Combining Frama-C and PathCrawler for C Program Debugging

Omar Chebaro[1,2], Nikolai Kosmatov[1], Alain Giorgetti[2,3], and Jacques Julliand[2]

[1] CEA, LIST, Laboratoire Sûreté des Logiciels, PC 94, 91191 Gif-sur-Yvette France
firstname.lastname@cea.fr
[2] LIFC, University of Franche-Comté, 25030 Besançon Cedex France
firstname.lastname@lifc.univ-fcomte.fr
[3] INRIA Nancy - Grand Est, CASSIS project, 54600 Villers-lès-Nancy France

Extended Abstract

Software validation remains a crucial part in software development process. Software testing accounts for about 50% of the total cost of software development. Automated software validation is aimed at reducing this cost. The increasing demand has motivated much research on automated software validation. Two major techniques have improved in recent years, dynamic and static analysis. Traditionally, they were viewed as separate domains.

Static analysis examines program code and reasons over all possible behaviors that might arise at run time. It is often necessary to use approximations. Static analysis is conservative and sound: the results may be weaker than desirable, but they are guaranteed to generalize to all executions. Dynamic analysis operates by executing a program and observing this execution. Dynamic analysis is efficient and precise because no approximation or abstraction needs to be done: the analysis can examine the actual, exact run-time behavior of the program for the corresponding test case. It can be as fast as program execution.

The pros and cons of the two techniques are apparent. If dynamic analysis detects an error then the error is real. However, it cannot in general prove the absence of errors. On the other hand, if static analysis reports a potential error, it may be a false alarm. However, if it does not find any error (of a particular kind) in the overapproximation of program behaviors then the analyzed program clearly cannot contain such errors.

Recently, there has been much interest in combining dynamic and static methods for program verification. Static and dynamic analyses can enhance each other by providing valuable information that would otherwise be unavailable. This paper reports on an ongoing project that aims to provide a new combination of static analysis and structural testing of C programs. We implement our method using two existing tools: Frama-C, a framework for static analysis of C programs, and PathCrawler, a structural test generation tool.

Frama-C [1] is being developed in collaboration between CEA LIST and the ProVal project of INRIA Saclay. Its software architecture is plug-in-oriented and allows fine-grained collaboration of analysis techniques. Static analyzers are implemented as plug-ins and can collaborate with one another to examine a C program. Let us introduce the value analysis plug-in based on abstract interpretation. This plug-in computes and stores supersets of possible value ranges of variables at each statement of the program. Among other applications, these over-approximated sets can be used to exclude the possibility of a run-time error. The value analysis is correct: it emits an alarm for an operation whenever it cannot guarantee the absence of run-time errors for this operation. The value analysis memorizes abstract states at each statement and provides an interface for other plug-ins to extract these states.

Developed at CEA LIST, PathCrawler [2] is a test generation tool for C functions respecting *the all-paths criterion,* which requires to cover all feasible program paths, or *the k-path criterion,* which restricts the generation to the paths with at most k consecutive iterations of each loop. The PathCrawler generation method is similar to the so-called *concolic,* or *dynamic symbolic execution.* The user provides the C source code of the function under test. The generator's main loop is rather simple: given a partial program path π, the main idea is to symbolically execute it using constraints. A solution of the resulting constraint solving problem will provide a test case exercising a path starting with π. Then concrete execution of the test case allows to obtain the complete path. The partial paths are explored in a depth-first search.

We present an original combination of static analysis and structural test generation for validation of C programs, in particular, for detection of run-time errors. The main idea is to call first a static analysis tool (Frama-C) in order to generate alarms for the statements for which the absence of run-time errors is not ensured by static analysis. Second, these alarms are transferred to a structural test generation tool (PathCrawler) where they guide test generation trying to confirm alarms by activating bugs on some test cases. Our ongoing implementation of this method, called SANTE, assembles two heterogeneous tools using quite different technologies (such as abstract interpretation and constraint logic programming).

We evaluate our method by several experiments on real-life C programs, and compare the results with other methods. Static analysis alone will in general just generate alarms (some of which may be false alarms), whereas our method allows to confirm some alarms as real bugs and provides a test case activating each bug. This is done automatically, avoiding time-consuming alarm analysis by the validation engineer, at least for confirmed alarms, the task which requires significant expertise, experience and deep knowledge of source code. Stand-alone test generation, when it is not guided by generated alarms for some statements, does not detect as many bugs as our combined method. When guided by the exhaustive list of alarms for all potentially threatening statements (not filtered by static analysis), test generation usually has to examine more infeasible paths and takes more time than our combined method (or even times/spaces out). In all cases, our method outperforms the use of each technique independently.

References

1. Frama-C: A framework for static analysis of C programs (2007-2010) http://frama-c.cea.fr/.
2. Botella, B., Delahaye, M., Hong-Tuan-Ha, S., Kosmatov, N., Mouy, P., Roger, M., Williams, N.: Automating structural testing of C programs: Experience with PathCrawler. In: AST'09, Vancouver, Canada (May 2009)