Software testing 2014

# ASSIGNMENT 2- REPORT

November 16, 2014

Stephane KI
Abo Akademy University
Student Number 71364
I.T - Software Engineering
stephane.ki@abo.fi

# Table of contents

# 1 Activity

Chose 5 of the mutants that could not be killed, compute the RIP conditions for each, and tell if any of them could be weakly killed.

# 2 Mutants

## 2.1 Mutant 1- AOIS_19

```
rslt=rslt * left;
//BECOMES
rslt=rslt++ * left;
```

1. Reachability

   ```
   right != 0 && right >=2
   ```

2. Infection

   ```
   false
   ```

3. Propagation

   ```
   false
   ```

   Infection and propagation are false because after the post incrementation of the variable, the value is override.

   Example: left=3 and right=2. The loop will be executed one time and the statement *rslt=rslt++ * left* can be translated like that:

   ```
   int u= rslt*left;
   rslt=rslt+1;
   rslt=u;
   return rslt;
   ```

   So the mutant is equivalent.

## 2.2 Mutant 2 - AOIS_20

```
rslt=rslt * left;
//BECOMES
rslt=rslt-- * left;
```

1. Reachability

```
right != 0 && right >=2
```

2. Infection

```
false
```

3. Propagation

```
false
```

Infection and propagation are false because after the post decrementation of the variable, the value is override.

Example: left=3 and right=2. The loop will be executed one time and the statement *rslt=rslt- - * left* can be translated like that:

```
int u= rslt*left;
rslt=rslt -1;
rslt=u;
return rslt;
```

So the mutant is equivalent.

## 2.3 Mutant 3 - AOIS_25

```
return rslt;
//BECOMES
return rslt++;
```

1. Reachability

```
true
```

2. Infection

```
true
```

3. Propagation

```
false
```

The propagation is *false* because the variable is incremenented after the return statement ; so the returned result is not modified.

### 2.4 Mutant 4 - AOIS_256

```
return rslt;
//BECOMES
return rslt--;
```

1. Reachability

   ```
   true
   ```

2. Infection

   ```
   true
   ```

3. Propagation

   ```
   false
   ```

   The propagation is *false* because the variable is decremenented after the return statement ; so the returned result is not modified.

### 2.5 Mutant 5 - ROR_4

```
if (right == 0) {
}
//BECOMES
if (right <= 0) {
}
```

1. Reachability

   ```
   true
   ```

2. Infection

   ```
   right < 0;
   ```

   But knowing that the precondition says that *right* should be greater or eqaul to zero so the previous condition is false.

   **The Mutant is equivalent!**

3. Propagation (Does not mater ! ! !)