Project Proposal

Symbolic Execution

Course: SE305 Software Project Lab – I

Submitted by

Toukir Ahammed	*Roll: 0806*	*2015-2016*

Document Version 1.0

Submitted to

SPL I Coordinators

Rezvi Shahariar, Assistant Professor, IITDU

Amit Seal Ami, Lecturer, IITDU

Institute of Information Technology

University of Dhaka

[09-02-2017]

Date: 09-02-2017

SPL I Coordinators,
Institute of Information Technology(IIT),
University of Dhaka. Dhaka, 1000.

Subject: Application for approving Software Project Lab(SPL-I) proposal.

Dear Sir,
With due respect, I have the honor to state that I am a student of 3rd semester of BSSE 08th batch. As my software project of SPL-I, I have selected a project titled "Symbolic Execution" supervised by Alim Ul Gias, Lecturer, IIT, University of Dhaka.

I hope that, you would be kind enough to approve my project proposal and oblige thereby.

Yours Sincerely,

Toukir Ahammed
Roll: BSSE0806

Supervisor's Name

Alim Ul Gias
Lecturer, IIT,
University of Dhaka.

01. Title: Symbolic Execution

02. Introduction:

Symbolic execution is a useful technique of analyzing a source code to identify which inputs cause each part of a program to execute. Symbolic execution means executing a program with symbols as input rather than concrete values. During a normal execution the program takes a concrete input value and the program proceeds according to that particular value. But during a symbolic execution the program takes a symbolic value (e.g., α) rather than a concrete value and then proceeds with that symbol as an input value. As the input value is concrete in case of normal execution, the control flow path is fixed for that particular value. But in symbolic execution the symbol (e.g., α) can take any value, so the program can take any feasible path and thus can explore multiple paths simultaneously.

03. Background:

In the field of software industry testing is a very important thing. Symbolic execution is very popular technique in software testing. It is used in software testing, bug detection, bug exploitation, test-input generation, worst case execution time estimation, program verification, authentication bypassing, detecting infeasible paths etc.

04. Description:

This project is about symbolic execution of a source code. The input of this program is a source code file written in C language. Then the task is to detect basic blocks and branching statement where the program can be divided into branches. Using this information the program will explore the

feasible paths that the source code can follow. Finally, the program will output a diagrammatic representation of the program flow graph showing feasible and unfeasible paths.

05. Motivation:

Software testing is very important in software industry. Because every year billions of dollars are lost due to software system failures. More than one third of this cost could be avoided if better software testing is performed. So it is necessary to test a software properly. Symbolic execution is very useful and effective technique for software testing. So I am interested in this project and I am hopeful that it will help me to achieving proper knowledge and skill regarding to software testing as well as my programming skill. This will also help me in my future project.

06. Objective and Scope:

The objective of this project is the symbolic execution of a program to generate diagrammatic representation of the program flow graph showing feasible and infeasible paths.

The scope of this project is the symbolic execution of a source file written in C programming language.

07. Dependencies:

To run this program, a runnable and well indented source code and C++14 compiler is required.

08. Risk and Limitations:

The program will not work when the numbers of infeasible paths are large, there are infinite loops in source code. This program can not handle a very complex source code as the number of feasible paths in a program grows exponentially with an increase in program size and can even be infinite in the case of programs with unbounded loop iterations.

09. Assumptions:

It is assumed that all source codes are written in C programming language and are properly indented. It is also assumed that all source codes are written following the "**Recommended C Style and Coding Standards".** [1]

10. Timeline:

Task / Week	01	02	03	04	05	06	07	08	09	10	11	12	13	14
Project Analysis	■	■	■											
Proposal Submission				■										
Implementation					■	■	■	■	■	■				
Testing											■	■		
Documentation						■	■	■	■	■	■	■		
Report Submission													■	■

11. Conclusions:

Throughout this project I can learn new things, develop my skill of programming and achieve knowledge about symbolic execution. After all, I can gather a lot of experiences to handle a project which will help me in the future project.

12. References:

01. "**Recommended C Style and Coding Standards"** https://www.doc.ic.ac.uk/lab/cplus/cstyle.html (Last accessed on: 2/9/2017 1:34:06 AM).

02. "**Symbolic execution systems – a review" -**by P. David Coward Software Engineering Journal 1988. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.475.6865&rep=rep1&type=pdf (Last accessed on: 2/9/2017 1:34:06 AM).

03. **"Symbolic execution for finding bugs"** –by Michael Hicks, University of Maryland and MC2 https://www.cs.umd.edu/~mwh/se-tutorial/symbolic-exec.pdf (Last accessed on: 2/9/2017 1:34:06 AM).

04. Anand, Saswat; Patrice Godefroid; Nikolai Tillmann (2008). "Demand-Driven Compositional Symbolic Execution". Tools and Algorithms for the Construction and Analysis of Systems, Lecture Notes in Computer Science. **4963**: 367–381. Retrieved 2013-04-03.

05. "**A Survey of Symbolic Execution Techniques"** https://arxiv.org/pdf/1610.00502.pdf (Last accessed on: 2/9/2017 1:34:06 AM).

06.Software Errors Cost U.S. Economy $59.5 Billion Annually, NIST report 2002. (Last accessed on: 2/9/2017 1:34:06 AM).