Chapter 6

Conclusions and Future Work

Software inevitably changes, however, well conceived and well written it initially may be. Operational failures expose faults to be repaired. Mistaken and changed requirements cause the software to be re-worked. Developers have responded to this need in many ways, including improving the process, increasing the attention on early development activities and using the formal methods for describing the requirements, specification and design. Although, all of these improvements contribute to better software, software still requires testing. For software to be correct, it is essential that it must comply with its specifications which describe the functional and non functional requirements of the software that is used as a basis of software development.

Boolean expressions have been used for defining the specification of the software. For verifying the correctness of the software specifications written in Boolean expressions, a number of techniques have been described in the literature. Experimental work conducted on the various Boolean expressions testing technique uses Mutation analysis to compare the performance and effectiveness of these techniques. For mutation analysis, generation of mutant is an important problem as it affects the effectiveness and cost of mutation testing. As noted by [005,084,087,144,145,146,147], if the mutants for the stronger fault classes are generated, the generation of mutants for corresponding weaker fault classes will not be necessary. This can significantly reduce the number of mutants generated to detect the fault.
The research work extends the earlier work by following the approach proposed in Kuhn [084]. Our specific contributions to the fault based testing include the following:

- Generation of mutants for operator and operand faults in Boolean expression
- Compute the necessary and sufficient conditions for detecting 11 different types of faults. This includes the fault detection criteria for ASF and CDF faults.
- Extension of fault hierarchy by including the ASF and CDF faults.

Mutation analysis is used for evaluating the efficiency of a test suite in fault based testing approach. One major obstacle in mutation analysis is the number of mutants generated for a given Boolean expression. The number of mutants reported for a fault class in literature for same Boolean expression differs indicating the non-uniformity in the generation of the mutants. As a solution, a mechanism for generation of mutants for operator and operand fault class has been derived for Boolean expression written in DNF and Canonical DNF (CDNF). The results obtained assess the maximum number of inequivalent mutants that can be created for Boolean expression in DNF and CDNF for various fault classes. The mechanism of establishment of the results is presented in a constructive fashion. Algorithm for generation of various mutants has also been derived. This will help the development community to build the tool for automatic generation of mutants.

The testing of all faults (that can be estimated) is very costly and impractical. The fault based testing concentrates on certain type of faulty classes that cover most of the errors that are made by programmer [079,081,104,105,137,138]. The existing work on Boolean expression in DNF does not describe the fault detection criteria for the ASF and CDF fault classes. In this work, a review of the detection of existing fault classes has been done and the fault detection criteria for the ASF and CDF fault class have been reported.
Fault hierarchy specifies the inter-relationship amongst the various fault classes in terms of their fault detection capability. Kuhn [084] has developed a fault hierarchy for Boolean expression in DNF which was complemented by Tsuchiya and Kikuno [131]. Lau and Yu [087] extended the fault hierarch by adding more fault classes in the hierarchy. This fault hierarchy has been further extended by including the ASF and CDF faults in the hierarchy. This has been achieved by establishing the inter-relationship of ASF and CDF faults with respect to other fault classes. The extended hierarchy is consistent with the hierarchy derived by various authors [084, 087, 131]. The work will facilitate the testing of the specifications of the software systems that are written using Boolean expressions. Inclusion of more fault classes in the fault hierarchy will be helpful in designing effective test cases that will detect more type of faults. The fault class hierarchy suggests a higher priority to the test cases that detect stronger fault class as these test cases will detect many other weaker classes of faults.

To conclude, the research work would be helpful to the testing community in the following ways:

- Generation of uniform mutants will be helpful in interpreting the previous empirical results.
- Establish a framework for evaluating the effectiveness of the various testing strategies for Boolean expression in DNF.
- The fault detection criteria and the extended fault hierarchy suggests the ways to design strategies that will use lesser test cases.
- Development of the tool for automatic generation of mutants for various fault classes.
6.1 Future Work

Most of the existing work in the fault class testing is based on the competent programmer hypothesis and coupling effect. Though it has been asserted in the literature that a test suite that detects simple faults in program will also detect complex fault \([043]\) but this statement has not been quantified for the multiple faults. The future work may involve the derivation of the fault detection criteria for the multiple faults in the Boolean expression.

The mechanism for generation of the mutants seeding a single fault for Boolean expression in DNF has been derived in the current work. This can be extended for Boolean expression having multiple faults.

To assess whether the existing fault hierarchy that is based on single fault assumptions also holds true for the multiple faults in the Boolean expression.

Mechanism for generation of mutants for the Boolean expression in DNF can also be further extended to the expression in general form.