UNIT-3: Semantic Analysis

→ Why / What Semantic Analysis

- Helps with getting more info on Sym table entries.
- SDT's are a representation formalism.
- Meaning of a Sentence is related to its parse-tree
  - We have attributes of grammar Syms by rep. lang constructs
  - Value for attrs are computed by Sem rules ass. with grammar prod.

for attaching rules:

- SDD : high level
- Translation Scheme : Also indicates order of eval of rules.

① SDD : generalisation of CFG's where:

- Each Symbol has a set of attrs
- Each prod has Sem rules for Computing attrs.

Each prod rule has rules like:

$A \rightarrow \alpha \quad b = f(c_1, c_2, \ldots c_k)$

b is either Synthesised or inherited attr of A.

$c_1, c_2 \ldots c_k$ are attrs of grammar Symbols (child → parent)

$c_1, c_2 \ldots c_k$ are attrs of Gramar Symbol that are present in α or attrs of A itself (parent → child / siblings)

term Symbols have Syn attrs from lexer.

a SDD with only Synthesis is called S-Attributed dfg.

for S-attr dfg, we can use post order eval

•) eval SDD:

- Construct parse tree
- " dependency graph
- Sort nodes of " "
- Produce outputs of annotate parse tree

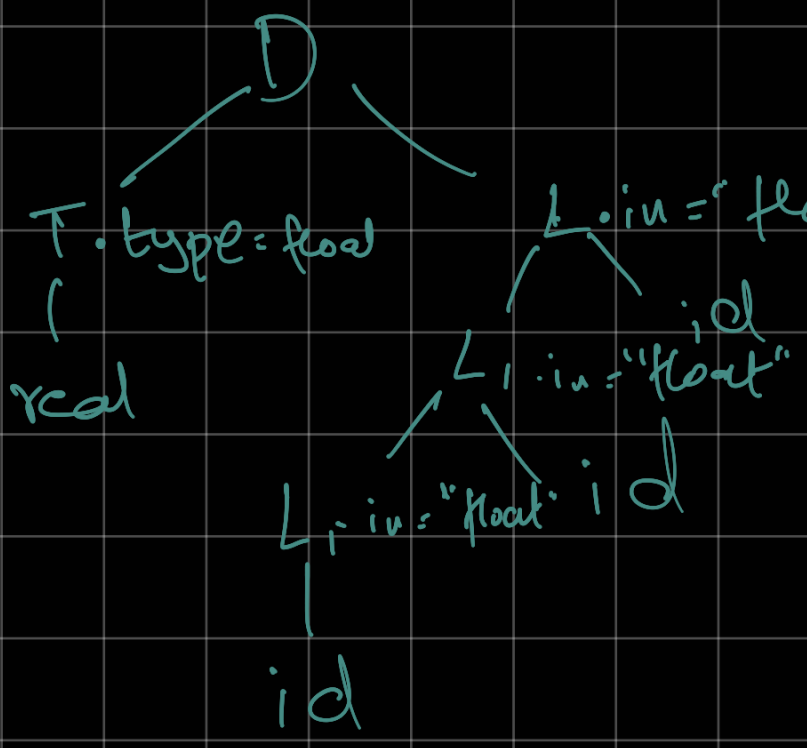
depency graph: if attrb is dependent on c then is link from $c \to b$ hence we need to give sem rule for c before b.

Q)

Production	Semantic Rule
D → T L	{ L.in = T.type;}
T → int	{ T.type = integer; }
T → real	{ T.type = float; }
L → L_1 , id	{ L_1.in= L.in; addType(id.entry, L.in); }
L → id	{ addType(id.entry, L.in); }

real id1, id2, id3

Parse tree with dependency (not really needed)



⇒ If we have circular dependency, can't eval. fine consuming due to dependency.

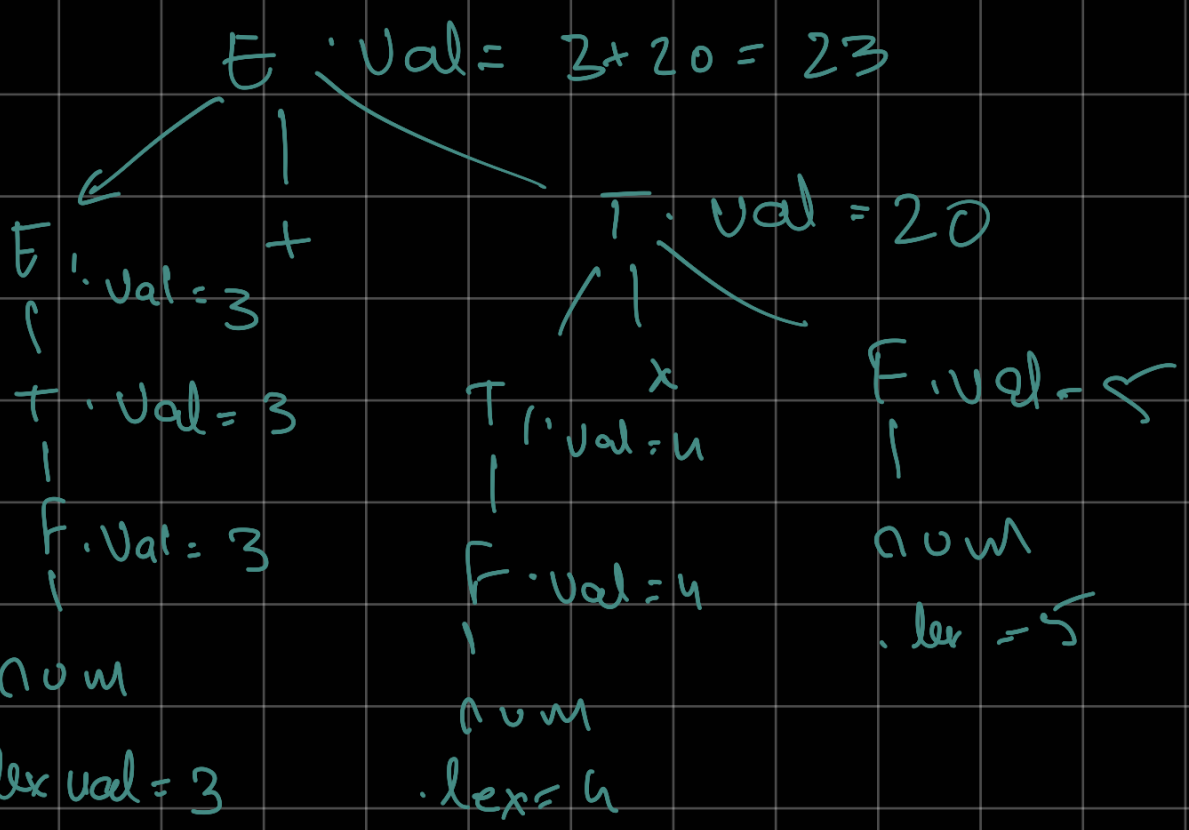
⇒ alternate: S attr SDD for fixed order Dep can be achieved using a Bup by extending the Stack of a LR parser.

Q)

- Evaluate the following SDD for the input 3 + 4 * 5

Production	Semantic Rule
$E \rightarrow E_1 + T$	{ E.val = E_1.val + T.val }
$E \rightarrow T$	{ E.val = T.val }
$T \rightarrow T_1 * F$	{ T.val = T_1.val * F.val }
$T \rightarrow F$	{ T.val = F.val }
$F \rightarrow num$	{ F.val = num.lexval }

3 + 4 * 5



SDD to Convert Bin to Dec:

$L \Rightarrow L_1 B$	$L.Val = 2 * L_1.Val + B.Val$
$L \Rightarrow B$	$L.Val = B.Val$
$B \Rightarrow 0$	$B.Val = 0$
$B \Rightarrow 1$	$B.Val = 1$

Binary with float → Dec.

$L \rightarrow L_1 . L_2$ { $L.Val = L_1.Val + L_2.Val / 2^{L_2.Count}$ }

$L \rightarrow L_1 B$ { $L.Val = 2 * L_1.Val + B.Val$;
$L.Count = L_1.Count + B.Count$ }

$L \rightarrow B$ $L.Count = B.Count$; $L.Val = B.Val$.

$B \rightarrow 0$ $B.Count = 1$ $B.Val = 0$

$B \Rightarrow 1$ " " " " $B.Val = 1$

Infix to prefix ? (Lets see later)
(You need SDTS)

SDD to find Sign of result:

$S \to E$

$E \to num$	$E.sign = POS$
$E \to +E_1$	$E.Sign = E_1.Sign$
$E \to -E_1$	if $(E_1.Sign == POS)$ $E.Sign = NEG$
	else $E.Sig = POS$
$E \to E_1 * E_2$	if $E_1.Sign == E_2.Sign$ $E.Sign = POS$
	else $E.Sign = NEG$

→ L-attr SDD

i) Synthesised
ii) Extended (itself)
iii) Siblings on left. Only on $(X_1, X_2, \dots X_{j-1})$

$A \to X_1 X_2 X_3 \dots X_j X_{j+1} \dots$
attrs of X_j depend

L-SDD for prefix eval:

$E \to +TE'$ $\quad E'.ival = T.val$; $E.val = E'.val$

$E' \to +TE_1'$ $\quad E_1'.ival = T.val + E'.ival$; $E^{-1}.val = E_1'.val$

$E \to \lambda$ $\quad E'.ival = E.val$

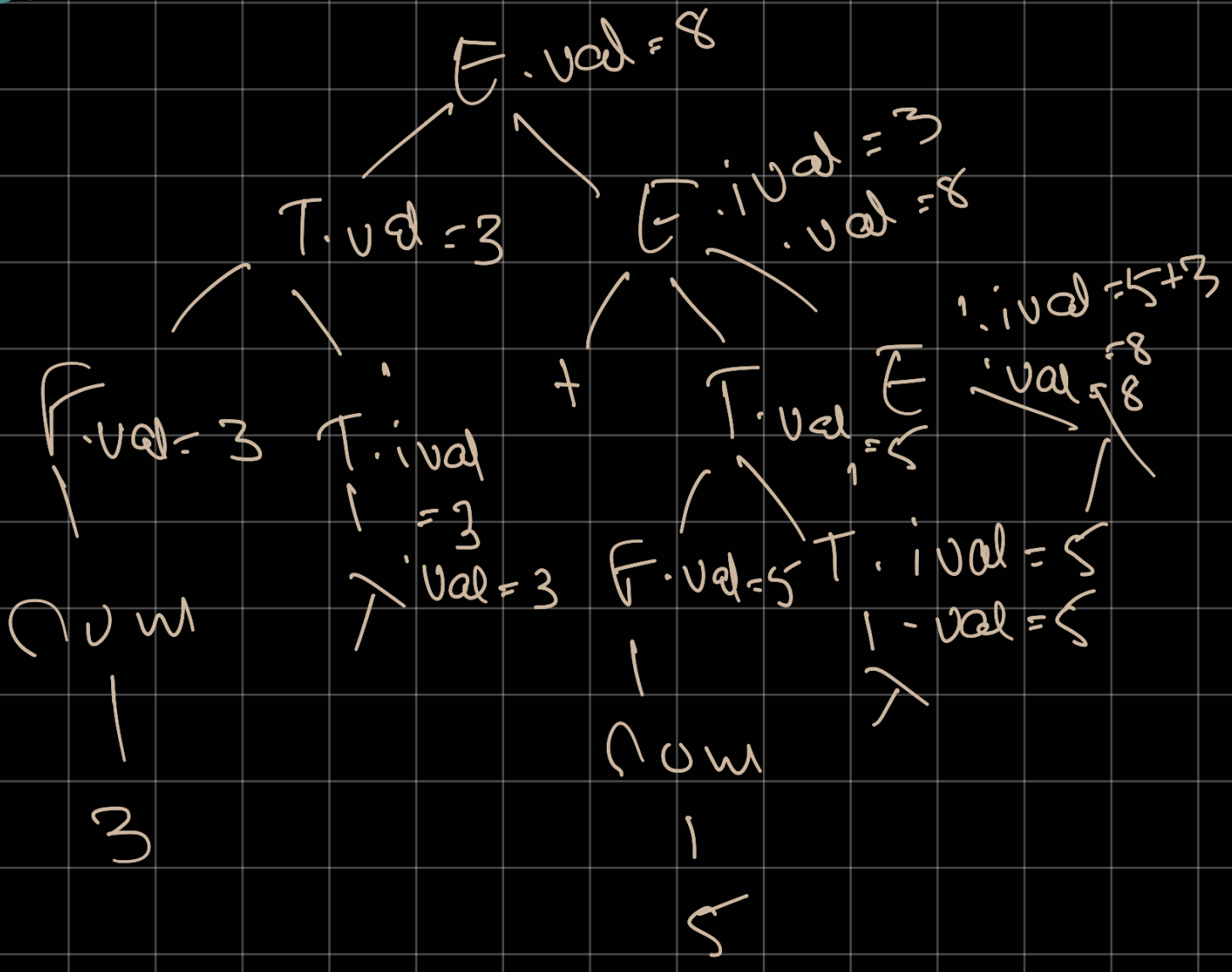
$T \to FT'$ $\quad T'.ival = F.val$; $T.val = T'.val$

$T \to *FT_1'$ $\quad T_1'.ival = T.ival * F.val$; $T.val = T_1'.val$

$T' \to \lambda$ $\quad T'.ival = T'.val$

$F \to num$ $\quad F.val = num.lexVal$

$3+5$:



Q) $D \to TL$ $L.itype = T.type$ $L.iwidth = T.width$

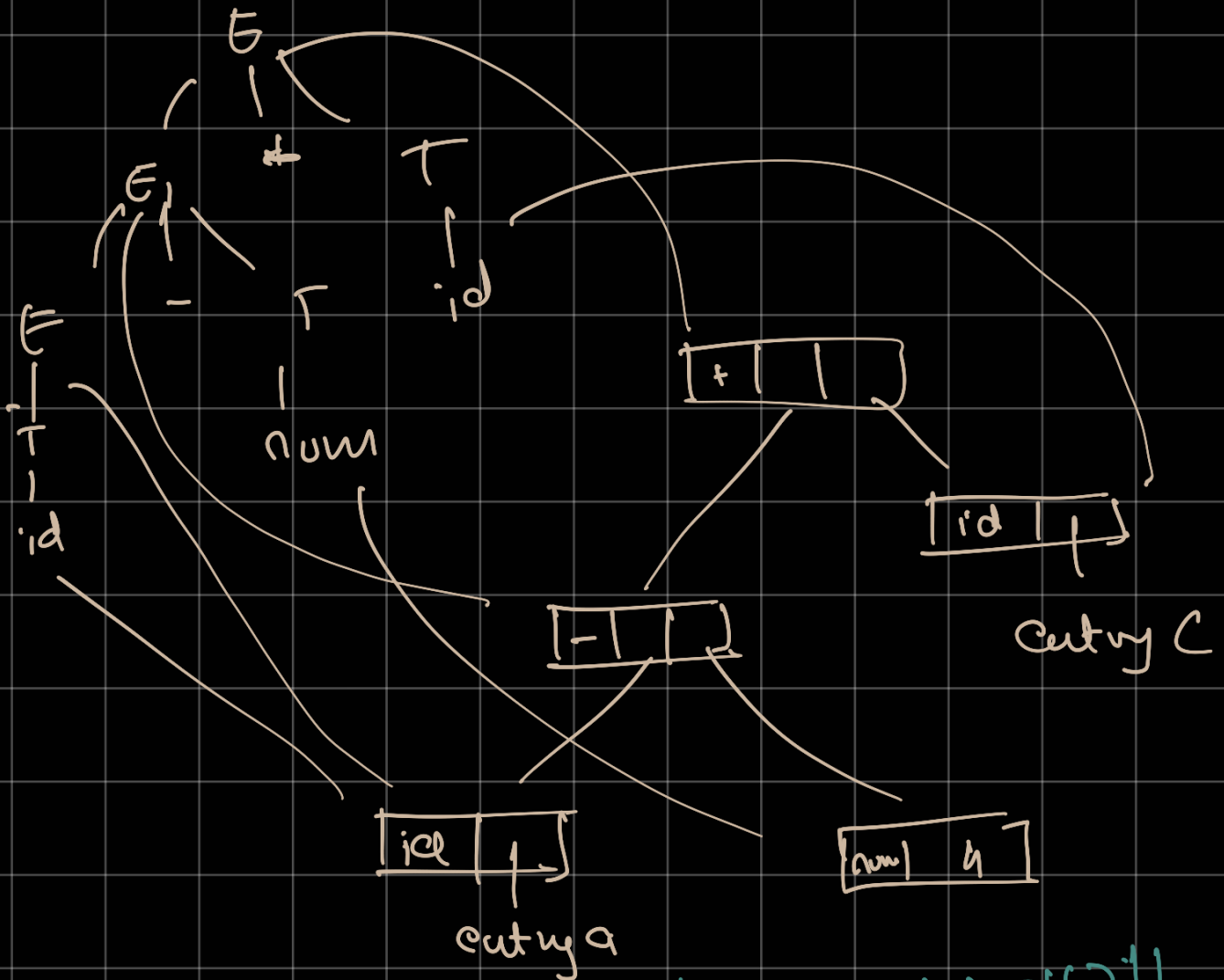
$T \to int$ $T.type = int$ $T.width = 4$

$T \to float$ $T.type = float$ $T.width = 8$

$L \to L_1, id$ $L_1.itype = L.itype$ $L_1.iwidth = L.iwidth$

$L \to id$

a) Slide 73 $a - 4 + C$



→ ICG using SDD

$S \to do\ S\ while\ B$ $begin = new\ label()$ || $S.code$ || $B.true = begin$ || $B.fal = S.next$

$B \to id1\ rel\ id2$ $B.code = S.code$ || gen(if id1.lex rel.op id2.lex goto B.true) || goto(B.false)

$rel \to >$ $rel.op$

$rel \to <$ $rel.op$

rel → >= rel.op

rel → <= rel.op

→ SDT

if action appears at end of production it is called a postfix SDT

SDD → SDT

place inherited attrs for NT before NT

place Synthesized attrs at end of the prod.