Mutation-based testing

May 07, 2019

Mutation testing: mutant generation

Program

lhs < rhs
lhs <= rhs
lhs < rhs
lhs != rhs
stmt no-op

Mutation testing: mutant generation

Program

lhs < rhs
lhs <= rhs
lhs < rhs
lhs != rhs
stmt no-op

Mutants
Mutation testing: mutant generation

Program

Mutants

lhs < rhs
lhs <= rhs
lhs < rhs
lhs != rhs
stmt no-op

Program

Mutants

lhs < rhs
lhs <= rhs
lhs < rhs
lhs != rhs
stmt no-op

Mutation testing

Program

Mutants

Tests

How expensive is mutation testing?

PollEv.com/renejust859
- Generating mutants is cheap
- Executing tests on mutants is cheap(ish)
- Writing tests is expensive

Is the mutation score meaningful?

- Generating mutants is cheap
- Executing tests on mutants is cheap(ish)
- Writing tests is expensive

Few mutants are effective

Engineers care about mutants that elicit effective tests!
A mutant is not effective if:
- it cannot be detected (semantically equivalent)
- it fails for any given test (trivial)
- it is dominated by other mutants (subsumed)
Killable vs. productive mutants

**Status quo**
- **Killable** mutants are good $\rightarrow$ tests
- **Equivalent** mutants are bad $\rightarrow$ no tests

**A more nuanced view**
- Killable vs. equivalent is too simplistic
- **Productive mutants** elicit effective tests, but
  - Killable mutants can be unproductive
  - Equivalent mutants can be productive
Killable vs. productive mutants

Status quo
- Killable mutants are good $\implies$ tests
- Equivalent mutants are bad $\implies$ no tests

A more nuanced view
- Killable vs. equivalent is too simplistic
- Productive mutants elicit effective tests, but
  - Killable mutants can be unproductive
  - Equivalent mutants can be productive

The notion of productive mutants is fuzzy!
A mutant is productive if it is
1. killable and elicits an effective test or
2. equivalent and advances code quality or knowledge

---

Original program
```java
public double getAvg(double[] nums) {
  double sum = 0;
  int len = nums.length;
  for (int i = 0; i < len; ++i) {
    sum = sum + nums[i];
  }
  return sum / len;
}
```

Mutant
```java
public double getAvg(double[] nums) {
  double sum = 0;
  int len = nums.length;
  for (int i = 0; i < len; ++i) {
    sum = sum * nums[i];
  }
  return sum / len;
}
```

Is the mutant is killable?

The mutant is killable, but is it productive?
**Killable vs. productive mutants (2)**

**Original program**

```java
public double getAvg(double[] nums) {
    int len = nums.length;
    double sum = 0;
    double avg = 0;
    for (int i = 0; i < len; ++i) {
        avg += nums[i] / len;
        sum += nums[i];
    }
    return sum / len;
}
```

**Mutant**

```java
public double getAvg(double[] nums) {
    int len = nums.length;
    double sum = 0;
    double avg = 0;
    for (int i = 0; i < len; ++i) {
        avg *= (nums[i] / len);
        sum += nums[i];
    }
    return sum / len;
}
```

**Is the mutant killable?**

The mutant is **not killable, but is it unproductive?**

**Killable vs. productive mutants (3)**

**Original program**

```java
Set cache = new HashSet(a * b);
```

**Mutant**

```java
Set cache = new HashSet(a + b);
```

**Is the mutant killable?**

The mutant is **killable, but is it productive?**
Many mutants but not all are productive

A mutant is **productive** if it is:

1. **killable** and elicits an effective test or
2. **equivalent** and advances code quality or knowledge

Existing selection strategies

Sample mutation operators

Effective tests

Sample generated mutants
Existing strategies are \textit{program-independent} and \textit{no better than random}.*

*Gopinath et al., ICSE’16, *Kurtz et al., FSE’16