Lab 0: Introduction to SootUp and PhASAR

1 Introduction to SootUp

Exercise 1

Converting Java to Jimple

- Have a look at `Exercise1.java` and `JimpleBodyPrinter.java` and try to understand the purposes of these classes.
- What does `print(JavaSootMethod method)` method in `JimpleBodyPrinter.java` do?
- Examine the jimple output of the methods in `SampleClass.java`, how does various statements (assignments, branches, loops) look like after conversion?
- Optionally write your own methods to try it out.

Exercise 2

Counting Statements

- Have a look at `Exercise2.java` and `StatementCounter.java` and try to understand the purposes of these classes.
- What does `countStmts(JavaSootMethod method)` method in `StatementCounter.java` do?

Exercise 3

Classifying Statements

- Have a look at `Exercise3.java` and `StatementClassifier.java` and try to understand the purposes of these classes.
- What does `classify(JavaSootMethod method)` method in `StatementClassifier.java` do?
- What types of statements can you find in a method body other than `JAssignStmt` and `JInvokeStmt`? Extend `classify(JavaSootMethod method)` method to find these statements. (You can write your own methods in `SampleClass.java`)

Exercise 4

Forbidden Method Detection

- Have a look at `Exercise4.java` and `ForbiddenMethodDetection.java` and try to understand the purposes of these classes.
- What does `accept()` method in `ForbiddenMethodDetection.java` do?
- Current implementation of `accept()` is imprecise, it finds a call to the actual `forbiddenMethod` in `forbiddenClass` but it also finds a call to one of the false `forbiddenMethods`. Extend the `accept()` method to make it more precise.

2 Introduction to PhASAR

Exercise 5

Converting C to LLVM IR

- Have a look at `unittests/BodyPrinterTest.cpp` and try to understand the purpose of the `Test01` function.
- What does `IR.emitPreprocessedIR()` do? How does its output differ from the content of `build/target/Sample_c_dbg.ll`?

Exercise 6

Counting Statements

- Have a look at `unittests/StatementCounterTest.cpp` and try to understand the purpose of the `Test01` function.
- What do you recognize when looking at the output?

Exercise 7

Classifying Statements

- Have a look at `unittests/StatementClassifierTest.cpp` and try to understand the purpose of the `Test01` function.
- What does the `classify` function do?
- What types of instructions can you find in a method body other than `AllocaInst` and `LoadInst`? Extend the `classify` function to find these instructions. (You can write your own functions in `Sample.c`)

Exercise 8

Forbidden Method Detection

- Have a look at `unittests/ForbiddenMethodDetectionTest.cpp` and try to understand the purpose of the `Test01` function.
- What does the `verify` function do?
- The current implementation of `verify` is imprecise. It finds a call to `forbiddenFn` where the parameter comes from `source()`, but it also finds others.
- Extend the `verify` function to make it more precise.