Infeasible Path Detection and Code Pattern Mining
Improving Quality of Automated Software Tests

Infeasible paths are execution paths which are impossible to traverse, determining infeasibility is a
NP-Complete problem. Being able to predict infeasible paths would allow software testing tools to
ignore these paths and improve quality of tests.

How? Leveraging on previous research that has proven correlations between infeasible paths and
certain code patterns.

What does this research improve? Previous research was not general enough. This
research would aim to form a general method of mining infeasible code patterns.

Simple Mining Example:

```java
01. public void f3(int x, int y) {
02.     int a = 10;
03.     if ( a != 10 ){
04.         }
05. }
```

Collected Expressions: [a != 10], [a == 10]
Collected Mappings: {a=10}

Theorem Solver:
[10 != 10] is unsat

Substituted Expressions: [10 != 10],[10 == 10]

Generalization Algorithm

Infeasible Code Pattern: [10 != 10]

Detection Phase of Mined Code Pattern:

```java
01. public void h1() {
02.     int b = 20;
03.     if ( b != 20 ){
04.         }
05. }
```

Infeasible Code Pattern: [10 != 10] = Line 4 cannot be reached

Student: Elgin Lee Wei Sheng
Supervisor: Associate Professor Yeo Chai Kiat