Introduction to Soot

Automated testing and verification

J.P. Galeotti - Alessandra Gorla
The Java virtual machine (JVM)

• The Java compiler translates a Java program into Java bytecode (input language of the JVM)
• The Java bytecode is similar to machine language (assembler) for the JVM
The SOOT framework

- Set of Java APIs to handle Java bytecode
  - Optimization
  - Annotation
- It was created by the Sable Research Group ([http://www.sable.mcgill.ca](http://www.sable.mcgill.ca))
- Web:
Intermediate representation

- Jimple: main Soot intermediate representation
- Grimp: Jimple + complex expressions
- Shimple: Jimple + SSA
- Baf: Java bytecode “for humans”
Intermediate representations

Java

- Grimple (closer to Java)
- Jimple/Shimple
- Baf (closer to Bytecode)

Bytecode
Jimple

- A Jimple representation can be created from:
  - Java source code
  - Java bytecode

- Main features:
  - 3-address code: all instructions use at most 3 addresses
  - Unstructured: while, if, for, etc. are replaced with GOTO statements
  - Typing: all local variables are typed
Example: Original Java and Jimple transformation

```java
if (x+y!=z)
    return;
else
    System.out.println("foo");

t = x+y;
if (t==z) goto label0;
return;
lable0:
    ref = System.out;
    ref.println("foo");
```
Soot: dataflow analysis

- **Eclipse plugin:**
  - Right click on Java file to analyze

- **Soot->Process Source File -> Run Soot...**
  - Output Options -> Output Format -> Jimple File
  - Phase Options -> Jimple Annotations Pack ->
    - Live Variables Tagger
    - Reaching Defs Tagger
    - Available Expressions Tagger
    - ...etc
Soot: dataflow analysis

- (demo)
Soot: dataflow analysis

- Interactive Mode
  - Run.. -> General Options -> Interactive Mode
- Executing a custom analysis
  - Run... -> Soot Main Class
Developing a Soot analysis

• Create new project
• Add to project build path the libraries:
  • jasminclasses,.jar
  • polyglot.jar
  • sootclasses.jar
• These libraries are stored in the lib/ plugin folder
• Javadoc:
  • http://www.sable.mcgill.ca/soot/doc/
Soot Dataflow Framework

• Direction: Backward or Forward?
• Approximation: May or Must?
• Transfer Function definition:
  • E.g. how x:=expr should be treated?
• Initial state definitions
  • Entry/exit node (depending on direction)
  • Intermediate nodes
1. Dataflow Direction

- Soot has 3 analysis implementations
  - ForwardFlowAnalysis
  - BackwardFlowAnalysis
  - ForwardBranchedFlowAnalysis
- The output is a object:
  - Map<Node,<IN set, OUT set>>
1. Dataflow direction

```java
public class MyFwdAnalysis
    extends ForwardFlowAnalysis<Unit, FlowSet> {

    public MyFwdAnalysis(DirectedGraph<Unit> g) {
        super(g);
        doAnalysis();
    }
}
```
2. Approximation

- Implement methods merge and copy

```java
protected void merge(FlowSet inSet1, FlowSet inSet2, FlowSet outSet) {
    inSet1.intersection(inSet2, outSet);
}

protected void copy(FlowSet srcSet, FlowSet dstSet) {
    srcSet.copy(dstSet);
}
```
3. Transfer Function

- Implement method `flowThrough`

```java
protected void flowThrough(FlowSet inSet, Unit node, FlowSet outSet) {
    Kill(inSet, u, outSet);
    Gen(outSet, u);
}
```

- Methods `kill` and `gen` are defined by the user
4. Initial flows

• The initial flow content for entry/exit points, as well as other nodes:

```java
protected FlowSet entryInitialFlow() {
    return new FlowSet();
}

protected FlowSet newInitialFlow() {
    return new FlowSet();
}
```
FlowSets

- Soot offers several FlowSet implementations:
  - ArraySparseSet
  - ArrayPackedSet
  - ToppedSet
Executing a custom analysis

```java
SootClass c = Scene.v().loadClassAndSupport("MyClass");
c.setApplicationClass();
SootMethod m = c.getMethodByName("myMethod");
Body b = m.retrieveActiveBody();
UnitGraph g = new ExceptionalUnitGraph(b);

MyFwdAnalysis an = new MyFwdAnalysis(g);
for (Unit unit : g) {
    FlowSet in = an.getFlowBefore(unit);
    FlowSet out = an.getFlowAfter(unit);
}
```
Flow Transformation

• The design pattern “Visitor” can be used to traverse the Jimple AST:
  • soot.jimple.AbstractStmtSwitch
  • soot.jimple.AbstractJimpleValueSwitch