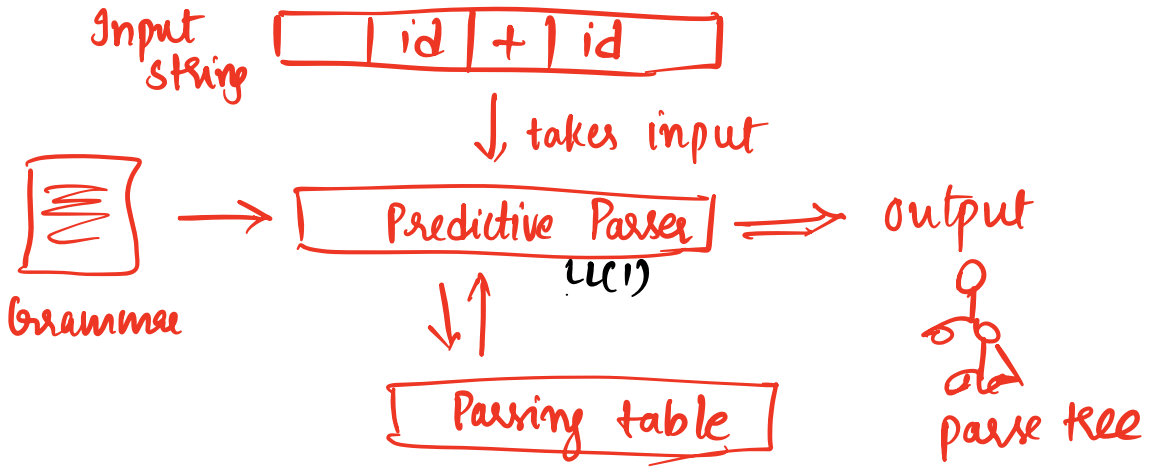
Predictive Parser

→ Non-Recursive descent parser / LL(1) / Table driven parser



Steps: for LL(1) Parser

* 1. Eliminate LR and also check common prefixes – L.Factor ✓
* 2. first & follow ✓
* 3. Parsing table ✓
* 4. Stack Implementation ✓
5. Parse tree

Q)

$E \rightarrow E+T/T$

$T \rightarrow T*F/F$

$F \rightarrow (E)/id$

$E \rightarrow E+T/T$

LHS — LM of RHS

Input string : "id + id * id"

$A \rightarrow A\alpha/\beta$

↓ Eliminate

$A \rightarrow \beta A'$

$A' \rightarrow \alpha A'/\varepsilon$

Step 1: LR & LF

$E \rightarrow E+T/T$

(A, A α, β)

1. $E \rightarrow TE'$
2. $E' \rightarrow +TE'/\varepsilon$

$T \rightarrow T*F/F$

(A, A α, β)

3. $T \rightarrow FT'$
4. $T' \rightarrow *FT'/\varepsilon$

1. $E \rightarrow TE'$
2. $E' \rightarrow +TE'/\varepsilon$ — $E' \rightarrow +TE'$, $E' \rightarrow \varepsilon$
3. $T \rightarrow FT'$
4. $T' \rightarrow *FT'/\varepsilon$ — $T' \rightarrow *FT'$, $T' \rightarrow \varepsilon$
5. $F \rightarrow (E)/id$

$F \rightarrow (E)$ & $F \rightarrow id$

LF:

$A \rightarrow \alpha\beta_1/\alpha\beta_2/\alpha\beta_3$

↓ LF

$A \rightarrow \alpha A'$

$A' \rightarrow \beta_1/\beta_2/\beta_3$

Step 2: ε ✓ ε ✗ $ added for 1st PR

	FIRST	FOLLOW
E	{ (, id }	{), $ }
E'	{ +, ε }	{), $ }
T	{ (, id }	{ +,), $ }
T'	{ *, ε }	{ +,), $ }
F	{ (, id }	{ *, +,), $ }

Step 3: parsing table

N.T ↓ / T →	id	+	*	(	)	$
① E	E→TE'			E→TE'		
② E'		E'→+TE'			E'→ε	E'→ε
③ T	T→FT'			T→FT'		
④ T'		T'→ε	T'→*FT'		T'→ε	T'→ε
⑤ F	F→id			F→(E)		

How to fill table

1. first of E is: {(, id} so fill the boxes of (, id with E production.
2. E' has + & ε, so check for follow E' and in follow

+ : $E' \rightarrow +TE'$

values), $ write $E' \rightarrow \varepsilon$

"id + id * id"

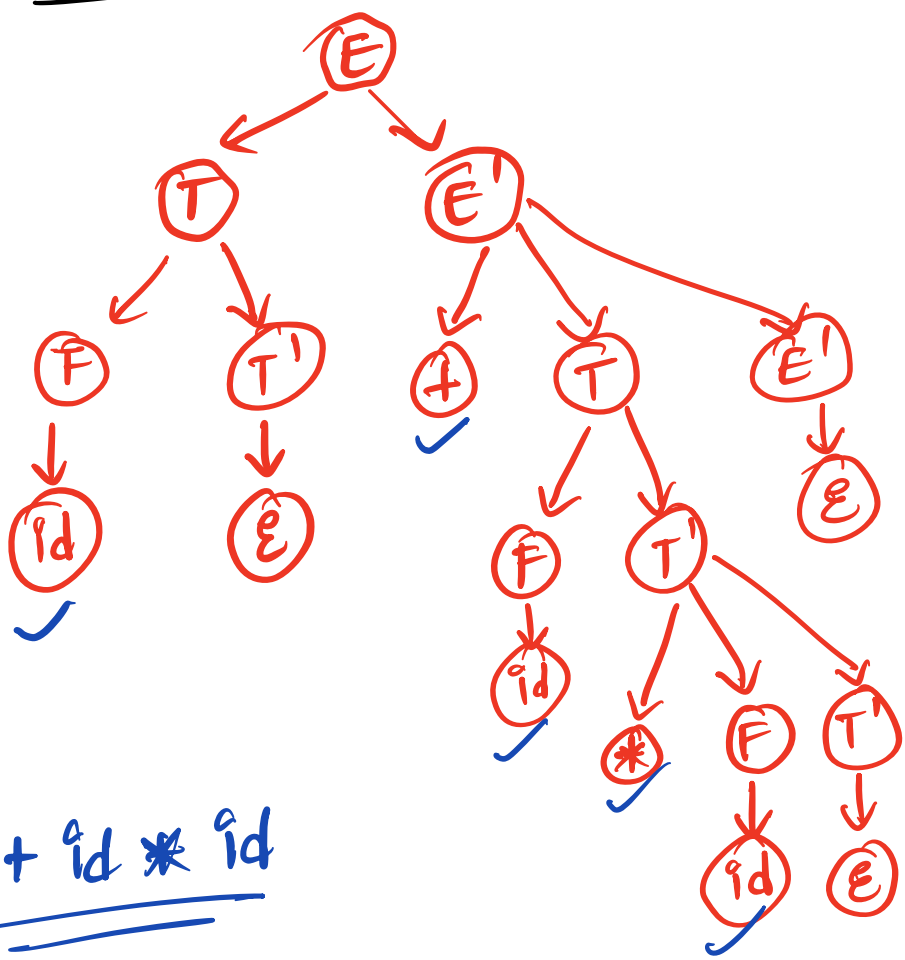
step 4: stack implementation

Stack	Input	Action - production Rules
E : root node $E\$$	id + id * id $	
compare left most value $TE'\$$	id + id * id $	$E \rightarrow TE'$
$FT'E'\$$	id + id * id $	$T \rightarrow FT'$
~~id~~ $T'E'\$$	~~id~~ + id * id $	$F \rightarrow id$
ε $E'\$$	+ id * id $	$T' \rightarrow \varepsilon$.
~~+~~ $TE'\$$	~~+~~ id * id $	$E' \rightarrow +TE'$
$FT'E'\$$	id * id $	$T \rightarrow FT'$
~~id~~ $T'E'\$$	~~id~~ * id $	$F \rightarrow id$
~~*~~ $FT'E'\$$	~~*~~ id $	$T' \rightarrow *FT'$
~~id~~ $T'E'\$$	~~id~~ $	$F \rightarrow id$
ε-Null $E'\$$	$	$T' \rightarrow \varepsilon$
ε-Null $\$$	$	$E' \rightarrow \varepsilon$

$ $ Accepted

Therefore, the given grammar can be passed using LL(1) passer.

Step 5: Parse Tree:



id + id * id

Q) : check whether given grammar can be passed by LL(1) passer / table driven passer

$S \rightarrow iEtS / iEtSeS / a$

$E \rightarrow b$

string is 'eab'