SDT Part-2

Complete Course on Compiler Design

Subbarao Lingamgunta • Lesson 15 • Sept 4, 2023

Syntax Directed Translation (SDT)

$S \rightarrow ABC$ { pf("hi") } — SDT

Gramm + Semantic action

```
E → E1 + E2 {  if(E1.type == E2.type)
                    E.type = E1.type
                 else
                    pf("type mismatch")
             }
```

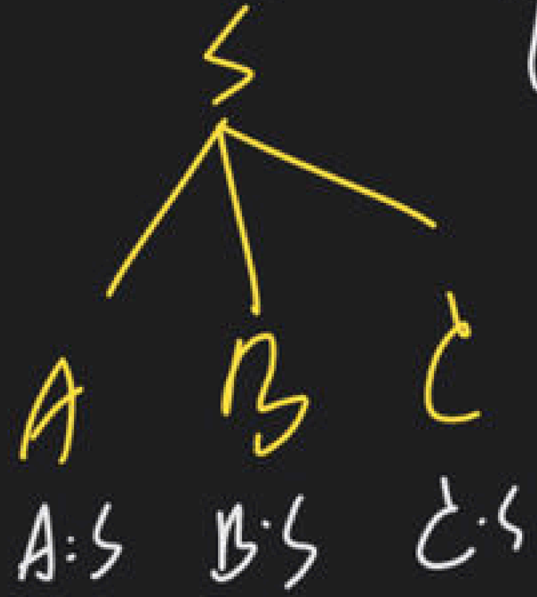
Attributes

Synthesized attribute

⇓

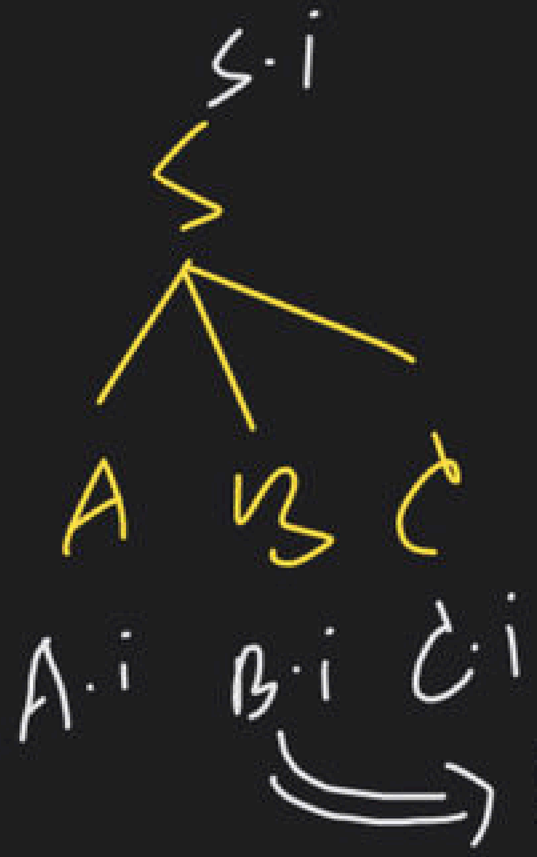
$$S \longrightarrow ABC \left\{ S.S = f(A.S \,|\, B.S \,|\, C.S) \right\}$$

$$S.S = \{ A.S * B.S \}$$



Inherited attribute

$$S \longrightarrow ABC \left\{ B.i = f(S.i \,|\, A.i \,|\, C.i) \right\}$$



$B.i = A.i + C.i * S.i$

SDT

S-attributed Definition

① It uses Synthesized attributes.

② It follows Bottom-Up evaluation (or) Postorder Evaluation

③ Semantic actions will be placed on RHS, Rmp

L-attributed Definition

① It uses both Syn && Inh but in inherited ⇒ P, L

② It follows Left to Right Depth First order

③ Semantic actions will be placed anywhere on RHS

Applications of SDT

1. Arithmetic expression Evaluation
2. Infix to postfix conversion
3. " " Prefix "
4. Binary to Decimal "
5. Type checking
6. Syntax Tree Creation
7. String into Symbol Table
8. Counting no. of reductions
9. intermediate code genera

⋮

Construct SDT to evaluate given A.P

~~Arithmetic expr Evaluation~~

i/p : 2 + 3 * 4

o/p : ~~14~~ 14

$E \longrightarrow E + T$ { $E.val = \cancel{E.val * T.val}$; pf(E.val) }

$\quad | \; T$ { $E.val = T.val$ }

$T \longrightarrow T * F$ { $T.val = \cancel{T.val * F.val}$ }

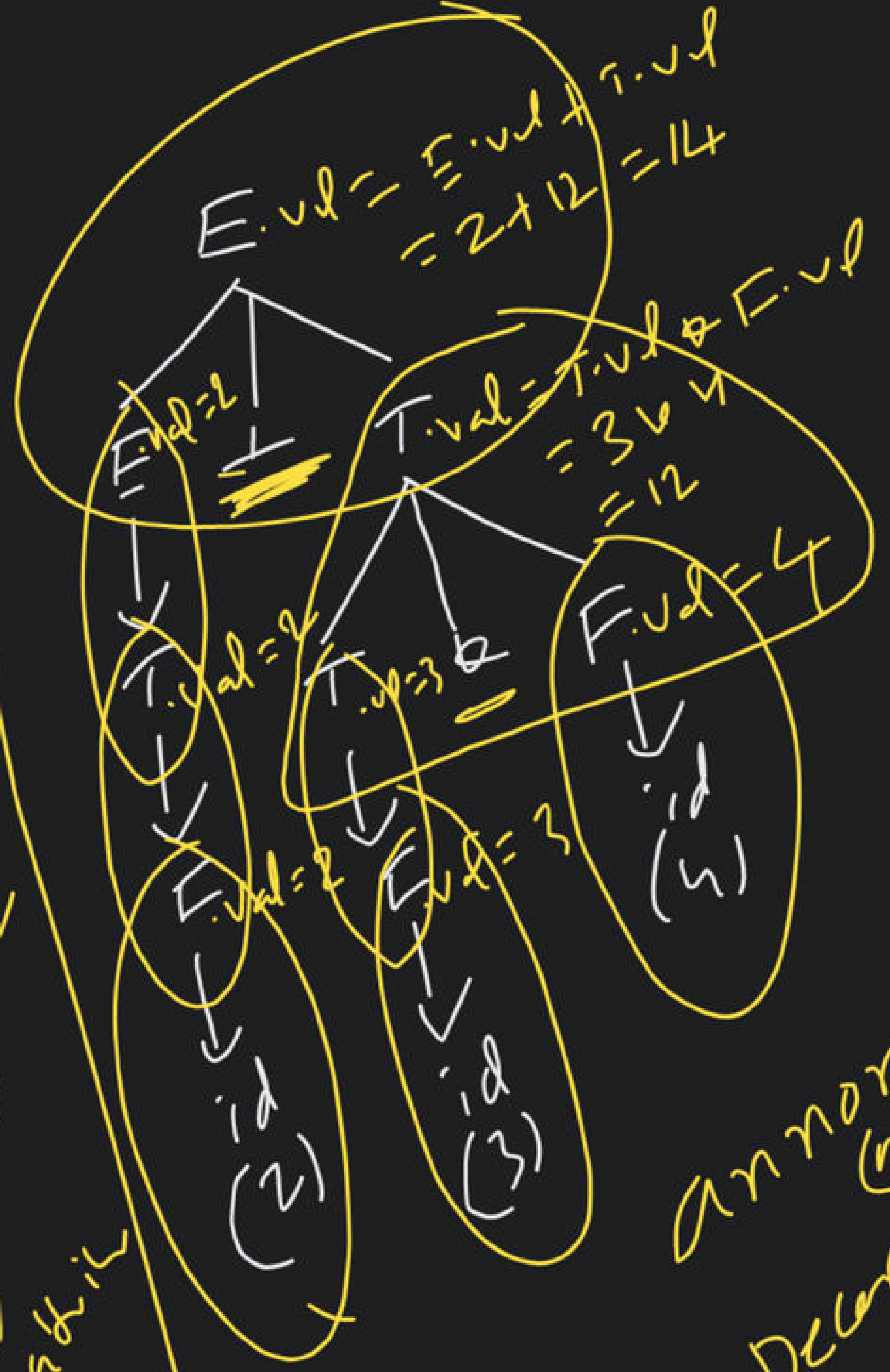
$\quad | \; F$ { $T.val = F.val$ }

$F \longrightarrow id$ { $F.val = id$ }

val — Syn. attr

||

S-attributed Def → L-attri.



Annotated P.T
(or)
Decorated P.T

Consider the following SDT

i/p = 8 # 2 & 4 # 16 & 12 # 4 & 2

o/p = 512

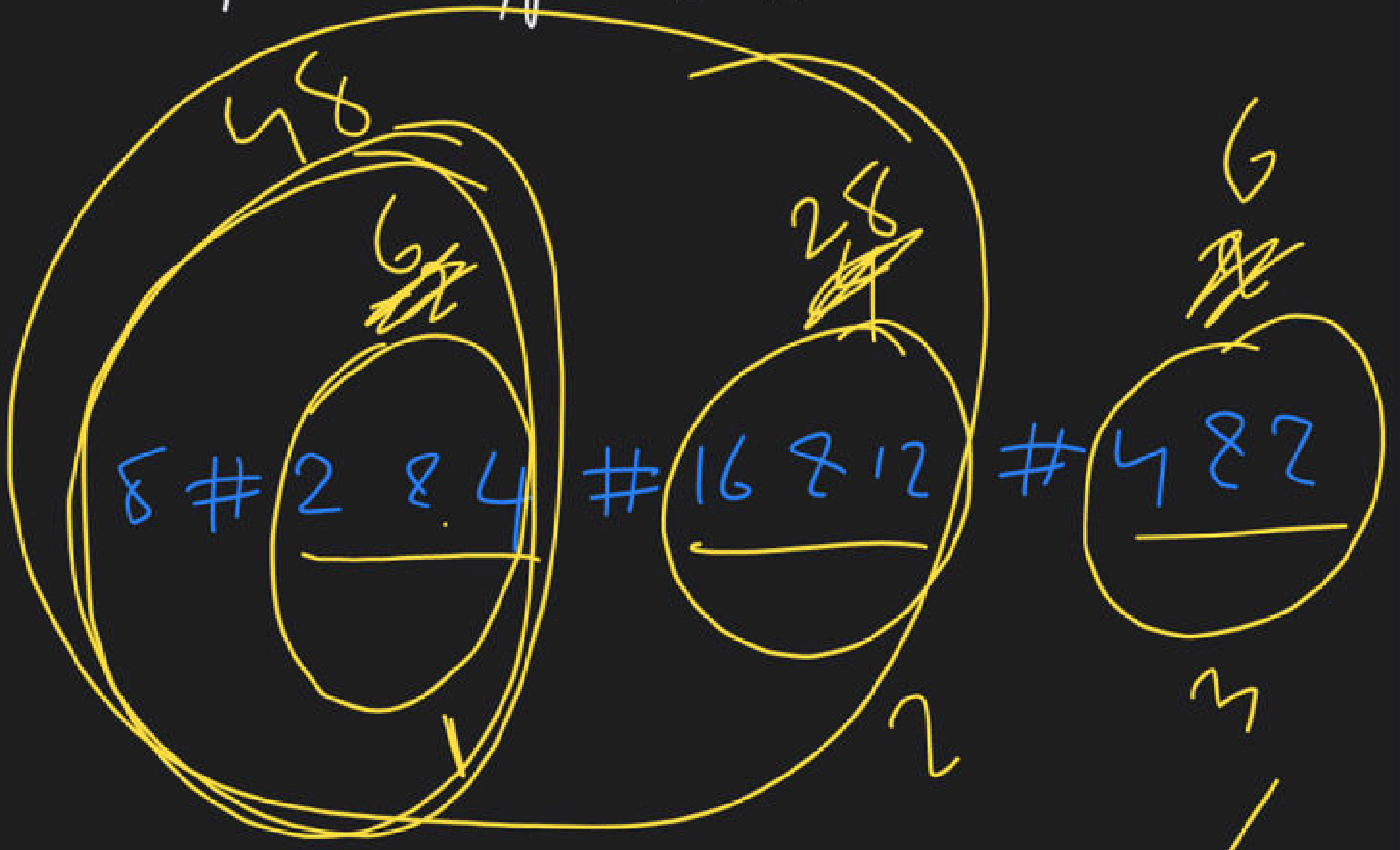
$E \rightarrow E_1 \# T$ { $E.val = E_1.val * T.val$ }

$| T$ { $E.val = T.val$ }

$T \rightarrow T_1 \& F$ { ~~?~~ $T.val = T_1.val \, \boxed{*} \, F.val$ }

$| F$ { $T.val = F.val$ }

$F \rightarrow num$ { $F.val = num$ }



(a) & ⇒ * $T.val = T_1.val * F.val$

(b) & ⇒ + $T.val = T_1.val + F.val$

(c) & ⇒ − $T.val = T_1.val - F.val$

[✓ (d)] None

$E \rightarrow E_1 \# T \{ E.vl = E_1.vl * T.vl \}$

$\quad | T \{ E.vl = T.vl \}$

$T \rightarrow T_1 \& F \{ T.vl = T_1.vl \textcircled{+} F.vl \}$

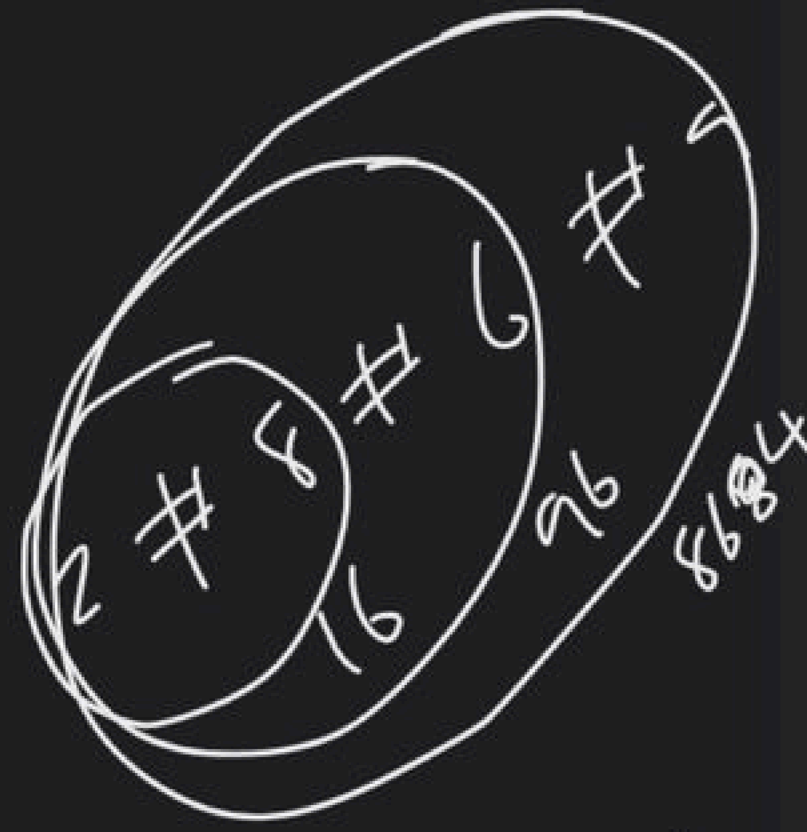
$\quad | F \{ T.vl = F.vl \}$

$F \rightarrow num \{ F.vl = num \}$

i/p: 2 # 3 & 5 # 6 # 4 & 2 & 3

8 6 9

o/p: 864



construct SDT to convert given Infix expression into postfix Expression

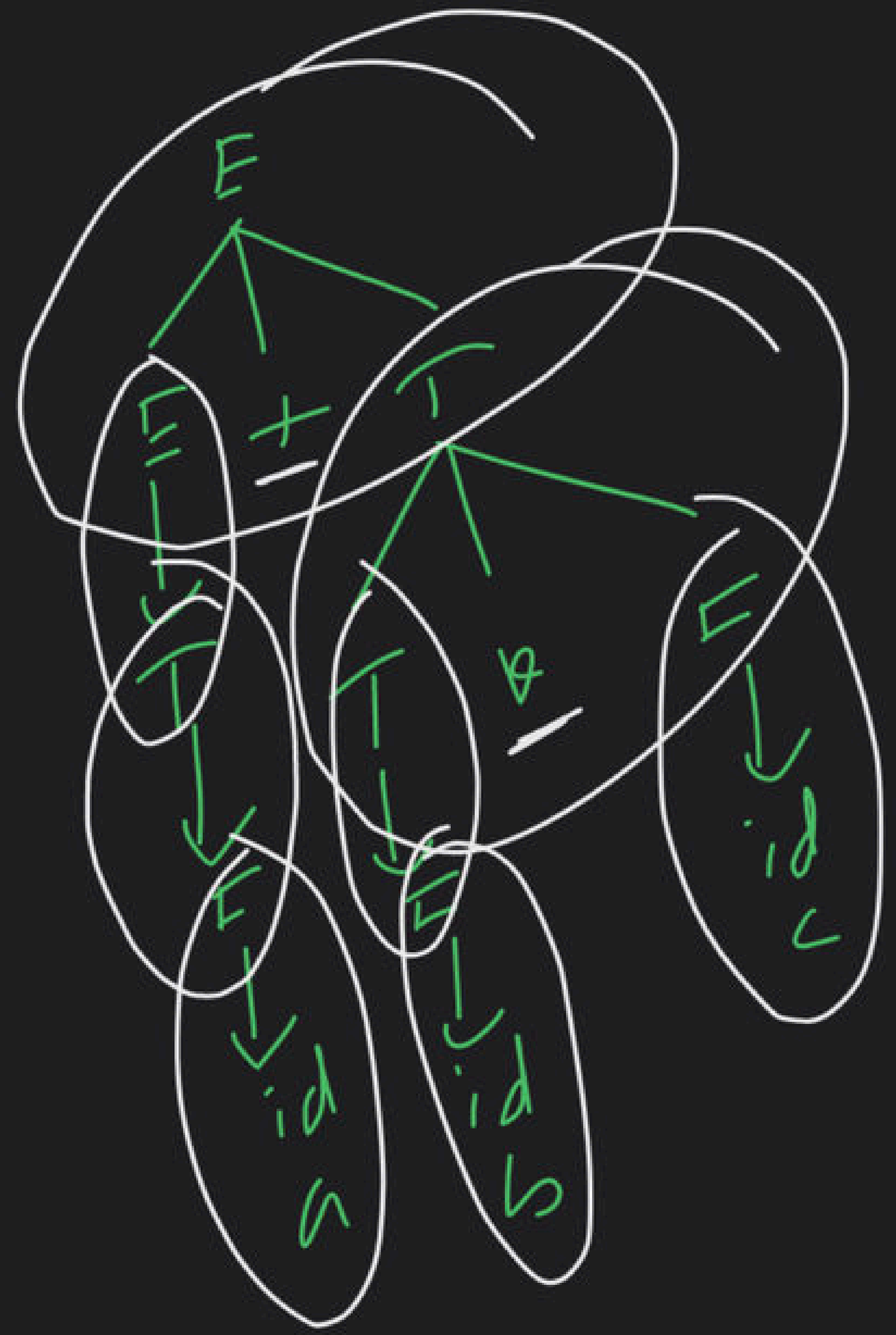
⟹ i/p: a+b*c

o/p: abc*+

E ⟶ E+T {rs(+)}
  | T

T ⟶ T*F {rs(*)}
  | F

F ⟶ id {rs(id)}



i/p : a+b*c

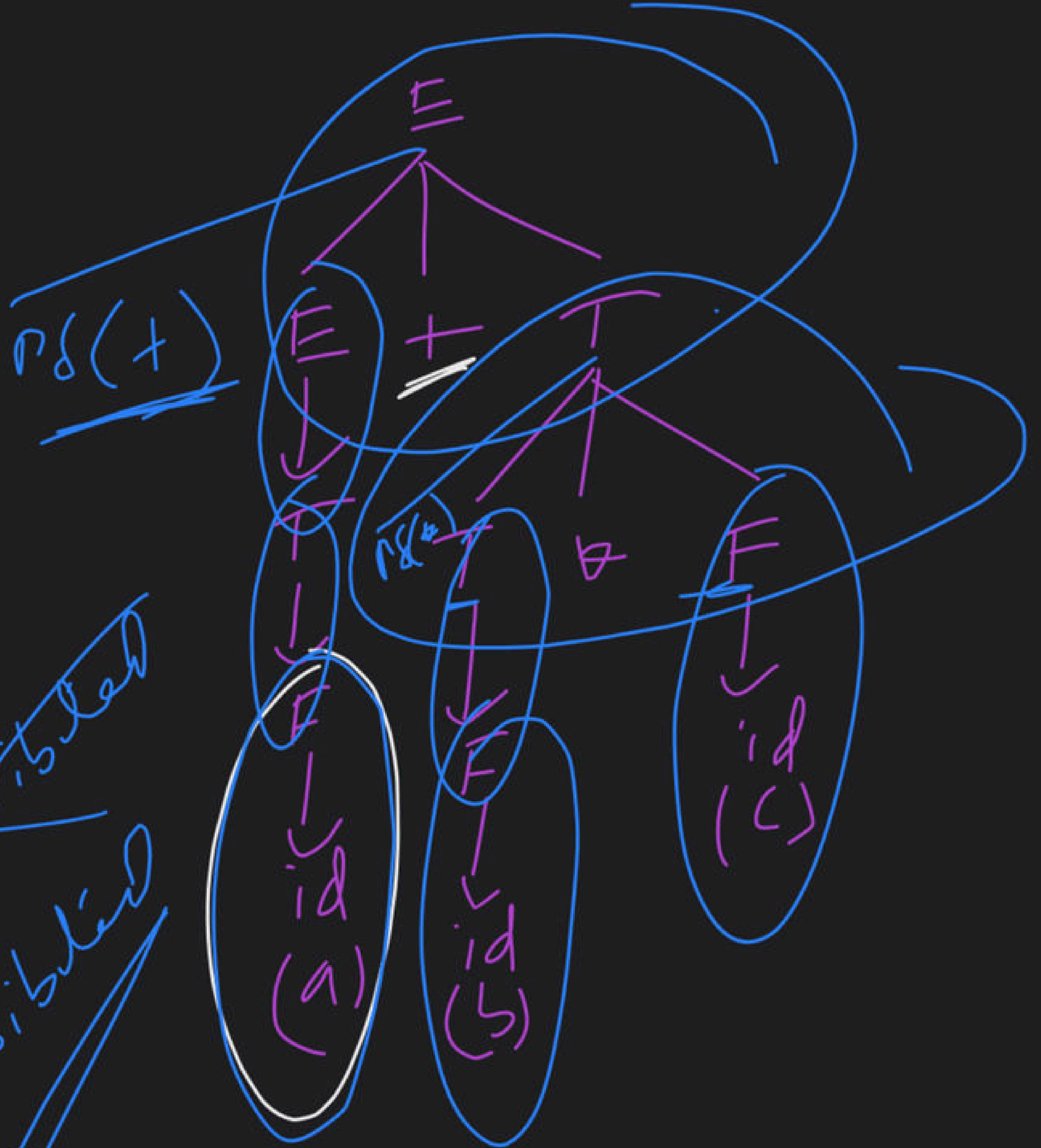
*bc

o/p: +a*bc

E → {pf(+)} E + T
 | T

T → {pf(*)} T * F
 | F

F → id {pf(id)}



~~S-attributed~~

~~L-attributed~~