Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Exercise 7

Bytecode Verification

November 8, 2013

Task 1

The method `f` of class `E` has the following signature:

```
void f();
```

and one local variable `v`. The maximal stack size is equal to 1.

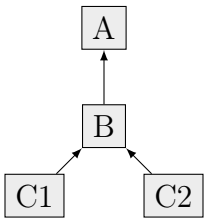
It has the following body:

```
0: iconst_5
1: istore_1
2: aload_0
3: astore_1
4: iload_1
5: iconst_1
6: iadd
7: istore_1
8: return
```

Can the provided byte code be verified? If so then verify it, otherwise explain which line of the code causes the problem and why.

Task 2

Consider the following type hierarchy:



Suppose that the method `f` of class `E` has the following signature:

```
A f(boolean b1, boolean b2);
```

and there are three local variables `x`, `y`, `z`. It is known that the initial state is:

```
([], [E,boolean,boolean,C1,C2,A])
```

The maximal stack size is equal to 1.

The method f has the following body:

```
0: iload_1
1: ifeq  22
4: iload_2
5: ifeq  12
8: aload_3
```

```
 9: goto  14
12: aload_4
14: astore_3
15: aload_5
17: astore_4
19: goto 0
22: aload_3
23: areturn
```

- Verify that the program is type safe.
- Provide the minimal type information that enables verification of the bytecode without a fixpoint computation.

Note: In this example, `ifeq x` pops an integer from the stack and jumps to line `x` if the integer is equal to zero.

Task 3

Consider the following Java code:

```
interface IFace {
   void m();
}
class Cl1 implements IFace {
   public void m() { System.out.println("Cl1.m"); }
}
class Cl2 implements IFace {
   public void m() { System.out.println("Cl2.m"); }
}
public class Test1 {
   public static void main( String[] args ) {
      xxx(true);
      xxx(false);
  }
   public static void xxx( boolean param ) {
      IFace iface = null;
      if( param ) { iface = new Cl1();}
      else { iface = new Cl2(); }
      iface.m();
   }
}
```

- What type will be calculated for the variable `iface` of the method `xxx` during the bytecode verification?
- When can we decide that `iface.m()` is safe to call? During bytecode verification, or execution?
- What if `IFace` was a class instead of an interface? What if it was an abstract class?

Task 4

The bytecode type inference algorithm rejects a verified program if there are different stack sizes for input values of a join point.

- Provide a bytecode program that is rejected because of this limitation but that does not cause runtime errors.
- Is it possible to construct a bytecode verification algorithm that avoids this limitation? If yes, then provide an updated algorithm. If no, then show that it can't be done.

- How serious is this restriction from a pragmatic perspective?