SMCDCT: A Framework for Automated MC/DC Test Case Generation Using Distributed Concolic Testing

Sangharatna Godboley, Subhrakanta Panda, and Durga Prasad Mohapatra

Department of CSE, National Institute of Technology,
Rourkela-769008, Odisha, India
sanghu1790@gmail.com, {511cs109,durga}@nitrkl.ac.in

Abstract. In this paper we propose a framework to compute MC/DC percentage for distributed test case generation. MC/DC stands for Modified Condition/Decison Coverage [1]. This approach uses several client nodes to generate the non-redundant test cases in a distributed and scalable manner. To achieve an increase in MC/DC, we transform the input C program, P, into its transformed version, $P^{'}$, using Ex-NCT. A coverage analyzer accepts P along with the generated test cases as input from SCORE framework and outputs the MC/DC percentage. The experimental studies show that SMCDCT approach achieves 6.5 % (approx.) of average increase in MC/DC. This increase in MC/DC percentage is achieved in an average computation time of 7.1622715 seconds.

Keywords: MC/DC, Distributed Concolic Testing, Coverage Analyser.

1 Introduction

We propose an approach to calculate MC/DC percentage using *SMCDCT (Scalable MC/DC percentage Calculator using Distributed Concolic Testing)* for structured C programs. SMCDCT consists of mainly three modules: i) *Code Transformer (Ex-NCT)* [2], ii) *Concolic Tester (SCORE) [3]*, and iii) *Coverage Anlyser (CA)* [2]. The experimental studies in [2] shows that Ex-NCT gives better MC/DC percentage as compared to Program Code Transformer (PCT). Hence, in SMCDCT we use Ex-NCT for code transformation of the target programs. In this approach, we use SCORE in SMCDCT to improve the effectiveness and efficiency of the framework, as compared to traditional concolic testing. SCORE is based on distributed concolic testing to generate test cases for the program under consideration. This process reduces the computation time for measuring MC/DC percentage. The coverage analyser takes the non-transformed program and the generated test cases to calculate the MC/DC percentage.

2 SMCDCT Framework

First we present an overview of the proposed SMCDCT framework in Section 2.1, then discuss the steps of our proposed approach in Section 2.2.

R. Natarajan et al. (Eds.): ICDCIT 2015, LNCS 8956, pp. 199–202, 2015.

2.1 Overview of SMCDCT Framework

As the CREST concolic tester supports only linear-integer arithmetic (LIA) formulae, the non linear arithmetic operations in a target C program may not be analyzed symbolically [4,5]. The concolic testing consumes a significant amount of time in exploring the possible execution paths, and this forms a big challenge in its reification. We have proposed a framework to overcome some of the above mentioned limitations. Our main objective is to achieve an increase in MC/DC percentage without affecting the time value. We have used six distributed computing nodes connected through network in order to decrease the time cost of traditional concolic testing. To achieve high scalability, SMCDCT framework enables distributed nodes to generate test cases independently. SMCDCT consists of three modules these are discussed below:

Code Transformer. The Code Transformer (Ex-NCT [2]) uses transformation technique to instrument the C program by augmenting it with additional nested if-else conditional statements. This augmentation of code with additional statements causes MC/DC to vary.

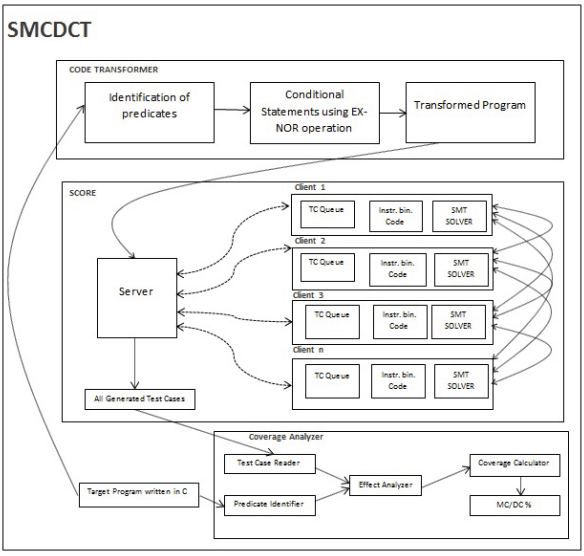

Fig. 1. Schematic representation of the proposed SMCDCT Framework

SCORE. SCORE consists of the *library code* and *symbolic execution engine* modules to achieve full path coverage without generating any redundant test cases. SCORE supports the bit vector symbolic path formulae by using Z3 2.19

SMT Solver [6] to solve the non-linear arithmetic operations symbolically. A distributed concolic algorithm in SCORE decreases the communication overhead among the distributed nodes and increases the speed of test case generation [3].

Coverage Analyser (CA). The module CA [2,7] accepts original C program under test and set of test cases generated from concolic tester as inputs to compute the MC/DC percentage.

As SMCDCT is based on SCORE, so it overcomes the limitations of traditional concolic testing. Therefore, SMCDCT is efficient to effectively perform a distributed and scalable concolic testing in less time.

2.2 Steps of Our Proposed Approach

In this section, we describe in detail the steps of our proposed approach to compute the MC/DC percentage difference. These steps are as follows:

Step1: Generate Test_Suite1 for the target program P.
Step2: Compute MC/DC_1 percentage.
Step3: Transform P into P'.
Step4: Generate Test_Suite2 for P'.
Step5: Compute MC/DC_2 percentage.
Step6: Compare MC/DC_1 percentage and MC/DC_2 percentage.

3 Experimental Study

In this section, we discuss the experimental analysis of the obtained results. The experimentation is carried out on two benchmark C programs (sed and grep) taken from SIR repository [8]. The results are presented under four different experimental scenarios as described below:

i. The first scenario corresponds to the experimentation carried out with the *CREST tool* that ran on a stand-alone machine.
ii. The second scenario corresponds to the experimentation carried out with *Ex-NCT* and *CREST tool* that ran on a stand-alone machine.
iii. The third scenario corresponds to the experimentation carried out with *SCORE tool* that ran on a client-server architecture implemented with six distributed nodes.
iv. The fourth scenario corresponds to the experimentation carried out with *Ex-NCT* and *SCORE tool* that ran on a client-server architecture implemented with six distributed nodes.

The readings in Table 1 show that we achieve 6.5 % (approx.) of average increase in MC/DC for both the experimental programs. The Table 2 shows, the different timing results for the two experimental programs. From Table 2, it can be observed that the average of total computation times for the two experimental programs is 7.1622715 seconds.

Table 1. Comparison of MC/DC percentages

S.No	Program	MCDC %(CREST)	MCDC %(Ex-NCT + CREST)	MCDC %(SCORE)	MCDC %(Ex-NCT + SCORE)	MCDC %(difference)
1	sed	58.5 %	79.7%	82.4%	89.7%	7.3%
2	grep	53.8%	76.2%	79.68%	85.37%	5.69%

Table 2. Timing Reqiurements

S.No	Program	Ex-NCT (Sec)	SCORE (Sec)	CA (Sec)	Total Time (Sec)
1	sed	2.213746	0.871246	3.812357	6.897349
2	grep	2.523712	1.114357	3.789125	7.427194

4 Conclusion and Future Work

We proposed a framework for distributed test case generation named SMCDCT that is based on coverage analysis of C programs. We discussed the detailed steps of our proposed approach along with the working principles of the modules (Code transformer, SCORE, and Coverage Analyzer) of SMCDCT. The experimental results show that the proposed approach achieves better MC/DC in comparison to the existing approaches. SMCDCT approach achieved 6.5 % of average increase in MC/DC. This increase in MC/DC percentage is achieved in an average computation time of 7.1622715 seconds. In future, we will aim at developing an approach to find MC/DC of OOPs.

References

1. Hayhurst, K.J., Veerhusen, D.S., Chilenski, J.J., Rierson, L.K.: practical tutorial on modified condition/decision coverage, Tech. rep. (2001)
2. Godboley, S.: Improved modified condition/ decision coverage using code transformation techniques, M. tech thesis. NIT Rourkela (2013)
3. Kim, Y., Kim, M.: Score: A scalable concolic testing tool for reliable embedded software. In: Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering, pp. 420–423. ACM (2011)
4. Kim, M., Kim, Y., Rothermel, G.: A scalable distributed concolic testing approach: An empirical evaluation. In: Software 2012 IEEE Fifth International Conference on Testing, Verification and Validation (ICST), pp. 340–349 (2012)
5. [link], `http://www.code.google.com/p/crest`
6. de Moura, L., Bjørner, N.: Z3: An efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 337–340. Springer, Heidelberg (2008)
7. Godboley, S., Mohapatra, D.P.: Time analysis of evaluating coverage percentage for c program using advanced program code transformer. In: Computer Society of India, 7 th CSI International Conference on Software Engineering, pp. 91–97 (2013)
8. Do, H., Elbaum, S., Rothermel, G.: Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact. Empirical Softw. Engg. 10(4), 405–435 (2005)