CS 6004 Assignment 4

Manish Kumawat (200050071) Utkarsh Ranjan (200050147)

April 25, 2024

Contents

1 Simple Constant Folding and Propagation

Before starting, here's the code repository. The optimization we implemented is a simple constant folder and propagator which also removes branches which are useless after constant propagation. The algorithm is really simple as the name suggests:

```
queue = all statements in the function
while queue is not empty:
    s = queue.front()
    queue.pop()
    if s is of the form x = Φ(c, c, ...) for some c:
        replace s with x = c
    if s is of the form x = c:
        delete s from the program
        for each use u of x:
            simplify(u)
            queue.add(u)
```

where $simplify$ just replaces x with c in u and further folds the expression. If u is an if statement and the condition simplifies to a constant expression, it also eliminates the useless branch. As you can see the algorithm uses SSA form of the code for which we used Soot's Shimple IR which is the SSA variant of Jimple.

For the following testcase,

```
public class Test {
    public static void main(String[] args) {
        foo();
    }
    public static int foo(){
        int x0 = 0;
        int y0 = 0;
        int iterations = 500000000;
        for (int i = 0; i < iterations; ++i){
            int x = 1;
            int y = 2;
            x = x * y;
            if (x <= 2 || x < 0 && x != 0 && x == 1
                && x >= 4 || x > 2){
                x = 3;
            } else {
                x = 4;
            }
            boolean b = false;
            if (x != 15){
                b = 2 > x;
            } else {
                y = -x / y;
            }
            x0 = x;
            y0 = y;
        }
        return x0 * y0;
    }
}
```

the results we got were,
Without the optimization:

```
real    0m12.627s
user    0m12.325s
sys     0m0.030s
```

With the optimization:

```
real    0m6.843s
user    0m5.676s
sys     0m0.050s
```

That's a speed-up of $2x$. There are not many situations to test this optimization since its focus is pretty narrow.

2 Building and Testing

We've provided the built **openj9** image in the repository. We've not made changes to the interpreter so if the provided image doesn't work, your version of the original **openj9** should work fine.

The easiest way to run the analysis is by using **run_analysis.sh** and **run_with_openj9.sh**. Use **run_analysis.sh** without any arguments to analyse all the files in **testcases** dir. To run a class file which is in dir **sootOutput**, say **test**, just do:

```
./run_with_openj9.sh sootOutput test
```

By default, **run_analysis.sh** will have the analysis turned on. To disable the optimization, just comment out the following line (line 13) in **PA4.java**:

```
"-via-shimple",
```