Mutation Testing and Automated Program Improvement

Mike Papadakis
University of Luxembourg
SnT Center

58th CREST Open Workshop

*Part of the presentation were made by Dr. Yue Jia
Mutation Testing
Mutation Testing
Code-based Mutation Testing

Tests → Original Program

Mutant Mutant Mutant Mutant Mutant Mutant Mutant
Code-based Mutation Testing

Tests ➔ Original Program

Mutant Mutant Mutant Mutant Mutant Mutant Mutant Mutant

gear icon next to each Mutant label
Code-based Mutation Testing

Original Program
Our test suite can capture faults represented by the killed mutants.
Our test suite **cannot** capture faults represented by the survived mutants.
Our test suite **cannot capture** faults represented by the survived mutants.
Mutation Score \approx \frac{\# \text{ Killed}}{\# \text{ All}}

\approx 0.43
**Mutation Testing:** Focuses on the behaviour differences between program variants

A **good** mutant is one that **leads** to test cases that reveal **erroneous** behaviour (of the original program).

**Program Improvement:** Focuses on the program variants with the “same” or “similar” behaviour.
Mutation Testing and Automated Program Improvement

- Same Behaviour
  - Improving other properties
- Different Behaviour
  - Bug Fixes
  - Likely Fault Revealing
- Program Variants
Mutation Testing and Automated Program Improvement

- **Same Behaviour**
  - Improving other properties
- **Different Behaviour**
  - Bug Fixes
  - Likely Fault Revealing

Program Variants
Feasibility and Scalability issues are the same!
Automated Program Improvement
In Mutation Testing, Context (code location and transformation) matters...
Improbable or surprising code is likely to be problematic

Original Program

Improbable code snippets

Variant

More probable code snippets

Ray et al. *On the "naturalness" of buggy code* ICSE'16
Build a model (learn from existing code)
Mutations - Transformations
Mutations-Transformations
Mutations - Transformations
Mutations-Transformations
Original Program

Variants

Variants

Variants

Multiple Predefined Syntactic Transformations

Tailored Mutations
Code Locations & Guidance...
These patterns can be used for identifying interesting locations & guide evolution.
Patterns are described in terms of Control & Data Dependencies and AST Graph elements.
In mutation testing only a few mutants (approximately 3%) are interesting...
Problem

- Set selection problem
- Prioritisation Problem

Variant
Variant
Variant
Variant
Variant
Selection Problem

Variants

Interesting
Prediction Modelling
Machine Learning

Learning phase

Mutants → Defects set

Program features Extraction → Classification Training

Classifier
Classifier that predicts mutants’ Utility
Learning-to-rank Mutants

Programs under test → Mutants → Program features Extraction → ML Classifier → Mutants ranked by probability of utility

Evolution Time
Learning-to-rank Mutants

**Learning phase**
- Mutants
- Defects set
- Program features Extraction
- Classification Training

**Evolution time**
- Mutants
- Programs under test
- Program features Extraction
- ML Classifier
- Mutants ranked by probability of utility
Features

- Depth in the CFG of $B$
- Complexity of $S$ as its number of mutants
- Mutant type of $M$
- Types of mutants on $P_S$
- AST type of $P_S$
- Number of predecessor/successor Basic Blocks of $B$ on the CFG
- Number of AST parents of $S$
- Type of $B$
- Type of $P_S$' basic block
- Number of mutants on $S$
- Number of mutants of statement control out-dependent to $S$
- Number of mutants of statement control in-dependent to $S$
- Number of mutants of statement data out-dependent to $S$
- Number of mutants of statement data in-dependent to $S$
- Number of mutants of statement control out-dependent to $P_S$
- Number of mutants of statement control in-dependent to $P_S$
- Number of mutants of statement data out-dependent to $P_S$
- Number of mutants of statement data in-dependent to $P_S$
- Mutant type and statement type of statement control out-dependent to $S$
- Mutant type and statement type of statement control in-dependent to $S$
- Mutant type and statement type of statement data out-dependent to $S$
- Mutant type and statement type of statement data in-dependent to $S$

$M$ denotes a mutant on statement $S$ in basic block $B$. $P_S$ denotes the AST parent (of $S$).
Learning to Rank Interesting Mutants

Use Gradient Boosted Decision Trees

LLVM-based mutation tool
Control and Data Dependencies are the most informative Features
Subsuming mutants and Faults
Mutation Testing and Automated Program Improvement

![Diagram showing the relationship between program variants, bug fixes, likely fault revealing, and improving other properties.]

- Same Behaviour
- Improving other properties
- Different Behaviour
- Bug Fixes
- Likely Fault Revealing
- Program Variants
Plausibility & Evaluation
Evaluate Solutions

Dynamic - Test-based Evaluation
Evaluate Solutions

Execution Hijack
Proposing & Documenting Changes
Differences in the data states

Differences Synthesise conditions-hints
Mutation Testing and Automated Program Improvement

Build a model (learn from existing code)

Thank you for your attention!

https://sites.google.com/site/mikepapadakis/
michail.papadakis@uni.lu