Generating Functions for Probabilistic Programs via the Weighted Relational Model of Linear Logic

Ugo Dal Lago[1], Guido Fiorillo[2], and Paolo Pistone[3]

[1] University of Bologna, Italy, ugo.dallago@unibo.it
INRIA Sophia Antipolis, France
[2] Université Claude Bernard Lyon 1, France guido.fiorillo@ens-lyon.fr
[3] Université Claude Bernard Lyon 1, France paolo.pistone@ens-lyon.fr

1 Introduction

The recent years have seen a surge of interest towards probabilistic programming. Probabilistic programs pose difficult challenges already when one only considers discrete probabilistic choices (i.e. Bernoulli random variables): as opposed to deterministic programs, each execution of a probabilistic program can give a different result both in term of convergence and of its output; hence, while for non-deterministic programs one can study *qualitative* properties (does the program terminate or not?), for probabilistic programs one has to consider *quantitative* generalizations of such properties (what is the probability that the program terminates?).

However, quantitative properties can be difficult or even intractable in many contexts. For instance, the literature on *probabilistic model-checking* has studied the problems of determining the probability of termination or the expected running time for different computational models, leading to several results, both positive and negative, regarding the decidability and tractability of the underlying verification problem (see [1] for a recent survey). Beyond termination, one can consider the problem of *exact inference*, that is, of finding the posterior distribution of the values computed by a program. While efficient methods are known for finitary programs, in the presence of *unbounded* iteration (and hence the possibility of non-termination), inference is undecidable [19], and a clear picture of the decidable or tractable cases is still missing.

Properties like the probability of termination or the expected running time are often sensitive to how much the program is able to *duplicate* its inputs. This is why ideas and methods from linear logic have been widely applied in this area. In particular, the recent literature has explored methods based on the weighted relational model [12] and the related probabilistic coherent spaces (PCOH) [4]. Our starting point is indeed the basic observation that the interpretations of programs in both such models can be represented as formal power series [2], [13]. For example, the interpretation of a program $\mathcal{G}$ that can reduce to a finite set of outputs $[\![A]\!] = \{a_1, \dots a_k\}$ yields a family of n *generating functions* $A_k(z) = \sum_n \mathbf{P}(\mathcal{G} \to^n a_k) z^n$, where $\mathbf{P}(\mathcal{G} \to^n a_k)$ indicates the probability of $\mathcal{G}$ reducing to the value a_k in n steps.

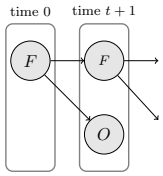
Our general goal is to explore the use of the concept of generating function, that is classical in enumerative and analytic combinatorics [5, 6], to identify classes of probabilistic programs whose quantitative properties can be extracted, in a computable way, from their interpretation in the weighted relational model or PCOH. Notably, while the combinatorial approach has already been explored in the field of imperative programming [7], through these linear logic models we manage to extend it to *higher-order* programming languages.

In the recent paper [11] we provided a first demonstration of this method by reproving (and extending) a result of [15] about probabilistic model checking using algebraic power series. In this abstract we first briefly recall this result, and then sketch a few new results and perspectives arising from this method.

$$S = Fe,$$
$$Fx = x \oplus_{\frac{1}{2}} F(Fx).$$

(a) PHORS computing a simple random walk: starting from Fe the number of F may either increase or decrease by one at each reduction, and termination occurs when this number reaches zero.

$$\begin{aligned} S &= HFO \\ F &= \mathsf{True} \oplus_q \mathsf{False} \\ O &= \mathsf{True} \oplus_p \mathsf{False} \\ Hxy &= \text{if } x \text{ then } H_{\mathrm{True}}xy \text{ else } H_{\mathrm{False}}xy \\ H_{\mathrm{True}}xy &= Hxy \oplus_r O \\ H_{\mathrm{False}}xy &= Hxy \oplus_s O \end{aligned}$$



(b) A Boolean PHORS and the corresponding iterated model: the functions $H, H_{\mathsf{True}}, H_{\mathsf{False}}$ describe the propagation of the model which, at each iteration, queries the Boolean variable F and, depending on the result, either produces an observable O or queries F again.

Figure 1: Examples of PHORS.

2 PHORS, aka. the Probabilistic λY-Calculus

In *higher-order* model-checking one studies the decidability and tractability of verifying properties of programs described in languages arising from the λ-calculus. As the termination property is undecidable for this language, the literature has mostly focused on restricted languages like the HORS (Higher Order Recursion Schemes), due to a seminal result [17] stating that all properties of HORS expressible in monadic second order logic (including termination) are decidable. A HORS is defined as the fixpoint of a finite system of recursive equations of the form $Fx_1 \dots x_n = t$. The functional variables F defined by such equations are called the *non-terminal symbols* of the PHORS. In the *probabilistic* HORS, or PHORS [8], equations $Fx_1 \dots x_n = t_L \oplus_p t_R$ involve probabilistic choices $\oplus_p$. The execution of a (P)HORS starts from a *source* non-terminal S and goes on by unrolling the equations. For example, the PHORS in Fig. 1a simulates a simple random walk. The (P)HORS are equivalent to the simply typed λY-calculus [18], that is, the simply typed λ-calculus enriched with fixpoints $Y : (\sigma \to \sigma) \to \sigma$ at all types σ (and with the choice operators $\oplus_p$ in the probabilistic case).

While qualitative properties like termination are decidable for the HORS, the corresponding quantitative properties of PHORS are not: establishing if a PHORS terminates with maximal probability (*almost sure termination*, AST) is undecidable [8]. It is then interesting to study for which classes of PHORS problems like AST (or the related PAST problem, asking whether the expected running time is finite) are tractable or, at least, decidable.

To study exact inference we also consider an extension of the PHORS with a type of Booleans and if-then-else, equivalent to a probabilistic version of finite PCF [16]. This language represents both finite graphical models (e.g. Bayesian networks) and *unbounded* models [9, Ch. 6], iterating some basic network template, as in the example in Fig. 1b. A Boolean PHORS $\mathcal{G}$ reduces with probability p to False and with probability q to True (with $p+q \leq 1$), noted $\mathcal{G} \sim \mathrm{Bernoulli}(p, q)$. The (generally undecidable) problem of exact inference is to find these p, q.

3 Finiteness and Algebraicity

The weighted relational model of linear logic is a well-studied denotational semantics in which a program $M : \sigma \to \tau$ is interpreted as a matrix $[\![M]\!] : !|\![\sigma]\!| \times |\![\tau]\!| \to \mathcal{Q}$ with coefficients in

a continuous semiring $\mathcal{Q}$, where $[\![\sigma]\!], [\![\tau]\!]$ are countable sets and $!X$ denotes the set of finite multisets on X. Our starting observation is that this matrix can be equivalently described as a family of formal power series $[\![M]\!](x)_b = \sum_{\mu \in ![\![\sigma]\!]} [\![M]\!]_{\mu,b} x^\mu$ (for $b \in [\![\tau]\!]$), where for all $x \in \mathcal{Q}^{[\![\sigma]\!]}$, $x^\mu = \prod_{a \in [\![\sigma]\!]} x_a^{\mu(a)}$.

Interpreting a PHORS in this model corresponds then to finding the *least fixpoint* of an *infinite* system $[\![\mathcal{G}]\!] = \Phi([\![\mathcal{G}]\!])$ of equations between power series. In particular, by choosing $\mathcal{Q} = \mathbb{R}_{\geq 0}^{+\infty}\{\!\{z\}\!\}$, the semiring of formal power series in z with positive real coefficients, the interpretation of the source symbol S of a PHORS yields a generating function $[\![S]\!](z) = \sum_{n \geq 0} \mathbf{P}(\mathcal{G} \Downarrow_n) z^n$ describing the probability of termination in n reduction steps.

Unfortunately, the infinitary systems produced by the PHORS are not, in general, tractable with algebraic or computational tools: this model is not, globally, computable. However, what if we could ensure, by some appropriate restrictions on the PHORS $\mathcal{G}$, that the resulting equational system is equivalent to some *finite* polynomial system? This is the approach we follow in [11], where we introduce a semantic condition of *finiteness* to this effect. For the technical definition of a *finitary family* see [11]; we here report only its fundamental property:

Proposition 1. *Suppose $(\Phi_i(\vec{x}, \vec{y}))_{i \in I}$ is a finitary family of power series. If $r_i(\vec{y})$ is the least solution of the system $r_i(\vec{y}) = \Phi_i(r_i(\vec{y}), \vec{y})$, then* all but finitely many *of the $r_i(\vec{y})$ are equal to 0 and the remaining ones are solution of a finite family of polynomial equations $(r_{j_\ell}(\vec{x}) = p_\ell(r_{j_1}(\vec{x}), \dots r_{j_n}(\vec{x}), \vec{y}))_{\ell \in \{1 \dots n\}}$.*

Now, the main point is to identify a class of PHORS whose interpretation satisfies the previous condition. We do this by introducing the class of *bounded PHORS* (PBHORS$^\infty$), which extends the *affine PHORS* from [15]. The bounded PHORS are defined using a linear type system involving *graded exponentials* (where a graded type of the form $!_k\varphi \multimap \psi$ describes a program that can use an input of type φ *at most k times* to produce an output of type ψ). More precisely, a type judgement in this system is of the form $\mathcal{N} \mid \Delta \vdash t : \sigma$, where:

- $\mathcal{N}$ is a *non-linear* environment for the non-terminal symbols;
- Δ is a *graded* environment for the other variables, comprising *both* finite and infinite grades.

The type system is designed to do two things: first, restrict the use of variables with an infinite grade (those which can be duplicated at will), as these may only play the role of the *parameters* $\vec{y}$ in the power series $r_i(\vec{y})$ above; second, enforce a *fixpoint condition* ensuring that for any equation $Fx_1 \dots x_n = t$, the non-terminal function symbol F and the term $\lambda x_1 \dots x_n.t$ agree on the grades assigned to $x_1, \dots, x_n$; this condition warrants that the least fixpoint $r_i(\vec{y})$ is determined by only a *finite* number of conditions. This leads to the following:

Theorem 1. *For all PBHORS$^\infty$ $\mathcal{G}$, the interpretation $[\![\mathcal{G}]\!]$ is the least solution of a finite polynomial system. In particular, the generating function $[\![S]\!](z) = \sum_{i \geq 0} \mathbf{P}(\mathcal{G} \Downarrow_n) z^n$ is algebraic (i.e. there is a polynomial $p(z, w)$ with rational coefficients such that $p(z, [\![S]\!](z)) = 0$).*

Using Tarski's decidability of the first-order existential theory of the reals, we obtain then:

Theorem 2. *For all PBHORS$^\infty$, the AST and PAST problems are decidable (EXPTIME if order is fixed).*

Example 1. *The PHORS in Fig. 1a is a PBHORS$^\infty$, its generating function $[\![S]\!](z)$ annihilates the polynomial $p(z, y) = \frac{1}{2}y^2 z - y + \frac{1}{2}z$, which can be used to show that it is AST but not PAST.*

4 New Results and Perspectives

We now list a few new applications and perspectives of the results in [11].

Rational PHORS It is well-known that algebraic numbers are computable, in the sense of being approximable to arbitrary degree. Similarly, algebraic power series come with several well-known exact or approximate computational methods. However, just like rational numbers are computable in an exact way, stronger (i.e. more tractable) methods exist when considering rational power series, that is power series that can be expressed as a ratio of two polynomials.

Starting from the type system described above, it is not difficult to design a more restricted discipline which enforces rationality. Indeed, it is well-known that rational power series $r(\vec{y}) = \frac{p(\vec{y})}{q(\vec{y})}$ can be characterized as the solution of *linear* recursive systems $X_i = a_i(\vec{y})X_i + b_i(\vec{y})$ [10]. Such systems can be solved effectively (and efficiently) by means of the Gauss algorithm [14].

In our system, this linearity constraint can be ensured by restricting the discipline of non-terminal symbols: in the typing context $\mathcal{N} \mid \Delta \vdash t : \sigma$ we now require that the variables in $\mathcal{N}$ be used in an *affine* way, that is, possibly discarded but never duplicated. Call PRatHORS$^\infty$ the class of PHORS typable in this restricted system. We then have the following:

Theorem 3. *For all PRatHORS$^\infty$ $\mathcal{G}$, the interpretation $[\![\mathcal{G}]\!]$ is the least solution of a finite linear system. In particular, the generating function $[\![S]\!](z) = \sum_{i\geq 0} \mathbf{P}(\mathcal{G} \Downarrow_n) z^n$ is rational and its closed form can be computed in polynomial time (when order is fixed). Hence, the probability of termination is a rational number which can be computed explicitly from the typing derivations.*

Exact Inference As suggested above, given the computational nature of rational and algebraic power series, we can exploit Theorems 1 and 3 to devise solutions to the exact inference problem for a Boolean PHORS.

Theorem 4. *For all Boolean PHORS $\mathcal{G} \sim$ Bernoulli(p, q):*

- *if $\mathcal{G}$ is a PBHORS$^\infty$, then for any given $0 < \epsilon \in \mathbb{Q}$ we can compute approximations $p', q' \in \mathbb{Q}$ such that $|p - p'| < \epsilon, |q - q'| < \epsilon$;*
- *if $\mathcal{G}$ is a PRatHORS$^\infty$, then p and q are rational and can be computed effectively.*

Example 2. *The PHORS in Fig. 1b is a PRatHORS$^\infty$, with rational generating functions $[\![S]\!]_{\mathrm{True}}(z) = p\frac{\alpha z^3}{1-\beta z^2}$ and $[\![S]\!]_{\mathrm{False}}(z) = \overline{p}\frac{\alpha z^3}{1-\beta z^2}$, where $\alpha = q\overline{r} + \overline{q}s$ and $\beta = qr + \overline{q}s$, yielding, for e.g. $p = q = \frac{1}{3}$ and $r = s = \frac{1}{2}$, $\mathcal{G} \sim$ Bernoulli$(\frac{1}{3}, \frac{2}{3})$.*

Asymptotic Estimates The algebraicity of the generating function of a PHORS is not only useful to answer exact problems like AST, but also to estimate *how fast* the program converges. We provide here a sketch of how, using the methods from analytic combinatorics [5], it might be possible to make such estimates. If the radius of convergence of the power series $[\![\mathcal{G}]\!](z)$ is $\rho > 1$, then $\mathbf{P}(\mathcal{G} \Downarrow_n) = O(1/\rho^n)$. On the other side, if $\rho = 1$, (the case $\rho < 1$ can be excluded by standard convergence results), the asymptotic behaviors is regulated by the nature of the dominant (i.e. smallest in modulus) singularities of $[\![\mathcal{G}]\!](z)$: for example, if $[\![S]\!](z)$ has a unique dominant singularity of 'square root type' (i.e. it has a Puiseaux expansion of the form $[\![S]\!](z) = \sum_{k\geq 1} c_k(z-1)^{k/2}$), one obtains that $\mathbf{P}(\mathcal{G} \Downarrow_n) = O(n^{3/2})$.

While the problem of finding the radius of convergence and the Puiseaux expansions is generally not computable, when the function is algebraic (so there exists an irreducible polynomial $p(z, w)$ s.t $p(z, [\![S]\!](z)) = 0$) singularities are located at the points (z, w) for which $\frac{\partial p(z,w)}{\partial w} = 0$ and algorithms exist to compute both [5], [3]. We obtain for example that:

Proposition 2. *Let $\mathcal{G}$ be a PBHORS$^\infty$ and let $p(z, w)$ be the irreducible annihilating polynomial of $[\![\mathcal{G}]\!](z)$ (that can be computed effectively). Let $\mathbf{P}(\mathcal{G} \Downarrow_{\leq n})$ be the probability of termination* in no more than n steps. *If for no $|z| < 1$ we can find a w s.t. $\frac{\partial p(z,w)}{\partial w} = 0$, then there exists a $r < 1$ s.t. $\mathbf{P}(\mathcal{G} \Downarrow) = \mathbf{P}(\mathcal{G} \Downarrow_{\leq n}) + O(r^{n+1})$ for $n \to +\infty$.*

References

[1] Christel Baier and Joost-Pieter Katoen. *Principles of Model-Checking*. The MIT Press, 2008.

[2] Davide Barbarossa and Paolo Pistone. Tropical mathematics and the lambda-calculus II: Tropical geometry of probabilistic programming languages. *Proc. ACM Program. Lang.*, 10(POPL), January 2026.

[3] D.V Chudnovsky and G.V Chudnovsky. On expansion of algebraic functions in power and puiseux series, i. *Journal of Complexity*, 2(4):271–294, 1986.

[4] Thomas Ehrhard, Michele Pagani, and Christine Tasson. Full abstraction for probabilistic PCF. *J. ACM*, 65(4):23:1–23:44, 2018.

[5] Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009.

[6] Manuel Kauers and Peter Paule. *The Concrete Tetrahedron - Symbolic Sums, Recurrence Equations, Generating Functions, Asymptotic Estimates*. Texts & Monographs in Symbolic Computation. Springer, 2011.

[7] Lutz Klinkenberg, Christian Blumenthal, Mingshuai Chen, Darion Haase, and Joost-Pieter Katoen. Exact bayesian inference for loopy probabilistic programs using generating functions. *Proceedings of the ACM on programming languages*, 8(OOPSLA1):pages 127, 2024.

[8] Naoki Kobayashi, Ugo Dal Lago, and Charles Grellois. On the termination problem for probabilistic higher-order recursive programs. *Log. Methods Comput. Sci.*, 16(4), 2020.

[9] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, Cambridge, Massachusetts, 2009.

[10] Werner Kuich and Arto Salomaa. *Semirings, Automata, Languages*, volume 5 of *EATCS Monographs on Theoretical Computer Science*. Springer, 1986.

[11] Ugo Dal Lago, Guido Fiorillo, and Paolo Pistone. On higher-order probabilistic verification via the weighted relational model of linear logic. To appear in Proceedings LICS 2026, 2026.

[12] Jim Laird, Giulio Manzonetto, Guy McCusker, and Michele Pagani. Weighted relational models of typed lambda-calculi. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*, pages 301–310. IEEE Computer Society, 2013.

[13] François Lamarche. Quantitative domains and infinitary algebras. *Theoretical Computer Science*, 94(1):37–62, 1992.

[14] Daniel J. Lehmann. Algebraic structures for transitive closure. *Theoretical Computer Science*, 4(1):59–76, 1977.

[15] Guanyan Li, Andrzej S. Murawski, and Luke Ong. Probabilistic verification beyond context-freeness. In Christel Baier and Dana Fisman, editors, *LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2 - 5, 2022*, pages 33:1–33:13. ACM, 2022.

[16] Ralph Loader. Finitary pcf is not decidable. *Theoretical Computer Science*, 266(1):341–364, 2001.

[17] C.-H. L. Ong. On model-checking trees generated by higher-order recursion schemes. In *Proceedings of the 21st Annual IEEE Symposium on Logic in Computer Science*, LICS '06, pages 81–90, USA, 2006. IEEE Computer Society.

[18] Richard Statman. On the lambda y calculus. In *Proceedings of the 17th Annual IEEE Symposium on Logic in Computer Science*, LICS '02, pages 159–166, USA, 2002. IEEE Computer Society.

[19] Mateo Torres-Ruiz, Robin Piedeleu, Alexandra Silva, and Fabio Zanasi. On Iteration in Discrete Probabilistic Programming. In Jakob Rehof, editor, *9th International Conference on Formal Structures for Computation and Deduction (FSCD 2024)*, volume 299 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20:1–20:21, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.