COMS W4115 Assignment 3

Programming Languages and Translators

Xijiao Li (xl2950)

November 30, 2020

Problem 1

a.

Scope 1: [13, 32]; Scope 2: [15, 29]; Scope 3: [18, 21]; Scope 4: [22, 25]

b.

Table 1: Scope 1 Symbol Table

Name	Kind	Type	Line Number
something	parameter	int	13
a	variable	int	14
b	variable	int	14
c	variable	int	14
x	variable	int	14

Table 2: Scope 2 Symbol Table

Name	Kind	Type	Line Number
something	variable	float	2.5
r	variable	int	2

Table 3: Scope 3 Symbol Table

Name	Kind	Type	Line Number
x	variable	float	19
r	variable	float	19

Table 4: Scope 4 Symbol Table

Name	Kind	Type	Line Number
x	variable	int	23

c.

The `r` in line 26 correspond to the global variable `r`, whose definition is at line 3.

In the 4 tables mentioned in part (b) and the global table, and line 26 is outside of Scope 3 and Scope 4. We first search Scope 2, and the only definition of `r` is after line 26. We then search Scope 1, and there is no definition of `r`. We finally search Global variable table, and found the definition of `r` at line 3.

d.

Table 5: Active identifiers Table (after line 19)

Name	Kind	Type	Line Number
a	variable	int	14
b	variable	int	14
c	variable	int	14
something	variable	float	17
x	variable	float	19
r	variable	float	19

e.

Table 6: Active identifiers Table (after line 26)

Name	Kind	Type	Line Number
r	variable	int	5
a	variable	int	14
b	variable	int	14
c	variable	int	14
x	variable	int	14
something	variable	float	17

f.

-3. With the codes being statically scoped, a variable always refers to its top level environment. Thus, `pltIsAwesome(5)` returns -3, and `foo(-3)` returns -3.

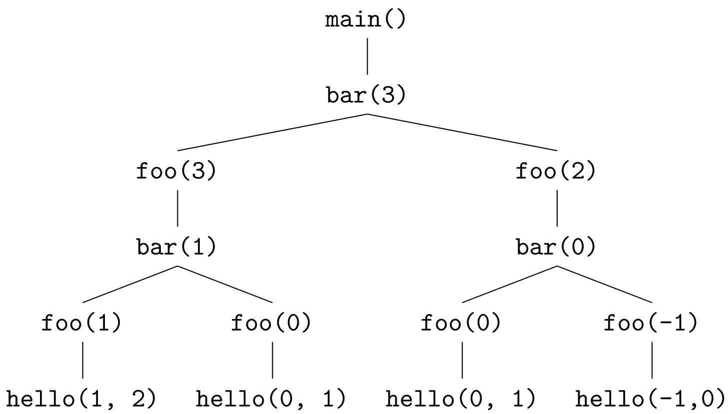
g.

7. With the codes being dynamically scoped, we first look for a local definition of a variable. If it isn't found, we look up the calling stack for a definition. Thus, `pltIsAwesome(5)` returns 2, and `foo(4)` returns 7.

Problem 2

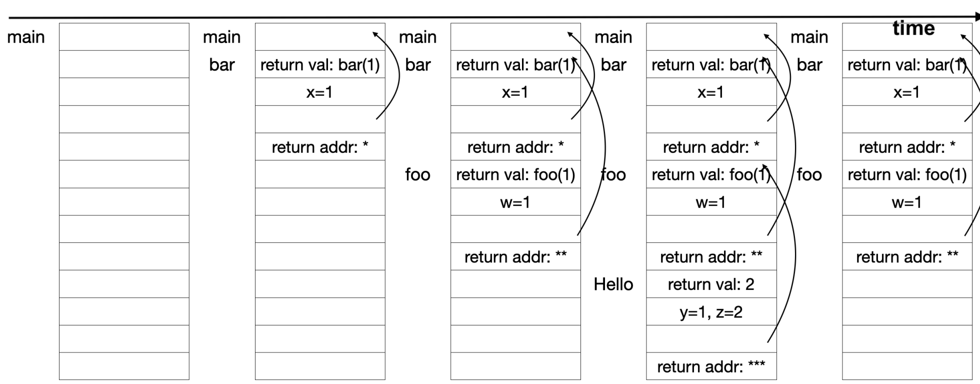
a.

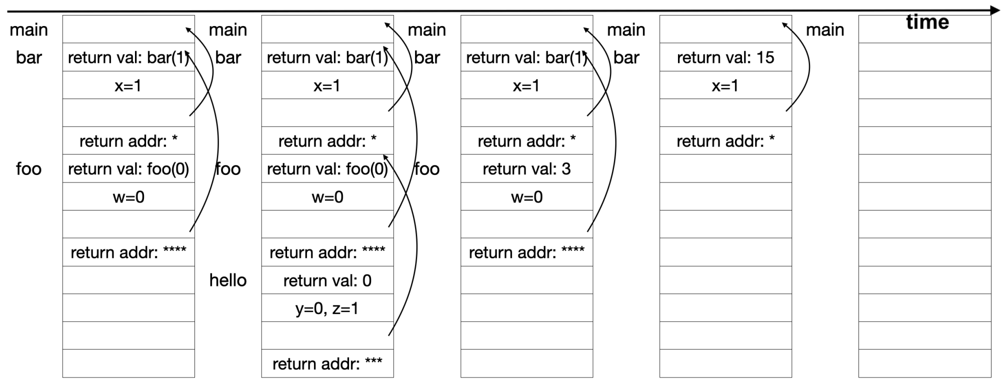
Tree:



b.

The stacks (the direction of expansion is from up to down in the image):





Given the pointer of addresses:

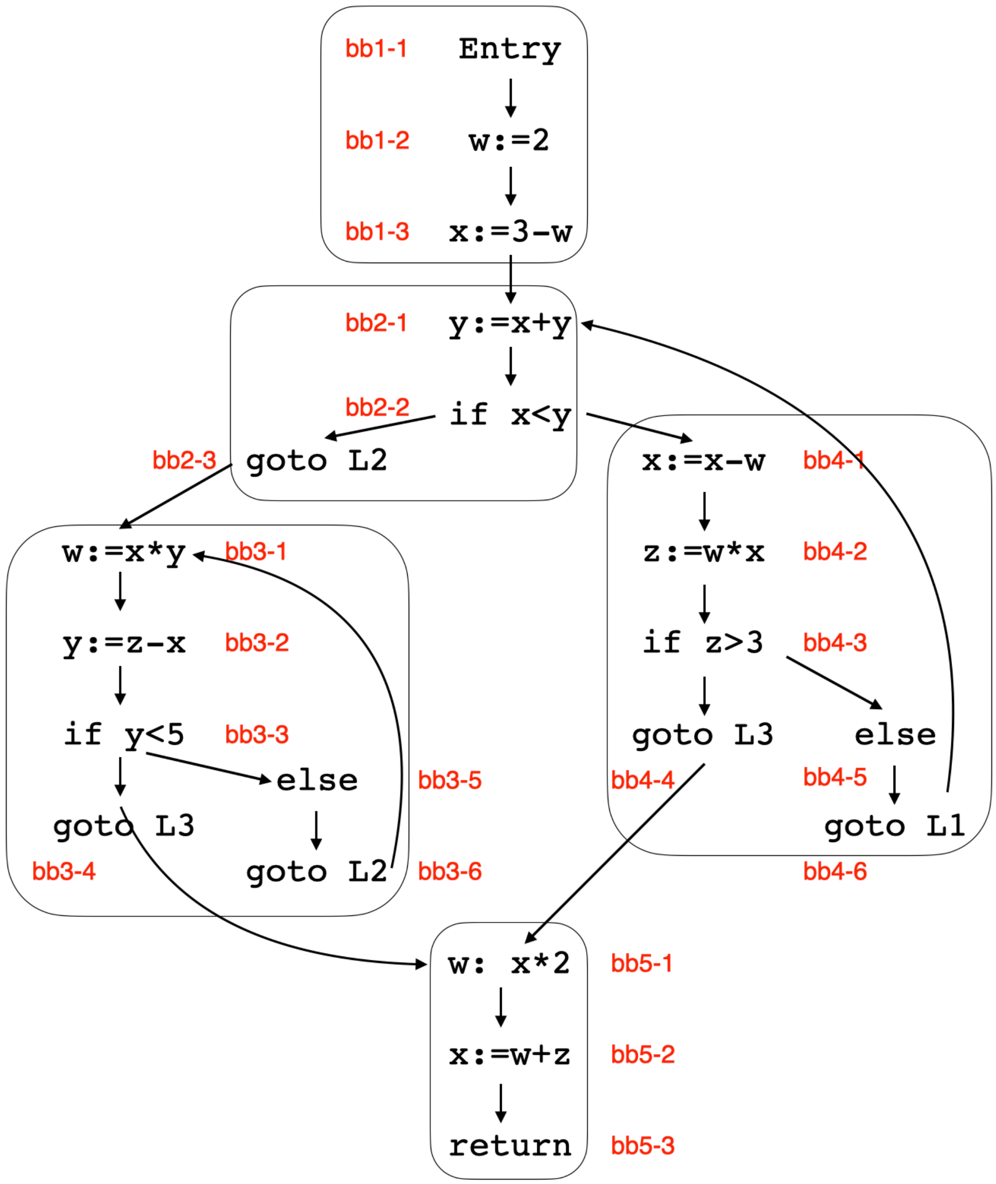
```
Class PLT {

        int foo(int w) {
            if(w < 2)
                                ***
                return hello(w, w + 1) + 3;
            return bar(w - 2);
        }

        int bar(int x)**{         ****
            return foo(x) * foo(x - 1);
        }

        int hello(int y, int z) {
            return y * z;
        }

        int main()*{
            bar(val);
            return 0;
        }
}
```

Problem 3

a.

{1,2,3}, {4,5}, {6,7,8,9}, {10,11,12,13}, {14,15,16}

b.



c.

$bb1-1:\{bb1-1\}$

$bb1-2:\{bb1-2,bb1-1\}$

$bb1-3:\{bb1-3,bb1-1,bb1-2\}$

$bb2-1:\{bb2-1,bb1-1,bb1-2,bb1-3\}$

$bb2-2:\{bb2-2,bb1-1,bb1-2,bb1-3,bb2-1\}$

$bb2-3:\{bb2-3,bb1-1,bb1-2,bb1-3,bb2-1,bb2-2\}$

$bb3-1:\{bb3-1,bb1-1,bb1-2,bb1-3,bb2-1,bb2-2,bb2-3\}$

$bb3-2:\{bb3-2,bb1-1,bb1-2,bb1-3,bb2-1,bb2-2,bb2-3,bb3-1\}$

$bb3-3:\{bb3-3,bb1-1,bb1-2,bb1-3,bb2-1,bb2-2,bb2-3,bb3-1,bb3-2\}$

$bb3-4:\{bb3-4,bb1-1,bb1-2,bb1-3,bb2-1,bb2-2,bb2-3,bb3-1,bb3-2,bb3-3\}$

$bb3-5:\{bb3-5,bb1-1,bb1-2,bb1-3,bb2-1,bb2-2,bb2-3,bb3-1,bb3-2,bb3-3\}$

$bb3-6:\{bb3-6,bb1-1,bb1-2,bb1-3,bb2-1,bb2-2,bb2-3,bb3-1,bb3-2,bb3-3,bb3-5\}$

$bb4-1:\{bb4-1,bb1-1,bb1-2,bb1-3,bb2-1,bb2-2\}$

$bb4-2:\{bb4-2,bb1-1,bb1-2,bb1-3,bb2-1,bb2-2,bb4-1\}$

$bb4-3:\{bb4-3,bb1-1,bb1-2,bb1-3,bb2-1,bb2-2,bb4-1,bb4-2\}$

$bb4-4:\{bb4-4,bb1-1,bb1-2,bb1-3,bb2-1,bb2-2,bb4-1,bb4-2,bb4-3\}$

$bb4-5:\{bb4-5,bb1-1,bb1-2,bb1-3,bb2-1,bb2-2,bb4-1,bb4-2,bb4-3,bb4-4\}$

$bb4-6:\{bb4-6,bb1-1,bb1-2,bb1-3,bb2-1,bb2-2,bb4-1,bb4-2,bb4-3\}$

$bb5-1:\{bb5-1,bb1-1,bb1-2,bb1-3,bb2-1,bb2-2\}$

$bb5-2:\{bb5-2,bb1-1,bb1-2,bb1-3,bb2-1,bb2-2,bb5-1\}$

$bb5-3:\{bb5-3,bb1-1,bb1-2,bb1-3,bb2-1,bb2-2,bb5-1,bb5-2\}$

d.

$bb3-21:\{bb5-1\}$

e.

$bb3-6\rightarrow bb3-1$
$bb4-6\rightarrow bb2-1$

f.

Yes, the CFG is reducible. All the back edges are those whose targets dominate sources. All the forward edges form an acyclic graph, and can reach every node.

Problem 4

a.

Table 7: Stack Machine for Problem 4

Step	*Accumulator*	*Stack*
acc $\leftarrow$ 12	12	<init>
push	12	12, <init>
acc $\leftarrow$ 3	3	12, <init>
acc $\leftarrow$ top $-$ acc	9	12, <init>
pop	9	<init>
push	9	9, <init>
acc $\leftarrow$ 8	8	9, <init>
push	8	8, <init>
acc $\leftarrow$ 2	2	8, 9, <init>
acc $\leftarrow$ top $+$ acc	10	8, 9, <init>
pop	10	9, <init>
acc $\leftarrow$ top $*$ acc	90	9, <init>
pop	90	<init>