PRACTICAL-10

AIM : Write a document to understand working of LL(1) Parsing.

INTRODUCTION:

- This top-down parsing algorithm is of non-recursive type. In this type of parsing 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.
- The data structures used by LL(1) are:
  I. Input buffer.
  II. Stack.
  III. 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 RHS of rule are pushed into the stack in reverse order i.e. from right to left. Thus use of stack makes this algorithm non-recursive.
- The table is basically a two dimensional array. The table has row for non-terminal and column for terminals. The table can be represented as M[A,a] where A is a non-terminal and a is current input symbol.

WORKING:

- The parsing program reads top of the stack and a current input symbol. With the help of these two symbols parsing action is determined. The parsing actions can be push, pop and error.
- The parser consults table M[A,a] each time while taking the parsing actions hence this type of parsing method is called table driven parsing algorithm.
- The configuration of LL(1) parser can be defined by top of the stack and a lookahead token.
- One by one configuration is performed and the input is successfully parsed if the parser reaches the halting configuration.
- When the stack is empty and next token is $ then it corresponds to successful parse.

CONSTRUCTION OF PREDICTIVE LL(1) PARSER:

- The construction of predictive LL(1) parser is based on two important functions and those are FIRST and FOLLOW. For construction of Predictive LL(1) parser we have to follow the following steps-
  Step-1: Computation of FIRST and FOLLOW function.
  Step-2:Construct the Predictive Parsing Table using FIRST and FOLLOW functions.
  Step-3:Parse the input string with the help of Predictive Parsing Table.

FIRST FUNCTION:

- FIRST(α) is a set of terminal symbols that are first symbols appearing at RHS in derivation of α.

FOLLOW FUNCTION:

- FOLLOW(A) is defined as the set of terminal symbols that appear immediately to the right of A. In other words FOLLOW(A) = { a | S ⇒ α Aa β where α and β are some grammar symbols may be terminal or non-terminal.}

EXAMPLE:

Consider the grammar:

E → TE’

E’ → +TE’ | e

T → FT’

T’ → *FT’ | e

F → (E) | id

Where e is epsilon

First we will compute FIRST and FOLLOW of given grammar,

Symbols	FIRST	FOLLOW
E	{ (,id }	{),$ }
E’	{ +,e}	{),$ }
T	{ (,id }	{),$,+}
T’	{ *,e }	{),$,+}
F	{ (,id }	{),$,+,*}

Now , we will construct LL(1) parsing table

	id	+	*	(	)	$
E	E→TE’			E→TE’		
E’		E’→+TE’			E’→e	E’→e
T	T→FT’			T→FT’		
T’		T’→e	T’→*FT’		T’→e	T’→e
F	F→id			F→(E)		

Here, in given parsing table no two cells consists of two values hence given grammar is LL(1).

For parsing the string id+id*id using above table.

Stack	Input	Action
$E’T	id+id*id$	E→TE’
$E’T’F	id+id*id$	E→FT’

$E’T’id	id+id*id$	F→id
$E’T’	+id*id$	
$E’	+id*id$	T’→e
$E’T+	+id*id$	E’→+TE’
$E’T	id*id$	
$E’T’F	id*id$	T→FT’
$E’T’id	id*id$	F→id
$E’T’	*id$	
$E’T’F*	*id$	T→*FT’
$E’T’F	id$	
$E’T’id	id$	F→id
$E’T’	$	T→e
$E’	$	
$	$	E’→e

Hence, given input string also got parsed and thus given grammar is LL(1).