1.a)

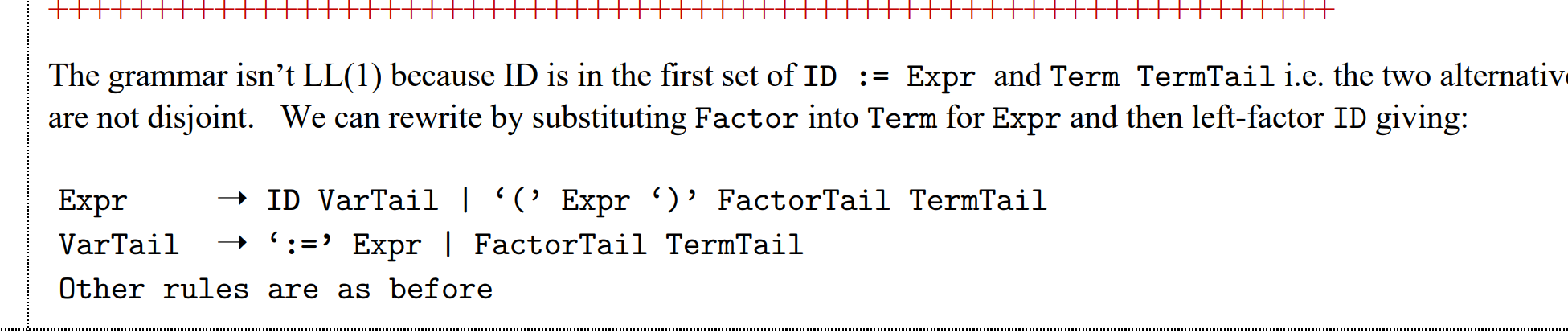
None.

LL(1) cannot since both First(S) & First(T) contain ‘z’

LR(1) cannot since the parse trees are ambiguous. E.g. ‘z’ could have a parse tree of G -> S -> z or G -> T -> z

Therefore also SLR(1) cannot.

b)



c)

Statically:

Global variables

Classes

Stack:

Local variables

Parameters/Formals

Heap:

Objects

Dynamic arrays

d)

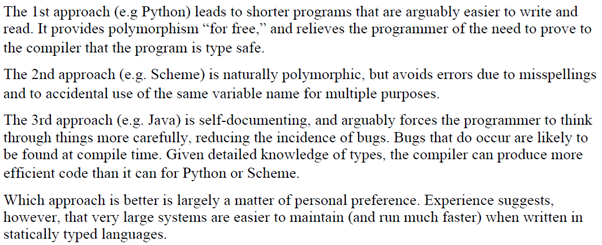
Dynamic binding is a requirement for method overriding. The binding of objects to overridden methods happens at runtime.

Example of dynamic binding:

|  |
| --- |
| **public** **class** **DynamicBindingTest** {  **public** **static** **void** **main**(String args[]) {  Vehicle vehicle = new Car(); //here Type is vehicle but object will be Car  vehicle.start(); //Car's start called because start() is overridden method  }  }  **class** **Vehicle** {  **public** **void** **start**() {  System.out.println("Inside start method of Vehicle");  }  }  **class** **Car** **extends** **Vehicle** {  @Override  **public** **void** **start**() {  System.out.println("Inside start method of Car");  }  } |

e)

(From tutorial 5 on Naranker’s side question 7)



2a (from 2016 paper)

transExp :: Exp -> [Register] -> [Instr]

transExp (Plus e1 e2) (r1:r2:rs) =

transExp e1 (r1:rs) ++

transExp e2 (r2:rs) ++

[Add (Reg r2) (Reg r1)]

transExp (Num n) (r1 : rs) =

[Mov (ImmNum n) (Reg r1)]

transExp (Ref (Var id)) (r1:rs) =

[Mov (Abs id) (Reg r1)]

transExp (Ref (Array id e)) (r1:r2:rs) =

transExp e (r1:rs) ++

[

Mul (ImmNum 4) (Reg r1)

Mov (ImmName id) (Reg r2)

Add (Reg r1) (Reg r2)

Mov (Ind r2) (Reg r1)

]

b) (from 2016 paper)

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

transStat (Assign (Var id) e) (r1:rs) =

transExp e (r1:rs) ++

[Mov (Reg r1) (ImmName id)]

transStat (Assign (Array id arrIdxExp) rhsExp) (r1: r2 : rs) =

transExp rhsExp (r1: r2 : rs) ++ # RHS expression => R1

transExp arrIdxExp (r2 : rs) ++ # Array index => R2

[

Mul (ImmNum 4) (Reg r2), # Offset = 4 \* arrIdx => R2

Add (ImmName id) (Reg r2), # indexed addr = base addr + offset => R2

Mov (Reg r1) (Ind r2) # Ind R2 => Reg R1

]

c) The range is usually fixed for a for loop and so we just need to check if the range of the counter variable in the for loop is within the bounds of the array, so check it once before entering the loop.

d)

e)

It is live because its current value is used on the next increment (where it is killed).

It would still need to be allocated to a register as although it’s not formally “used” it will be used when it comes to the number of iterations needed to complete the for loop