefore they are absolutely convergent and, as functions of z, analytic for $|z| < 1$.

4. Polynomial systems

From pPDAs to polynomial systems We now characterize the generating functions $A_{p\beta}(z)$, $R_{p\beta q}(z)$ and $E_{p\beta}(z)$ as solutions of a polynomial system. To this purpose, let us introduce a set of formal *indeterminates*

$$\mathcal{X} := \{A_{pX} : p \in Q,\ X \in \mathcal{Z}\} \cup \{R_{pXq} : p, q \in Q,\ X \in \mathcal{Z}\} \cup \{E_{pX} : p \in Q,\ X \in \mathcal{Z}\}.$$

We shall work in the polynomial ring $\mathbb{C}[z, \mathcal{X}]$. To avoid notational ambiguities, we will distinguish an indeterminate $U \in \mathcal{X}$ from the eponym generating function $U(z)$, defined in Section 3, by always making the argument z explicit in the second case. This convention extends to polynomials in $\mathbb{C}[z, \mathcal{X}]$ as expected. We now introduce auxiliary polynomials $R_{p\beta q}, E_{p\beta} \in \mathbb{C}[z, \mathcal{X}]$, as follows: $R_{p\epsilon q} := \mathbf{1}_{p=q}, E_{p\epsilon} := 0$; for $\beta = X$, $R_{p\beta q}, E_{p\beta}$ are indeterminates in $\mathcal{X}$; and for $\beta = X\gamma$ ($\gamma \in \mathcal{Z}^+$), we have inductively: $R_{pX\gamma q} := \sum_{s\in Q} R_{pXs}R_{s\gamma q}$, and $E_{pX\gamma} := E_{pX} + \sum_{s\in Q} R_{pXs}E_{s\gamma}$. Finally, define $A_{p\epsilon} := 1$ and $A_{p\beta} := E_{p\beta} + \sum_{r\in Q} R_{p\beta r}$ for $|\beta| \geq 2$. For each $U \in \mathcal{X}$, we define the polynomial (equation) $\mathcal{P}_U$, as follows.

$$\mathcal{P}_{A_{pX}} := A_{pX} - (E_{pX} + \sum_{q\in Q} R_{pXq}) \qquad \mathcal{P}_{R_{pXq}} := R_{pXq} - \sum_{pX \xrightarrow{\pi,\sigma} r\beta} \pi w(\sigma) R_{r\beta q}$$

$$\mathcal{P}_{E_{pX}} := E_{pX}\Big(\sum_{pX \xrightarrow{\pi,\sigma} r\beta} \pi w(\sigma) E_{r\beta} + \sum_{pX \xrightarrow{\pi,a}_{\mathrm{rst}} r\gamma} \pi z\, A_{r\gamma}\Big).$$

Let $\mathcal{P} := \{\mathcal{P}_U : U \in \mathcal{X}\}$, which we see as a set of polynomial equations. Note that $\mathcal{P}_{A_{pX}}$ is an instance of (1), while $\mathcal{P}_{R_{pXq}}, \mathcal{P}_{E_{pX}}$ mimic the definitions of sets $\mathcal{R}^{(n)}_{pXq}, \mathcal{E}^{(n)}_{pX}$, respectively. A tuple of formal power series $x(z) \in \mathbb{C}[[z]]^{\mathcal{X}}$ is a *solution* of $\mathcal{P}$ if all polynomials in $\mathcal{P}$ vanish after substituting $X_i = x_i(z)$ for every $X_i \in \mathcal{X}$. Below, the first part follows from the definition of the summary series; uniqueness follows from the classical Implicit Function Theorem (IFT).

Theorem 1. *Let A be a pPDA. The tuple of formal power series $(U(z))_{U\in\mathcal{X}}$, as defined in Sect. 3, is a solution of $\mathcal{P}$. If $A = A_{\mathcal{T}}$ for some GCP specification $\mathcal{T}$, then this is the* unique *analytic solution $x(z)$ of $\mathcal{P}$ near $z = 0$ that satisfies $x(0) = (U(0))_{U\in X}$.*

Example 2. Consider again our running GCP example. In what follows, for the sake of notation we denote q_i by i and omit 'X' from subscripts. The translation from the pPDA gives rise to the following system $\mathcal{P}$ of 15 equations (here we read polynomials as equations). For $s, i, j \in \{1, 2, t\}$

$$A_i = E_i + \sum_s R_{is}, \qquad R_{1j} = \frac{1}{2}z\,\mathbf{1}_{j=1} + \frac{1}{2}z\sum_s R_{2s}R_{sj}, \qquad R_{2j} = \frac{1}{6}z\,\mathbf{1}_{j=t} + \frac{2}{3}z\sum_s R_{2s}R_{sj}, \qquad R_{tj} = \mathbf{1}_{j=1},$$
$$E_1 = \frac{1}{2}z(E_2 + \sum_s R_{2s}E_s), \qquad E_2 = \frac{1}{6}z + \frac{2}{3}z(E_2 + \sum_s R_{2s}E_s), \qquad E_t = 0.$$

It is immediate to check that $R_{12} = R_{22} = R_{1t} = 0$, $R_{2t} = \frac{1}{6}z$ and that $R_{t1} = 1, R_{t2} = R_{tt} = 0$. We now focus on the equation for $A_1 = A_{q_1X}$. Let $S = R_{11}, U = E_1$, and define $L := \sum_s R_{2s}R_{s1} = R_{2,XX,1}$, and $W := E_2 + \sum_s R_{2s}E_s = E_{2,XX}$. Focusing on A_1, the system reduces to

$$A_1 = S + U, \quad S = \frac{1}{2}z + \frac{1}{2}zL, \quad U = \frac{1}{2}zW, \quad L = \frac{1}{6}z + \frac{2}{3}zLS, \quad W = \frac{1}{6}z + \frac{2}{3}zW + \frac{2}{3}zLU. \tag{2}$$

From this system, one can already build a Taylor series[1] of $A_1(z)$ hence the coefficients $a^{(n)}_{q_1X}$ up to any finite order: $A_1(z) = \frac{1}{2}z + \frac{1}{6}z^2 + \frac{1}{18}z^3 + \frac{7}{108}z^4 + \frac{11}{324}z^5 + \frac{31}{972}z^6 + \cdots$.

[1] E.g. from an ODE system obtained via IFT.

The general case $A_{p\beta}(z)$ with $\beta = X^d$ ($d \geq 1$) can be expressed in terms of $A_{pX}(z), R_{pXq}(z), E_{pX}(z)$ as follows. Let $R_X(z)$ be the $Q \times Q$ matrix of entries $R_{pXq}(z)$, and $E_X(z)$ the column vector of entries $E_{pX}(z)$ ($p \in Q$). Then $A_{pX^d}(z) = e_p^T(R_X^d(z)\mathbf{1} + \sum_{i=0}^{d-1} R_X^i(z)E_X(z))$, where $\mathbf{1}$ is the all-1 column vector, e_p is the standard basis vector for state p, and e_p^T denotes its transpose. We omit the details.

5. Asymptotic analysis

We now analyze the system $\mathcal{P}$ to find the *dominant* singularities of components $A_{pX}(z)$ of its unique solution, i.e. the (non-removable) singularities in $\mathbb{C}$ of smallest modulus of $A_{pX}(z)$. This will be instrumental to obtain an asymptotic expansion of the coefficients $a_n := a_{pX}^{(n)}$, using the classical transfer theorems of [18, Ch.6,7]. As this is a standard procedure, we only outline its main steps, using $A_1(z) = A_{q_1X}(z)$ from our running example as an illustration.

1. From $\mathcal{P}$, or directly from (2), we find a nonzero polynomial $P(z, Y)$ such that $P(z, A_1(z)) = 0$ identically. This can be done algorithmically using either Groebner bases or resultants, see e.g. [17]. For our example, we obtain $P(z, Y) = b(z)Y^2 + (-2z^4 + 10z^3 - 18z^2 + 42z - 36)Y + z^4 - 2z^3 - 15z^2 + 18z$, where $b(z) = (6z^3 - 28z^2 + 24z)$ is the leading coefficient.

2. From $P(z, Y)$, we build a finite set of *candidate singularities* Σ, which includes all those of $A_1(z)$. Basically, Σ is the set of $z \in \mathbb{C}$ where the hypotheses of the IFT for $P(z, Y) = 0$ fail for some Y (branch points), union $\{z : b(z) = 0\}$ (poles). Again, this set can be characterized algorithmically (details omitted).

3. Σ is scanned in order of increasing modulus in search of singularities. $\zeta \in \Sigma$ is a singularity of $A_1(z)$ iff the power series expansion of $A_1(z)$ in $z - \zeta$ is non-analytic, i.e. it has either negative or fractional exponents. Technically, such expansions are known as *Puiseux series*, and can be computed given P and ζ by Newton's polygon algorithm [18, Ch.7]. Using this method, in our example the only dominant singularity is found to be $\zeta = \frac{7-\sqrt{13}}{3} \approx 1.131$.

4. Consider the Puiseux series of $A_1(z)$ in $z - \zeta$, say $S(z)$. Write the *dominant singular* term — the one of smallest negative or fractional exponent — of $S(z)$ in the form $c \cdot \left(1 - \frac{z}{\zeta}\right)^{\alpha}$, for $\alpha \notin \mathbb{N}$. The transfer theorem of [18, Th.VI.4] gives the asymptotic expansion $a_n \sim c\frac{\zeta^{-n}}{\Gamma(-\alpha)}n^{-\alpha-1}$ for the coefficients of $A_1(z)$. In our case, the dominant singular term is $c \cdot (1 - \frac{z}{\zeta})^{-1}$, for $c \approx 0.01703$ and ζ defined above. We get the expansion

$$a_n \sim c \cdot \zeta^{-n} = \frac{26 - 5\sqrt{13}}{468}\left(\frac{7 + \sqrt{13}}{12}\right)^n = 0.01703... \times 0.88386...^n$$

The general case of multiple dominant singularities is only slightly more complicated, see [18, Th.VII.8]. Finally, building the Puiseux series of $A_1(z)$ in $z - 1$, we can read off from its first two coefficients the exact probability of termination $A_1(1) = 1$ and expected running time $A_1'(1) = \frac{5}{2}\sqrt{2} \approx 3.5355$.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT in order to check spelling and grammar, paraphrase and reword. The authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] A. D. Gordon, T. A. Henzinger, A. V. Nori, S. K. Rajamani, Probabilistic programming, in: Proceedings of the on Future of Software Engineering, 2014, pp. 167–181. doi:10.1145/2593882.2593900.

[2] G. Barthe, J.-P. Katoen, A. Silva, Foundations of Probabilistic Programming, Cambridge University Press, 2020. doi:10.1017/9781108770750.
[3] R. Motwani, P. Raghavan, Randomized Algorithms, Cambridge University Press, 1995. doi:10.1017/CBO9780511814075.
[4] D. Koller, N. Friedman, Probabilistic Graphical Models: Principles and Techniques, MIT Press, 2009.
[5] C. Baier, J.-P. Katoen, Principles of Model Checking, MIT Press, 2008.
[6] M. Z. Kwiatkowska, G. Norman, D. Parker, PRISM 4.0: Verification of probabilistic real-time systems, in: CAV 2011, volume 6806 of *LNCS*, Springer, 2011, pp. 585–591. doi:10.1007/978-3-642-22110-1_47.
[7] S. Junges, N. Jansen, J.-P. Katoen, U. Topcu, Probabilistic verification for cognitive models, in: AAAI Fall Symposia, AAAI Press, 2016.
[8] B. L. Kaminski, J. Katoen, C. Matheja, On the hardness of analyzing probabilistic programs, Acta Informatica 56 (2019) 255–285. URL: https://doi.org/10.1007/s00236-018-0321-1. doi:10.1007/S00236-018-0321-1.
[9] A. V. Nori, C.-K. Hur, S. K. Rajamani, S. Samuel, R2: An efficient MCMC sampler for probabilistic programs, in: AAAI 2014, AAAI Press, 2014, pp. 2476–2482. doi:10.1609/AAAI.V28I1.9060.
[10] M. Boreale, L. Collodi, Guaranteed inference for probabilistic programs: A parallelisable, small-step operational approach, in: VMCAI 2024, volume 14500 of *LNCS*, Springer, 2024, pp. 141–162. doi:10.1007/978-3-031-50521-8_7.
[11] M. Boreale, Analysis of probabilistic systems via generating functions and Padé approximation, in: ICALP 2015, volume 9135 of *LNCS*, Springer, 2015, pp. 82–94. doi:10.1007/978-3-662-47666-6_7.
[12] L. Klinkenberg, K. Batz, B. L. Kaminski, J.-P. Katoen, J. Moerman, T. Winkler, Generating functions for probabilistic programs, in: Logic-Based Program Synthesis and Transformation 2020, Springer, 2021, pp. 231–248. doi:10.1007/978-3-030-68446-4_12.
[13] L. Klinkenberg, C. Blumenthal, M. Chen, D. Haase, J.-P. Katoen, Exact bayesian inference for loopy probabilistic programs using generating functions, Proceedings of the ACM on Programming Languages 8 (2024) 923–953. doi:10.1145/3649844.
[14] M. Chen, J.-P. Katoen, L. Klinkenberg, T. Winkler, Does a program yield the right distribution? verifying probabilistic programs via generating functions, in: CAV 2022, volume 13371 of *LNCS*, Springer, 2022, pp. 79–101. doi:10.1007/978-3-031-13185-1_5.
[15] J. Giesl, P. Giesl, M. Hark, Computing expected runtimes for constant probability programs, in: CADE 2019, volume 11716 of *LNCS*, Springer, 2019, pp. 286–296. doi:10.1007/978-3-030-29436-6_16.
[16] J. E. Hopcroft, J. D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, 1979.
[17] D. A. Cox, J. Little, D. O'Shea, Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra, Undergraduate Texts in Mathematics, 5 ed., Springer, 2025. doi:10.1007/978-3-031-91841-4.
[18] P. Flajolet, R. Sedgewick, Analytic Combinatorics, Cambridge University Press, 2009. URL: https://doi.org/10.1017/CBO9780511801655. doi:10.1017/CBO9780511801655.
[19] D. Kozen, Semantics of probabilistic programs, Journal of Computer and System Sciences 22 (1981) 328–350. doi:10.1016/0022-0000(81)90036-2.
[20] A. McIver, C. Morgan, Abstraction, Refinement and Proof for Probabilistic Systems, Monographs in Computer Science, Springer, 2005. doi:10.1007/B138392.
[21] F. Gretz, J.-P. Katoen, A. McIver, Operational versus weakest pre-expectation semantics for the probabilistic guarded command language, Performance Evaluation 73 (2014) 110–132. doi:10.1016/J.PEVA.2013.11.004.
[22] A. Di Pierro, H. Wiklicky, Semantics of probabilistic programs: A weak limit approach, in: APLAS 2013, volume 8301 of *LNCS*, Springer, 2013, pp. 241–256. doi:10.1007/978-3-319-03542-0_18.
[23] B. Bichsel, T. Gehr, M. T. Vechev, Fine-grained semantics for probabilistic programs, in: ESOP 2018,

volume 10801 of *LNCS*, Springer, 2018, pp. 145–185. doi:10.1007/978-3-319-89884-1_6.

[24] F. Olmedo, F. Gretz, N. Jansen, B. L. Kaminski, J.-P. Katoen, A. McIver, Conditioning in probabilistic programming, ACM Transactions on Programming Languages and Systems 40 (2018) 4:1–4:50. doi:10.1145/3156018.

[25] T. Gehr, S. Misailovic, M. T. Vechev, Psi: Exact symbolic inference for probabilistic programs, in: CAV 2016, volume 9779 of *LNCS*, Springer, 2016, pp. 62–83. doi:10.1007/978-3-319-41528-4_4.

[26] V. C. Ngo, Q. Carbonneaux, J. Hoffmann, Bounded expectations: Resource analysis for probabilistic programs, in: PLDI 2018, ACM, 2018, pp. 496–512. doi:10.1145/3192366.3192394.

[27] E. Bartocci, L. Kovács, M. Stankovic, Automatic generation of moment-based invariants for prob-solvable loops, in: ATVA 2019, volume 11781 of *LNCS*, Springer, 2019, pp. 255–276. doi:10.1007/978-3-030-31784-3_15.

[28] T. Brázdil, J. Esparza, S. Kiefer, A. Kučera, Analyzing probabilistic pushdown automata, Formal Methods in System Design 43 (2013) 124–163. URL: https://doi.org/10.1007/s10703-012-0166-0. doi:10.1007/s10703-012-0166-0.

[29] T. Brázdil, S. Kiefer, A. Kučera, I. H. Vareková, Runtime analysis of probabilistic programs with unbounded recursion, Journal of Computer and System Sciences 81 (2015) 288–310. URL: https://doi.org/10.1016/j.jcss.2014.06.005. doi:10.1016/j.jcss.2014.06.005.

[30] H. S. Wilf, Generatingfunctionology, A. K. Peters, 2006.

[31] L. Collodi, M. Boreale, A. Pompa Di Gregorio, Towards algebraic analysis of probabilistic programs, in: Proceedings of the Italian Conference on Theoretical Computer Science (ICTCS), Pescara, Italy, 2025. URL: https://ceur-ws.org/Vol-4039/paper12.pdf.

[32] M. Boreale, L. Collodi, A. Pompa Di Gregorio, On the algebraic analysis of runtime distribution of probabilistic programs, 2026. Submitted.

[33] M. Boreale, Algorithms for exact and approximate linear abstractions of polynomial continuous systems, in: Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (HSCC 2018), ACM, 2018, pp. 207–216. doi:10.1145/3178126.3178137.

[34] M. Boreale, Complete algorithms for algebraic strongest postconditions and weakest preconditions in polynomial ODE's, in: SOFSEM 2018, volume 10706 of *LNCS*, Springer, 2018, pp. 442–455. doi:10.1007/978-3-319-73117-9_31.

[35] M. Boreale, Algebra, coalgebra, and minimization in polynomial differential equations, Logical Methods in Computer Science 15 (2019). doi:10.23638/LMCS-15(1:14)2019.

[36] M. Boreale, On the coalgebra of partial differential equations, in: Mathematical Foundations of Computer Science (MFCS 2019), volume 138 of *LIPIcs*, 2019, pp. 24:1–24:13. doi:10.4230/LIPIcs.MFCS.2019.24.

[37] M. Boreale, Automatic pre- and postconditions for partial differential equations, Information and Computation 285 (2022) 104860. doi:10.1016/j.ic.2021.104860.

[38] M. Boreale, D. Gorla, Algebra and coalgebra of stream products, in: S. Haddad, D. Varacca (Eds.), 32nd International Conference on Concurrency Theory, CONCUR 2021, Virtual Conference, August 24-27, 2021, volume 203 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, pp. 19:1–19:17. URL: https://doi.org/10.4230/LIPIcs.CONCUR.2021.19.

[39] M. Boreale, L. Collodi, Bayesian parameter estimation with guarantees via interval analysis and simulation, in: Verification, Model Checking, and Abstract Interpretation (VMCAI 2023), volume 13881 of *LNCS*, Springer, 2023, pp. 106–128. doi:10.1007/978-3-031-50521-8_7.

[40] M. Boreale, M. Paolini, On formally bounding information leakage by statistical estimation, in: Information Security (ISC 2014), volume 8783 of *LNCS*, Springer, 2014, pp. 216–236. doi:10.1007/978-3-319-13257-0_13.

[41] M. Boreale, F. Pampaloni, Quantitative information flow under generic leakage functions and adaptive adversaries, in: Formal Techniques for Distributed Objects, Components, and Systems (FORTE 2014), volume 8461 of *LNCS*, Springer, 2014, pp. 166–181. doi:10.1007/978-3-662-43613-4_11.

[42] M. Boreale, F. Pampaloni, Quantitative information flow under generic leakage functions and

adaptive adversaries, Log. Methods Comput. Sci. 11 (2015). URL: https://doi.org/10.2168/LMCS-11(4:5)2015.

[43] M. Boreale, F. Corradi, C. Viscardi, Relative privacy threats and learning from anonymized data, IEEE Trans. Inf. Forensics Secur. 15 (2020) 1379–1393. URL: https://doi.org/10.1109/TIFS.2019.2937640.

[44] L. Acciai, M. Boreale, G. Zavattaro, Behavioural contracts with request-response operations, Sci. Comput. Program. 78 (2013) 248–267. URL: https://doi.org/10.1016/j.scico.2011.10.007.

[45] M. Boreale, R. Bruni, R. D. Nicola, M. Loreti, CaSPiS: a calculus of sessions, pipelines and services, Mathematical Structures in Computer Science 25 (2015) 666–709. doi:10.1017/S0960129512000953.

[46] M. Boreale, D. Gorla, Approximate model counting, sparse XOR constraints and minimum distance, in: The Art of Modelling Computational Systems, volume 11760 of *LNCS*, Springer, 2019, pp. 363–378. doi:10.1007/978-3-030-31175-9_21.

[47] H. D. Menéndez, M. Boreale, D. Gorla, D. Clark, Output sampling for output diversity in automatic unit test generation, IEEE Transactions on Software Engineering 48 (2022) 295–308. doi:10.1109/TSE.2020.2987377.