Dynamic symbolic execution (concolic execution)

Seminar: Understanding of configurable software systems

Bipin Oli

Advisors: Prof. Sven Apel,
Christian Hechtl

Saarland Informatics Campus,
Saarland University

1 Abstract

Concolic execution [6] is a software verification technique that performs symbolic execution together with concrete input values. Concrete values are selected with the help of a constraint solver to guide a program flow in a particular direction. The selection of concrete values helps to scale the verification to a larger program as it makes the symbolic constraints smaller by selecting specific branches in the program. Compared to random execution, this allows us to guide the analysis in a direction likely to have bugs which makes this technique powerful. However, in doing so we sacrifice the completeness of the analysis in favor of the depth of analysis. The sheer number of branches in a large program makes it difficult to perform a complete analysis, so we have to prioritize the branches likely to contribute to finding a bug. There have been a lot of studies to deal with this path explosion problem. In this paper, I have presented state-of-the-art methods to deal with this problem.

2 Introduction

- What is it?
- Where did it start?
- How does it work?
- Give an example
- Why is it important?
- It's contributions
- It's limitations
- Example of the use of this technique in finding bugs in configurable system eg result of SAGE, EXE, etc.

3 Different catagorical bodies

- summerize, give an overview of the main points of each source and combine them into a coherent whole

– Analyze and interpret: don't just paraphrase other researchers—add your own interpretations where possible, discussing the significance of findings in relation to the literature as a whole
– Critically evaluate: mention the strengths and weaknesses of your sources
– Write in well-structured paragraphs: use transition words and topic sentences to draw connections, comparisons and contrasts

4 Conclusion

5 Papers

– 2006: they worked on backtracking algorithms for search heuristics [16]
– 2007: they combined the fuzzing techniques to improve the coverage [10] [5] [7]
– 2008: Heuristic based approach to select the branches [1]
– 2009: they worked on the fitness guided approach to improve the coverage [15]
– 2013: they boosted concolic testing by subsuming paths that are guaranteed to not hit a bug with their interpolation technique [8]
– 2014: they introduced a concept of context guided search strategy [12]
– 2018: automatic selection of suitable heuristic [2]
– 2018: template guided approach [3]
– 2018: based on probability of program paths and the cost of constraint solving [13]
– 2018: they improved the speed of SMT solver by removing the IR layer making it more practical to keep bigger constraints [17]
– 2019: fuzzy search strategy [11]
– 2019: adaptably changing search heuristics [4]
– 2021: Pathcrawler: proposed different strategies to improve the performance of concolic execution on exhaustive branch coverage [14]
– 2022: Dr. Pathfinder [9] combined concolic execution with deep reinforement learning to prioritize deep paths over shallow ones for hybrid fuzzing

References

1. Burnim, J., Sen, K.: Heuristics for scalable dynamic test generation. In: 2008 23rd IEEE/ACM International Conference on Automated Software Engineering. pp. 443–446 (2008)
2. Cha, S., Hong, S., Lee, J., Oh, H.: Automatically generating search heuristics for concolic testing. In: Proceedings of the 40th International Conference on Software Engineering. pp. 1244–1254 (2018)
3. Cha, S., Lee, S., Oh, H.: Template-guided concolic testing via online learning. In: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering. pp. 408–418 (2018)

4. Cha, S., Oh, H.: Concolic testing with adaptively changing search heuristics. In: Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. pp. 235–245 (2019)
5. Godefroid, P.: Compositional dynamic test generation. In: Proceedings of the 34th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages. pp. 47–54 (2007)
6. Godefroid, P., Klarlund, N., Sen, K.: Dart: Directed automated random testing. In: Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation. pp. 213–223 (2005)
7. Godefroid, P., Levin, M.Y., Molnar, D.: Sage: Whitebox fuzzing for security testing: Sage has had a remarkable impact at microsoft. Queue 10(1), 20–27 (2012)
8. Jaffar, J., Murali, V., Navas, J.A.: Boosting concolic testing via interpolation. In: Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering. pp. 48–58 (2013)
9. Jeon, S., Moon, J.: Dr. pathfinder: hybrid fuzzing with deep reinforcement concolic execution toward deeper path-first search. Neural Computing and Applications 34(13), 10731–10750 (2022)
10. Majumdar, R., Sen, K.: Hybrid concolic testing. In: 29th International Conference on Software Engineering (ICSE'07). pp. 416–426. IEEE (2007)
11. Sabbaghi, A., Rashidy Kanan, H., Keyvanpour, M.R.: Fsct: A new fuzzy search strategy in concolic testing. Information and Software Technology 107, 137–158 (2019), `https://www.sciencedirect.com/science/article/pii/S0950584918302428`
12. Seo, H., Kim, S.: How we get there: A context-guided search strategy in concolic testing. In: Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering. pp. 413–424 (2014)
13. Wang, X., Sun, J., Chen, Z., Zhang, P., Wang, J., Lin, Y.: Towards optimal concolic testing. In: Proceedings of the 40th International Conference on Software Engineering. pp. 291–302 (2018)
14. Williams, N.: Towards exhaustive branch coverage with pathcrawler. In: 2021 IEEE/ACM International Conference on Automation of Software Test (AST). pp. 117–120 (2021)
15. Xie, T., Tillmann, N., De Halleux, J., Schulte, W.: Fitness-guided path exploration in dynamic symbolic execution. In: 2009 IEEE/IFIP International Conference on Dependable Systems & Networks. pp. 359–368. IEEE (2009)
16. Yan, J., Zhang, J.: Backtracking algorithms and search heuristics to generate test suites for combinatorial testing. In: 30th Annual International Computer Software and Applications Conference (COMPSAC'06). vol. 1, pp. 385–394. IEEE (2006)
17. Yun, I., Lee, S., Xu, M., Jang, Y., Kim, T.: QSYM: A Practical Concolic Execution Engine Tailored for Hybrid Fuzzing. In: Proceedings of the 27th USENIX Security Symposium (Security). Baltimore, MD (Aug 2018)