International Journal of Advance Research in Science and Engineering
Vol. No.5, Issue No. 03, March 2016
www.ijarse.com
IJARSE
ISSN 2319 - 8354

SURVEY ONAUTOMATIC TEST CASE GENERATION USING MC/DCTESTING

Annu Rani[1], SukhdipSingh[2]

[1,2]*Department of Computer Science & Enggineering*

Deenbandhu Chhotu Ram University of Science &Technology, Sonepat (India)

ABSTRACT

Given the large domain of inputs and possibly too many possible execution paths, the software is often tested using a sampled set of test cases. A variety of coverage criteria have been proposed to assess the effectiveness of the sampled set of test cases. As far as structural testing involving predicate evaluation is concerned, criteria exercising aspects of control flow, such as statement, branch and path coverage have been the most common. Although useful, these criteria are often susceptible to the problem of masking. Addressing this issue, this paper explores to adoption of mc/dc as the necessary criteria for structural testing.

I. INTRODUCTION

Software testing relates to process of finding errors and of validating the software system against its specification.Apart from reducing the risk of software failures, software testing gives a direct indirection of quality.Structural testing involving predicate evaluation is concerned, criteria exercising aspects of control flow, such as statement, branch and path coverage have been the most common.Mc/dc criterion as a strategy to systematically minimize the number of test cases for testing.In nut shell mc/dc, is a white a box testing criterion ensuring each condition within a predicate can independently influence outcome of the decision-while the outcome of all other conditions remains constant.An mc/dc test predicates exist in pairs. Each one of the pair differs only by the boolean value of one condition but gives different result for the decision statement.To ensure quality software that conforms to specification the software needs to be thoroughly tested.One key aspect to be tested is on structural testing and in current state of the art on existing work involving mc/dc is also highlighted.

II. OVERVIEW OF MODIFIED CONDITION/DECISION COVERAGE

As running example, consider the following if statements involving and and or operations (see figure 1). For both and or operations in figure 1,

Decision coverage is registered at 100% even without the need to change the value of y. Specifically, x is masking y and giving misleading coverage. To illustrate further, consider the equivalent if statements for both and and or operation as shown in figure 2. Given the same inputs for x and y (i.e. X=15, y =3 and x=5, y=3), there are parts of the program which has not been covered (as in shaded regions in figure 2).

International Journal of Advance Research in Science and Engineering
Vol. No.5, Issue No. 03, March 2016
www.ijarse.com
IJARSE
ISSN 2319 - 8354

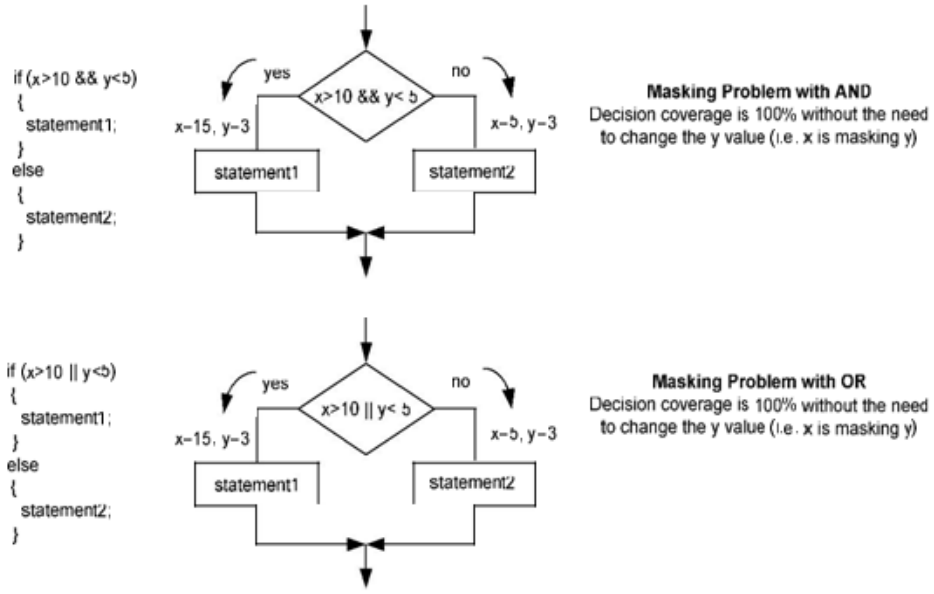

Fig. 1. Masking problem

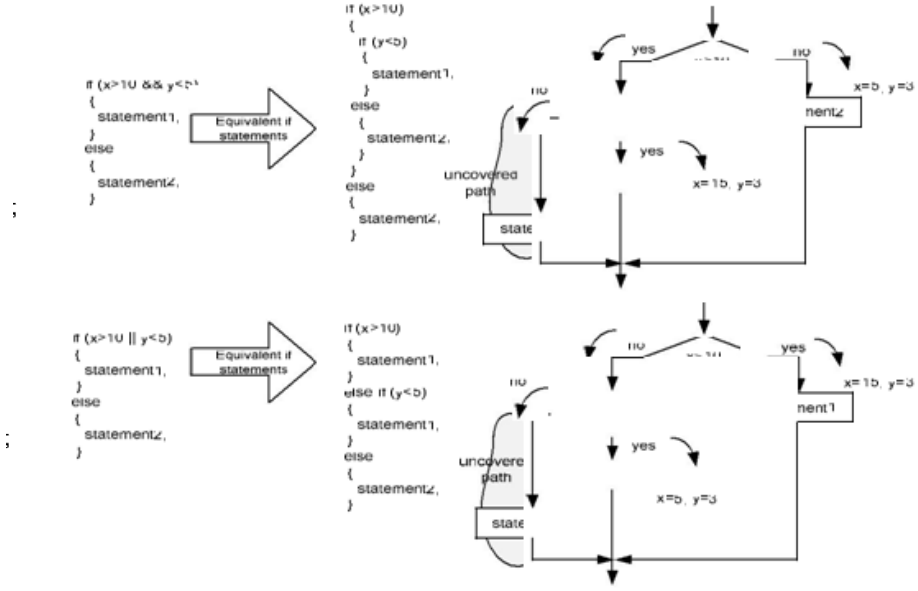

Fig. 2. Equivalent if statements

The main concerns here is on how to cover the uncovered path and hence eliminate masking problems for both and and or operations. Exhaustive combinations (often termed as multiple condition coverage (mcc)) are the most desirable alternatives. However, considering mcc is practically infeasible especially when the combinations are large. Here, the number of conditions grew with 2^n where n is the number of boolean variables.

Condition coverage (cc) and condition/decision coverage(c/dc) are also possible. Cc dictates that every condition in a decision has taken all possible outcomes at least once. C/dc requires cc and also dictates the true and false decision outcome at least once. Despite being useful, cc and c/dc does not consider independence as the criteria for selecting test cases.

Summing up in table 1, it is clear that mc/dc is the most viable alternative but with significantly reduced test size

as compared to exhaustive combination, mcc. Here, mc/dc dictates that each condition within a predicate can independently influence the outcome of the decision. Mc/dc is a stricter form of decision coverage. For decision coverage, each decision statement must evaluate to true on some execution of the program and must evaluate to false on some execution of the program. Mc/dc, however, requires execution coverage at the condition level.

An mc/dc test predicates exist in pairs. Each one of the pair differs only by the boolean value of one condition, but gives a different result for the decision statement. For and operation, mc/dc pairs are {{f,t}, {t,t}}, {{t,f}, {t,t}}. As the entry {t,t} is redundant, the complete mc/dc compliant test predicate is reduced to {f,t}, {t,f} and {t,t}. In similar manner, for or operation, the mc/dc pairs are compliant test predicates are {{f,f}, {t,f}}, {{f,t}, {t,f}}.

As the entry {t,f} is redundant, the complete mc/dc compliant test predicate is reduced to {f,f}, {f,t} and {t,f}.converting the mc/dc compliant predicates into test cases values for and and or operation, figure 3 revisits the masking problem in figure 2. Here, the test cases fulfilling the mc/dc criterion are able to cover all the paths.

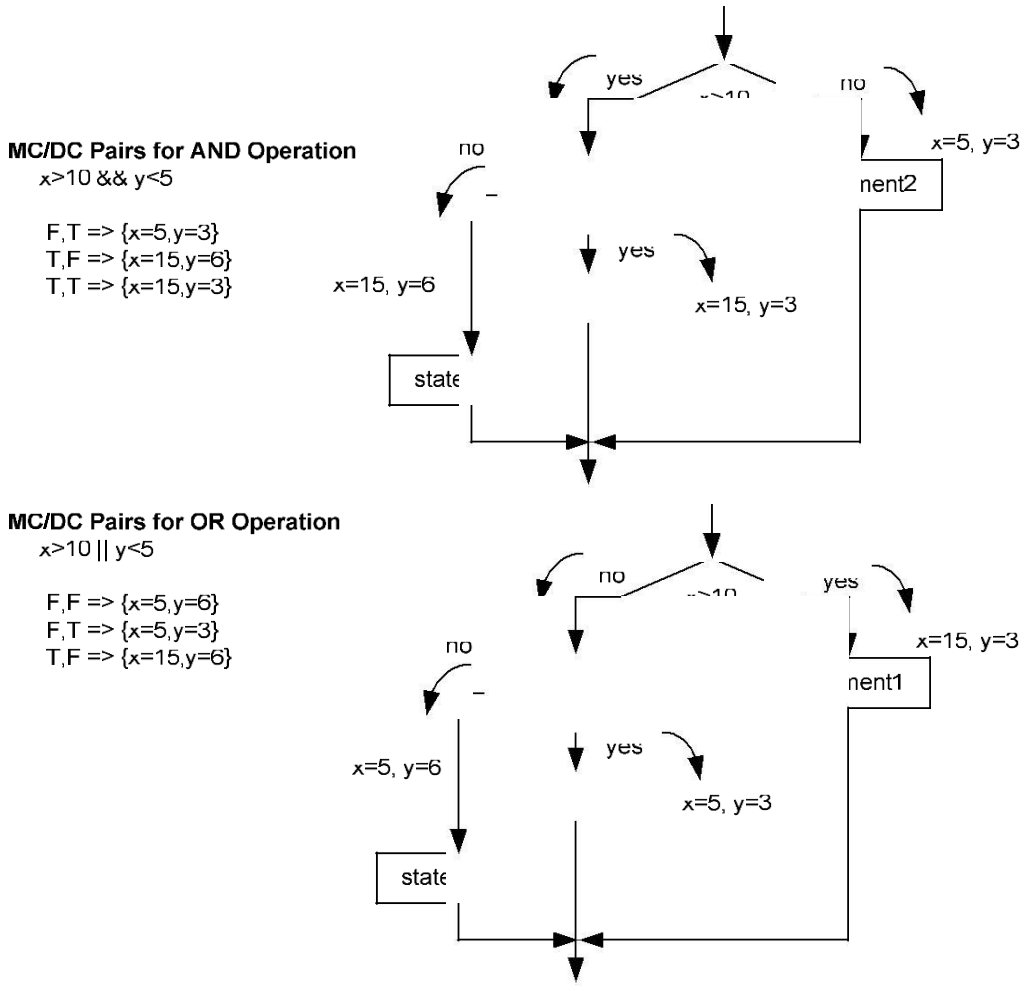

Fig.3. Mc/dc coverage

III. REFLECTION ON RELATED WORK

There is already a number of related works that deals with test case generation for mc/dc coverage. Jones and harrold [3] introduce two strategies for generating mc/dc compliant test cases. The first strategy is based on the breakdown algorithm whilst the second strategy is based on the prioritization algorithm. At the start, both strategies generate the exhaustive mc/dc pairs as the basis for selection. For the first strategy, the selection of the test candidates is based on iterative generation of essential test cases. Here, essential test cases are established by summing up contribution of each test case towards mc/dc coverage. In each iteration, the least contributing test case is systematically removed leaving only available for selection. For the second strategy, the selection of test candidates is Also done iteratively. In this case, in each iteration, the contribution for each candidate test case is prioritized based on greedy ordering, that is, to cover the most pairs. The iteration stops when no more pairs are available for selection. Although helpful, both strategies appear unsuitable for handling large predicates owing to the need to generate all exhaustive mc/dc pairs.

In other work, jun-ru and chin-yu [4] usefully exploit n-cube graph in order to generate appropriate mc/dc compliant test data. In this case, the vertex of the cube represents the resultant boolean enumeration for predicates under evaluation. Each vertex is traversed and arranged and evaluated using gray code sequence ordering until all the required sequences are covered. As the sequence of ordering for mc/dc pairs are non-unique (and not generalizabile to only gray code sequence), this strategy appears not optimized as far as the number of test cases is concerned.

Ghani and clark [5] are perhaps the pioneer researchers that adopt optimization algorithm based on simulated annealing (sa) for mcc and mc/dc test generation. Sa works based on the process of maximizing material's crystal size via heating and slow cooling [9, 10]. The heating process excites the atom to move from its initial position (to avoid a local minima of inte/,rnal energy) while the slow cooling process allows the atom to settle for lower internal energy configurations for better crystal size. Analogous to the physical process, sa based strategy starts with a randomly generated mc/dc pair of test cases (as initial state) and applies a series of transformations according to a pre-defined probability equation. Here, the probability equation depends heavily on parameter t (namely, the controlling temperature of the simulation) to simulate the heating and cooling process.

Complementing the work from ghani and clark, awedikian et al [2] adopt two optimization algorithms based on hill climbing (hc) and genetic algorithm (ga) respectively to generate mc/dc compliant test cases. For hc, the algorithm starts by choosing a random test case as an initial solution. The quality of the test case is evaluated based on the defined fitness function. Hc attempts to improve the current test case by moving to better points in a neighborhood of the current solution. This iterative process continues until a termination criterion. There are two termination conditions. First, for the given major clause, hc terminates if test case satisfying the mc/dc clause assignment are found. If after a fixed number of attempts, the algorithm is not able to satisfy the mc/dc major clause constraints, the search is stopped and another set of possible mc/dc assignments is selected. Concerning ga, the algorithm starts by creating an initial population of n test cases chosen randomly. Each chromosome represents a test case; genes are values of the input variables. In an iterative process, ga tries to improve the population from one generation to another. Test cases in a generation are selected according to their

fitness in order to perform reproduction, that is, through crossover and/or mutation. Then, a new generation is constituted by the fittest test cases of the previous generation and the offspring obtained fromcrossover and mutation. The iterative process continues until a stopping criterion is met. Here, two stopping criteria are defined. First, for the given major clause, ga terminates if test input data satisfying the mc/dc clause assignment are found. Ga is also stopped when an upper limit in computation is reached.

IV. DISCUSSION AND CONCLUSION

To ensure the quality software that conforms to specifications, the software needs to be thoroughly tested. One key aspect to be tested is on structural testing. Here, like most testing endeavor, exhaustive structural testing is not always feasible as it consumes significant resources in term of costing and man power.
Existing work on mc/dc for structural testing has been useful, but they are not without limitation.

4.1 Future Scope

In future we need to be proposed to make a generic tool by using various other coverage metrices.
We plan to automate the actual generation of test from the mc/dc generated pairs.

REFERENCES

[1] minimization using modified condition/decision coverage(mc/dc)", international journal of software engi[1] ajaykumarjena, santoshkumar swain and durgaprsadmohapatra, "model-based test-suite neering and its applications,vol. 9,no. 5(2015),pp-61-74

[2] zalanszugyi and zoltanporkolab, "necessary test cases for decision coverage and modified condition/decision coverage", electrical engineering 52/3-4(2008) 187-195,web:http://www.pp.bme.hu/ee

[3] s.shanmugapriya and shebakeziamalarchelvi, "smart test case quantifier using mc/dc coverage criterion", international journal of advanced computer research(issn (print): 2249-7277 issn(online):2277-7970) volume-4, number-1, issue-14 march-2014

[4] sanjalrayadurgam and mats p.e.heimdahl, "generation mc/dc adequate test sequences through model checking" computer science and engineering university of minnesota

[5] kamranghani and john a.clark, "automatic test data generation for multiple condition and mc/dc coverage", department of computer science university of york,york yo10 5dd,uk

[6] arifulhaque, imrankhalil and kamal z. Zamli, "an automated tool for mc/dc test data generation" , faculty of computer science and software engineering university malaysiapahang

[7] kamal z. Zamli, abdulrahman a.al-sewari and mohd hafiz mohdhassin, "on test case generation satisfying the mc/dc criterion", int. J. Advance soft compu. Appl, vol. 5,no. 3, december 2013, issn 2074-8523;copyright scrg publication,2013

[8] porshiamitra, shreyachatterjee and nikitaali, "graphical analysis of mc/dc using automated software testing" , 978-1-4244-8679-3/11/s26.00 © 2011ieee