UNIVERSITY OF AMSTERDAM

Informatics Institute
Systems and Networking Lab
Parallel Computing Systems Group

Compiler Construction
2019/20
Dr. Clemens Grelck
February 25, 2020

Assignment 4

Semantic Analysis

Assignment 4.1: Scoping and symbol tables

Consider the following CiviC nested function definition:

```
int d = 2;
int foo( int a)
{
  int b = 1;
  int c;
  int f( int x) {
    int b = x + b;
    return b;
  }
  int g( int x) {
    c = f( x - b);
    return c + a;
  }
  c = c + g( d - b);
  return c;
}
```

a) What is the value of `foo(8)`, and, more importantly, why? Describe the relevant execution steps of running above code. Do not feed the code into the reference compiler!

b) Mark every occurrence of a variable identifier in statement position by an arrow to the corresponding declaration, i.e. global variable declaration, function parameter or local variable declaration.

c) Annotate each scope (level) with its symbol table.

d) Annotate each variable identifier in statement position by a number indicating the relative scope distance to the corresponding declaration, starting with zero for local variables and parameters.

Assignment 4.2: Lambda lifting

Manually apply the lambda lifting transformation to the code example of Assignment 4.1.

Assignment 4.3: Function overloading

Assume we would extend CiviC by function overloading.

Function overloading supports the presence of multiple functions within the same scope bearing the same name, as long as they are distinguished by different arity (number of parameters) or by different parameter types. Function calls are dispatched to the matching function definition according to the argument types inferred.

Describe how this extension would affect semantic analysis in the CiviC compiler in general and how you would solve the corresponding problems in detail.

Assignment due date: Friday, March 6, 2020