Automated Refutation of Probabilistic and non-Probabilistic Program Equivalence

Krishnendu Chatterjee[1], Ehsan Kafshdar Goharshady[1], Petr Novotný[2], and Đorđe Žikelić[3]

[1] Institute of Science and Technology Austria, Klosterneuburg, Austria
[2] Masaryk University, Brno, Czech Republic
[3] Singapore Management University, Singapore

Programs and probabilistic programs. Classically, imperative non-probabilistic programs are programs whose execution is fully determined by the input: given a fixed input, the program always produces the same output. Probabilistic programs (PPs) extend non-probabilistic programs by incorporating *probabilistic fragments* [29,37,11], i.e. constructs that involve randomness. The two most common such constructs are *probabilistic branchings*, where a branch is taken with a certain probability, and *sampling statements*, where a value is drawn at random from a probability distribution. Rather than producing a single deterministic output, a PP gives rise to an *output distribution* over possible outputs. They have been adopted in many application domains including machine learning [28], security [7,8] and robotics [43], and formal analysis of PPs has become a very active research area.

Static analysis of PPs. Given that PPs produce output distributions, in comparison to deterministic programs that produce a single output value for a given input, PPs are inherently more difficult to analys both in theory [32] and in practice [39,25].

Recent years have seen much work on static analysis of PPs, where the aim is to formally prove temporal or input/output properties by analyzing the source code directly and instead of repeatedly sampling randomized executions of PPs. There have been significant developments on static analysis with respect to termination [16,19,33,36,20,1,21], reachability [42], safety [22,41,14,13,5], cost [40,47,45], input/output [23], runtime [35], productivity for infinite streams [3], sensitivity [6,46,2] or differential privacy [4] properties.

Equivalence and similarity refutation. In this work, we focus on *relational analysis* of programs. The goal of relational analysis is to prove properties of *pairs* of programs. A prominent example of relational property is *equivalence*: two programs are equivalent if they produce the same outputs. In this paper, we consider static analysis of equivalence and similarity of output distributions of PP pairs. Equivalence and similarity are two fundamental properties of probabilistic programming systems that are essential for their correctness *both in implementation and in compilation*. We study the following two problems:

1. *Equivalence refutation problem.* Given a pair of probabilistic programs, prove that their output distributions are not equivalent.

2. *Similarity refutation problem.* Given a pair of probabilistic programs, prove a lower bound on Kantorovich distance [44] between their output distributions.

We emphasise that while our work is focused on PPs, equivalence of non-probabilistic programs is a special case of the equivalence refutation problem in PPs and our techniques apply to it as well.

Relevance. Equivalence checking and refutation are crucial for ensuring program correctness or for bug detection. For instance, if we have two different implementations of a model or two (randomized) algorithms designed to solve a given problem, the equivalence refutation analysis allows us to detect whether the two programs give rise to different outputs [38]. Such an analysis allows, e.g., bug detection in samplers from probability distributions [17] or in implementations of cryptographic protocols [9]. Equivalence refutation analysis also allows bug detection in probabilistic program compilers. For instance, it was observed by [26] that a 10-line probabilistic program in Stan [27] executes over 6000 lines of code of Stan implementation. Hence, detecting compilation bugs by testing may be a challenging task even for small programs. Static equivalence refutation analysis allows bug detection in compilation by comparing the source code to its intermediate representation without program execution.

While equivalence refutation analysis only proves that output distributions of two programs are not equivalent, similarity analysis provides more fine-grained information and *quantifies the difference* between two output distributions (e.g., the difference between the output distribution induced by a sampler and the ground probability distribution whose samples we wish to generate [17]).

Prior work. Equivalence and similarity are *relational properties* that are defined with respect to a program *pair*. The prior work on relational reasoning about probabilistic programs focused on developing logical systems for such reasoning [24] rather than on automation; or on sensitivity analysis [6,2,46,31,4,7], whose aims and assumptions differ from equivalence analysis. *Automated* methods for formal analysis of probabilistic program equivalence have been developed for *finite-state* probabilistic systems [38,34,10]. However, probabilistic programs defined over real or integer-valued variables or containing sampling from continuous probability distributions (such as normal or uniform) all give rise to infinite-state programs.

On the other hand, there is a huge body of work on sampling-based statistical testing of equivalence and similarity of two probability distributions [12,18], see [15] for a survey. While these methods provide extremely useful information and do not impose syntactic restrictions on probability distributions that they can analyze, they suffer from two key limitations. First, guarantees on the correctness of their analyses are *statistical*, meaning that there is always a non-zero probability that the analysis results are incorrect. Second, sampling-based methods suffer from scalability issues if the probabilistic program needs to be executed for a long time. To the best of our knowledge, no prior work has proposed an *automated* method for equivalence and similarity refutation analyses in *infinite-state PPs* that provide *formal guarantees* on the correctness of their results.

Our approach – automated formal analysis via expectation martingales. We present a new method for static equivalence and similarity refutation analyses of probabilistic program pairs. To the best of our knowledge, we present the first method that provides the following desired features:

1. *Automation.* Our method is fully automated.
2. *Infinite-state programs.* Our method is applicable to infinite-state probabilistic programs.
3. *Formal guarantees.* Our method provides formal guarantees on the correctness of its results. Specifically, we introduce a sound and complete proof rule for equivalence refutation.

Technical challenges. Given two programs, our method refutes their equivalence by computing a function f over their output variables such that the expected value of f at the output of the two programs differs. Our method searches for such a function by computing it simultaneously with an *upper expectation supermartingale (UESM)* for the first program and a *lower expectation submartingale (LESM)* for the second program. UESMs and LESMs, notions similar to cost supermartingales [47] or super- and sub-invariants [30], provide sound proof rules for deriving upper and lower bounds on the expected value of a function on program output in probabilistic programs. We show that UESMs and LESMs together with the function f over outputs provide a sound and complete proof rule for refuting equivalence of programs. To the best of our knowledge, no martingale-based approach has been used in prior work for static analysis of *relational* properties of probabilistic program pairs. The non-trivial challenge is to simultaneously compute the function f, along with two martingales (one submartingale and other supermatingale), which we achieve via a constraint solving-based approach.

Contributions. Our contributions can be summarized as follows:

1. We present a method for *static equivalence and similarity refutation* of PP pairs, which is *automated*, applicable to *infinite-state* PPs and provides *formal guarantees* on the correctness of its results.
2. We formulate *a sound and complete proof rule for equivalence the refutation problem* via UESMs and LESMs.
3. We present *fully automated algorithms* for equivalence and similarity refutation analyses in PPs, based on the above proof rules. The algorithms simultaneously compute a UESM and an LESM for two PPs together with a function over their output variables. They are applicable to numerical PPs with polynomial arithmetic expressions that may contain sampling instructions from both discrete and continuous probability distributions.
4. Our *experimental evaluation* demonstrates the ability of our method to refute equivalence and compute lower bounds on Kantorovich distance for a variety of program pairs.

References

1. Agrawal, S., Chatterjee, K., Novotný, P.: Lexicographic ranking supermartingales: an efficient approach to termination of probabilistic programs. Proc. ACM Program. Lang. **2**(POPL) (2018)
2. Aguirre, A., Barthe, G., Hsu, J., Kaminski, B.L., Katoen, J., Matheja, C.: A pre-expectation calculus for probabilistic sensitivity. Proc. ACM Program. Lang. **5**(POPL) (2021)
3. Aguirre, A., Barthe, G., Hsu, J., Silva, A.: Almost sure productivity. In: ICALP (2018)
4. Albarghouthi, A., Hsu, J.: Synthesizing coupling proofs of differential privacy. Proc. ACM Program. Lang. **2**(POPL) (2018)
5. Bao, J., Trivedi, N., Pathak, D., Hsu, J., Roy, S.: Data-driven invariant learning for probabilistic programs. In: CAV (2022)
6. Barthe, G., Espitau, T., Grégoire, B., Hsu, J., Strub, P.: Proving expected sensitivity of probabilistic programs. Proc. ACM Program. Lang. **2**(POPL) (2018)
7. Barthe, G., Gaboardi, M., Grégoire, B., Hsu, J., Strub, P.: Proving differential privacy via probabilistic couplings. In: LICS (2016)
8. Barthe, G., Gaboardi, M., Hsu, J., Pierce, B.C.: Programming language techniques for differential privacy. ACM SIGLOG News **3**(1) (2016)
9. Barthe, G., Grégoire, B., Béguelin, S.Z.: Formal certification of code-based cryptographic proofs. In: POPL (2009)
10. Barthe, G., Jacomme, C., Kremer, S.: Universal equivalence and majority of probabilistic programs over finite fields. ACM Trans. Comput. Log. **23**(1) (2022)
11. Barthe, G., Katoen, J.P., Silva, A.: Foundations of probabilistic programming. Cambridge University Press (2020)
12. Batu, T., Fortnow, L., Rubinfeld, R., Smith, W.D., White, P.: Testing closeness of discrete distributions. J. ACM **60**(1) (2013)
13. Batz, K., Chen, M., Junges, S., Kaminski, B.L., Katoen, J., Matheja, C.: Probabilistic program verification via inductive synthesis of inductive invariants. In: TACAS (2023)
14. Beutner, R., Ong, C.L., Zaiser, F.: Guaranteed bounds for posterior inference in universal probabilistic programming. In: PLDI (2022)
15. Canonne, C.L.: A survey on distribution testing: Your data is big. but is it blue? Theory of Computing (2020)
16. Chakarov, A., Sankaranarayanan, S.: Probabilistic program analysis with martingales. In: CAV (2013)
17. Chakraborty, S., Meel, K.S.: On testing of uniform samplers. In: AAAI (2019)
18. Chan, S., Diakonikolas, I., Valiant, P., Valiant, G.: Optimal algorithms for testing closeness of discrete distributions. In: SODA (2014)
19. Chatterjee, K., Fu, H., Novotný, P., Hasheminezhad, R.: Algorithmic analysis of qualitative and quantitative termination problems for affine probabilistic programs. ACM Trans. Program. Lang. Syst. **40**(2) (2018)
20. Chatterjee, K., Goharshady, A.K., Meggendorfer, T., Zikelic, D.: Sound and complete certificates for quantitative termination analysis of probabilistic programs. In: CAV (2022)
21. Chatterjee, K., Goharshady, E.K., Novotný, P., Zárevúcky, J., Zikelic, D.: On lexicographic proof rules for probabilistic termination. In: FM (2021)
22. Chatterjee, K., Novotný, P., Zikelic, D.: Stochastic invariants for probabilistic termination. In: POPL (2017)

23. Chen, M., Katoen, J., Klinkenberg, L., Winkler, T.: Does a program yield the right distribution? - verifying probabilistic programs via generating functions. In: CAV (2022)
24. Culpepper, R., Cobb, A.: Contextual equivalence for probabilistic programs with continuous random variables and scoring. In: ESOP (2017)
25. Dutta, S., Legunsen, O., Huang, Z., Misailovic, S.: Testing probabilistic programming systems. In: FSE (2018)
26. Dutta, S., Zhang, W., Huang, Z., Misailovic, S.: Storm: program reduction for testing and debugging probabilistic programming systems. In: FSE (2019)
27. Gelman, A., Lee, D., Guo, J.: Stan: A probabilistic programming language for bayesian inference and optimization. Journal of Educational and Behavioral Statistics **40**(5) (2015)
28. Ghahramani, Z.: Probabilistic machine learning and artificial intelligence. Nat. **521**(7553), 452–459 (2015)
29. Gordon, A.D., Henzinger, T.A., Nori, A.V., Rajamani, S.K.: Probabilistic programming. In: FOSE (2014)
30. Hark, M., Kaminski, B.L., Giesl, J., Katoen, J.: Aiming low is harder: induction for lower bounds in probabilistic program verification. Proc. ACM Program. Lang. **4**(POPL) (2020)
31. Huang, Z., Wang, Z., Misailovic, S.: Psense: Automatic sensitivity analysis for probabilistic programs. In: ATVA (2018)
32. Kaminski, B.L., Katoen, J., Matheja, C.: On the hardness of analyzing probabilistic programs. Acta Informatica **56**(3) (2019)
33. Kaminski, B.L., Katoen, J., Matheja, C., Olmedo, F.: Weakest precondition reasoning for expected runtimes of randomized algorithms. J. ACM **65**(5) (2018)
34. Kiefer, S., Murawski, A.S., Ouaknine, J., Wachter, B., Worrell, J.: Language equivalence for probabilistic automata. In: CAV (2011)
35. Kura, S., Urabe, N., Hasuo, I.: Tail probabilities for randomized program runtimes via martingales for higher moments. In: TACAS (2019)
36. McIver, A., Morgan, C., Kaminski, B.L., Katoen, J.: A new proof rule for almost-sure termination. Proc. ACM Program. Lang. **2**(POPL) (2018)
37. van de Meent, J., Paige, B., Yang, H., Wood, F.: An introduction to probabilistic programming. CoRR **abs/1809.10756** (2018), `http://arxiv.org/abs/1809.10756`
38. Murawski, A.S., Ouaknine, J.: On probabilistic program equivalence and refinement. In: CONCUR (2005)
39. Nandi, C., Grossman, D., Sampson, A., Mytkowicz, T., McKinley, K.S.: Debugging probabilistic programs. In: MAPL@PLDI (2017)
40. Ngo, V.C., Carbonneaux, Q., Hoffmann, J.: Bounded expectations: resource analysis for probabilistic programs. In: PLDI (2018)
41. Sankaranarayanan, S., Chakarov, A., Gulwani, S.: Static analysis for probabilistic programs: inferring whole program properties from finitely many paths. In: PLDI (2013)
42. Takisaka, T., Oyabu, Y., Urabe, N., Hasuo, I.: Ranking and repulsing supermartingales for reachability in randomized programs. ACM Trans. Program. Lang. Syst. **43**(2) (2021)
43. Thrun, S.: Probabilistic algorithms in robotics. AI Mag. **21**(4) (2000)
44. Villani, C.: Topics in optimal transportation, vol. 58. American Mathematical Soc. (2021)
45. Wang, D., Hoffmann, J., Reps, T.W.: Central moment analysis for cost accumulators in probabilistic programs. In: PLDI (2021)

46. Wang, P., Fu, H., Chatterjee, K., Deng, Y., Xu, M.: Proving expected sensitivity of probabilistic programs with randomized variable-dependent termination time. Proc. ACM Program. Lang. **4**(POPL) (2020)
47. Wang, P., Fu, H., Goharshady, A.K., Chatterjee, K., Qin, X., Shi, W.: Cost analysis of nondeterministic probabilistic programs. In: PLDI (2019)