Statistical Symbolic Execution with Informed Sampling*

Antonio Filieri[a], Corina S. Păsăreanu[b], Willem Visser[c], and
Jaco Geldenhuys[c]

[a]University of Stuttgart, Stuttgart, Germany
[b]Carnegie Mellon Silicon Valley, NASA Ames, Moffet Field, CA, USA
[c]Stellenbosch University, Stellenbosch, South Africa

Abstract:
Probabilistic program analysis aims at quantifying the probability of a target event to occur during a program execution. Recent approaches exploit symbolic execution to compute the constraints on the inputs leading to the occurrence of a target event; the solution space for such constraints is then *quantified* given a probabilistic usage profile, which characterizes each input variable by a probability distribution over its possible values. Despite their generality and accuracy, these exhaustive approaches suffer scalability issues for large programs.

To address this issue, we propose a statistical symbolic execution technique that performs Monte Carlo sampling of the symbolic program paths and uses the obtained information for Bayesian estimation and hypothesis testing with respect to the probability of reaching the target events. To speed up the convergence of the statistical analysis, we propose Informed Sampling, an iterative symbolic execution that first explores the paths that have high statistical significance, prunes them from the state space and guides the execution towards less likely paths. The technique combines Bayesian estimation with a partial exact analysis for the pruned paths leading to provably faster convergence of the statistical analysis. We have implemented statistical symbolic execution with informed sampling in the Symbolic PathFinder tool and evaluated experimentally its effectiveness.

Several techniques have been proposed recently for the probabilistic analysis of programs [BFd$^+$14, FPV13, GDV12, LPD$^+$14]. These techniques have multiple applications, ranging from program understanding and debugging, to computing reliability of software operating in uncertain environments, to probabilistic programming. For example, in previous work [FPV13, GDV12], we described a bounded symbolic execution of a program that uses a quantification procedure over the collected symbolic constraints to *count* the inputs that follow the explored program paths. These counts are then used to compute the probability of executing different paths through the program (or of violating program assertions), under given probabilistic usage profiles. While promising, these exact techniques have scalability issues due to the large number of symbolic paths to be explored.

To address this problem we describe a statistical symbolic execution technique that uses randomized sampling of the symbolic paths. For deciding termination of sampling we investigate two different criteria: Bayesian estimation and hypothesis testing. The first is used to estimate the probability of executing designated program paths while the latter is used to test a given hypothesis about such probability. Unlike in a typical statistical setting where one samples randomly across a concrete input domain, our samples are

*This paper reports a summary of [FPVG14]. Please refer to the original paper for a complete exposition.

done in the context of symbolic execution, according to conditional probabilities computed at each branching point in the program. This approach is similar to statistical model checking [YS02, ZPC13], with the difference that we work with code and compute the probability of executions based on the provided probabilistic usage profile.

When using Bayesian estimation, the randomized sampling terminates when pre-specified confidence and error bounds (accuracy) have been achieved. The answer to the analysis problem is not guaranteed to be correct, but the probability of a wrong answer can be made arbitrarily small [ZPC13]. However, in practice, the convergence to an answer might be very slow. Hypothesis testing can be faster [ZPC13], but both techniques may require a very large number of sample paths to achieve the desired statistical confidence.

To speed up both methods, we propose *Informed Sampling* (IS), an iterative technique combining statistical methods with partial exact analysis. At each iteration, IS randomly samples a set of execution paths and performs a statistical analysis of the sample. The probability of sampling each path is proportional to the number of input values following it under the specified usage profile not to bias the sample. If the statistical method converged, its result is returned. Otherwise, the already sampled paths are pruned out from the execution tree and analyzed exactly. The next iteration will then focus on the analysis of only the remaining part of the execution tree, increasing also the chances of selecting low probability paths that might have not been sampled (and pruned) previously.

For pruning the sampling space we propose an efficient procedure that leverages the *counts* of the inputs associated with each explored symbolic path and subtracts them from the counts of all the prefixes along the path. For estimating the probability results we propose a combination of exact analysis (for the paths that are pruned in previous iterations) and Bayesian statistical analysis (for the paths sampled in the current iteration over the pruned state space). The combined estimator converges faster to the prescribed confidence goals, it is provably unbiased, and guarantees the termination of the analysis [FPVG14].

For a complete description of the approach and its experimental evaluation see [FPVG14].

References

[BFd$^+$14] Mateus Borges, Antonio Filieri, Marcelo d'Amorim, Corina S. Păsăreanu, and Willem Visser. Compositional Solution Space Quantification for Probabilistic Software Analysis. PLDI, pages 123–132. ACM, 2014.

[FPV13] Antonio Filieri, Corina S. Păsăreanu, and Willem Visser. Reliability Analysis in Symbolic Pathfinder. ICSE, pages 622–631. IEEE, 2013.

[FPVG14] Antonio Filieri, Corina S. Păsăreanu, Willem Visser, and Jaco Geldenhuys. Statistical Symbolic Execution with Informed Sampling. FSE, pages 437–448. ACM, 2014.

[GDV12] Jaco Geldenhuys, Matthew B. Dwyer, and Willem Visser. Probabilistic Symbolic Execution. ISSTA, pages 166–176. ACM, 2012.

[LPD$^+$14] Kasper Luckow, Corina S. Păsăreanu, Matthew B. Dwyer, Antonio Filieri, and Willem Visser. Exact and Approximate Probabilistic Symbolic Execution for Nondeterministic Programs. ASE, pages 575–586. ACM, 2014.

[YS02] Håkan L. S. Younes and Reid G. Simmons. Probabilistic Verification of Discrete Event Systems Using Acceptance Sampling. CAV, pages 223–235. Springer, 2002.

[ZPC13] Paolo Zuliani, Andr Platzer, and Edmund M. Clarke. Bayesian statistical model checking with application to Stateflow/Simulink verification. *Form Method Syst Des*, 43(2):338–367, 2013.