Predictive LL(1) Parser

- → This top-down parsing algorithm is of non-recursive type.
- → In this of parsing a table is built.
- → For LL(1) - the first L means the input is scanned from left to right. The second L means it uses leftmost derivation for input string. And the number (1) in the input symbol means it uses only one input symbol (lookahead) to predict the parsing process.

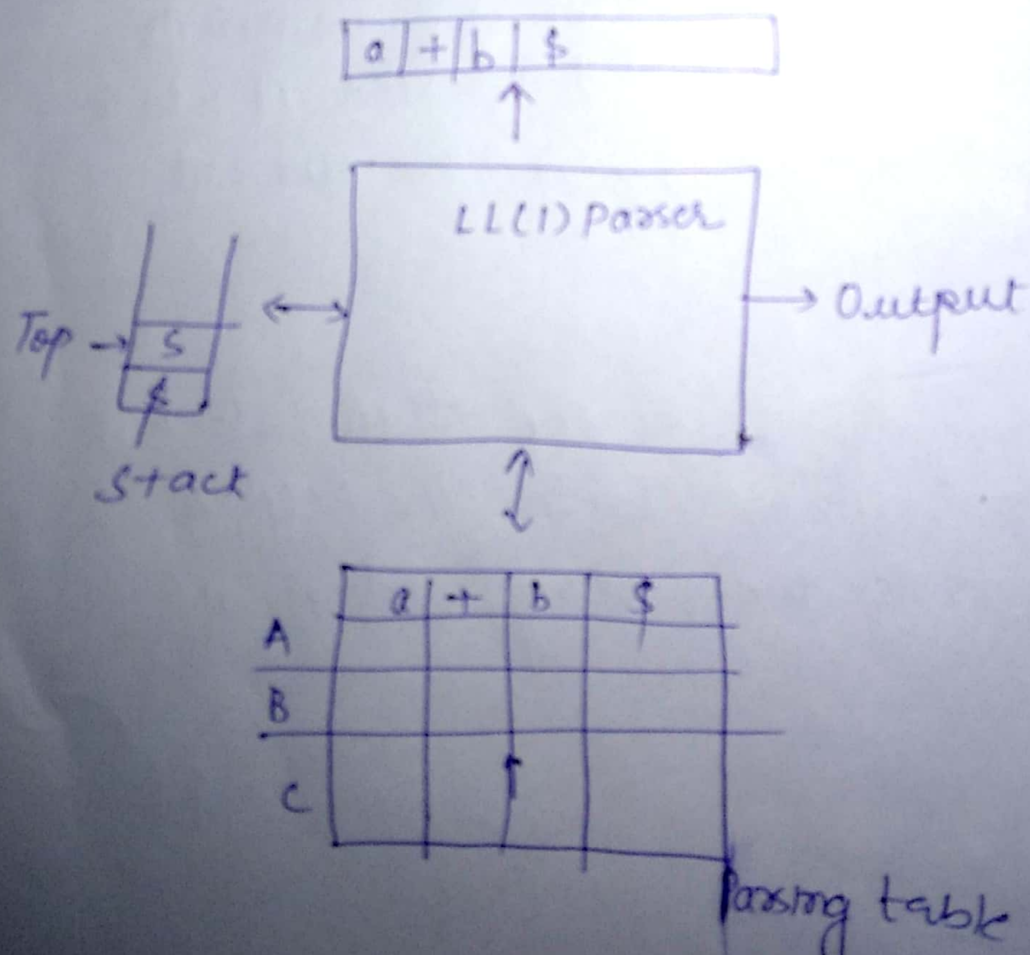

fig:- Model for LL(1) Parser

The data structures used by LL(1) are (1) input buffer (2) stack (3) parsing table. The LL(1) parser uses input buffer to store the input tokens.

→ The stack is used to hold the left sentential form. The symbols in R.H.S of rule are pushed into the stack in reverse order i.e from ~~left~~ ~~to~~ right to left. Thus use of stack makes this algorithm non-recursive.

→ The table is basically a 2-D array. The table has row for non-terminal & column for terminals. The table can be represented as M[A, a] where A is a non-terminal & a is the current input symbol.

→ The parsing program reads top of the stack & a current i/p symbol. With the help of these two symbols the parsing action is determined.

Construction of Predictive LL(1) Parser

Step ① Computation of FIRST & FOLLOW FUNCTION

Step 2 Construct the predictive parsing table using FIRST & FOLLOW functions

Step-3 Parse the input string with the help of Predictive Parsing table.

Algorithm for predictive Parsing table

Input: Context free Grammar G.

Output: Predictive Parsing table M.

Algorithm

For the rule $A \rightarrow \alpha$ of grammar G

1) For each a in FIRST(α) create entry $M[A, a] = A \rightarrow \alpha$ where a is terminal symbol.

2) for ϵ (→ epsilon) in FIRST(α) create $M[A, b] = A \rightarrow \alpha$ where b is the symbols from FOLLOW(A).

3) If ϵ is in FIRST(α) & \$ is in FOLLOW(A) then create ~~tab~~ entry in table $M[A, \$] = A \rightarrow \alpha$

(4) All the remaining entries are marked as error.

Q Create the parsing table for the above grammar.

$E \to TE'$

$E' \to +TE' / \epsilon$

$T \to FT'$

$T' \to *FT' / \epsilon$

$F \to (E) / id$

Sol First check for left recursion & left factoring

There is no left recursion & no left factoring required

Now Calculate first & follow

~~First~~ FIRST (E) = FIRST (T) = FIRST(F)

~~As~~

FIRST(F) = { (, id }

FIRST(E') = { +, ε }

FIRST (T) = FIRST (F) = { (, id }

FIRST (T') = { *, ε }

∴ FIRST(E) = { (, id }

Now calculate FOLLOW

$FOLLOW(E) = \{ \$,) \}$

As E is the starting symbol so we add \$ in follow (E).

$$FOLLOW(E') = FOLLOW(E) \cup FOLLOW(E')$$
$$= FOLLOW(E)$$
$$= \{ \$,) \}$$

~~FOLLOW(T') = FOLLOW(T) ∪ FOLLOW(T')~~

~~= FOLLOW(T)~~

$$FOLLOW(T) = \{ FIRST(E') - \epsilon \} \cup FOLLOW(E) \cup FOLLOW(E')$$
$$= \{ + \} \cup \{ \$,) \} \cup \{ \$,) \}$$
$$= \{ +, \$,) \}$$

$$FOLLOW(T') = FOLLOW(T) \cup FOLLOW(T')$$
$$= Follow(T)$$
$$= \{ \$,) \}$$

$$FOLLOW(F) = \{ FIRST(T') - \epsilon \} \cup Follow(T') \cup Follow(T)$$
$$= \{ * \} \cup \{ \$,) \} \cup \{ +, \$,) \}$$
$$= \{ *, +, \$,) \}$$

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

Now first create the table

	id	+	*	(	)	$
E						
E'						
T						
T'						
F						

Now consider each rule one by one to fill the parsing table.

$$E \rightarrow TE'$$

according to Parsing table algo we map the above the grammar with the rul $A \rightarrow \alpha$

~~$A \rightarrow \alpha$~~ $A = E$, $\alpha = TE'$

$FIRST(TE') = FIRST(T) = \{ (, id \}$

$M[E, (] = E \rightarrow TE'$ & $M[E, id] = E \rightarrow TE'$

	id	+	*	(	)	$
E	$E \to TE'$	error	error	$E \to TE'$	error	error
E'	error	$E' \to +TE'$	error	error	$E' \to \epsilon$	$E' \to \epsilon$
T	$T \to FT'$	error	error	$T \to FT'$	error	error
T'	error	$T' \to \epsilon$	$T' \to *FT'$	error	$T' \to \epsilon$	$T' \to \epsilon$
F	$F \to id$	error	error	$\cancel{T \to FT'}$ $F \to (E)$	error	error

Now (1) $E' \to +TE'$

Map the grammar using the rule $A \to \alpha$

$A = E'$

$\alpha = +TE'$

$FIRST(\alpha)$ i.e $FIRST(+TE') = \{+\}$

$\therefore \quad M[E', +] = E' \to +TE'$

Now (2) $E' \to \epsilon$

Here according to rule $A \to \alpha$

$A = E'$

$\alpha = \epsilon$

then $FOLLOW(E') = \{), \$ \}$

$M[E',)] = E' \to \epsilon$

$M[E', \$] = E' \to \epsilon$

Now

(3) $T \to FT'$

Map it according to rule $A \to \alpha$

$A = T$

$\alpha = FT'$

Now $FIRST(FT') = FIRST(F) = \{(, id\}$

Hence $M[T, (] = T \to FT'$

$M[T, id] = T \to FT'$

(4) $T' \to *FT'$

Map it according to rule $A \to \alpha$

$A = T'$

$\alpha = *FT'$

$FIRST(\alpha) = FIRST(*FT')$

$= FIRST(*)$

$= *$

$\therefore M[T', *] = T' \to *FT'$

(5) $T' \to \epsilon$

Here

$A = T'$

$\alpha = \epsilon$

$\therefore FOLLOW(T') = \{+,), \$\}$

$M[T', +] = T' \to \epsilon$

$M[T',)] = T' \to \epsilon$

$M[T', \$] = T' \to \epsilon$

(6) $F \to (E)$

Map it according to the rule $A \to \alpha$

$A = F$

$\alpha = (E)$

$FIRST(\alpha) = \cancel{(E)}\ FIRST((E))$

$= \{ (\}$

$\therefore M[\cancel{F}, (] = F \to (E)$

(7) $F \to id$

Map it according to rule $A \to \alpha$.

$A = F$

$\alpha = id$

$FIRST(\alpha) = FIRST(id)$

$= id$

$\therefore M[F, id] = F \to id$