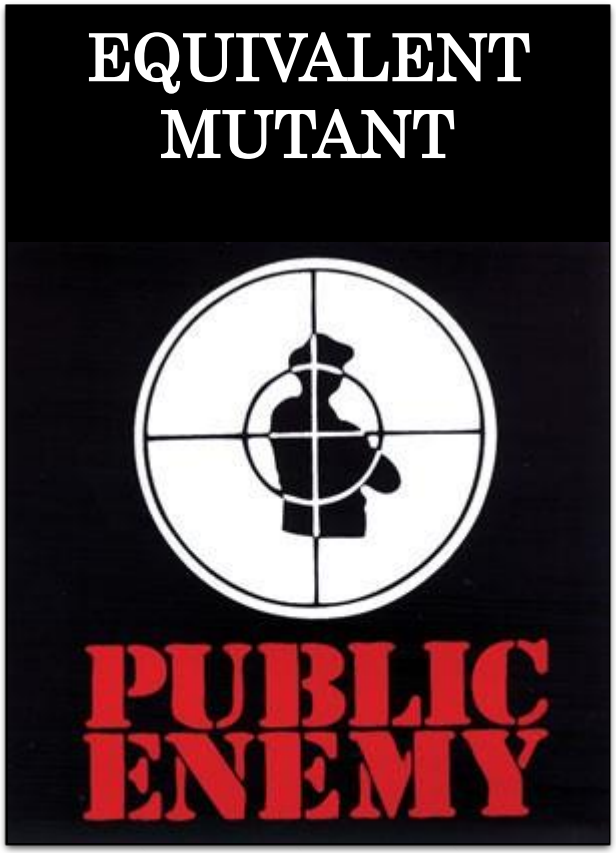
Rehabilitating equivalent mutants as static anomaly detectors in software artifacts

Paolo Arcaini[1], Angelo Gargantini[2], Elvinia Riccobene[3], Paolo Vavassori[2]

1 Charles University in Prague, Czech Republic
2 University of Bergamo, Italy
3 University of Milan, Italy

Rehabilitating equivalent mutants



Equivalent mutants are seen as an inconvenience:

- considered one of the main causes why mutation testing is seldom used in practice
- several attempts try to eliminate or to avoid them

In this work:
exploring the positive side of equivalent mutants

Software (static) anomalies

- **Software anomaly [IEEE]** Any condition that deviates from the expected based on requirements specifications, design documents, user documents, standards, etc. or from someone's perceptions or experiences
- We focus on **static anomalies**, i.e., anomalies that can be removed without changing the "meaning" of the artifact
  - Static anomalies regard the structure of the artifacts and they relate to qualities that may be statically measured

- **Is it possible to remove static anomalies using equivalent mutants?**

Defining anomalies in terms of equivalent mutants

- Assuming
  - a quality **Q** over artifacts and that **Q** induces a partial order (of better quality) $>_Q$
  - possible to define and check equivalence ≡ between artifacts

Static anomaly. Given an artifact A and its mutation A', if A' is equivalent to A but $A' >_Q A$, then A contains a static anomaly. The anomaly is the difference between A' and A

Mutation operators as anomaly detectors and removers

Thesis it is possible to use a suitable mutation operator that detects and removes anomalies

- Finding anomalies:

1. Build a mutation A' for A
2. Compute their qualities Q_A and $Q_{A'}$
3. Check the equivalence between A' and A

IF $Q_{A'} > Q_A$ and $A' \equiv A$

THEN anomaly found (and removed)

Mutation operators as anomaly detectors

Ingredients for an anomaly detector

Thesis it is possible to use a suitable mutation operator that detects and removes anomalies

- In the paper many examples that confirm our thesis
- For every example:

Anomaly

Mutation operator

Quality

Equivalence checking

Source code

- Anomaly: (dead code) DD - A recently defined variable is redefined
- Quality: code compactness
- Mutation operator: Statement deletion operator (SDL)

```
public int m(int b) {
 int a;
 a = 2;
 a = b;
 return a;
}
```

SDL

≡

```
public int m(int b) {
 int a;

 a = b;
 return a;
}
```

- Equivalence checking: Very hard. There are several attempts to automatize the solution of this problem. Some incomplete solutions are acceptable (e.g. Papadakis et al.'s work at ICSE2015)

Boolean expressions

- Anomaly: redundant conditions
- Quality: simplicity (1/ # conditions)
- Mutation operator: Missing Variable Fault (MVF)

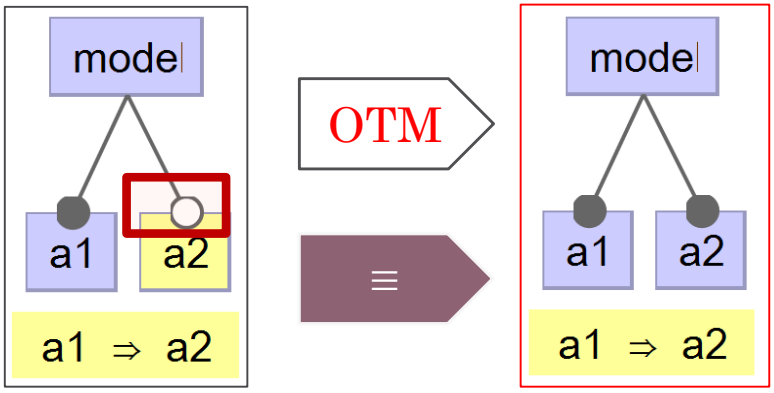
$x \leq 10 \wedge x \leq 5$ → MVF → $x \leq 5$

≡

- Equivalence checking: simple with SAT/SMT – taking into account constraints can be challenging

Feature models

- Anomaly: false optional: if a feature is marked as optional but it is present in all the products of the FM
- Quality: solvability, $\frac{\#\text{mandatory features}}{\#\text{features}}$
- Mutation operator: Optional To Mandatory (OTM)



- Equivalence checking: Translation to SAT

Other examples (not in the paper)

- We found that also for other formalisms equivalent mutants can be used to detect static anomalies
- NuSMV model checker models
  - **Anomaly: vacuity**
  - Equivalence checking: using NuSMV itself
  - See:
    - Paolo Arcaini, Angelo Gargantini, Elvinia Riccobene: A model advisor for NuSMV specifications. ISSE 7(2): 97-107 (2011)
- Combinatorial interaction testing models
  - **Anomaly: vacuity**
  - Equivalence checking: SMT solver
  - See:
    - Paolo Arcaini, Angelo Gargantini, Paolo Vavassori, Validation of Models and Tests for Constrained Combinatorial Interaction Testing. ICST Workshops 2014

Not all the mutation operators are equal

Some mutation operators

1. may both decrease the quality and produce a non-equivalent mutant – **both quality and equivalence must be checked**
2. always increase the quality but can produce non-equivalent mutants – **equivalence must be checked**
   - Example: Statement Deletion mutation operator (SDL) always improves code compactness but may change the behavior
3. always produce equivalent mutants, but they may decrease the quality – **quality must be measured**
   - Example: Refactoring produces an equivalent mutant but must be used in a way that increases the quality

Goal: mutation operators applications that guarantee equivalence and better quality

Conclusions

- Exploring the positive side of equivalent mutants
- **Is it possible to define static anomalies using equivalent mutants?**

Thesis it is possible to use mutation operators to detect and remove anomalies

Examples: source code, Boolean expressions, feature models, ….

Thank you