How to use Soot

Patrick Lam
University of Waterloo

In Memoriam

Laurie Hendren
December 13, 1958 - May 27, 2019

Raja Vallée-Rai
1975 - 2004

https://github.com/patricklam/soot-spg

For more information on Soot:

https://soot-oss.github.io/soot/

Documentation:

https://github.com/soot-oss/soot/wiki/Tutorials

A Survivor's Guide to Java Program Analysis with Soot:

https://www.brics.dk/SootGuide/

I. Soot as a command-line tool

Get & use pre-built Soot

https://repo1.maven.org/maven2/org/soot-oss/soot/4.2.1/soot-4.2.1-jar-with-dependencies.jar

Produce abbreviated Jimple output (cmdline/simple-example):

```
java -jar soot-4.2.1-jar-with-dependencies.jar -f j -pp -cp . Example
```

Afterwards, get sootOutput/Example.jimp.

More details at https://github.com/soot-oss/soot/wiki/Running-Soot.

What you can get out of Soot

Outputs:

● jimp/jimple:	Three-address code
● shimp/shimple:	SSA version of Jimple
● b/baf:	Soot version of stack-based bytecode
● grimp/grimple:	Jimple with aggregated expressions
● dex:	Dalvik VM files
● class:	Back to Java .class files
● dava:	Decompile Jimple to Java

… and others.

More sophisticated example:

Uses pointer analysis and static method binding, no inlining (cmdline/complex-example):

```
java -jar soot-4.2.1-jar-with-dependencies.jar -W -app -f j -p jb
use-original-names:true -p cg.spark on -p cg.spark
simplify-offline:true -p jop.cse on -p wjop.smb on -p wjop.si off -pp
-cp . -process-dir multi-classes
```

II. Building tools with Soot

IIa. Setting up a Benchmark

I put a microbenchmark in driver-generator/Benchmarks/microbenchmark.

```
    $ mvn package
```

generates:

```
    driver-generator/Benchmarks/microbenchmark/target/payroll-test
-0.0.1-SNAPSHOT-tests.jar
```

We can look at it (`cmdline/view-tests-cmdline`):

```
java -jar ../../../cmdline/soot-4.2.1-jar-with-dependencies.jar
-f j -print-tags -p jb use-original-names:true -pp
-process-jar-dir mvn_dependencies -process-jar-dir target
```

IIb. Building on Soot

1. Create a `pom.xml` that includes Soot (see example).
2. Create a custom Main class that calls Soot.

```
$ mvn clean compile assembly:single
```

3. Run the resulting jar

```
$ java -jar
target/standalone-jar-with-dependencies.jar
```

IIc. Live Coding a Driver Generator

Other things we can talk about

- Dataflow analysis
  https://github.com/soot-oss/soot/wiki/Implementing-an-intra-procedural-data-flow-analysis-in-Soot
- Pointer analysis
- Tamiflex
- Android apps