Static v. Dynamic Scoping

Eddy Varela
Taylor Beebe

November 2018

1 Code Snippet

```
1 let result() =
2 let x = 2
3 let f = fun y -> x + y
4 let x = 7
5 x +
6 f x
```

1.1 Static Scoping

The value of x + f x will be 7+(2+7) = 16

Line 3: x =2
Line 5: x=7
Line 6: x = 7

In a statically scoped languages, whenever a variable is referenced we look for the last time that variable was referenced in the program (look for an earlier time in the source code). For example in line 3 we reference x+y so this is in reference to x = 2 in line 2. However, in line 5 we reference the assignment to line 4 where x = 7.

1.2 Dynamic Scoping

The value of x + f x would be 7 + (7+7) = 21

Line3 : x = 7 Because we need to trace when f x was called. When we define f this does not get put on the stack. Thus we look at when f x get called and find the last time x was defined on the stack.

Line5 : x = 7 If we follow the trace/look on the call stack, we can see that on the line above, x is now 7.

Line6 : x = 7 Finally for the last occurrence of x, we look to the last place where x was defined which was line 4

1.3 F#

Static scoping because F# uses variable names to hold values. Thus if you define a function and call it f, you can redefine f inside of f and F# will consider the inner f as a new function.