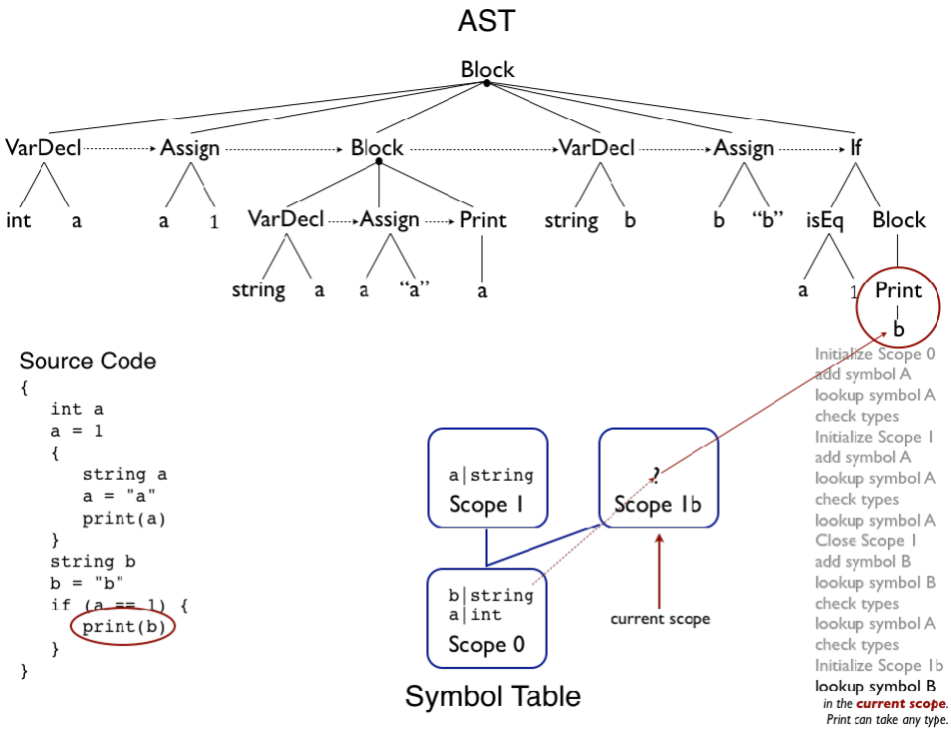
Gianna Julio

CMPT432 - Design of Compilers
Lab 7

1. **Describe in detail what is happening in the diagram below.**



This diagram depicts a depth-first, in-order traversal of an abstract syntax tree. During this traversal, a symbol table (a tree of hash tables) is built, scope is checked, and variable type is checked all in a single pass. The following is a detailed description of a traversal of this AST:

Block: The current scope pointer is set to 0.

- ***VarDecl:*** A new symbol *a* of type int is added to the symbol table with the current scope of 0.
- ***Assign:*** The symbol *a* is looked up in the symbol table with a current scope of 0 and found. A type check occurs, verifying that the left child *a* of type int and right child of the value 1 are type ***compatible*** for assignment, which is true.
- ***Block:*** A new scope is initialized as scope 1 and the current scope pointer is moved to this child.
  - ***VarDecl:*** A new symbol *a* of type string is added to the symbol table with the current scope of 1.
  - ***Assign:*** The symbol *a* is looked up in the symbol table with the current scope 1 and found. Another type check occurs, verifying that the left child *a* of type string and right child "a" are type ***compatible*** for assignment, which is true.

  - ***Print:*** Print statements can take any type in this grammar, so there is no type check, but the scope was still checked. The symbol a is looked up in the symbol table with the current scope of 1 and found.

- ***Block (close):*** Scope 1 is closed and the current scope pointer is moved to its parent of scope 0.
- ***VarDecl:*** A new symbol b of type string is added to the symbol table with the current scope of 0.
- ***Assign:*** The symbol b is looked up in the symbol table with a current scope of 0 and found. A type check occurs, verifying that the left child b of type string and right child "b" are type ***compatible*** for assignment, which is true.
- ***If:***
  - ***isEq:*** The symbol a is looked up in the symbol table with a current scope of 0 and found. A type check occurs, verifying that the left child a of type int and right child of the value 1 are type ***comparable*** for assignment, which is true.
  - ***Block:*** A new scope is initialized as scope 1b and the current scope pointer is moved to this child.
    - ***Print:*** Print statements can take any type in this grammar, so there is no type check, but the scope was still checked. The symbol b is looked up in the symbol table with a current scope of 1b and ***not*** found. The symbol b is looked up in the symbol table with the parent scope of 1 and found.
  - ***Block (close):*** Scope 1b is closed.

Block (close): Scope 0 is closed.

Within this AST traversal, errors like type mismatches, undeclared identifiers, and redeclared identifiers in the same scope are detected. In addition, warnings like, variables declared but unused, variables used without being initialized, and variables declared and initialized but never used are also detected. Collision checking between variables added to the symbol table also occurs during this traversal