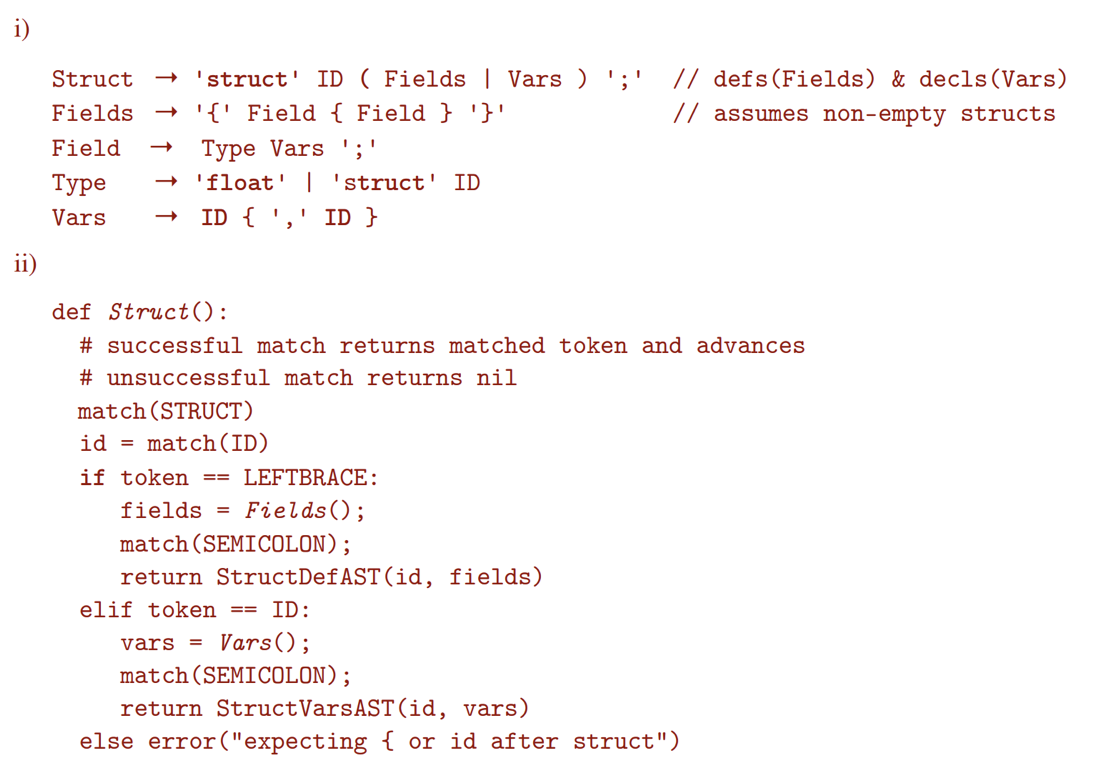
9

Struct -> 'struct' ID ('{' Fields '}' | Vars) ';'

Fields -> Field {Field}

Field -> Type Vars ';'

Type -> 'struct' ID | 'float'

Vars -> ID {',' ID}

def parseStruct() -> ASTNode:

match(STRUCT)

structName = token.get\_id()

match(ID)

if token == ID:

vars = parseVars()

match(SEMI\_COMMA)

return DeclareStructAST(structName, vars)

else:

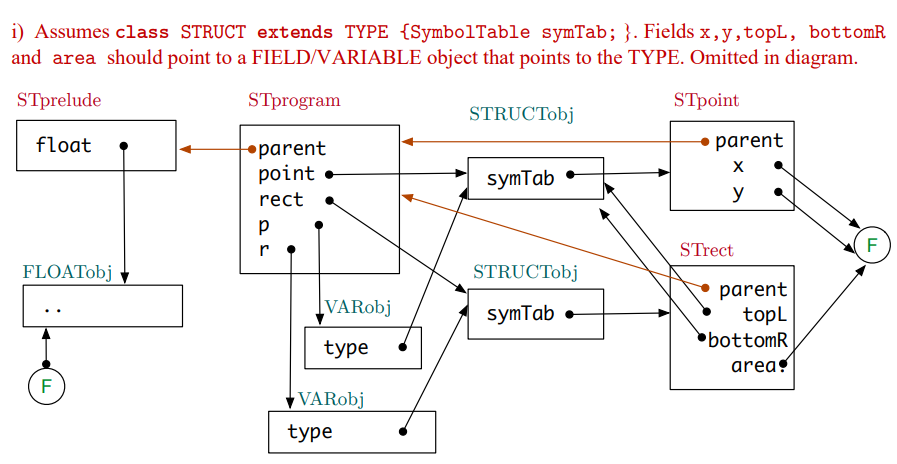
match(OPEN\_BRACES)

fields = parseFields()

match(CLOSE\_BRACES)

match(SEMI\_COMMA)

return NewTypeAST(struct\_name, fields)

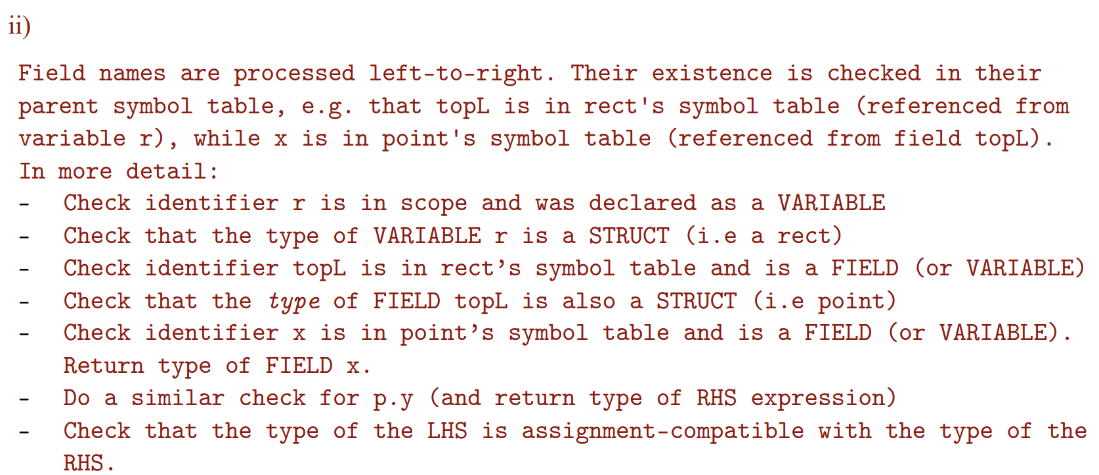


The semantic checker first look at the type of this statement, which is an assignment.

`r` is refered in LHS, it will check if it is in local scope (as a variable). Then it will check if it has attribute `.topL`. `r` has type `struct rect`, thus it has attribute `topL`. Expression `r.topL` has type `struct point`, then it has attribute `x`. `r.topL.x` has type float.

`p` is in local scope as a (variable). And it has type `struct point`. `p.y` has type float.

LHS and RHS has the same type.



Stack frame for get

Get.y

Get.x

r.area

r.bottomR.y

r.bottomR.x

r.topL.y

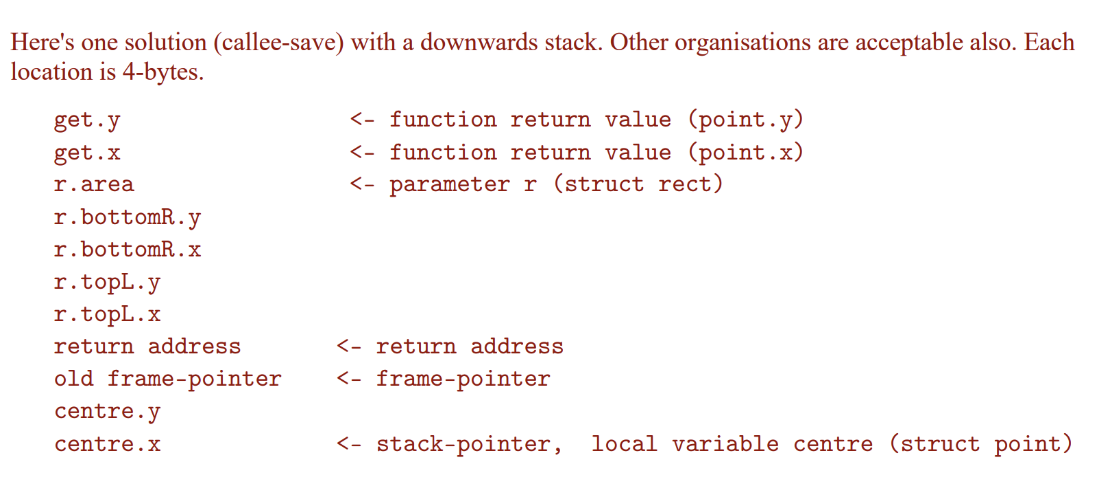
r.topL.x

Return address

Old frame-pointer (frame pointer)

Centre.y

Centre.x (stack pointer)



2. (I) only S3

(ii) S0, S1, S3, S4

(iii) there isn’t a natural loop inside the code

To see this, we just need to check if there is a header of a loop

For S0, S1, S2, S5 there is not instruction jump back to them

For S3, S5 can jump to S3, however S3 doesn’t dominate S5

For S4, S2 can jump to S4, but S4 doesn’t dominate S2.

b.

transCond :: Cond -> regs -> String -> [Intrs]

transCond (LessThan e1 e2) (r1:r2:rs) label =

transExp e1 (r1:r2:rs) ++

transExp e2 (r2:rs) ++

[Cmp (Reg r1) (Reg r2),

Blt label]

transDoWhile :: Stat -> regs -> [Intrs]

transDoWhile (DoWhile stats cond) (r1:r2:rs) =

[Define label] ++

concatMap (flip transStat (r1:r2:rs)) st statement. When “continue” is executed in a loop, control immediately ats ++

transCond cond (r1:r2:rs) label

where

label = get\_unique\_label()

transContinue :: Stat -> regs -> [Intrs]

transContinue Continue \_ = [Bra label]

where

label = get\_current\_end\_loop\_label()

transDoWhile' :: Stat -> regs -> [Intrs]

transDoWhile' (DoWhile stats cond) (r1:r2:rs) =

[Define label] ++

concatMap (flip transStat (r1:r2:rs)) stats ++

[Define curren\_end\_loop\_label]

transCond cond (r1:r2:rs) label :

(misplaced label? Should be before transCond?)

where

curren\_end\_loop\_label = create\_current\_end\_loop\_label()

-- there is an external state telling you what is the most inner end loop label when we have a nested loop

label = get\_unique\_label()

transStat :: Stat -> [Register] -> Maybe String -> [Instr]

transStat (DoWhile stats cond) (r0:rs) \_

= [Define beginLabel] ++

concatMap (\s -> transStat s (r0:rs) beginLabel) stats ++

transCond cond (r0:rs) beginLabel

where beginLabel is fresh label

transStat Continue (r0:rs) (Just label)

= [Bra label]

transStat Continue (r0:rs) Nothing

= error "can not use continue outside loop"

**Alternate Solution:-** (doesn’t syntax check for continue outside doWhile - don’t think that’s the point of the question or the responsibility of the below functions. The TransDowhile TransContinue TransAssign basically also return the leftover registers.)

transCond (LessThan e1 e2) label regs

= transExp e1 regs ++ transExp e2 (tail regs) ++ [Cmp (Reg r1) (Reg r2),Blt label]

where r1 : r2 : \_ = regs

transDowhile (DoWhile stats cond) regs

= (regs', Define l1: IR ++ transCond cond l1 regs')

where

(regs', IR) = transStats stats l1 regs

l1 = (getNewLabel)

transContinue label regs = regs, BRA label

transAssign (Assign str e) reg@(r1:rs)

= (rs, (transExp e reg ++ [Mov r1 (Abs str)]))

transStat stat l1 regs

case stat of

a@(Assign \_ \_) -> transAssign a regs

Continue -> transContinue l1 regs

d@(DoWhile \_ \_) -> transDoWhile d regs

transStats [] \_ regs

= (regs, [])

transStats (stat:stats) label regs

= (rs1, irs ++ irs1)

where

rs, irs = transStat stat label regs

rs1, irs1 = transStats stats label rs