Inferring Mutant Utility from Program Context

Abstract :

The paper addresses the issue of mutation analysis producing a large number of equivalent, trivial, and redundant mutants. Selective mutation strategies aim to mitigate this by reducing redundancy, but effectiveness varies across different programs due to the varying utility of mutants produced by specific mutation operators. This research hypothesizes that the utility of a mutant, in terms of equivalence, triviality, and dominance, can be predicted using context information from the program. The study introduces an approach for modeling program context via the program's abstract syntax tree and evaluates various models for predicting mutant utility. The results show that context information significantly enhances the ability to predict mutant utility, emphasizing the importance of individual mutation operators over operator groups.

Chapter-by-Chapter Summary:

1. Introduction :

Mutation analysis, a method for evaluating test suites, produces many mutants, leading to high computational demands. The paper highlights the limitations of existing selective mutation strategies, noting their inability to consistently outperform random selection due to not accounting for program context. The authors argue that effective mutant selection must consider the specific context within a program, as the utility of mutants varies significantly depending on where and how they are generated.

1.1 Rationale of Our Approach :

Existing selective mutation approaches fail because they do not consider program context. The paper provides examples demonstrating how mutations of the same operator can have vastly different utilities depending on their program context, suggesting that a context-aware approach could effectively reduce the generation of low-utility mutants.

1.2 Contributions :

The paper's contributions include highlighting the importance of program context in assessing mutant utility, proposing a model of mutant utility considering equivalence, triviality, and dominance, and introducing a method to model program context. An empirical study validates the hypothesis that program context is a strong predictor of mutant utility.

2. Background on Mutation Analysis :

This section explains mutation testing fundamentals, defining mutants, mutation operators, and the concepts of equivalent mutants, trivial mutants, and dominator mutants. It discusses how mutation operators generate more mutants than necessary and introduces the idea of minimal mutation sets and dominator sets, albeit noting the challenge in directly finding dominator sets due to undecidability.

2.1 Equivalent Mutants :

Discusses mutants that behave identically to the original program on all inputs, noting their inability to be killed by any test case.

2.2 Trivial Mutants :

Defines trivial mutants as those easily killed by any test case that executes the mutated code location, thus offering little to no value in assessing test suite effectiveness.

2.3 Dominator Mutants :

Introduces the concept of dominator mutants, a minimal subset of mutants that, if adequately tested, implies comprehensive testing for the entire mutant set. The section details the challenges in identifying such mutants and the approach to approximate dominator sets.

3. Mutant Utility :

Explores the concept of mutant utility across three dimensions: equivalence, triviality, and dominance. It proposes a metric, dominator strength, to assess the value of mutants in contributing to effective mutation analysis.

4. Program Context :

The paper presents a method for modeling program context using a program's abstract syntax tree (AST), discussing how various aspects of the AST (parent and children nodes, data types) can influence mutant utility.

4.1 Motivational Example :

Provides a practical example to illustrate how the utility of mutants varies based on their program context, demonstrating the need for context-aware mutant selection.

4.2 Modeling Program Context :

Details the approach to model program context, focusing on parent statement context, children context, and data type context as predictors of mutant utility.

5. Evaluation :

Outlines an empirical study conducted to assess the impact of program context on mutant utility, utilizing the Defects4J benchmark and the Major mutation framework. The study explores the utility of mutants in terms of equivalence, triviality, and dominance, highlighting the predictive power of program context.

6. Related Work :

Reviews literature related to mutation testing, selective mutation strategies, and approaches to dealing with equivalent and trivial mutants. It contextualizes the presented research within the broader mutation testing discourse, noting contributions towards a more effective selection of high-utility mutants.

7. Conclusions

The paper concludes that considering program context significantly improves the selection of effective mutants, advocating for a context-sensitive mutation approach. It suggests future research directions for refining program context models and exploring the trade-offs between different dimensions of mutant utility.