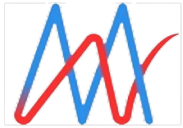
SC-MCC Test Case Generation using Dynamic Symbolic Execution Engines

[1]Golla Monika Rani, [2]Sangharatna Godboley, [3]Joxan Jaffar, [4]Rasool Maghareh

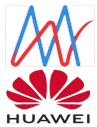
[1,2]NITMINER Technologies, National Institute of Technology Warangal, India
[3]National University of Singapore, [4]Huawei Canada Research Centre, Canada
gm720080@student.nitw.ac.in, sanghu@nitw.ac.in, joxan@comp.nus.edu.sg,
rasool.maghareh@huawei.com



KLEE Workshop 2024

Co-Organised with
46th International Conference on Software Engineering (ICSE 2024) ,
15-16 April, 2024,
Lisbon, Portugal

Plan of the talk

Introduction

- Exploring all the feasible paths in order to generate test cases is costly when dynamic symbolic execution is considered.
- Hence, there comes the interpolation concept that minimizes the cost to some extent.
- The earlier work on custom interpolation introduced a resource annotator that instruments the program and generates multiple meta programs (LLVM IRs) in order to generate optimal mc/dc-based test cases.

Introduction

- The proposed approach leverages a Meta Program Generator (MPG) to create a single meta program that encapsulates SC-MCC sequences within "klee_assert" statements, aligned with their corresponding predicates.
- The effectiveness of this approach is demonstrated through experiments on benchmark programs, comparing it with traditional methods.
- The results indicate improved efficiency and the generation of a higher number of feasible SC-MCC sequences, making our approach a promising advancement in software testing and symbolic execution.

Proposed Idea

- Software testing plays a crucial role in ensuring the reliability and robustness of software applications.
- Among various testing methods, Dynamic Symbolic Execution (DSE) has gained significant attention as it can systematically explore different program paths and generate test cases automatically.
- In DSE [1], generating Multiple Condition Coverage with Short-Circuit (SC-MCC) [3] sequences is essential to create meaningful test cases.
- However, the existing custom interpolation approach [2,4] generates multiple meta programs at the binary level, making it a time-consuming process to execute each binary file in order to obtain SC-MCC Score (%).
- Our proposed approach aims to generate a single Meta Program in source code format, specifically C programs, to avoid multiple executions and generate the SC-MCC Score % efficiently.

Proposed Idea

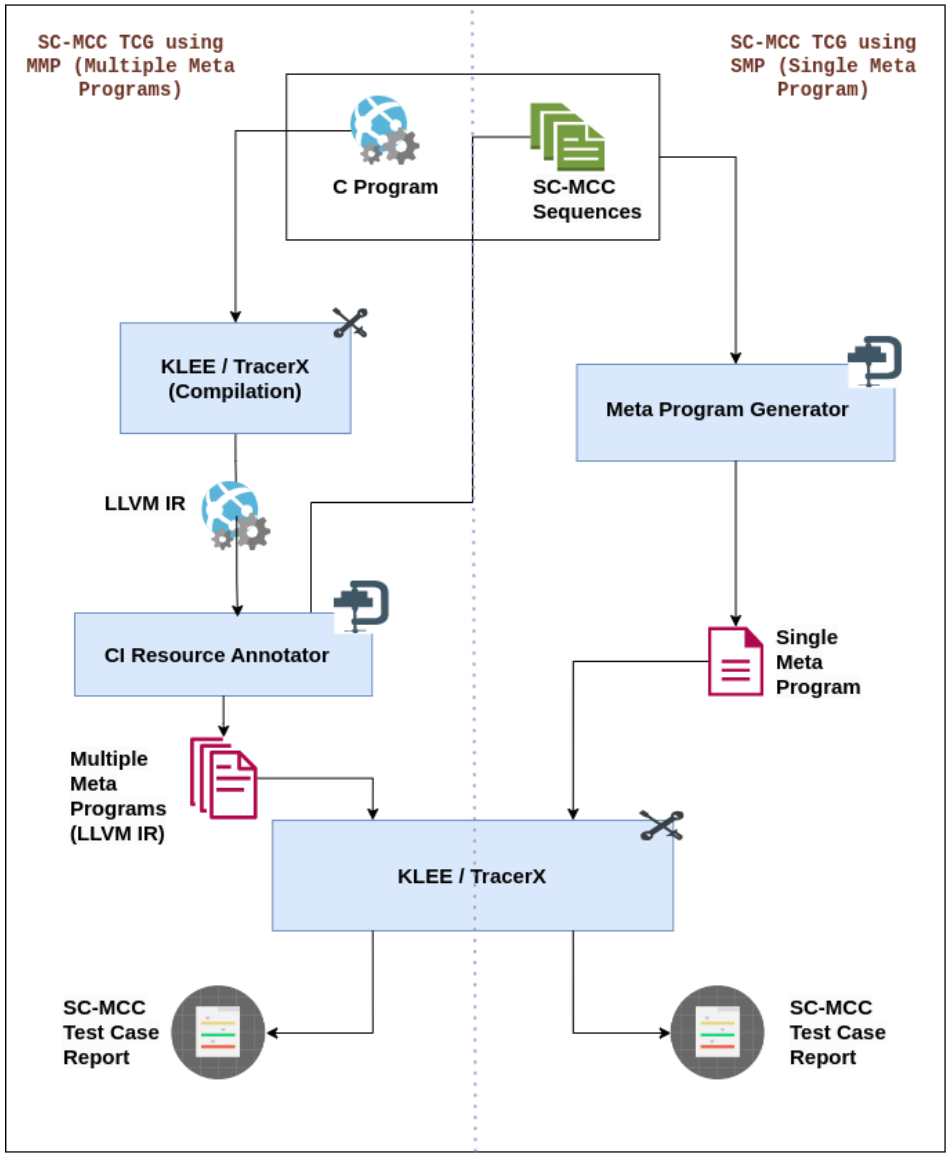


Figure 1: Framework of the proposed approach.

Implementation Details

- The framework of our proposed approach is depicted in Fig. 1.
- It is to be noted that the existing custom interpolation approach annotates the program at binary level i.e., KLEE/TracerX compiled version – LLVM-IR, whereas the proposed approach generates a meta program in the source code format i.e., C program.
- The traditional approach flow starts from compiling *C program* using *KLEE/TracerX* which produces an *LLVM IR*.
- Then, this LLVM-IR is given as input to the *Resource Annotator*.
- The *Resource Annotator* takes SC-MCC sequences as input and then annotates the SC-MCC sequences of each predicate into the LLVM-IR separately and thus generates *Multiple Meta Programs*.

Implementation Details

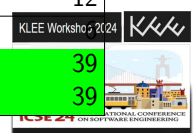
- These *Meta Programs* upon executed with KLEE/TracerX, produces SC-MCC Test Case report.
- Now, the proposed approach flow starts from *Test Cases* along with the *SC-MCC sequences* being imparted into *Meta Program Generator* in order to generate a *Single Meta Program*.
- The *Meta Program Generator* encloses the SC-MCC sequences inside the *klee_assert* statements and places these statements just above its corresponding predicate.
- Finally, the SC-MCC score is calculated by dividing the no. of feasible sequences out of the total SC-MCC sequences.
- Thus, the *SC-MCC Score%* is generated.

Experimental Study

- We performed the experiments on a 64-bit Ubuntu machine with 8GB RAM and Intel (R) Core (TM)-i5 processor.
- We considered *30* benchmark (Coolant, Pals and Psyco series) programs to produce the results.
- The results are shown in Table 1.
- Columns 1, 2 and 3 contain the serial number of the programs, the benchmark and their corresponding program name information, respectively.
- Column 4 constitutes the number of SC-MCC sequences that are instrumented into the Meta Program(s).
- Columns 5 and 6 contain the results of the traditional approach (Custom-KLEE, Custom-TX) whereas Columns 7 and 8 correspond to the proposed approach results (Meta-KLEE and Meta-TX).

Table 1: Results for Custom-KLEE, Custom-TX, Meta-KLEE and Meta-TX. Note: #TSeq: Total no. of SC-MCC sequences #FeasSeq: No. of feasible SC-MCC sequences.

Sl. No	Benchmark	Program	#TSeq	Custom-KLEE		Custom-TX		Meta-KLEE		Meta-TX	
				Time	#FeasSeq	Time	#FeasSeq	Time	#FeasSeq	Time	#FeasSeq
1	COOLANT	test33	3	3600	3	3600	3	3600	3	3600	3
2		test34	3	3600	3	3600	3	3600	3	3600	3
3		test35	3	3600	3	17	3	3600	3	31	3
4		test36	3	3600	3	381	3	3600	3	647	3
5		test37	3	3600	3	384	3	3600	3	668	3
7	PALS	pals1	9	55	6	48	6	19	6	17	6
8		pals2	9	53	6	47	6	19	6	18	6
9		pals3	9	54	6	51	6	18	6	17	6
10		pals4	24	2616	16	4	16	310	16	4	16
11		pals5	24	2438	16	4	16	315	16	5	16
12		pals6	24	2411	16	4	16	324	16	4	16
13		pals7	36	3600	4	2	24	2116	4	0	24
14		pals8	33	3600	6	2	22	1566	6	0	22
15		pals9	36	3600	6	2	22	1799	6	1	22
16		pals10	42	3600	10	74	14	448	10	27	16
20		pals14	60	3600	0	3600	0	3600	0	3600	16
21		pals15	60	3600	0	3600	0	3600	0	3600	14
22		pals16	60	3600	0	3600	0	3600	0	3600	14
23		pals17	18	3600	2	860	12	3600	2	224	12
24		pals18	36	3600	2	3600	0	3600	2	3600	16
25		pals19	60	3600	0	3600	0	3600	0	3600	16
26		pals22	18	378	12	42	12	65	12	23	12
27		pals23	18	370	12	41	12	63	12	21	12
28		Wtest1-B10	72	3600	0	3	6	3600	0	7	
29		Wtest2-B10	88	3600	1	3600	23	3600	1	3600	39
30		Wtest4-B10	88	3600	1	3600	23	3600	1	1291	39



Result Analysis

- It can be noticed that out of *30* programs, the first five programs belong to the Coolant benchmark, the next 21 programs belong to the Pals series programs and the remaining four programs belong to the Psyco series benchmark programs.
- The time-out period set for the KLEE/TracerX execution is 3600s. The total no. of SC-MCC sequences instrumented is highest (88) in the case of the Psyco series and the lowest (3) is observed w.r.t., Coolant series.
- The Custom-KLEE execution is timed out (3600s) for most of the programs whereas the Meta-KLEE execution took less time to complete the execution.
- The least Custom-KLEE time out (53s) is observed w.r.t., *pals2* program whereas the pals3 program finishes in less time in the case of Meta-KLEE i.e., 18s.
- The win cases of the Meta-KLEE over Custom-KLEE are highlighted in green colour.

Result Analysis

- Now, the efficiency of Custom-TX and Meta-TX is noticed to be more or less the same.
- Because, out of 30 programs, eight programs are timed-out (3600s), two programs have the same number (4s), 9 programs are efficient w.r.t., Custom-TX (highlighted in red colour) and the remaining 11 programs are efficient w.r.t., Meta-TX (highlighted in green colour).
- Thus, multiple verses single meta program didn't work well with the programs considered. However, the results are great when feasible sequences are considered.

Conclusion

- Precisely, the Meta-TX generates more number of feasible sequences than Custom-TX in the case of 12 programs.
- These cases are highlighted in green colour corresponding to the last column.
- This proves the single meta-program generation contributes to the greater number of feasible SC-MCC sequences.

Acknowledgement

This work is sponsored by IBITF, Indian Institute of Technology (IIT) Bhilai, under the grant of PRAYAS scheme, DST, Government of India.

References

1 Cristian Cadar, Daniel Dunbar, and Dawson Engler. 2008. 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 (San Diego, California) (OSDI'08). USENIX Association, USA, 209–224.

2 Sangharatna Godboley, Joxan Jaffar, Rasool Maghareh, and Arpita Dutta. 2021. Toward Optimal Mc/Dc Test Case Generation. In Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis (Virtual, Denmark) (ISSTA 2021). Association for Computing Machinery, New York, NY, USA, 505–516. https://doi.org/10.1145/3460319.3464841

References

3 Monika Rani Golla and Sangharatna Godboley. 2024. Automated SC-MCC Test Case Generation using Bounded Model Checking for Safety-Critical Applications. Expert Systems with Applications 238 (2024), 122033. https://doi.org/10.1016/j.eswa.2023.122033

4 Joxan Jaffar, Rasool Maghareh, Sangharatna Godboley, and Xuan-Linh Ha. 2020. TracerX: Dynamic Symbolic Execution with Interpolation (Competition Contribu-tion). In Fundamental Approaches to Software Engineering, Heike Wehrheim and Jordi Cabot (Eds.). Springer International Publishing, Cham, 530–534.


Thank You!