Detection of Infeasible Paths: Approaches and Challenges

Sun Ding and Hee Beng Kuan Tan

Division of Information Engineering, Block S2, School of Electrical & Electronic Engineering, Nanyang Technological University, 639798, Singapore
{ding0037, ibktan}@ntu.edu.sg

Abstract. Each execution of the program follows one path through its control flow graph. In general, a program has a large number of such paths. Most of the programs have an infinite number of these paths. Regardless of the quality of the program and the programming language used to develop it, in general, a sizable number of these paths are infeasible—that is no input can exercise them. Detection of these infeasible paths has a key impact in many software engineering activities including code optimization, testing and even software security. This paper reviews the approaches for detecting infeasible paths, discusses the challenges and proposes to revisit this important problem by considering also empirical aspect in conjunction to formal program analysis.

Keywords: Survey, Path Infeasibility, Symbolic Evaluation, Program Analysis, Software Testing.

1 Introduction

Control flow graph (CFG) is the standard model to represent the execution flow between statements in a program. In the CFG of a program, each statement is represented by a node and each execution flow from one node to another is represented by a directed edge, where this edge is out-edge of the former node and the in-edge of the latter node. Each path through the CFG from the entry node to the exit node is a logic path in the program. In order for an execution to follow a path in the CFG, the input submitted to the program must satisfy the constraint imposed by all the branches that the path follows. An infeasible path is a path in the CFG of a program that cannot be exercised by any input values. Figure 1 shows an infeasible path p = (entry, 1, 2, 3, 4, 5, 6, exit) in a CFG. This is because we cannot find any input x satisfying $x \geq 0$ and $x < 0$ jointly.

The existence of infeasible paths has major impact to many software engineering activities. Code can certainly be optimized further if more infeasible paths can be detected during the process of optimization. In software testing, the structural test coverage can be much accurately computed if infeasible paths can be detected more accurately. In the automated generation of structured test cases, much time can be saved if more infeasible paths can be detected. In code protection, it can also help in code deobfuscation to identify spurious paths inserted during obfuscation. In software verification, detecting and eliminating infeasible paths will help to enhance the verification precision and speed. There are many more areas like security analysis [37],...
web application verification [30]; [29]; [28]; [27], database application design [34]; [35] that can be helped by the detection of infeasible paths.

To detect infeasible paths in real programs, one needs to deal with complex data structures and dependency. Additional effort is required to formally present them in symbolic expressions or constraints for further verification by heuristics, predefined rules or even standard theorem provers. If the verification returns negative results (e.g.: “Invalid” answer from theorem provers), the path is then considered as infeasible. Such verification model is undecidable in general. But it is still possible to have practical approaches that are not theoretically complete to detect infeasible paths.

The purpose of this article is to familiarize the reader with the recent advances in infeasible paths detections and its related applications. Concepts and approaches will be introduced informally, with citations to original papers for those readers who preferring more details. Information about tools and implementation is also introduced. The paper is organized as below: the literals for infeasible paths detection is reviewed in section 2. Information of tool implementation is introduced in section 3. We discussed remaining problems and future challenges in section 4. Section 5 summarizes the entire paper.

2 Detection of Infeasible Path

During software developing or testing, infeasible path detection usually appears as an essential step for the final project goal. A variety of approaches have been proposed. Based on the ways that they detect infeasible paths, we classify them into six types: (1) data flow analysis; (2) path-based constraint propagation; (3) property sensitive data flow...