Killing Stubborn Mutants with Symbolic Execution

Thierry Titcheu Chekam  Mike Papadakis  Maxime Cordy  Yves Le Traon

*ACM Transactions on Software Engineering and Methodology (TOSEM)* Journal
**Mutation**

**Program**

Maximum \((a, b, c)\)

1. \(\text{max} = a;\)
2. \(\text{if } (b > \text{max})\)
3. \(\text{max} = b;\)
4. \(\text{if } (c > \text{max})\)
5. \(\text{max} = c;\)
6. \(\text{return max};\)

**Mutation operators**

- \(\text{if } (b > \text{max})\) → \(\text{if } (b \neq \text{max})\)
- \(\text{if } (c > \text{max})\) → \(\text{if } (c = \text{max})\)
- \(>\) → \(<\)

**Mutants**

Mutant 1

Maximum \((a, b, c)\)

1. \(\text{max} = a;\)
2. \(\text{if } (b \neq \text{max})\)
3. \(\text{max} = b;\)
4. \(\text{if } (c > \text{max})\)
5. \(\text{max} = c;\)
6. \(\text{return max};\)

Mutant 2

Maximum \((a, b, c)\)

1. \(\text{max} = a;\)
2. \(\text{if } (b > \text{max})\)
3. \(\text{max} = b;\)
4. \(\text{if } (c = \text{max})\)
5. \(\text{max} = c;\)
6. \(\text{return max};\)

Mutant 3

Maximum \((a, b, c)\)

1. \(\text{max} = a;\)
2. \(\text{if } (b > \text{max})\)
3. \(\text{max} = b;\)
4. \(\text{if } (c > \text{max})\)
5. \(\text{max} = c;\)
6. \(\text{return max};\)
Maximum (a, b, c)
1. max = a;
2. if (b > max)
3. max = b;
4. if (c > max)
5. max = c;
6. return max;

Mutant (M)
1. max = a;
2. if (b > max)
3. max = b;
4. if (c > max)
5. max = c;
6. return max;

Generated tests
Mutant Killing

| Input | Output |
|-------|--------|
| Original (P) | State\(_P\) \(=\) State\(_M\) |
| Mutant (M) | State\(_M\) \(\neq\) State\(_M\) |
Symbolic Mutant Test Generation

Original

Mutant

Path ID

Original Mutant Path ID

1 2 3 4 5 6 7

1 2 3 4 5 6 7

$M_1$
Symbolic Mutant Test Generation

Original

Mutant

\((\text{Out}_{\text{Original}} \neq \text{Out}_{\text{Mutant}}) \land \phi_{\text{Original}} \land \phi_{\text{Mutant}}\)
Symbolic Mutant Test Generation

Original Mutant State

Mutant

\[ (\text{State}_{\text{Original}} \neq \text{State}_{\text{Mutant}}) \land \phi_{\text{Original}} \land \phi_{\text{Mutant}} \]

\[ M_1 \]
Symbolic Mutant Test Generation

Program

```c
int func (int x) {
    int n = 0;
    if (x < 0) {
        n = x; // M1 (n = x + 1);
        if (n)
            n++;
    }
    return n;
}
```

### Original
- `x_0 = x_0`, `n = 0`
- `x_0 < 0`

### Mutant M₁
- `x_0 = x_0`, `n = x_0 + 1`
- `x_0 < 0`

The mutant M₁ is not killed by the test.

Z3 solver

Out\_Original = 0
Out\_Mutant = 0

`x_0 = -1`

\[(x_0 \neq x_0 + 1) \land x_0 < 0 \land x_0 < 0\]
Symbolic Mutant Test Generation

Program
```
int func (int x) {
    int n = 0;
    if (x < 0) {
        n = x; // M
    }
    if (n) {n++;}
    return n;
}
```

Infeasible
```
Z3 solver
```
```
(x_0 + 1) \neq (x_0 + 1) \land x_0 < 0 \land x_0 = -1
```

Symbolic Mutant Test Generation

Program

```c
int func(int x) {
    int n = 0;
    if (x < 0) {
        n = x; // M₁ (n = x + 1);
    }
    if (n)
        n++;
    return n;
}
```

The mutant $M₁$ is killed by the test

$Out_{Original} = -1$

$x₀ = -2$

$Z3$ solver

$(x₀ + 1) \neq x₀ + 2) \land x₀ < 0 \land x₀ < -1$
SEMu implements heuristics that enable scalable and deeper mutant-error propagation
Valuable Mutants

SEMu Cost-control Heuristics

(a)

(b)

(c)
Completeness Preserving Optimizations

**Program**

```c
int func (int x) {
    int n = 0;
    if (x < 0) {
        n = x; // M_1 (n = x + 1) M_2 (n += x)
        if (n)
            n++;
    }
    return n;
}
```

**Meta-mutant Program**

```c
unsigned MUTANT_ID;
int func (int x) {
    int n = 0;
    if (x < 0) {
        switch(MUTANT_ID){
            case 1: n = x + 1; break;
            case 2: n += x; break;
            default: n = x;
        }
        if (n)
            n++;
    }
    return n;
}
```

**Shared**
**SEMu** is implemented on KLEE

Our implementation added/modified more than 8,000 lines of code.

https://github.com/thierry-tct/KLEE-SEMu
We evaluate *SEM*u and compare with KLEE and the state-of-the-art-approach (*infect-only*)

*Subjects:* 36 Programs from GNU Coreutils
Valuable Mutants Evaluation - Procedure

DEV + Auto Stmt cov tests = Initial Test Suite

4,434 Mutants

New Tests: SEMu, Infect-only, KLEE

60,018

36,229 (61.5%)

23,789
Evaluation - Results

Proportion of Mutants Killed (%)

- 

semu: 37.2

infection-only: 17.2

klee: 17.2
Valuable Mutants Evaluation - Results
Valuable Mutants Evaluation - Results
