Heuristic Guided Selective Path Exploration for Loop Structure in Coverage Testing

Xu-zhou Zhang, Beijing University of Post and Telecommunication, Beijing, China
Yun-zhan Gong, Beijing University of Posts and Telecommunications, Beijing, China
Ya-Wen Wang, Beijing University of Posts and Telecommunications, Beijing, China

ABSTRACT

Static program analysis is a strong technique for analyzing program behavior, but suffers from scalability problem, such as path explosion which is caused by the presence of loops and function calls. This article applies the selective execution mechanism and heuristic strategy on exploring paths through loops. This combinatorial strategy tries to alleviate the path explosion problem from three aspects: 1) exploring loops with different approaches according to their relative position to a specific target; 2) combining static analysis, dynamic execution, and symbolic execution to deal with the separated program; 3) applying a heuristic strategy on offering guidance for the path exploration. These approaches are integrated to automatically generate paths for specified targets in loop structure. Experimental results show that the authors’ proposed strategy is available for combination of different loops. It outperforms some existing techniques on achieving better coverage for programs containing loops, and is applicable in engineering.

KEYWORDS

Heuristic Strategy, Hybrid Strategy, Path Generation, Program Analysis, Symbolic Execution

1. INTRODUCTION

Coverage testing is one of the most common techniques for ensuring software quality by covering all program elements of the program such as statements, predicates or branches. Many automated testing tools have emerged to release human from hard work on achieving code coverage. The two dominating approaches of automated coverage testing are constraint-based testing (Zhang, 2017) and search-based testing.

Symbolic execution (Godefroid, 2005) has gained popularity in the last decade as the fundamental technique of constraint-based testing, which performs path exploration and constraints collection, then turns to modern constraint solvers for solutions. Search-based testing (Malburg, 2011) casts program covering problem as a combinatorial problem and applies meta-heuristic search techniques to find appropriate solutions.

Both methods suffer from path explosion problem caused by loop structure, the arbitrary iterations of input-dependent loops make path exploration in symbolic execution unable to complete within a reasonable range of time, meanwhile cause search strategy to lose precision.

To cope with the scalability problems caused by function calls in symbolic execution, selective symbolic execution (Chipounov, 2009) or compositional dynamic execution (Godefroid, 2007) are introduced to perform symbolic execution within user specified scope of program space, meanwhile
concrete execution in other parts. This mechanism assists symbolic execution in creating full system simulation at the expense of losing completeness.

This paper concentrates on the path explosion problem caused by loop structure in path generation. In the presence of input-dependent loops, the number of paths is exponential in the unpredictable iterations of loop. Following the general idea of selective symbolic execution, the authors regard loop structures as bounded codes with inputs and outputs, apply concrete execution or symbolic execution on loop structures depending on their relative positions to user specified target program element. In order to eliminate the randomness of concrete execution, heuristic search strategy is introduced to generate promising solutions as an optimization. All these strategies are combined in the heuristic guided K+1 path exploration framework. The integration of concrete execution and heuristic optimization allow path exploration to generate prioritized paths through complex loop structures.

The experimental results of generating path for program target in loop with our prototype tool show that compared to traditional constraint-based testing, the efficiency of path generation is significantly higher. At the same time, the number of execution necessary for achieving path is reduced comparing to search-based testing.

The main contributions of this paper are listed below.

1. It draws on the general idea of selective execution to find feasible paths through loop structure, which extends the scalability of symbolic execution on exploring complex loop structures for coverage testing.
2. The meta-heuristic search is integrated with selective strategy to generate more promising paths during path exploration as an optimization, which eliminates the randomness of the concrete phase in selective execution.

The rest of the paper is organized as follows. The next section presents the background and motivation. An overview of the heuristic guided K+1 path exploration strategy is presented in the third section. The implementation of the strategy is illustrated in the fourth section with an example. The fifth section performs the experimental results. The sixth section elaborates related work on alleviating the path explosion problem in automated coverage testing. The seventh section concludes this paper and states the future work.

2. BACKGROUND AND MOTIVATION

In this section, the research background and motivation are described by briefly introducing the selective symbolic execution and heuristic search strategy.

2.1. Symbolic Execution

Symbolic execution is the core portion of constraint-based testing. In classic symbolic execution, the tester has to select a path beforehand, and execute a program statically with symbolic input values instead of concrete values.

During the execution, a constraint system, consisting of symbolic expressions, is derived by collecting all path conditions and handling variable operations. Then it is solved by a constraint solver to achieve solutions, which make the program to follow the same path during concrete execution.

2.2. Selective Symbolic Execution

The main limitation of pure symbolic execution is poor scalability of analyzing large programs involving function calls. Selective symbolic execution alleviates this limitation by selective executing concretely or statically within user appointed scopes. The general idea of selective execution is based on the observation that only part of the code are of interest. One may want to exhaustively explore all
Internet Policy Issues and Digital Libraries' Management of Intellectual Property

Open Growth: The Impact of Open Source Software on Employment in the USA
[www.igi-global.com/article/open-growth/104678?camid=4v1a](www.igi-global.com/article/open-growth/104678?camid=4v1a)