Probabilistic Symbolic Execution*

No Author Given

No Institute Given

Abstract. The abstract should briefly summarize the contents of the paper in 150–250 words

Keywords: Probabilisitic Programming · Symbolic Execution · Logic · Verification.

1 Introduction

2 Overview

3 Algorithm

Sumit: Layout the Algorithm text and Algorithm in package Sumit: In 4 pages

Algorithm 1: Probabilistic Symbolic Execution Algorithm

```
1 set the current state to T_sc^→;
2 E^→ ← getStateList(T_sc^→);
3 for s^→ ∈ E^→ do
4 |   getSolve(s^→)
```

4 Implementation

We build on top of KLEE [1], use Z3 [2] for solving the SMT formula for computing the $\mathbb{E}[v]$ values

5 Experimental Evalutaion

Sumit: What experiments and examples to work on? Sumit: In 4 pages

* We submit an online anonymous docker image of our tool along with a ZIP file containing all experimental artifacts and scripts for regeneration.

5.1 (RQ1) AxProf (Baseline) vs. PSETool

- We run AxProf with default settings where it samples the values of the forall variables at random and simulates multiple runs of the program with different values of the probabilistic variables for each sampled value of the forall variables.
- PSETool is also run with default settings, the values of the forall variables are determined by solving the path constraints via symbolic execution using KLEE and for each setting of the forall variables, the program is run multiple times with different values of the probabilistic variables.

5.2 (RQ2) Hybrid Strategy

We run PSETool under a hybrid strategy here. Symbolic Execution is used to find the worst case forall settings that lead to the violation of the probabilistic assert in the program. For each of the worst case setting found using symbolic execution, AxProf runs the program multiple times over different values of the probabilistic variables. Any error detected by AxProf is considered a violation of the probabilistic assert.

6 Related Work

7 Conclusion

References

1. Cadar, C., Dunbar, D., Engler, D.: Klee: Unassisted and automatic generation of high-coverage tests for complex systems programs. In: Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation. p. 209–224. OSDI'08, USENIX Association, USA (2008)
2. de Moura, L., Bjørner, N.: Z3: An efficient smt solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. pp. 337–340. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)