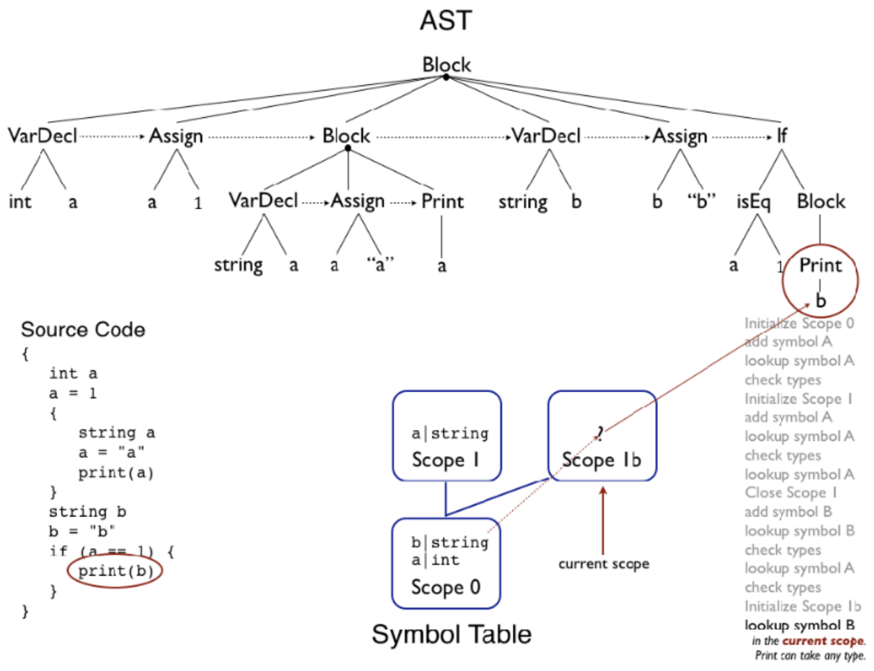
Lab 7 – Semantic Analysis

Ryan Munger
Ryan.Munger1@marist.edu

March 21, 2025

0.1 Scope and Type Checking

Explain in detail what is happening in the diagram below.



```
{
   int a
   a = 1
   {
      string a
      a = "a"
      print(a)
   }
   string b
   b = "b"
   if (a == 1) {
      print(b)
   }
}
```

```
Initialize Scope 0
add symbol A
lookup symbol A
check types
Initialize Scope 1
add symbol A
lookup symbol A
check types
lookup symbol A
Close Scope 1
add symbol B
lookup symbol B
check types
lookup symbol A
check types
Initialize Scope 1b
lookup symbol B
  in the current scope.
  Print can take any type.
```

Figure 1: AST Diagram

The diagram depicts an Abstract Syntax Tree (AST) and some evaluations of it. As we can see, we examined the source code to produce the Symbol Table and build the tree. We start with an open block. This is the root of our tree and initializes Symbol Table 0. As we go through the code, we detect statements and add them accordingly. For example, the first thing we see after the open block is *int a.* We then add a *VarDecl* to our AST and the symbol *a* as type *int* in our current symbol table (after ensuring *a* is not already in use!). This process continues very similarly to our parser. When *a = 1* is reached, we add an *Assign* to our AST. Since we are not declaring a new symbol, we must verify that we are using the existing entry correctly. We look up *a* and find that it is of type *int.* Since we are assigning a value of the correct type to *a*, we may continue. Skipping ahead, we can see that *Print b* is of special interest in this diagram. Since *if* statements have their own symbol table (they possess scope), we are in scope 1b. We now look up the symbol *b*, which is to be printed, in the most local scope. Since *b* is not declared in scope 1b, we must move up. Scope 1b's parent scope, scope 0, is checked next. Scope 0 has symbol *b* defined as a string. Since *print* may take any type and symbol *b* is defined, we are good to go! Happy semantic analysis!