What is Algol-style name scoping?

Analyzing Aliases of Reference Formal Parameters

The analysis used in this paper is formulated as a data flow analysis problem, allowing the compiler write to use any of the standard techniques from global data flow analysis to solve the problem.

When a program has multiple names for the same storage location, these names are said to be aliases. These aliases can arise in multiple ways, such as call-by-reference binding and pointers.

Correctly identifying aliases can allow for compiler optimizations.

For example this piece of code: a = 10; b = 12; c = a \* b. If we know a and b point to different locations, we can assign 10 and 12 to different registers and use those in the computation for c. This is however not always correct if a and b could be aliases. If a and b are aliases, but the previous optimization of storing a and b in different registers is used, c would incorrectly be assigned 120.

The goal of alias analysis is to annotate each procedure n in the program with set Alias(n) containing all the aliases which can hold an entry to n. Each potential alias is represented by an alias pair (x,y) and the presence of such a pair indicates that x and y can point to the same storage location when n is invoked, but it does not mean that those two are definitely the same.

In this paper, the only ways of introducing aliases are as follows: a single parameter is passed in more than one parameter position, or a global variable is passed as a parameter. This creates an alias between the global variable and the parameter within the function body. These are the only ways to introduce aliases through procedure calls.

The introduction of aliases only uses computations local in nature, whereas the propagation of those aliases through the program uses machinery of data flow analysis to account for interactions between procedures.