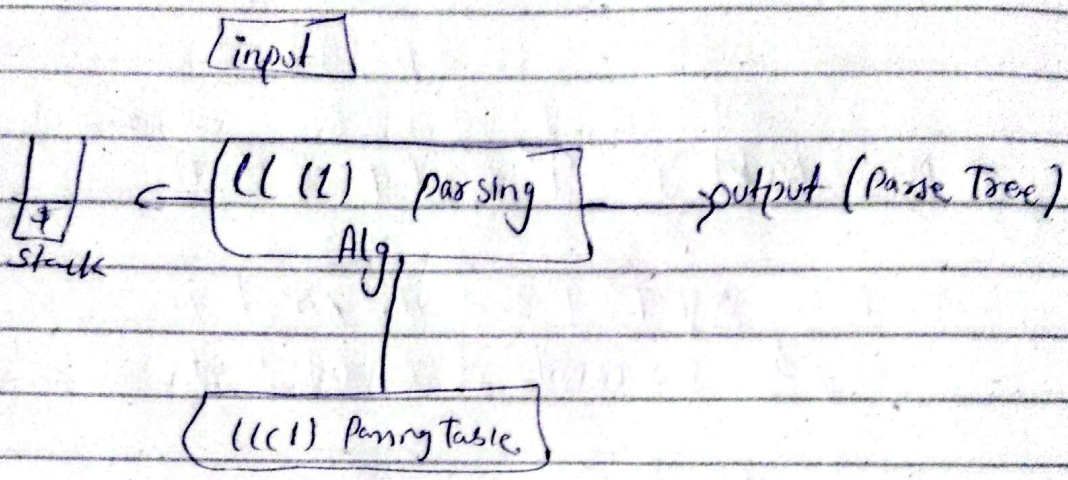
→ LL(1) Parser.

also known as Non Recursive Descent Parser. or predective parser.

Block diagram of LL1 passing



step1 Note that LL1 passing tree ka jo ~~grammar~~ table ~~hoti hai~~ ki size hoti hai wo grammar to grammar vary karti hai.

ex: Given grammar:

	first()	follow()
$E \to TE'$	{id, c}	{$,)}
$E' \to \varepsilon / +TE'$	{ε, +}	{$,)}
$T \to FT'$	{id, c}	{+, $,)}
$T' \to \varepsilon / *FT'$	{ε, *}	{+, $,)}
$F \to id / (E)$	{id, c}	{*, +, $,)}

step 2 count no of variable where i) S and there are S terminals aur ek column extra bana ga for $.

so

S x (S+1)

= 36

Page 6

so total 30 cells for that particular grammar.

1/ Rule 1

Add $A \to \alpha$ under $M[A, a]$ (table name, terminals, variable)

where $a \in$ First (α)

2/

Add $A \to \alpha$ under $M[A, \$]$

M	id	+	*	(	)	$
E	$E \to TE'$			$E \to TE'$		
E'		$E' \to +TE'$			$E' \to \varepsilon$	$E' \to \varepsilon$
T	$T \to FT'$			$T \to FT'$		
T'			$T' \to *FT'$		$T' \to \varepsilon$	$T' \to \varepsilon$
F	$F \to id$			$F \to (E)$		

like in first production rule

$E \to TE'$

(α, Means Find First of T which is (id, ( ) in the First() column. So write this production rule to there in E variable

→ Now second production Rule E', it's first is ε (Null) Null wali rule ko bilkul skip kar lo.

and its second production is: $+TE$ and it's first is +

So E' kai row kai + wali column mai us cfg ko rakh lo.

→ Now

$T \rightarrow FT'$

and it's first is:

{id, (}

So simple is ko {id, (} mai substitute kar lai.

→ Now Next is $T' \rightarrow \varepsilon$ skip ε (Null) and

next is ☆FT' and its first is.

☆ so write this on that

and Now Next is:

$F \rightarrow id / (E)$

so its first is id for first production rule.

Now ε (Null) which we skip first was

$E \rightarrow (E)$ so in Null jo left hand side mai jo us ka variable hai hum us kai follow likhy gai.

→ Note LL1 (Parser is the Top down parser)

Now parsing tree.

This is the variable:-

	Stack	Input	Production.
step1	$E	id+id $	E → TE′
step2	$E′T	id+ id $	T → FT′
step3	$E′T′F	id+id $	F → id
step4	$ E′T′id	id+id $	Pop
step5	$E′T′	+id $	T′ → ε
step6	E′	+id $	E′ → TE′
step6	E′T+	+id $	pop.
step6	E′T	id $	T → FT′
	$ E′T′F	id $	F → id
	$ E′T′id	id $	Pop
	$ E′T′	$	T′ → ε
	$ E′	$	E′ → ε
	$	$	Accept

→ step1 check the E (capital) ki id wadly column mai jo production parhi hai hum wo use kary gai.

jo hai E → TE′

→ step2 so Now variable khai asaan hin production rule asaan replace kandoeyi but reverse order mai which is E′T.

and again we have same input id+id $

step and we have Top of the stack is capital(T)

so write its productions which is

Now again which we use production rule make it rever and kee the preverse (E) as it is

so $ E′T′F

and again check what we have the top of the stack.? which is (f) and f k id mai hai F→id

Note:- Now we have Top of the stack and the input same aso simplify we will write the (pop) in the production rule and hum wo id remove k ar dai gai stack mai sai.

Like this.

$ E' T' aur

input mai sai bi hum remove kar dai gai.

So we will write

+id$.

So Now our input ~~is~~ pointer is +.

and Now will check T' kai pluss mai hamary pass hai T'→ε to T' remove ho jae gaa.

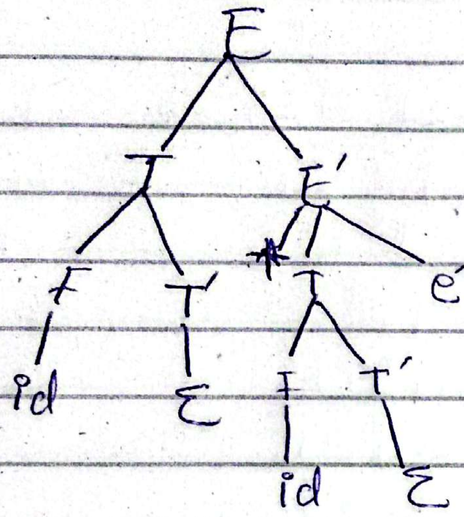
→ Now E' kai plus mai

is E'→TE'

we will write

→ Now lets construct the parse tree.

sol. Start from stack, jis mai first variable parha hai. wo hai E aur uski production hai aur next hai homogra.



Note LL1 parsing generate the left most Tree.