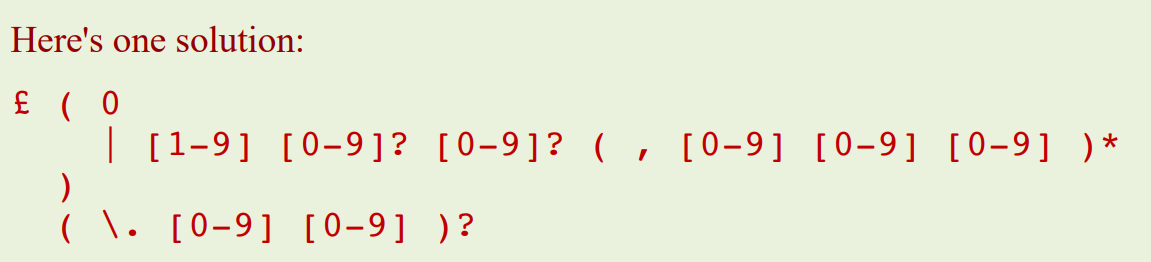
Tom’s version is out

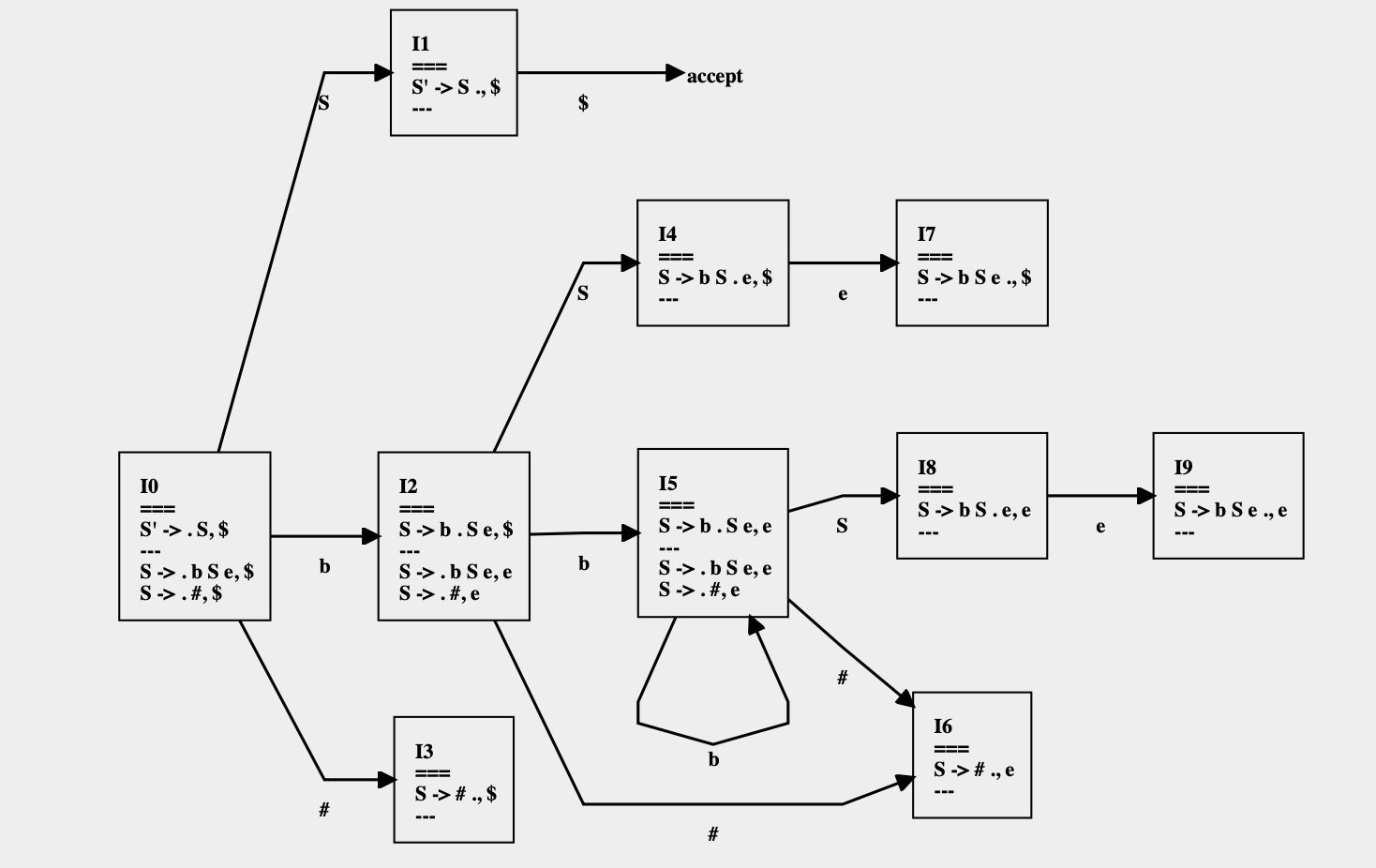
1a) £ -- sign

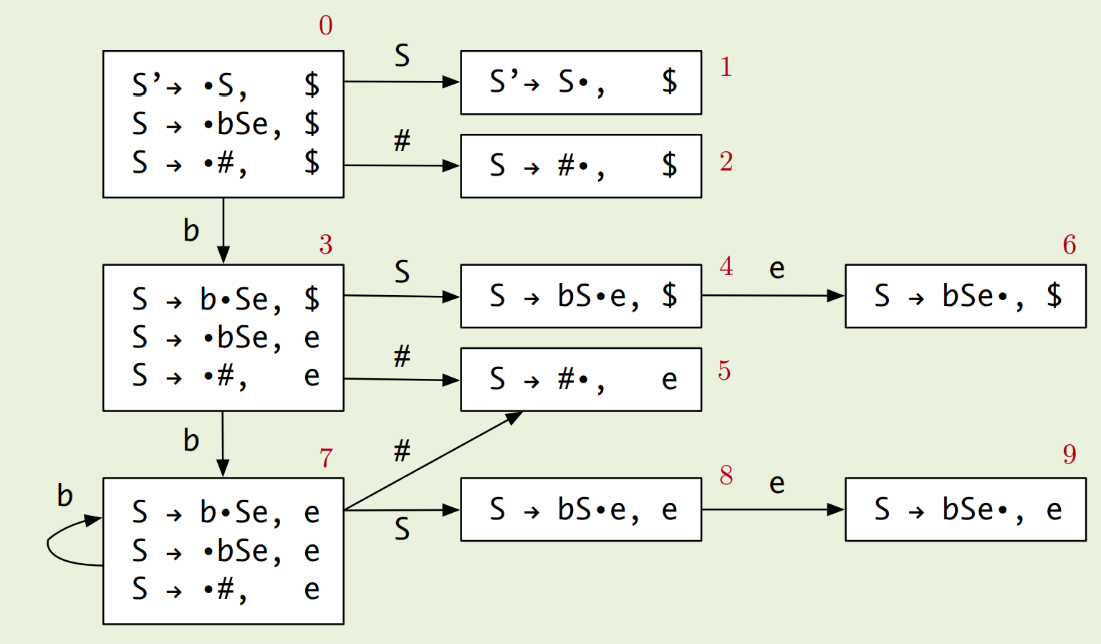
( ( [1-9][0-9] ?[0-9]? (,[0-9][0-9][0-9])\* ) | 0 ) -- Whole part

(\.[0-9][0-9])? -- Fractional part



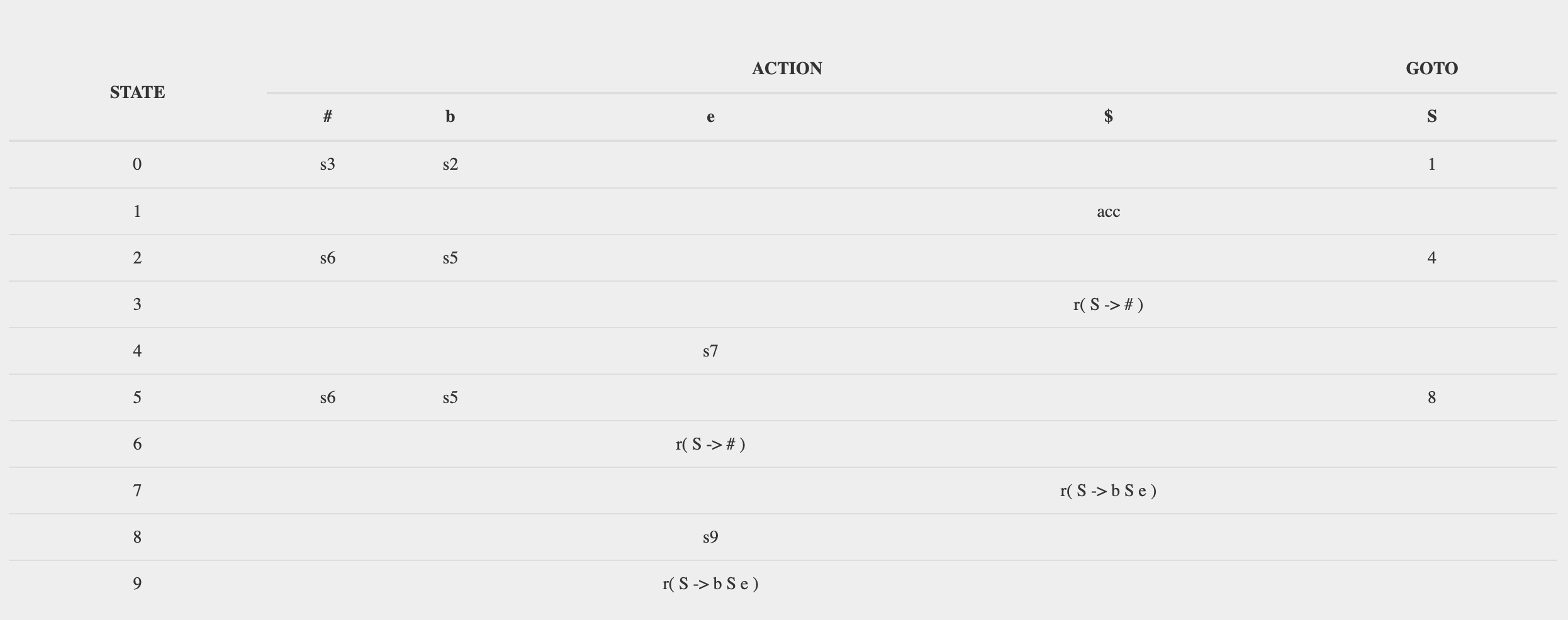
1b)



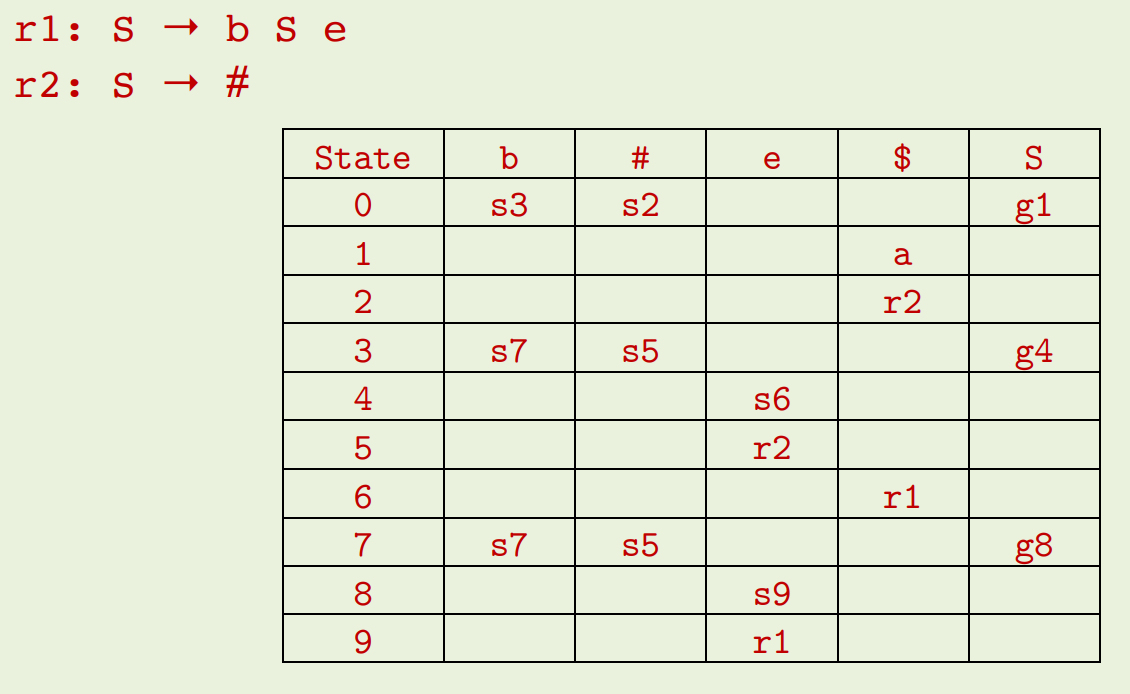


1c) In the diagram above, combine (2,5), (4,8), (7,9), (3,6)

1d)



<https://cyberzhg.github.io/toolbox/lr1?grammar=UyAtPiBiIFMgZSB8ICM=>



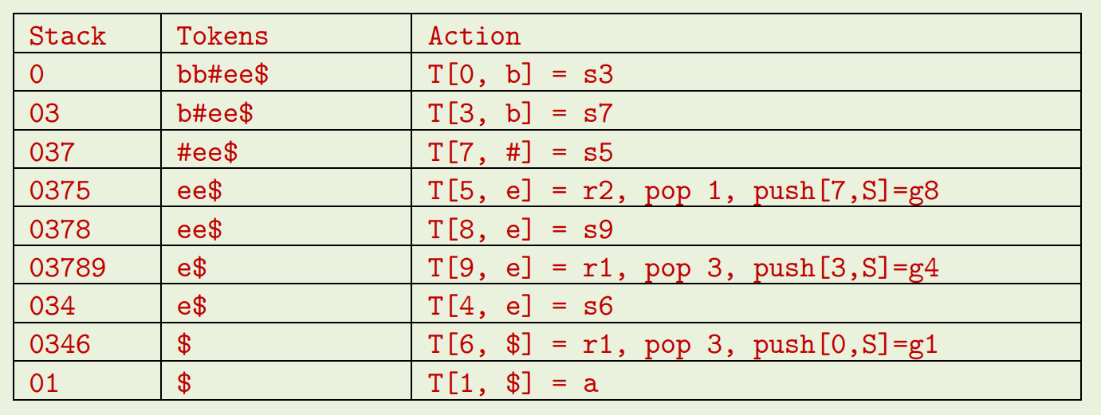
1e)

S’ -> S

r1: S -> b S e

r2: S -> #

|  |  |  |
| --- | --- | --- |
| Stack | Input | Action |
| 0 | bb#ee$ | T[0, b] = s2 – push 2 |
| 02 | b#ee$ | T[2, b] = s5 – push 5 |
| 025 | #ee$ | T[5, #] = s6 – push 6 |
| 0256 | ee$ | T[6, e] = r2 – pop 1 item and push T[5, S] |
| 0258 | ee$ | T[8, e] = s9 – push 9 |
| 02589 | e$ | T[9, e] = r1 – pop 3 items and push T[2, S] |
| 024 | e$ | T[4, e] = s7 – push 7 |
| 0247 | $ | T[7, $] = r1 – pop 3 items and push T[0, S] |
| 01 | $ | accept |



1f) S

/ | \

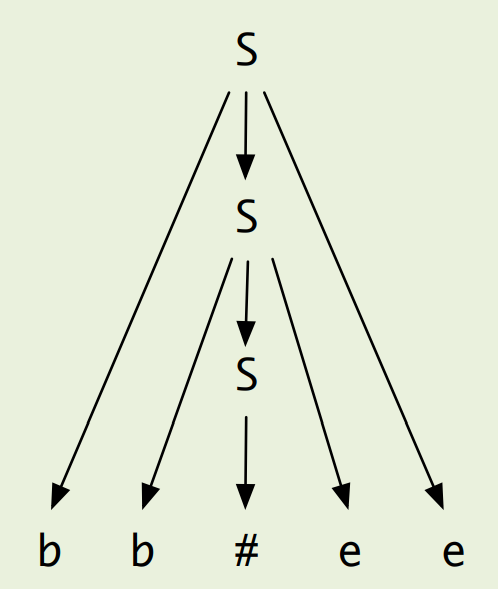
b S e

/ | \

b S e

|

#



2a)

If a loop header has more than one predecessor then the loop invariant code must be pulled into a pre-header to maintain control flow structure.

2b)

It is killed when the definition(variable) is re-defined(assigned again)

2c)

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

Takes in the expr to translate, the available registers, the current register, and outputs generated instructions.

TransExp (Num i) (r : \_)

= [Mov (ImmNum i) (Reg r)]

TransExp (Var s) (r : \_)

= [Mov (Abs s) (Reg r)]

TransExp (Plus e1 e2) (r1 : r2 : rs)

= transExp e1 (r1 : r2 : rs)

++ transExp e2 (r2 : rs)

++ [Add (Reg r2) (Reg r1)]

2d) ?

TransStat :: Stat -> [Instr]

TransStat (Assign str e) (r1:rs)

= transExp e (r1:rs)

++ [Mov (Reg r1) (Abs str)]

2e)

TransStat (For var e1 e2 stats) (r1:rs)

= transStat (Assign var e1) (r1:rs) -- this doesnt acc use a register

++ [Define exprLabel]

++ transExp e2 (r1:rs) -- this uses r1-

++ [Cmp (Abs var) (Reg r1)]

++ [Bge endLabel]

++ concat (map (\s -> transStat s rs) stats)

++ transExp (Plus (ImmNum 1) (Var var)) rs

++ [Bra exprLabel]

++ [Define endLabel]

2f)

TransStat (For var (Num i) (Num j) stats) (r1:rs)

| i >= j = []

| otherwise = transStat (Assign var e1)) (r1:rs)

++ replicate (j-i) (concat (map (\s -> transStat s rs) stats) ++ transExp (Plus (ImmNum 1)(Var var)) rs)

2g)

1. Need to check that the cost of increasing the code size will outweigh the possibility of icache misses that will impact performance.
2. Unnecessary branch conditionals can clog up the database, e.g. large switch statement in a redundant else will never be used but will increase code size for no reason.
3. We may run out of registers if we have to keep storing many temporary variables.
4. T

I don’t know if this is correct but: One advantage of unrolling the loops is that many moderns processors pre-execute instructions which wouldn’t be the case with a branch if the branch condition depends on the currently being executed code. Much like in the meltdown attack from OS.

Google’s Answer:

1. Increased program code size, which can be undesirable.
2. Possible increased usage of register in a single iteration to store temporary variables which may reduce performance.
3. **Unrolled loops** that contain branches are even slower than recursions.