CS846
Machine Learning for Software Engineering

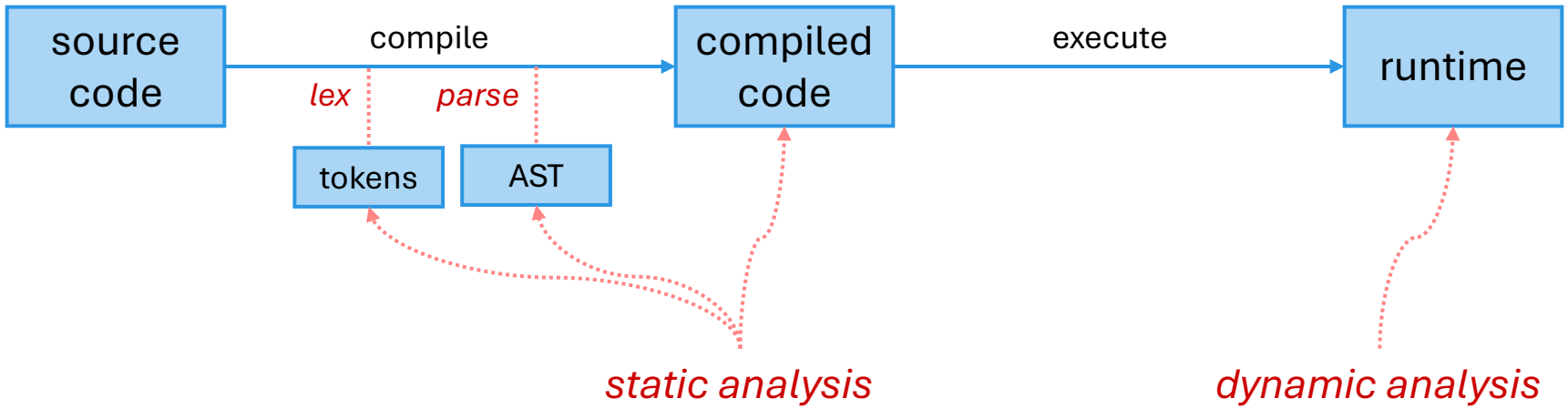
Pengyu Nie

Dynamic Analysis

Comparison to static analysis

Instrumentation / bytecode manipulation

Program Analyses Overview



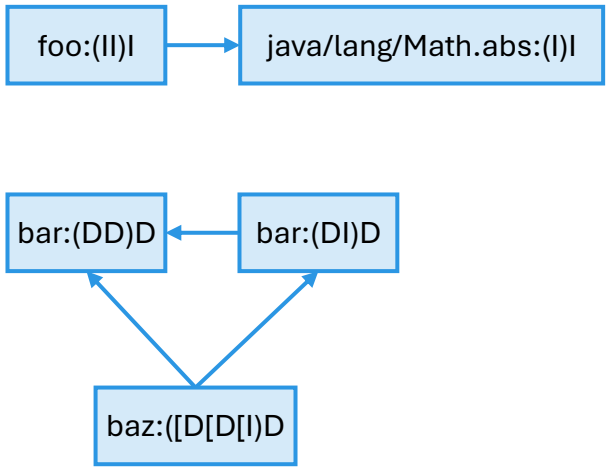
Recap: Call Graph from Static Analysis

```
public int foo(int x, int y) {
  int a = Math.abs(x);
  int b = Math.abs(y);
  …
}

public double bar(double t, int p) {
  return bar(t, (1 - p / 100.0d));
}
```

```
public int foo(int, int)
  descriptor: (II)I
  Code:
    0: iload_1
    1: invokestatic java/lang/Math.abs:(I)I
    4: istore_3
    5: iload_2
    6: invokestatic java/lang/Math.abs:(I)I
    9: istore 4
    …

public double bar(double, int)
  descriptor: (DI)D
  Code:
    0: aload_0
    1: dload_1
    2: dconst_1
    3: iload_3
    4: i2d
    5: ld2_w 100.0d
    8: ddiv
    9: dsub
   10: invokevirtual bar:(DD)D
   13: dreturn
```

👆 Command to view bytecode:
javap -c -p -v target/classes/ca/uwaterloo/cs846/Subject.class

```
ca/uwaterloo/cs846/Subject.<init>:()V ->
  java/lang/Object.<init>:()V
ca/uwaterloo/cs846/Subject.foo:(II)I ->
  java/lang/Math.abs:(I)I
ca/uwaterloo/cs846/Subject.bar:(DI)D ->
  ca/uwaterloo/cs846/Subject.bar:(DD)D
ca/uwaterloo/cs846/Subject.baz:([D[D[I)D ->
  ca/uwaterloo/cs846/Subject.bar:(DI)D
  ca/uwaterloo/cs846/Subject.bar:(DD)D
```



Problem with Static Analysis

Dynamic dispatch?

```
public interface Shape {
  public void draw();
}

public class Line implements Shape {
  @Override
  public void draw() { … };
}

public class Rectangle implements Shape {
  @Override
  public void draw() { … };
}

public class Circle implements Shape {
  @Override
  public void draw() { … };
}
```

?

```
public class ShapeMain {
  public static void drawShape(Shape shape) {
    shape.draw();
  }

  public static void main(String... args) {
    Shape[] shapes = ...;
    for (Shape shape: shapes) {
      drawShape(shape);
    }
  }
}
```

Static Analysis Results

```java
public interface Shape {
  public void draw();
}

public class Line implements Shape {
  @Override
  public void draw() { … };
}

public class Rectangle implements Shape {
  @Override
  public void draw() { … };
}

public class Circle implements Shape {
  @Override
  public void draw() { … };
}
```

call

inherit

```java
public class ShapeMain {
  public static void drawShape(Shape shape) {
    shape.draw();
  }

  public static void main(String... args) {
    Shape[] shapes = ...;
    for (Shape shape: shapes) {
      drawShape(shape);
    }
  }
}
```

call

static analysis need to overestimate call edges!

```
ca/uwaterloo/cs846/exp/Circle.<init>:(III)V ->
  java/lang/Object.<init>:()V
ca/uwaterloo/cs846/exp/Circle.draw:()V ->
  java/io/PrintStream.println:(Ljava/lang/String;)V
ca/uwaterloo/cs846/exp/Rectangle.<init>:(IIII)V ->
  java/lang/Object.<init>:()V
ca/uwaterloo/cs846/exp/Rectangle.draw:()V ->
  java/io/PrintStream.println:(Ljava/lang/String;)V
ca/uwaterloo/cs846/exp/ShapeMain.<init>:()V ->
  java/lang/Object.<init>:()V
ca/uwaterloo/cs846/exp/ShapeMain.drawShape:(Lca/uwaterloo/cs846/exp/Shape;)V ->
  ca/uwaterloo/cs846/exp/Shape.draw:()V
ca/uwaterloo/cs846/exp/ShapeMain.main:([Ljava/lang/String;)V ->
  ca/uwaterloo/cs846/exp/Rectangle.<init>:(IIII)V
  ca/uwaterloo/cs846/exp/ShapeMain.drawShape:(Lca/uwaterloo/cs846/exp/Shape;)V
  ca/uwaterloo/cs846/exp/Line.<init>:(IIII)V
ca/uwaterloo/cs846/exp/Line.<init>:(IIII)V ->
  java/lang/Object.<init>:()V
ca/uwaterloo/cs846/exp/Line.draw:()V ->
  java/io/PrintStream.println:(Ljava/lang/String;)V
```

Instrumentation / Bytecode Manipulation

- Collect the exact methods called during execution
- How?
  Insert additional logging instructions into bytecode

caller

```
// ShapeMain
public static void drawShape(ca.uwaterloo.cs846.exp.Shape)
  descriptor: (Lca/uwaterloo/cs846/exp/Shape;)V
  Code:
    0: aload_0
    1: invokeinterface ca/uwaterloo/cs846/exp/Shape.draw:()V, 1
    6: return
```

before each invoke instruction

```
+ ldc “ShapeMain.drawShape”
+ invokestatic DynamicAnalyzer.logCaller
```

callee

```
// Line (same for Rectangle / Circle)
public void draw()
  descriptor: ()V
  Code:
    0: getstatic java/lang/System.out:Ljava/io/PrintStream
    3: aload_0
    …
```

at the beginning of each method

```
+ ldc “Line.draw”
+ invokestatic DynamicAnalyzer.logCallee
```

Bytecode Analysis/Manipulation Resources (for Java)

- Libraries
  - ASM https://asm.ow2.io/
  - ByteBuddy https://bytebuddy.net/#/
- References
  - Java instrumentation API https://docs.oracle.com/en/java/javase/21/docs/api/java.instrument/java/lang/instrument/Instrumentation.html
  - List of bytecode instructions Wikipedia https://en.wikipedia.org/wiki/List_of_Java_bytecode_instructions
  - Java specifications https://docs.oracle.com/javase/specs/
    source code: "The Java Language Specification, Java SE xxx Edition"
    bytecode: "The Java Virtual Machine Specification, Java SE xxx Edition"