

Syntax-Directed Translation

- Context-free grammar with synthesized and/or inherited attributes.
- The showing of values at nodes of a parse tree is called “annotating the parse tree.”
- Computing values to be stored at the annotated nodes is called “decorating the parse tree.”

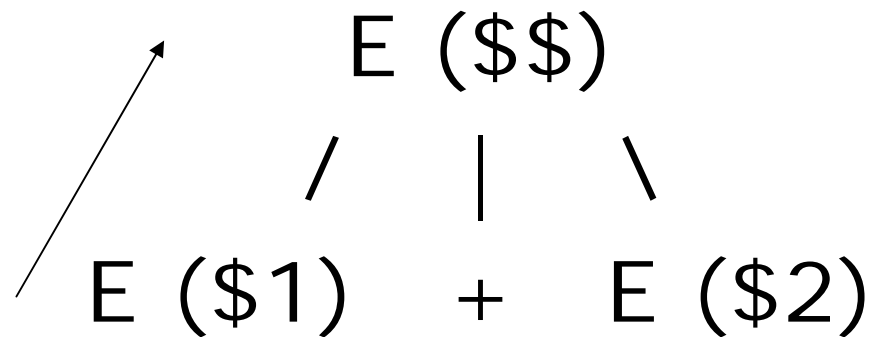
Synthesized Attributes

- Attributes passed up the parse tree.
- For example, the following involves synthesizing attributes:

`expr: expr '+' expr { $$ = $1; }`

Synthesized Attribute

$$$ = \$1;$



Inherited Attributes

- Attributes passed downwards or sideways in the parse tree.
- The following involves passing attributes down the parse tree:

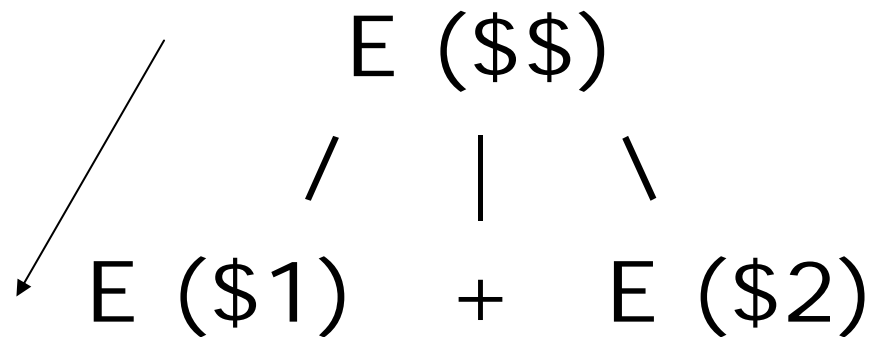
expr: expr '+' expr { \$1 = \$\$; }

- The following involves passing sideways the parse tree:

expr: expr '+' expr { \$1 = \$2; }

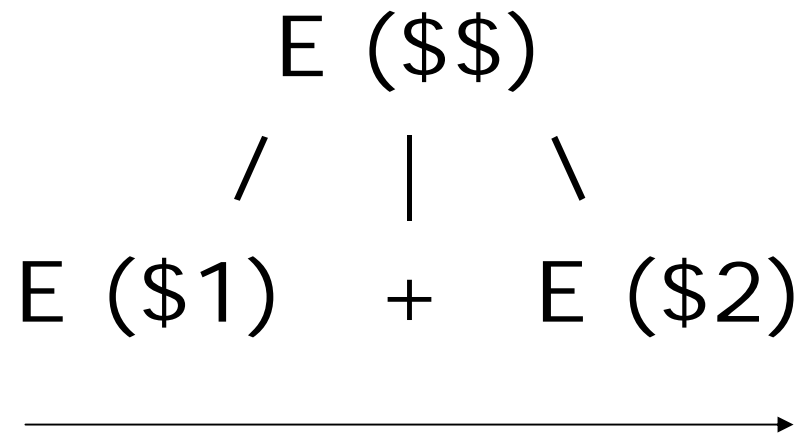
Attribute Inherited Downwards

$\$1 = \$\$;$



Attribute Inherited Sideways

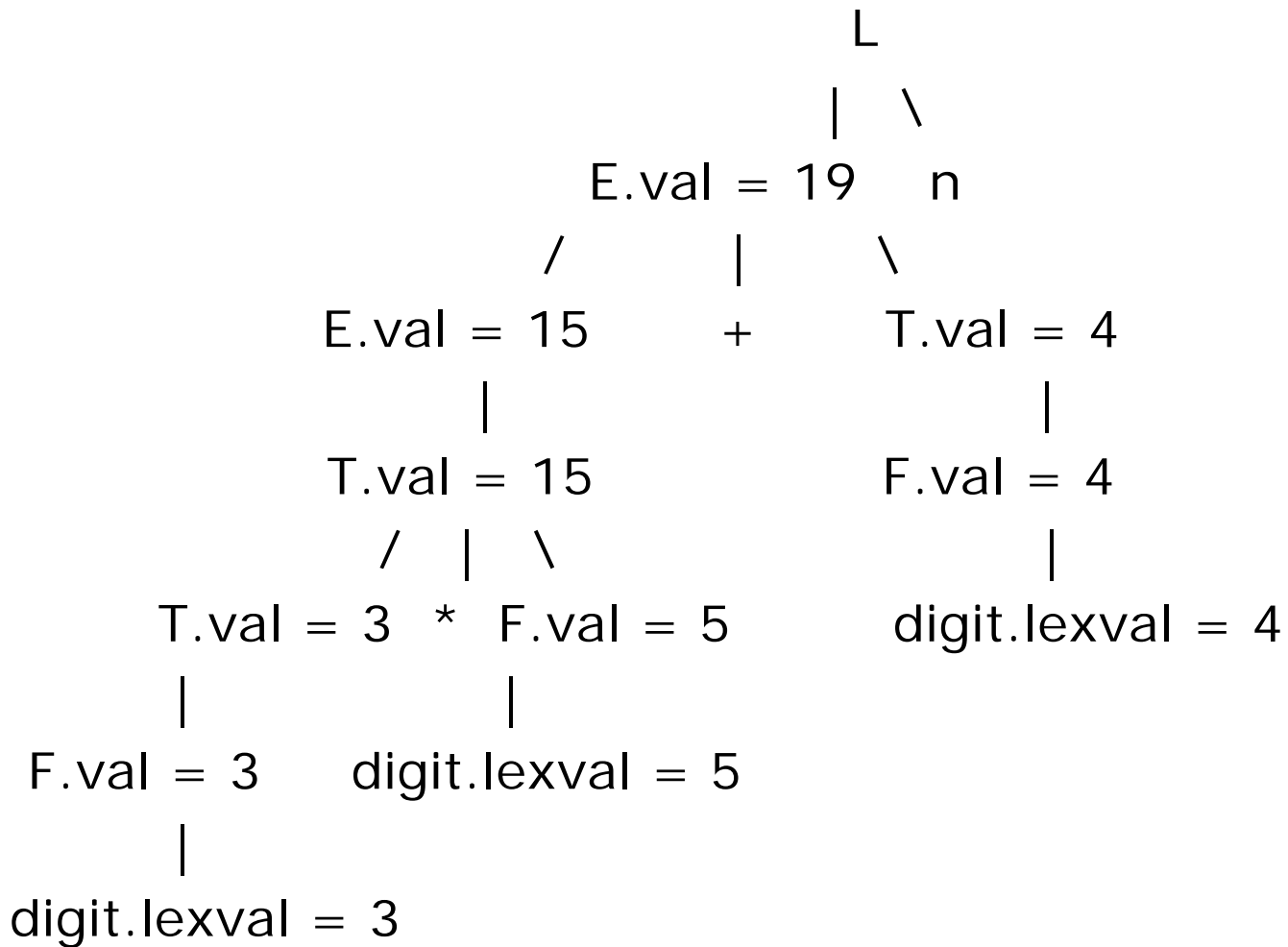
$$\$2 = \$1;$$



Desk Calculator - Semantic Rules

Production	Semantic Rules
$L \rightarrow E n$	<code>print(E.val)</code>
$E \rightarrow E_1 + T$	<code>E.val = E₁.val + T.val</code>
$E \rightarrow T$	<code>E.val = T.val</code>
$T \rightarrow T_1 * F$	<code>T.val = T₁.val * F.val</code>
$T \rightarrow F$	<code>T.val = F.val</code>
$F \rightarrow (E)$	<code>F.val = E.val</code>
$F \rightarrow \text{digit}$	<code>F.val = digit.lexval</code>

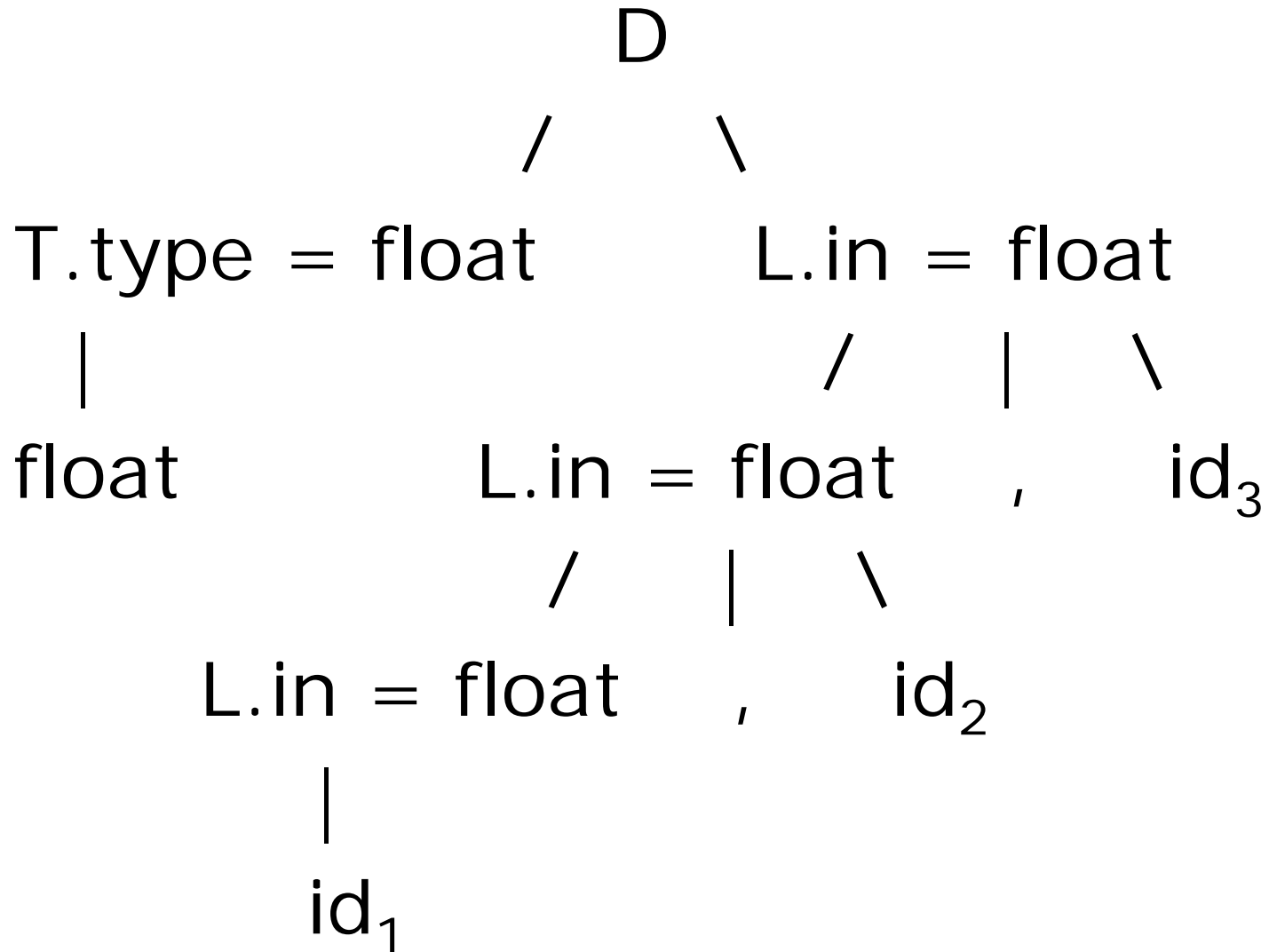
Parse Tree for $3 * 5 + 4 n$



Declarations – Semantic Rules

production	Semantic Rules
$D \rightarrow T L$	$L.in = T.type$
$T \rightarrow int$	$T.type = integer$
$T \rightarrow float$	$T.type = float$
$L \rightarrow L_1 , id$	$L_1.in = L.in$ $addtype(id.entry, L.in)$
$L \rightarrow id$	$addtype(id.entry, L.in)$

Parse Tree for float id₁, id₂, id₃



Abstract Syntax Tree (AST)

$(a - b) + ((a - b) + c)$

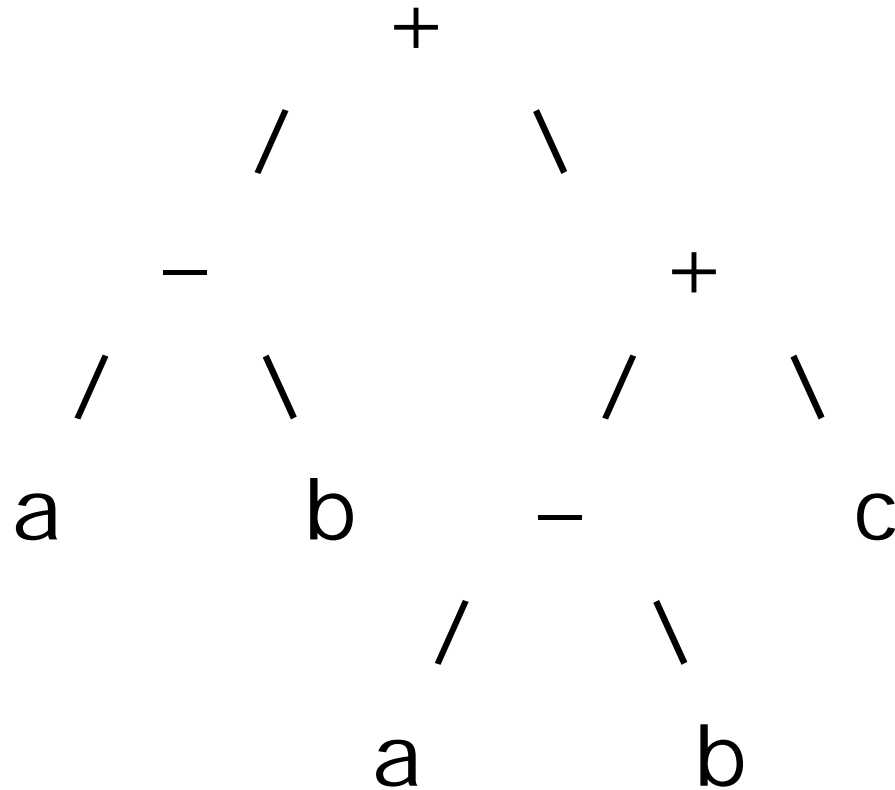


Table Representation of AST

(1)	-	a	b
(2)	-	a	b
(3)	+	(2)	c
(4)	+	(1)	(3)

Directed Acyclic Graph (DAG)

$$(a - b) + ((a - b) + c)$$

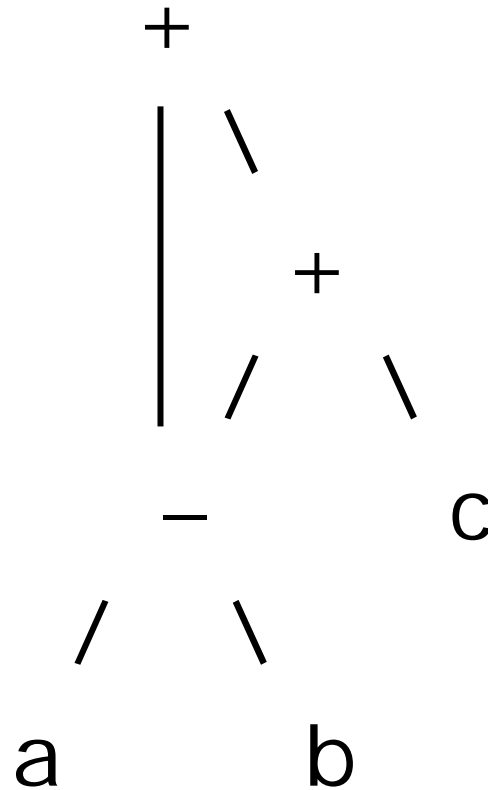


Table Representation of DAG

(1)	-	a	b
(2)	+	(1)	c
(3)	+	(1)	(2)

Generate AST from Productions

Production	Semantic Rules
$E \rightarrow E_1 + T$	$E.nptr = \text{mknode}('+', E_1.nptr, T.nptr)$
$E \rightarrow E_1 - T$	$E.nptr = \text{mknode}('-', E_1.nptr, T.nptr)$
$E \rightarrow T$	$E.nptr = T.nptr$
$T \rightarrow (E)$	$T.nptr = E.nptr$
$T \rightarrow \text{id}$	$T.nptr = \text{mkleaf}(\text{id}, \text{id.entry})$
$T \rightarrow \text{num}$	$T.nptr = \text{mkleaf}(\text{num}, \text{num.val})$

Prod to AST Example

$(a - b) + ((a - b) + c)$

p_1 mkleaf(id,a)

p_2 mkleaf(id,b)

p_3 mknnode('-', p_1 , p_2)

p_4 mkleaf(id,a)

p_5 mkleaf(id,b)

p_6 mknnode('-', p_4 , p_5)

p_7 mkleaf(id,c)

p_8 mknnode('+', p_6 , p_7)

p_9 mknnode('+', p_3 , p_8)