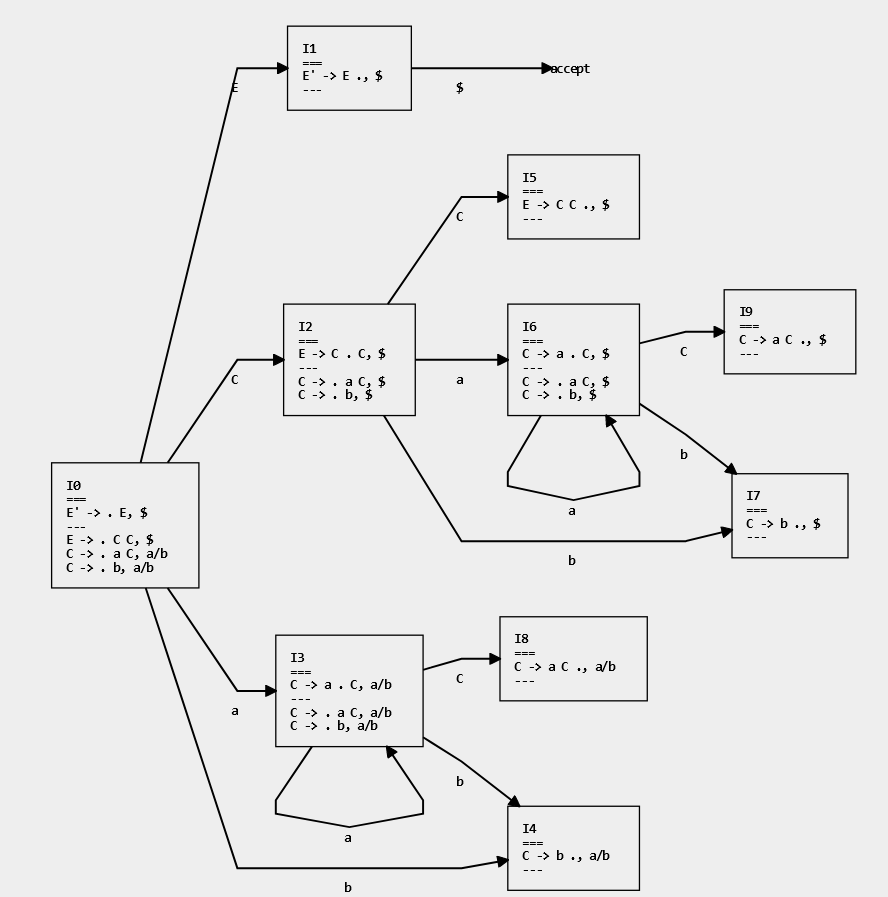
Compilers 2023

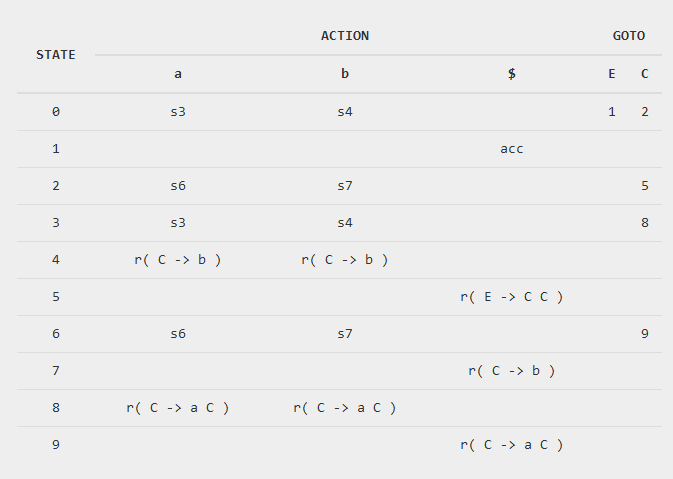
A picture containing human face, forehead, person, person

Description automatically generated

1a)



1bi)



1bii)

I3 & I6

I4 & I7

I8 & I9

1c)

1. Is “int” in scope and defined as a type?
2. Can you declare a 2D array of “int”s?
3. Are 5 and 8 valid integer values?
4. Have identifiers “m1” and “m2” already been used in this scope?

1d)

1. Are “m1” and “m2” in scope?
2. Can you assign an element of the type of “m2” to that of “m1”?
3. Are the ranges “a..b” and “c..d” the same sizes as ranges “e..f” and “g..h” respectively?
4. Are "m1” and “m2” the same matrix type? E.g., is “m1” a 2D “int” array, but “m2” is a 2D “string” array?

1e)

for (int m1i = a, m2i = e; m1i <= b; m1i++, m2i++) {

for (int m1j = c, m2j = g; m1j <= d; m1j++, m2j++) {

m1[m1i][m1j] = m2[m2i][m2j];

}

}

2a) The compiler should allocate to a caller saved register if the variable is important to the caller, and there is a chance that it may be clobbered by the callee (?)

2b)

*Induction variables:*

*• A variable which is incremented by a loop-invariant amount*

*• A variable which is a multiple of an induction variable*

**n** is not an induction variable as its value is unchanged (could you argue it \*is\* an induction variable as it is technically increased by 0 each time?)

**s** is not an induction variable as it is not incremented by a loop-invariant amount (the value of x on each iteration determines the value of s)

**r** \*is\* an induction variable \*if s is\*, otherwise r is not an induction variable

**x** is an induction variable as its value depends only on another induction variable (i)

**i** is an induction variable as it is incremented by 1 on each iteration

2c) There is only 1 natural loop. Both end S12 and S13 (the two paths in the if statement) are dominated by S14, meaning the single natural loop has header S7 and final node S14

2d) Relevant reaching definitions for S9:

S3 (initial definition of s for first iteration)  
S4 (initial definition of r for first iteration)  
S12 (redefines s)  
S13 (redefines s)

2e) *An instruction is loop-invariant if its operands (inputs) only arrive from outside the loop (moving it out of the loop does not change the semantics).*

S12 and S13 are candidates for loop-invariant code motion as they only depend on values defined outside of the loop (n, defined in S2).

2f) Header = S7, Pre-Header = S6

1. The value that s takes in any given iteration is dependent on the value of x, whose value is changed during each iteration
2. I forgor what I put as second reason