Symbolic execution for Java executables using angr

Sebastiano Mariani && Sydney Ma

OUTLINE

1

INTRODUCTION

What is symbolic execution?

2

BUILDING BLOCKS

Angr and Soot IR

3

IMPLEMENTATION DETAILS

Execution states and a simple example

4

DEMO

Good old fauxware

5

PROBLEMS AND LIMITATIONS

Array with symbolic indexes, reflection, etc.

6

CONCLUSION

It's just the beginning

WHAT IS SYMBOLIC EXECUTION

Introduction

A method of analyzing a program to determine what inputs cause each part of the program to execute

```
void foobar(int a) {
    int x = 4, y = 0;
    if(a != 0) {
        y = a + 3;
    }
    assert(x-y != 0);
}
```

WHAT IS SYMBOLIC EXECUTION

Introduction

A method of analyzing a program to determine what inputs cause each part of the program to execute

```
void foobar(int a) {
    int x = 4, y = 0;
    if(a != 0) {
        y = a + 3;
    }
    assert(x-y != 0);
}
```

WHAT IS SYMBOLIC EXECUTION

Introduction

A method of analyzing a program to determine what inputs cause each part of the program to execute

```
void foobar(int a) {
    int x = 4, y = 0;
    if(a != 0) {
        y = a + 3;
    }
    assert(x-y != 0);
}
```

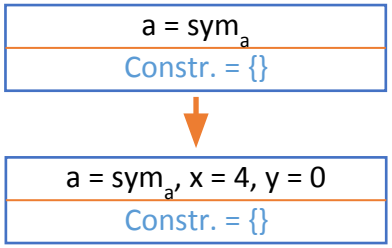
$a = sym_a$
Constr. = {}

WHAT IS SYMBOLIC EXECUTION

Introduction

A method of analyzing a program to determine what inputs cause each part of the program to execute

```
void foobar(int a) {
    int x = 4, y = 0;
    if(a != 0) {
        y = a + 3;
    }
    assert(x-y != 0);
}
```

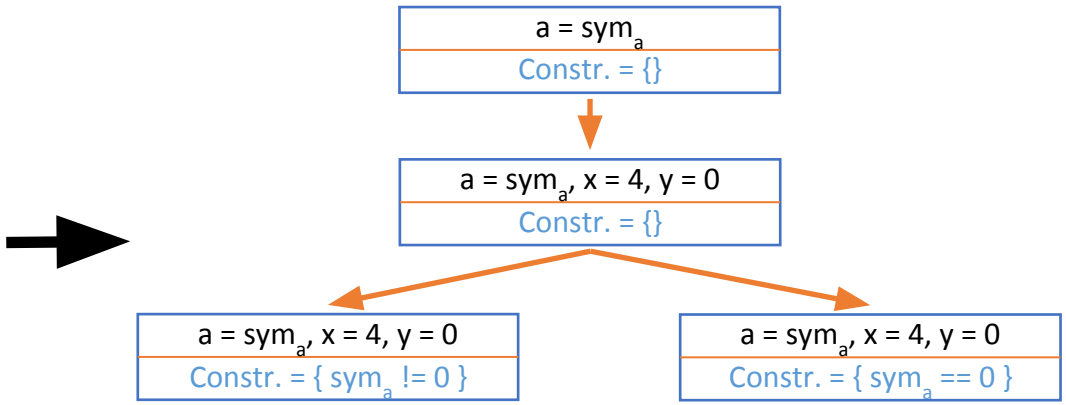


WHAT IS SYMBOLIC EXECUTION

Introduction

A method of analyzing a program to determine what inputs cause each part of the program to execute

```
void foobar(int a) {
    int x = 4, y = 0;
    if(a != 0) {
        y = a + 3;
    }
    assert(x-y != 0);
}
```

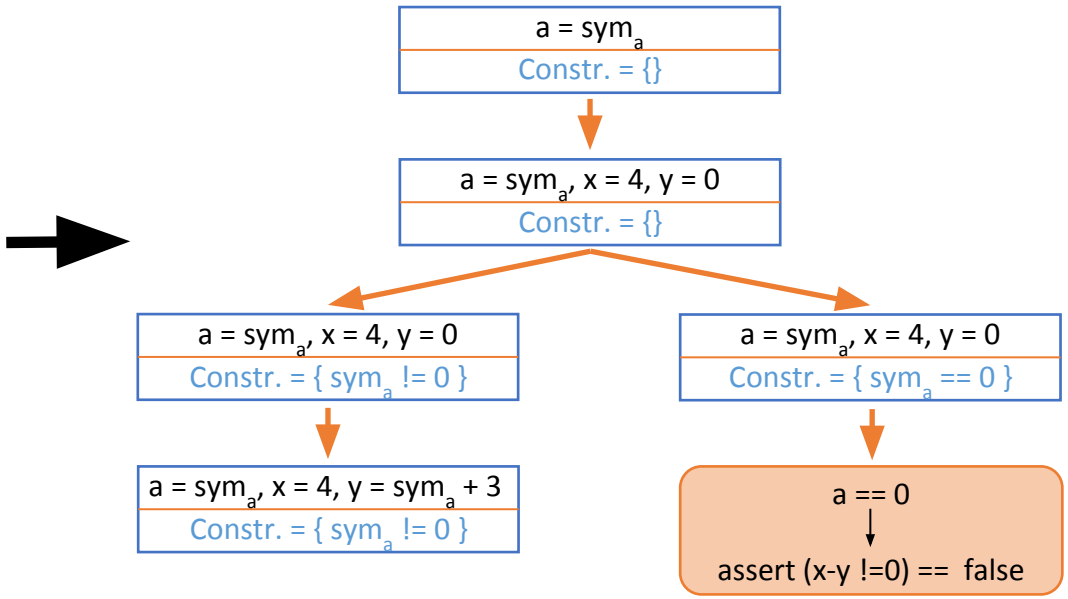


WHAT IS SYMBOLIC EXECUTION

Introduction

A method of analyzing a program to determine what inputs cause each part of the program to execute

```
void foobar(int a) {
    int x = 4, y = 0;
    if(a != 0) {
        y = a + 3;
    }
    assert(x-y != 0);
}
```

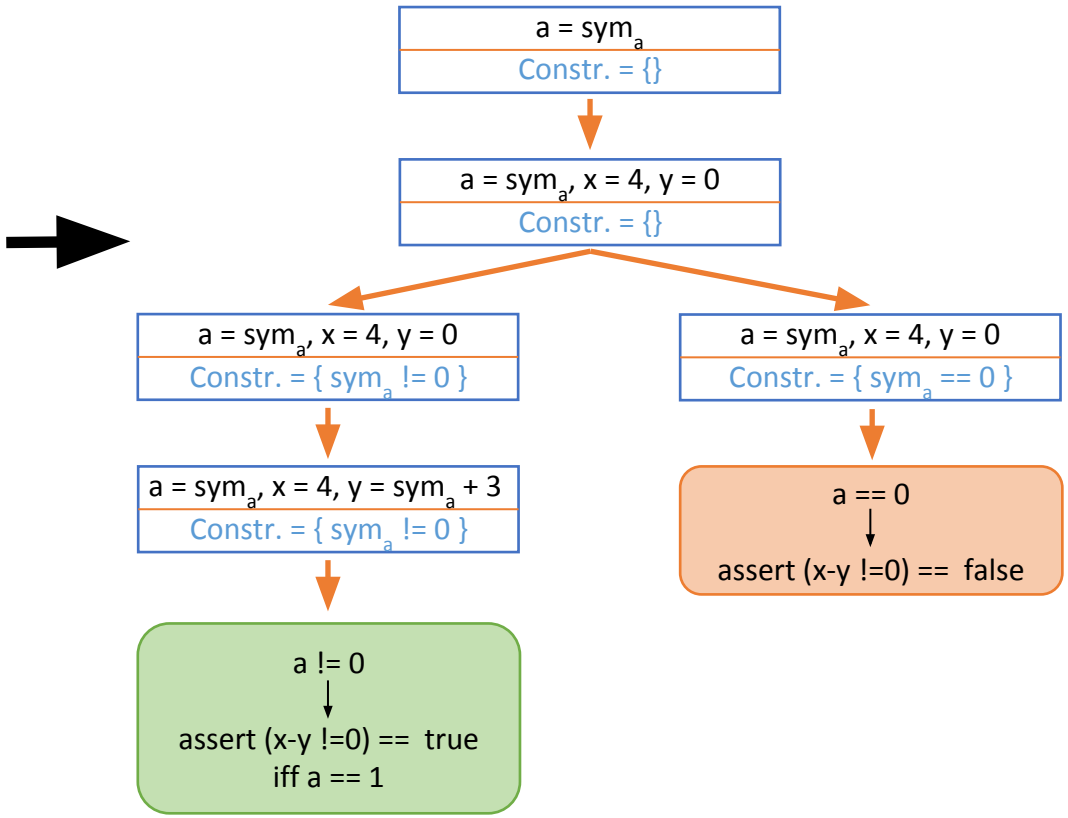


WHAT IS SYMBOLIC EXECUTION

Introduction

A method of analyzing a program to determine what inputs cause each part of the program to execute

```
void foobar(int a) {
    int x = 4, y = 0;
    if(a != 0) {
        y = a + 3;
    }
    assert(x-y != 0);
}
```



ANGR AND SOOT IR

Building blocks

- **Angr**
  - A python framework for analyzing binaries using symbolic execution and static analysis
- **Soot Intermediate representation**
  - Soot is a Java optimization framework, written in Java, which provides four intermediate representations for analyzing and transforming Java bytecode

ANGR AND SOOT IR

Building blocks

PROBLEMS!!!

- **Angr**
  - A python framework for analyzing **BINARIES** using symbolic execution and static analysis
- **Soot Intermediate representation**
  - Soot is a Java optimization framework, **WRITTEN IN JAVA**, which provides four intermediate representations for analyzing and transforming Java bytecode

ROADMAP

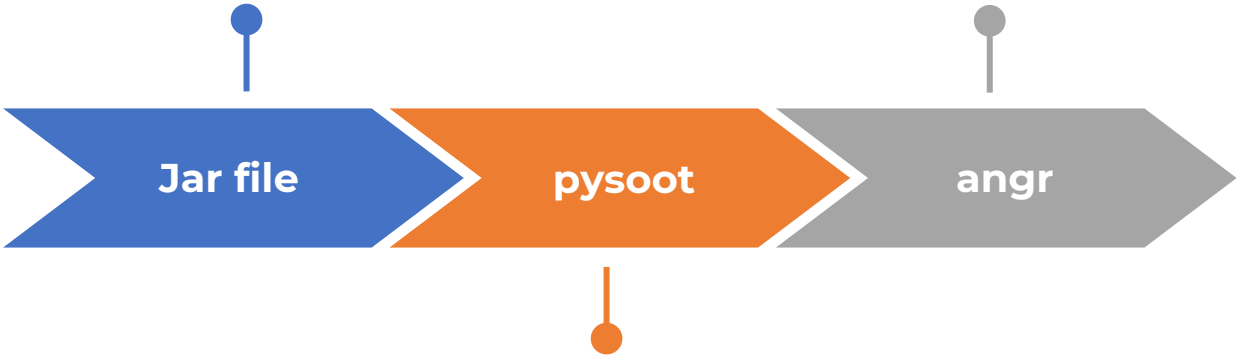
Building blocks

TARGET PROGRAM

Program that has to be analyzed

SYMBOLIC EXECUTION ENGINE

Symbolically execute the lifted bytecode



LIFTER

Lift Java bytecode to soot IR

EXECUTION STATE

Implementation details

At every step of execution, the state of the program is uniquely identified by:

- **STACK**
  - Set of functions' stack frames active at that given point
- **HEAP**
  - Dynamically allocated objects at that given point
- **STATIC TABLE**
  - Static fields of all the classes loaded and used by the program at that given point
- **OTHER STUFF**
  - Current instruction pointer, jump kind, etc.

A SIMPLE EXAMPLE

Implementation details

```
Public static void main(String[] args) {
    String str_1 = “Ciao”;
    int x = 4, y = 0;
    if(str_1.equals(args[1])) {
        y = 4;
    }
    assert(x-y != 0);
}
```

A SIMPLE EXAMPLE

Implementation details

How do we model command line arguments ?

```
Public static void main(String[] args) {
    String str_1 = “Ciao”;
    int x = 4, y = 0;
    if(str_1.equals(args[1])) {
        y = 4;
    }
    assert(x-y != 0);
}
```

How do we represent Strings?

What do we “execute” here?

A SIMPLE EXAMPLE

Implementation details

How do we model command line arguments ?

```
Public static void main(String[] args) {
    String str_1 = “Ciao”;
    int x = 4, y = 0;
    if(str_1.equals(args[1])) {
        y = 4;
    }
    assert(x-y != 0);
}
```

How do we represent Strings?

What do we “execute” here?

JUST MODIFY THE EXECUTION STATE ACCORDINGLY TO THE “SIDE EFFECTS” PRODUCED BY THESE OPERATIONS!!!

GOOD OLD FAUXWARE

Demo

ARRAY WITH SYMBOLIC INDEXES

Problems and limitations

```
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    int index = scanner.nextInt();
    int[] arr = {0,1,2, …, 1000000};
    int ele = arr[index];
}
```

ARRAY WITH SYMBOLIC INDEXES

Problems and limitations

```
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    int index = scanner.nextInt();
    int[] arr = {0,1,2, …, 1000000};
    int ele = arr[index];
}
```

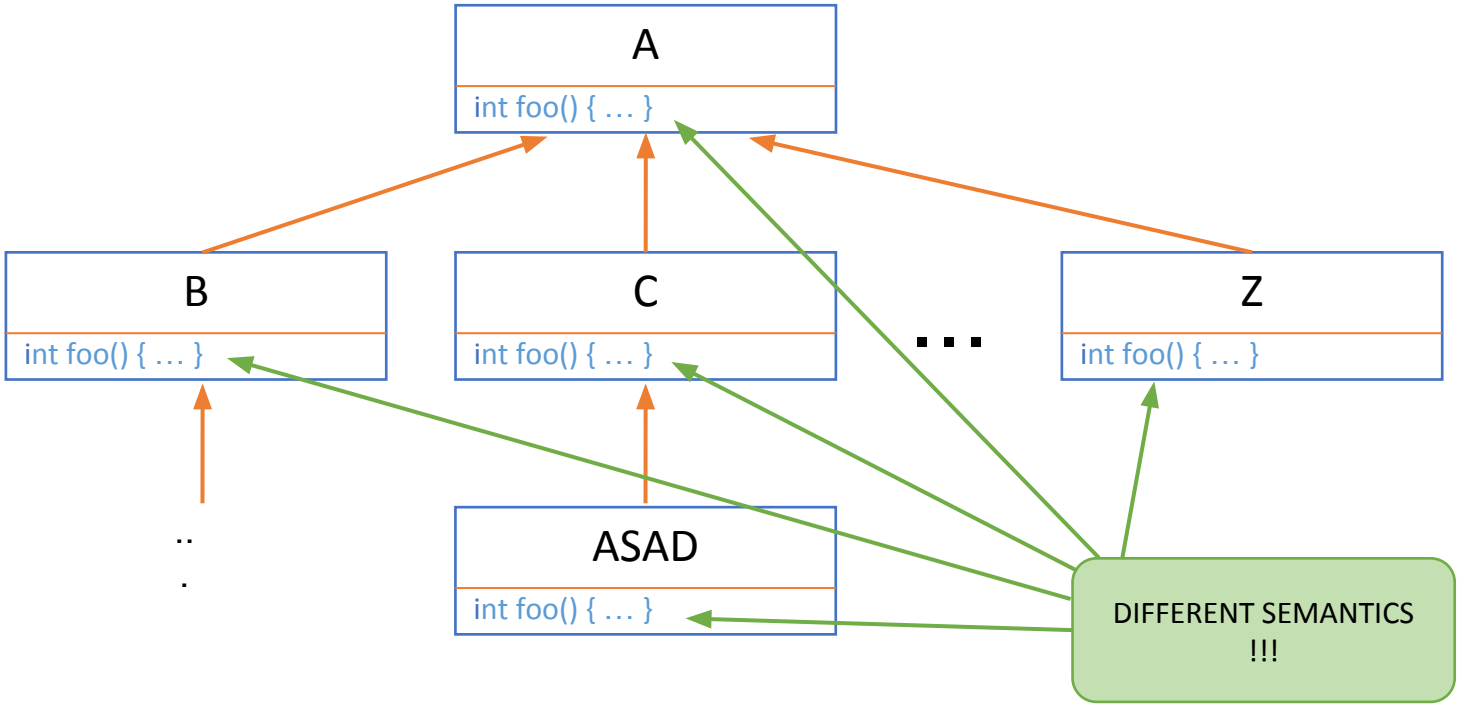
What is the value of index?

Which value should we pick from the array?

INHERITANCE PROBLEM

Problems and limitations

```
public static void foo(A param) {
    int res = param.foo();
}
```

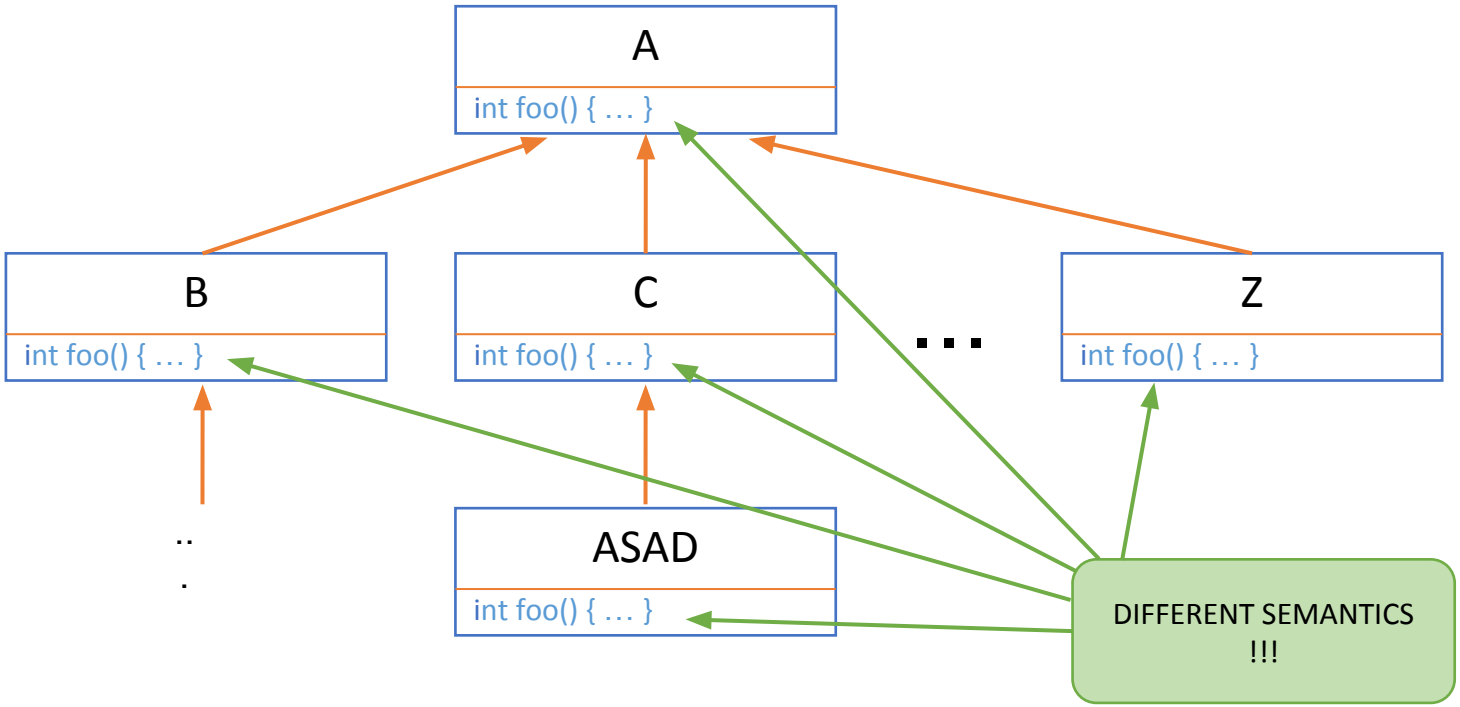


INHERITANCE PROBLEM

Problems and limitations

```
public static void foo(A param) {
    int res = param.foo();
}
```

Which foo() should we invoke?



LET'S NOT TALK ABOUT REFLECTION...

Problems and limitations

IT'S JUST THE BEGINNING

Conclusions

- Still have a lot work that needs to be done...
- ... But a lot of interesting research projects available!

QUESTIONS?

Thank you!