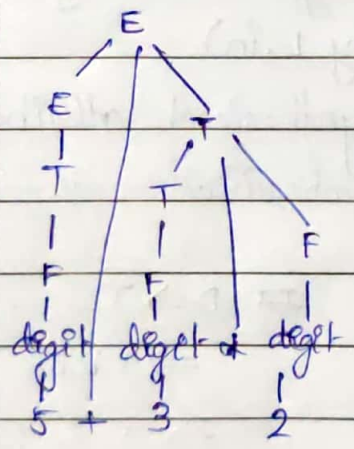
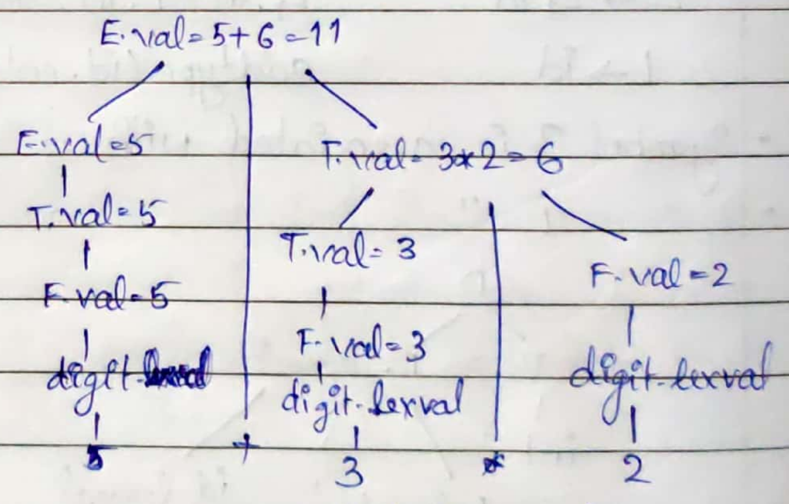
Productⁿ	Semantic Rules
$E \rightarrow E_1 + T$	E.loc = newtemp(), E.code = E_1.code || T.code || add E_1.loc, T.loc, E.loc
$E \rightarrow T$	E.loc = T.loc, E.code = T.code
$T \rightarrow T_1 * F$	T.loc = newtemp(), T.code = T_1.code || F.code || mult T_1.loc, F.loc, T.loc
$T \rightarrow F$	T.loc = F.loc, T.code = F.code
$F \rightarrow (E)$	F.loc = E.loc, F.code = E.code
$F \rightarrow id$	F.loc = id.name, F.code = ""

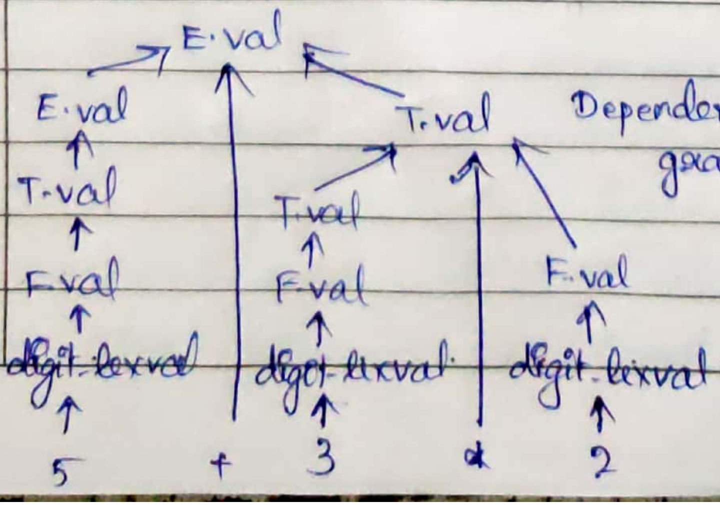
- Symbols E, T & F are associated with synthesized attributes loc & code.
- The token id has a synthesized attribute name
- It is assumed that || is the string concatenation operator



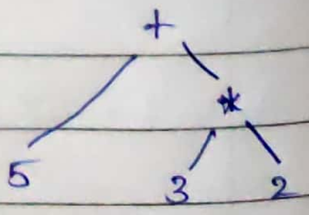
Parse tree



Annotated Parse tree



Dependency graph



Syntax tree

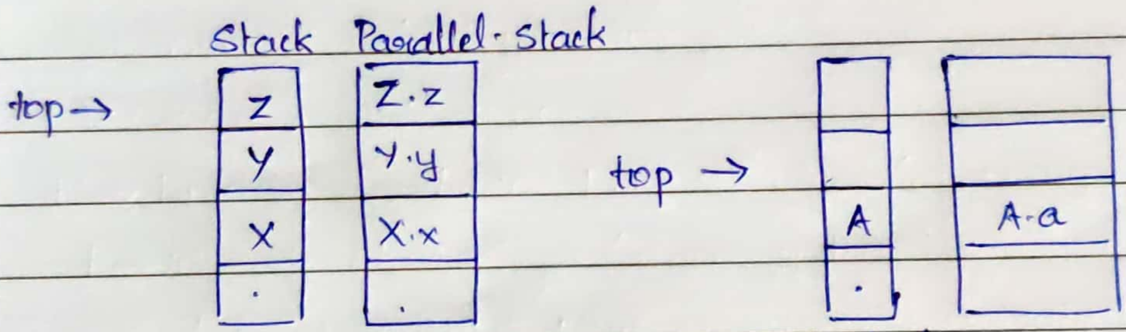
Bottom-Up Evaluatⁿ of S-Attributed Definitⁿs

- It is done using shift-reduce parser
- We evaluate values of attributes during reductⁿs
- 2 stacks:

→ Stack with grammar symbols

→ Parallel stack with synthesized attribute(s) of corresponding grammar symbols

~~eg~~ A → XYZ A.a = f(X.x, Y.y, Z.z) where all attributes are synthesized.

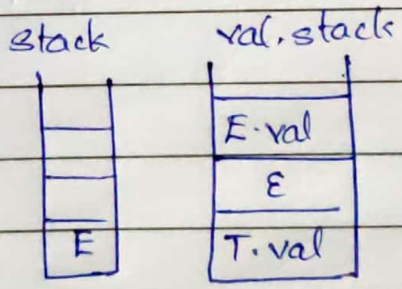


eg: E → E+T

E.val = E.val + T.val

val.stack [top]

val.stack [top-2]



val.stack[ntop] = val.stack[top] + val.stack[top-2]

Productⁿ	Semantic Rules
L → E return	print(val[top-1])
E → E₁ + T	val[ntop] = val[top-2] + val[top]
E → T	
T → T₁ * F	val[ntop] = val[top-2] * val[top]
T → F	

11/2/20

F → (E) val[ntop] = val[top-1]

F → digit

Scanned by CamScanner