Improving Evolutionary Class Testing in the Presence of Non-Public Methods

Evolutionary Class Testing
Automatic Generation of Unit Test Suites with High Code Coverage:
- Transformation to Optimization Problems
- Application of Genetic Programming (Evolutionary Algorithms)

Random Generation of Candidate Test Sequences
Fitness Evaluation
Selection
Termination?
Distance Calculation
Execution & Monitoring
Test Sequence Crossover
Test Sequence Mutation

Encoding of Test Sequences by Method Call Trees

Testing Non-Public Methods
Increased Coverage of Non-Public Methods and hence Improved Test Thoroughness and Quality without Breaking Data Encapsulation (Using the Public Class Interface only):
- Static Call Point Analysis
- Extended Fitness Functions using Call Points

Call Point Identification

Fitness Function Landscape for Non-Public Targets

Benefit: Higher Code Coverage and Test Quality
Empirical Case Study suggests Effectiveness of Improvement

Stefan Wappler
Technical University of Berlin, Daimler Center for Automotive IT Innovations, Ernst-Reuter-Platz 7, D-10587 Berlin, Germany, stefan.wappler@tu-berlin.de

Ina Schieferdecker
Technical University of Berlin, Sekretariat 5-14, Franklinstr. 28/29, D-10587 Berlin, Germany, ina.schieferdecker@fokus.fhg.de

```java
public class ClassA {
    private int x;
    public void m1(int a, int b) {
        if (a == 0)
            m3(b);
    }
    public void m2(int a, int b) {
        if (a == 1)
            m3(b);
    }
    private void m3(int x) {
        if (x > -10 && x < 10)
            this.x = x;
    }
}
```

```java
BigInteger i1 = BigInteger.valueOf(10387);
BigInteger i2 = i1.not();
Random r1 = new Random();
BigInteger i3 = new BigInteger(-23, r1);
BigInteger i4 = i2.divide(i3);
```