LR Grammars: -

A Grammar for which a parsing table can be constructed and for which every entry is uniquely defined is said to be an LR Grammar.

All CFG's are not a LR Grammar.

Closure Operation:-

If I is a set of items for a grammar G then Closure (I) is the set of items constructed from I by the two rules.

1. Initially every item in I is added to closure (I) .
2. If A→α **.**Bβ is in closure(I) and B→γ is a production, then add the item B→ **.** γ to I, if it is not already there.

Augmented Grammar:-

If G is a grammar with start symbol S, then G´, the augmented grammar for G, in G with a new start symbol S´ and production S´→S

Ex:-1. Consider the grammar given below,

E→E+T / T
T→T*F / F
F→(E) / id

Construct an LR Parsing table for the above grammar.

Solution:-

(i) Elimination left Recursion

E→TE´
E´ →+T E´/ε

T→FT´

T´ →*F T´/ ε

F→(E)/id

(ii) Finding FIRST and FOLLOW:-

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

FIRST(E´) = { + , ε}

FIRST(T´) = { * , ε}

FOLLOW(E) = {) , $ }

FOLLOW(E´) = FOLLOW(E) = {) , $ }

FOLLOW(T) = FIRST(E´) = { + , ε} + FOLLOW(E´)

= { + ,) ,$ }

FOLLOW(T´) = FOLLOW(T) = { + ,) ,$ }

FOLLOW(F) = FIRST(T´) = { * , ε} + FOLLOW(T´) = { *, + ,) ,$ }

(iii) Numbering the Grammar:-

1. $E \rightarrow E+T$
2. $E \rightarrow T$
3. $T \rightarrow T*F$
4. $T \rightarrow F$
5. $F \rightarrow (E)$
6. $F \rightarrow id$

<u>Augmented Grammar</u>

$E' \rightarrow E$

$E \rightarrow E+T$

$E \rightarrow T$

$T \rightarrow T*F$

$T \rightarrow F$

$F \rightarrow (E)$

$F \rightarrow id$

<u>Closure (I´)</u>

$$\left.\begin{array}{l} E' \rightarrow .E \\ E \rightarrow .E+T \\ E \rightarrow .T \\ T \rightarrow .T*F \\ T \rightarrow .F \\ F \rightarrow .(E) \\ F \rightarrow .id \end{array}\right\} I_0$$

<u>GO TO(I_0, E)</u>

$$\left.\begin{array}{l} E' \rightarrow E. \\ E \rightarrow E.+T \end{array}\right\} I_1$$

<u>GO TO (I_0, T)</u>

$$\left.\begin{array}{l} E \rightarrow T. \\ T \rightarrow T.*F \end{array}\right\} I_2$$

<u>GO TO(I_0, F)</u>

$T \rightarrow F.$ $\quad I_3$

GO TO(I_0, ( )

F→(.E)
E→.E+T
E→.T
T→.T*F
T→.F
F→.(E)
F→.id

} I_4

GO TO(I_0, id)

F→id . I_5

GO TO(I_1, +)

E→E+.T
T→.T*F
T→.F
F→.(E)
F→.id

} I_6

GO TO(I_2, *)

T→T*.F
F→.(E)
F→.id

} I_7

GO TO(I_4, E)

F→(E.)
E→.E+T

} I_8

GO TO(I_4, T)

E→T.
T→T.*F

} I_2

GO TO(I_4, F)

T→F. I_3

GO TO(I_4, ( )

F→(.E)
E→.E+T
E→.T
T→.T*F
T→.F
F→.(E)
F→.id

} I_4

GO TO(I_4, id)

$F \rightarrow id .$ I_5

GO TO(I_6, T)

$E \rightarrow E+T.$
$T \rightarrow T.*F$
} I_9

GO TO(I_6, F)

$T \rightarrow F.$ I_3

GO TO(I_6, ( )

$F \rightarrow (.E)$
$E \rightarrow .E+T$
$E \rightarrow .T$
$T \rightarrow .T*F$
$T \rightarrow .F$
$F \rightarrow .(E)$
$F \rightarrow .id$
} I_4

GO TO(I_6, id)

$F \rightarrow id .$ I_5

GO TO(I_7, F)

$T \rightarrow T*F.$ I_{10}

GO TO(I_7, ( )

$F \rightarrow (.E)$
$E \rightarrow .E+T$
$E \rightarrow .T$
$T \rightarrow .T*F$
$T \rightarrow .F$
$F \rightarrow .(E)$
$F \rightarrow .id$
} I_4

GO TO(I_7, id)

$F \rightarrow id .$ I_5

GO TO(I_8,))

$F \rightarrow (E) .$ I_{11}

GO TO(I_8, +)

$E \rightarrow E+.T$
$T \rightarrow .T*F$
$T \rightarrow .F$
$F \rightarrow .(E)$
$F \rightarrow .id$
} I_6

GO TO(I_9, *)

T→T*.F
F→.(E)   } I_7
F→.id

Reduce:-

E→T. (I_2)

ACTION(2,FOLLOW(E)) = (2,)), (2,$) → r_2

T→F. (I_3)

ACTION(3,FOLLOW(T)) = (3,+),(3,)), (3,$) → r_4

F→id. (I_5)

ACTION(5,FOLLOW(F)) = (5,*), (5,+), (5,)), (5,$) → r_6

E→E+T. (I_9)

ACTION(9,FOLLOW(E)) = (9,*), (9,$) → r_1

T→T*F. (I_{10})

ACTION(10,FOLLOW(T)) = (10,+), (10,)), (10,$) → r_3

F→(E). (I_{11})

ACTION(11,FOLLOW(F)) = (11,*), (11,+), (11,)), (11,$) → r_5

State	ACTION						GOTO		
	+	*	(	)	id	$	E	T	F
0			S_4		S_5		1	2	3
1	S_6					acc			
2		S_7		r_2		r_2			
3	r_4			r_4		r_4			
4			S_4		S_5		8	2	3
5	r_6	r_6		r_6		r_6			
6			S_4		S_5			9	3
7			S_4		S_5				10
8	S_6			S_{11}					
9		S_7		r_1		r_1			
10	r_3			r_3		r_3			
11	r_5	r_5		r_5		r_5			

Ex:-2. Consider the grammar given below,

S→CC

C→cC / d

Construct a CLR Parsing table for the above grammar.

Solution:-

(i) Elimination left Recursion

S→CC

C→cC / d

(ii) Finding FIRST and FOLLOW:-

FIRST(S)= FIRST(C) = { c , d }

FOLLOW(S) = { $ }

FOLLOW(C) = FIRST(C) = { c,d , $ }

(iii) Numbering the Grammar:-

1. S→CC
2. C→cC
3. C→d

Augmented Grammar

S´→S
S→CC
C→cC
C→d

} I´

Closure (I´)

S´→ ·S,$
S→ ·CC,$
C→ · cC,c/d
C→ ·d,c/d

} I_0

GOTO (I_0,S)

S´→ S ·,$ I_1

GOTO (I_0,C)

S→C·C, $
C→·cC, $
C→·d, $

} I_2

GOTO (I_0,c)

C→c·C, c/d
C→· cC, c/d
C→·d, c/d

GOTO (I_0,d)

C→d ·, c/d I_4

GOTO (I_2,C)

S→ CC ·,$ I_5

GOTO (I_2,c)

C→c·C, $
C→· cC, $ } I_6
C→·d, $

GOTO (I_2,d)

C→d ·, $ I_7

GOTO (I_3,C)

C→ cC ·,c/d I_8

GOTO (I_3,c)

C→c·C, c/d
C→· cC, c/d   } I_3
C→·d, c/d

GOTO (I_3,d)

C→d ·, c/d I_4

GOTO (I_6,C)

C→ cC ·,$ I_9

GOTO (I_6,c)

C→c·C, $
C→· cC, $ } I_6
C→·d, $

GOTO (I_6,d)

C→d ·, $ I_7

Reduce:-

C→d., c/d   (I_4)
ACTION(4,c/d) = (4,c), (4,d) → r_3

S→CC. , $ (I_5)
ACTION(5,$) = (5,$) → r_1

C→d., c/d   (I_4)
ACTION(4,c/d) = (4,c), (4,d) → r_3

$C \rightarrow cC.$, c/d (I_8)
ACTION(8,c/d) = (8,c) , (8,d) $\rightarrow r_2$

$C \rightarrow cC.$, \$ (I_8)
ACTION(9,\$) = (9,\$) $\rightarrow r_2$

Reference

1. J. Tremblay, P.G. Sorenson,"The Theory and Practice of Compiler Writing ", McGRAW-HILL,1985.
2. W.M. Waite, L.R. Carter,"An Introduction to Compiler Construction",Harper Collins,New york,1993
3. A.W. Appel,"Modern Compiler Implementation in, CambridgeUniversity Press,1998
4. Internet Papers
5. Aho, R. Sethi, J.D. Ullman," Compilers- Principles, Techniques and Tools"Addison-Weseley, 2007