Given grammar G is :

1) E 🡪 E + T | T

2) T 🡪 T \* F | F

3) F 🡪 ( E ) | id

After elimination of Left Recursion to the above grammar

1. E 🡪 TE’
2. E’ 🡪 + TE’ | ε
3. T 🡪 FT’
4. T’ 🡪\*FT’ | ε
5. F 🡪 ( E )
6. F 🡪 id

FIRST ( E ) = { ( ,id } M[A,a]=A🡪α

FIRST ( E’ )= { + , ε }

FIRST ( T ) = { ( , id }

FIRST ( T’) = { \* , ε }

FIRST ( F ) = { (, id }

1. FOLLOW( E )

F 🡪 ( E )

A🡪 α B β

A= F , α= ( , B= E , β = )

FIRST(β) = FIRST ( ‘)’ )

= { ) }

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

1. FOLLOW ( T )

E 🡪 T E’

A🡪 α B β

A= E , α = ε , B = T , β= E’

FIRST ( E’ ) = FIRST( +TE’)

= { + }

FIRST ( E’ ) = FIRST ( ε )

= { ε }

FIRST ( E’ ) = { + , ε }

FOLLOW( T ) = { + , ε }

FOLLOW ( E ) = { ) , $ }

FOLLOW( T )= FOLLOW( E ) + FIRST(β)non epsi

FOLLOW(T) = { ) , $ } + { + }

= { + , ) , $ }

FOLLOW (E) = { ) , $}

FOLLOW(T) = { + , ) , $ }

FOLLOW(F)= {+, \* , ) , $ }

FOLLOW(E’) = { ) , $ }

FOLLOW(T’)= { + , ) , $ }

E🡪TE’ FIRST (E ) ={(,id}

M[E,(]= E🡪TE’

M[E,id]= E🡪TE’

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | id | + | \* | ( | ) | $ |
| E | E🡪TE’ |  |  | E🡪TE’ |  |  |
| E’ |  |  |  |  |  |  |
| T | T🡪FT’ |  |  | T🡪FT’ |  |  |
| T’ |  |  |  |  |  |  |
| F | F🡪id |  |  | F🡪(E) |  |  |

**FOLLOW ( X )** where X a non terminal

1,Place $ in the result of FOLLOW( B ) if B is the start symbol in the given Grammar.

2, if the production is in the form of A🡪αBβ then find FIRST(β) , assign the result of FIRST(β) to FOLLOW(B)

3,if the production is in the form of A🡪αB then find FOLLOW(A) add this result to FOLLOW(B).

4.if the production is in the form of A🡪αBβ then find FIRST(β), if FIRST(β) contains epsilon symbol then find FOLLOW(A)

FOLLOW (B)= FOLLOW(A) + FIRST(β)(non epsilon symbols)

S → aBDh

B → cC

C → bC / ∈

D → EF

E → g / ∈

F → f / ∈

FIRST(S)= { a }

FIRST(B)= { c }

FIRST(C) = { b, ∈}

FIRST(D)= {g, ∈}

FIRST( E ) ={ g, ∈}

FIRST( F ) = {f, ∈}

FOLLOW (S) = { $ }

FOLLOW(B) = { }

S → aBDh

**A🡪αBβ**

**FIRST(β) = FIRST(Dh)= FIRST(D)**

**=** {g, ∈}

FOLLOW( A )= FOLLOW(S) ={ $ }

FOLLOW(B) ={ $ } + {g}

={g, $}

FOLLOW( C ) :

B → cC

A 🡪 **α B , A=B, α=c, B=C**

**Find FOLLOW (A)**

FOLLOW (B) ={g, $}

**FOLLOW(C) = {g, $}**

FOLLOW(D) :

S → aBDh

**A🡪α B β**

A=S, B=D, **α=aB , β= h**

**Find FIRST (β ) = FIRST (h)**

**= { h }**

**FOLLOW ( D ) = { h }**

FOLLOW ( E ) =

FOLLOW( F ) =

Algorithm 4.31 : Construction of a predictive parsing table.

INPUT: Grammar G.

OUTPUT: Parsing table M.

METHOD: For each production A ! \_ of the grammar, do the following:

1. For each terminal a in FIRST(\_), add A ! \_ to M[A; a].

2. If \_ is in FIRST(\_), then for each terminal b in FOLLOW(A), add A ! \_

to M[A; b]. If \_ is in FIRST(\_) and $ is in FOLLOW(A), add A ! \_ to

M[A; $] as well.

**Stack Input Output**

$ id\*id+id$

$E id\*id+id$ E🡪TE’

$E’T id\*id+id$ T🡪FT’

$E’T’F id\*id+id$ F🡪id

$E’T’id id\*id+id$ pop id stack

$E’T’ \*id+id$ T’🡪\*FT’

$E’T’F\* \*id+id$ pop \* from stack

$E’T’F id+id$ F🡪id

$E’T’id id+id$ pop id from stack

$E’T’ +id$ T’🡪ε

$E’ +id$ E’🡪+TE’

$E’T+ +id$ pop + from stack

$E’T id$ T🡪FT’

$E’T’F id$ F🡪id

$E’T’id id$ pop id from stack

$E’T’ $ T’🡪 ε

$E’ $ E’🡪 ε

$ $ ACCEPT

**LL(1) GRAMMER or NOT**

**Ex:**

Grammer G :

S🡪iEtS | iEtSeS |a

E🡪b

1. Eliminate left factoring from the above grammar

S🡪iEtSS’ | a

S’🡪eS | ε

E🡪b

1. Find FIRST()

FIRST(S)= { i, a }

FIRST(S’)= { e , ε }

FIRST(E) = { b }

1. Find FOLLOW()

FOLLOW(S) =

S🡪iEt**S**S’

A🡪α B β then find FIRST(β)

A=S , α = iEt , B= S , β = S’

FIRST(S’) = FIRST( eS )

= FIRST( e ) ={ e }

= FIRST(ε ) = { C}

FIRST(S’)= {e, ε }

Because if epsilon in the FRIST(β) I should find FOLLOW(A)

FOLLOW(A) = FOLLOW(S)

FOLLOW(S) =

S’🡪eS

**SHIFT REDUCE PARSER**

Grammar G :

E🡪E+T | T

T🡪T\*F | F

F🡪( E ) | id

Input string : **id \* id**

**STACK INPUT OUTPUT**

$ id \*id $ shift

$id \*id$ reduce

$F \*id$ reduce

$T\* id$ shift

$T\*id $ shift

$T\*F $ reduce

$ T $ reduce

$ E $ **accept**

**EXAMPLE-2**

S 🡪 S S + | S S \* | a

Input string : 1) SSS + a \*+ 2) aaa \*a++.

**STACK INPUT OUTPUT**

**$ S**SS+a\*+$ shift

$S SS+a\*+$ shift

$SS S+a\*+$ shift

$SSS +a\*+$ shift

$S**SS+** a\*+$ reduce

$SS a\*+$ shift

$SS**a** \*+$ reduce

$SSS \*+$ shift

$S**SS\*** +$ reduce

$SS +$ shift

$**SS+** $ reduce

$S $ **accept**

**EXAMPLE-3**

S → (L) | a

L → L ,S | S

Input String : ( a , ( a , a ) )

Solution :

**Example -4**

S → T L ;

T → int | float

L → L , id | id

Input string : **int id,id;**

**STACK INPUT OUTPUT**

$ int id,id;$ shift

**SLR PARSING TECHNQUE**

Grammar :

1,E🡪E+T

2,E🡪T

3,T🡪T\*F

4,T🡪F

5,F🡪( E )

6,F🡪id

Input string : id + id \* id

STEP 1: Prepare Augumented Grammar

**E’🡪E**

E🡪E+T

E🡪T

T🡪T\*F Augumented Grammar

T🡪F

F🡪( E )

F🡪id

STEP 2: Place dot at starting of the right sentential of the grammar

E’🡪.E

E🡪.E+T

E🡪.T

T🡪.T\*F **I0**

T🡪.F

F🡪.( E )

F🡪.id

**STEP 4 :**

APPLY GOTO() on I0 i.e goto(I0, E)

Before goto() the productions are

E’🡪.E

E🡪.E+T

After goto() the productions are

E’🡪E.

E🡪E.+T

Apply Closure() on goto(I0, E)

**Closure(goto(I0, E))**

Closure( E’🡪E. E🡪E.+T )

Closure( E’🡪E.)

E’🡪E.

Closure(E🡪E.+T)

E🡪E.+T

E’🡪E. **I1**

E🡪E.+T

Goto(I0,T)

E🡪.T

T🡪.T\*F

After goto(I0,T)

E🡪T.

T🡪T.\*F

Closure(goto(I0,T)

Closure(E🡪T.)

E🡪T.

Closure(T🡪T.\*F)

T🡪T.\*F

E🡪T.  **I2**

T🡪T.\*F

Goto(I0,F)

T🡪.F

After goto(I0,F)

T🡪F.

Closure(goto(I0,F))

T🡪 F. **I3**

Goto(I0, ‘(‘ )

Before goto()

F🡪.(E)

After goto()

F🡪(.E)

Closure(Goto(I0, ‘(‘ ) )

F🡪(.E)

E🡪.E+T

E🡪.T **I4**

T🡪.T\*F

T🡪.F

F🡪.(E)

F🡪.id

Goto(I0,id)

F🡪id.

Closure(goto(I0,id))

F🡪id. I5

Goto( I1, + ) where E🡪E.+ T

After goto()

E🡪E+.T

Closure(goto( I1, + ))

Closure(E🡪E+.T)

E🡪E+.T

T🡪.T\*F

T🡪.F I6

F🡪.(E )

F🡪.id

Goto(I2 ,\* )

T🡪T\*.F

Closure(goto(I2 ,\* ))

Closure(T🡪T\*.F)

T🡪T\*.F

F🡪.(E) **I7**

F🡪 .id

Goto(I4,E )

F🡪(.E)

E🡪.E+T

After goto()

F🡪(E.)

E🡪E.+T

Closure(goto(I4,E )

Closure(F🡪(E.) , E🡪E.+T)

F🡪(E.)

E🡪E.+T **I8**

Goto(I4,T)

E🡪.T

T🡪.T\*F

After goto()

E🡪T.

T🡪T.\*F

Closure(Goto(I4,T))

E🡪T. I2

T🡪T.\*F

Goto(I4,F)

T🡪.F

After goto()

T🡪F.

Closure(Goto(I4,F))

T🡪F. I3

Goto(I4,’(‘)

F🡪.( E )

After goto()

F🡪(.E )

Closure(Goto(I4,’(‘))

F🡪(.E )

E🡪.E+T

E🡪.T I4

T🡪.T\*F

T🡪.F

F🡪.( E )

F🡪.id

Goto(I4,id)

---------------------------------🡪 I5

Goto(I6,T)

E🡪E+.T

T🡪.T\*F

After goto()

E🡪E+T.

T🡪T .\*F

Apply closure on Goto(I6,T)

Closure( Goto(I6,T) )

E🡪E+T.

T🡪T .\*F **I9**

Goto(I6,F )

T🡪F.

Closure(Goto(I6,F ))

T🡪F. -------🡪 I3

Goto(I6, ‘(‘ )

Closure(Goto(I6, ‘(‘ ))

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-🡪 I4

Goto(Goto(I6,id)

Clousure(Goto(I6,id))

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_--> I5

Goto(I7 ,F )

T🡪T\*.F

After goto()

T🡪T \* F.

Closure(Goto(I7 ,F ))

Closure(T🡪T \* F.)

T🡪T \* F. --------------------------🡪 **I10**

Goto(I7 ,’(‘ )

Closure(Goto(I7 ,’(‘ )

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_--🡪 I4

Goto(I7 ,’id‘ )

Closure(Goto(I7 ,id )

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ -🡪 I5

Goto(I8,’)’)

F🡪(E.)

After goto()

F🡪( E ).

Closure(Goto(I8,’)’)

F🡪( E ). ---------------------------🡪 **I11**

Goto(I8, +)

E🡪E . + T

After goto()

E🡪E+.T

Closure(Goto(I8, +))

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-🡪I6

Goto(I9,\*)

Closure(Goto(I9,\*)) --------------------🡪 I7

Goto( I0 ,E) = I1

Goto( I0 ,T) = I2

Goto( I0, F) = I3

Goto( I0, ‘(‘) = I4

Goto( I0,id ) = I5

Goto( I1, + ) = I6

Goto(I2 ,\* ) = I7

Goto(I4 ,E ) = I8

Goto(I4,T) = I2

Goto(I4,F) = I3

Goto(I4,’(‘ )=I4

Goto(I4,id ) = I5

Goto(I6,T) = I9

Goto(I6,F ) = I3

Goto(I6,’(‘) =I4

Goto(I6,id ) = I5

Goto(I7 ,F ) =I10

Goto(I7 ,’(‘)=I4

Goto(I7,id)=I5

Goto(I8,’)’) = I11

Goto(I8,+)= I6

Goto(I9,\*) = I7

CONSTRUCT FINITE AUTOMATA

0 E 1 + 6

T

2 \* 7

F 3

( 4 E I8

Id 5

Table consists of 2 parts

1. Action -----------🡪 a) shift b) reduce c)accept d) error
2. Goto

SLR PARSING TABLE

ACTION GOTO

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | + | \* | ( | ) | Id | | $ | | E | | T | | F |
| 0 |  |  | S4 |  | S5 | |  | | 1 | | 2 | | 3 |
| 1 | S6 |  |  |  |  | | Acc | |  | |  | |  |
| 2 | r2 | S7 |  | r2 |  | | r2 | |  | |  | |  |
| 3 | r4 | r4 |  | r4 |  | | r4 | |  | |  | |  |
| 4 |  |  | S4 |  | S5 | |  | | 8 | | 2 | | 3 |
| 5 | r6 | r6 |  | r6 |  | | r6 | |  | |  | |  |
| 6 |  |  | S4 |  | S5 | |  | |  | | 9 | | 3 |
| 7 |  |  | S4 |  | S5 | |  | |  | |  | | 10 |
| 8 | S6 |  |  | S11 |  | |  | |  | |  | |  |
| 9 | r1 | S7 |  | r1 |  | | r1 | |  | |  | |  |
| 10 | r3 | r3 |  | r3 |  | r3 | |  | |  | |
| 11 | r5 | r5 |  | r5 |  | | r5 | |  | |  | |  |

E🡪E+T FIRST(E) =( ‘(‘,id )

E🡪T FIRST(T) =( ‘(‘,id )

T🡪T\*F FIRST(F) =( ‘(‘,id )

T🡪F FOLLOW(E) =

F🡪( E ) E🡪E+T

F🡪id A🡪αBβ

FIRST(β)=FIRST(+T)

=FIRST(+)

={+}

F🡪( E )

A🡪αBβ

FIRST(β)=FIRST( ‘)’ )

={ ) }

**FOLLOW(E ) ={+,),$}**

**FOLLOW(T) ={ +,\*,),$}**

**FOLLOW(F) ={ +,\*,),$}**

Stack implementation || SLR Parsing

|  |  |  |  |
| --- | --- | --- | --- |
| **Stack** | **Symbol** | **Input** | **Action** |
| 0  0 5  0 3  0 2  0 1  0 1 6  0 1 6 5  0 1 6 3  0 1 6 9  0 1 6 9 7  0 1 6 9 7 5  0 1 6 9 7 10  0 1 6 9  0 1 | Id  F  T  E  E +  E + id  E + F  E + T  E + T \*  E + T \* id  E + **T \* F**  **E + T**  E | **id**+id\*id$  **+** id \* id $  + id \* id $  + id \* id $  + id \* id $  id \* id $  \*id $  \*id $  \*id $  id $  $  $  $  $ | (S5) shift  (r6) reduce  (r4) reduce  (r2) reduce  (S6) shift  (S5) shift  (r6) reduce  (r4) reduce  (S7) shift  (S5) shift  (r6) reduce  (r3) reduce  (r1) reduce  **acc** |

**CLR(1)**

**Canonical LR(1)**

----------------------------------------------------------------------------

Example

Grammar G :

S🡪CC

C🡪cC

C🡪d

Input string **ccd**

Step1 : Augmented grammar

S’🡪.S,$

**// \*\*\* If the production is A🡪α.Bβ,a**

**Then add B productions i.e B🡪.γ,b where b is in FIRST(βa) \*\* //**

Apply closure( S’🡪**.**S,$ )

**A🡪α.Bβ,a**

A=S’ , α=ε , B= S , β= ε, a=$

S’🡪.S ,$ b=FIRST(βa)=FIRST(ε$)=$

S🡪.CC,$

C🡪.cC,c/d **I0**A=S, α=ε,B=C, β=C,a=$

C🡪.d,c/d b=FIRST(βa)=FIRST(C$)

=FIRST( C)

={c,d}

Goto(I0,S)

S’🡪S. ,$

Closure(Goto(I0,S))

S’🡪S. ,$ -------------------- **I1**

Goto(I0,C)

S🡪C.C,$

Closure(Goto(I0,C))

S🡪C.C,$

C🡪.cC,$ **I2**

C🡪.d, $

Goto(I0,c)

C🡪c.C,c/d

Closure(Goto(I0,c))

Closure(C🡪c.C,c/d)

C🡪c.C,c/d

C🡪.cC, c/d **I3**

C🡪.d, c/d

Goto(I0,d)

C🡪d.,c/d

Closure(Goto(I0,d))

Closure(C🡪d.,c/d )

C🡪d.,c/d ----------------- **I4**

Goto(I2,C)

S🡪CC. ,$

Closure(Goto(I2,C))

S🡪CC. ,$ --------------------- **I5**

Goto(I2,c)

C🡪c.C,$

Closure(Goto(I2,c))

Closure(C🡪c.C,$)

C🡪c.C,$

C🡪.cC, $ **I6**

C🡪.d, $

Goto(I2,d)

C🡪d., $

Closure(goto(I2,d))

Closure(C🡪d., $)

C🡪d., $ --------- **I7**

Goto(I3 ,C)

C🡪cC.,c/d

Closure(Goto(I3 ,C) )

Closure(C🡪cC.,c/d )

C🡪cC.,c/d ---------- **I8**

Goto(I3 ,c)

C🡪c.C, c/d

Closure(Goto(I3 ,c))

Closure(C🡪c.C, c/d )

C🡪c.C, c/d

C🡪 .cC, c/d I3

C🡪.d , c/d

Goto(I3 ,d)

C🡪d., c/d

Closure(Goto(I3 ,d) )

Closure(C🡪d., c/d )

C🡪d., c/d ------------ I4

Goto(I6 ,C)

C🡪cC.,$

Closure(goto(I6 ,C)

Closure(C🡪cC.,$ )

C🡪cC.,$ ------------------ **I9**

Goto(I6 , c)

C🡪c.C, $

Closure(goto(I6 , c))

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ I6

Goto(I6,d)

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ I7

Goto(I0,S)=I1

Goto(I0,C)=I2

Goto(I0,c)=I3

Goto(I0,d)=I4

Goto(I2,C)=I5

Goto(I2,c) = I6

Goto(I2,d) =I7

Goto(I3 ,C)=I8

Goto(I3 ,c) =I3

Goto(I3,d) =I4

Goto(I6 ,C)=I9

Goto(I6 ,c)=I6

Goto(I6 ,d)=I7

CLR(1) TABLE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| state | c | d | $ | S | C |
| 0 | S3 | S4 |  | 1 | 2 |
| 1 |  |  | acc |  |  |
| 2 | S6 | S7 |  |  | 5 |
| 3 | S3 | S4/R2 |  |  | 8 |
| 4 | R3 | R3 |  |  |  |
| 5 |  |  | R1 |  |  |
| 6 | S6 | S7 |  |  | 9 |
| 7 |  |  | R3 |  |  |
| 8 | R2 | R2 |  |  |  |
| 9 |  |  | R2 |  |  |

LALR(Look Ahead LR)

1.S🡪CC

2.C🡪cC

3.C🡪d

It is common From step1 to table as we have done in CLR method

Step1 : Augmented grammar

S’🡪.S,$

**// \*\*\* If the production is A🡪α.Bβ,a**

**Then add B productions i.e B🡪.γ,b where b is in FIRST(βa) \*\* //**

Apply closure( S’🡪**.**S,$ )

**A🡪α.Bβ,a**

A=S’ , α=ε , B= S , β= ε, a=$

S’🡪.S ,$ b=FIRST(βa)=FIRST(ε$)=$

S🡪.CC,$

C🡪.cC,c/d **I0**A=S, α=ε,B=C, β=C,a=$

C🡪.d,c/d b=FIRST(βa)=FIRST(C$)

=FIRST( C)

={c,d}

Goto(I0,S)

S’🡪S. ,$

Closure(Goto(I0,S))

S’🡪S. ,$ -------------------- **I1**

Goto(I0,C)

S🡪C.C,$

Closure(Goto(I0,C))

S🡪C.C,$

C🡪.cC,$ **I2**

C🡪.d, $

Goto(I0,c)

C🡪c.C,c/d

Closure(Goto(I0,c))

Closure(C🡪c.C,c/d)

C🡪c.C,c/d

C🡪.cC, c/d **I3**

C🡪.d, c/d

Goto(I0,d)

C🡪d.,c/d

Closure(Goto(I0,d))

Closure(C🡪d.,c/d )

C🡪d.,c/d ----------------- **I4**

Goto(I2,C)

S🡪CC. ,$

Closure(Goto(I2,C))

S🡪CC. ,$ --------------------- **I5**

Goto(I2,c)

C🡪c.C,$

Closure(Goto(I2,c))

Closure(C🡪c.C,$)

C🡪c.C,$

C🡪.cC, $ **I6**

C🡪.d, $

Goto(I2,d)

C🡪d., $

Closure(goto(I2,d))

Closure(C🡪d., $)

C🡪d., $ --------- **I7**

Goto(I3 ,C)

C🡪cC.,c/d

Closure(Goto(I3 ,C) )

Closure(C🡪cC.,c/d )

C🡪cC.,c/d ---------- **I8**

Goto(I3 ,c)

C🡪c.C, c/d

Closure(Goto(I3 ,c))

Closure(C🡪c.C, c/d )

C🡪c.C, c/d

C🡪 .cC, c/d I3

C🡪.d , c/d

Goto(I3 ,d)

C🡪d., c/d

Closure(Goto(I3 ,d) )

Closure(C🡪d., c/d )

C🡪d., c/d ------------ I4

Goto(I6 ,C)

C🡪cC.,$

Closure(goto(I6 ,C)

Closure(C🡪cC.,$ )

C🡪cC.,$ ------------------ **I9**

Goto(I6 , c)

C🡪c.C, $

Closure(goto(I6 , c))

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ I6

Goto(I6,d)

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ I7

Goto(I0,S)=I1

Goto(I0,C)=I2

Goto(I0,c)=I3

Goto(I0,d)=I4

Goto(I2,C)=I5

Goto(I2,c) = I6

Goto(I2,d) =I7

Goto(I3 ,C)=I8

Goto(I3 ,c) =I3

Goto(I3,d) =I4

Goto(I6 ,C)=I9

Goto(I6 ,c)=I6

Goto(I6 ,d)=I7

CLR(1) TABLE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | C | D | $ | S | C |
| 0 | S3 | S4 |  | 1 | 2 |
| 1 |  |  | acc |  |  |
| 2 | S6 | S7 |  |  | 5 |
| 3 | S3 | S4 |  |  | 8 |
| 4 | R3 | R3 |  |  |  |
| 5 |  |  | R1 |  |  |
| 6 | S6 | S7 |  |  | 9 |
| 7 |  |  | R3 |  |  |
| 8 | R2 | R2 |  |  |  |
| 9 |  |  | R2 |  |  |

Now identify the common items from LR(1)

LALR(1) items are

C🡪c.C,c/d/$

C🡪.cC, c/d/$ **I36**

C🡪.d, c/d/$

C🡪d.,c/d/$ ----------------- **I47**

C🡪cC.,c/d/$ ---------- **I89**

3=6=36,4=7=47, 8=9=89

LALR(1) PARSING TABLE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | c | d | $ | S | C |
| 0 | S36 | S47 |  | 1 | 2 |
| 1 |  |  | acc |  |  |
| 2 | S36 | S47 |  |  | 5 |
| 36 | S36 | S47 |  |  | 89 |
| 47 | R3 | R3 | R3 |  |  |
| 5 |  |  | R1 |  |  |
| 89 | R2 | R2 | R2 |  |  |

**Dangling Else problem**

Example :

Grammar G :

S 🡪 i S e S | i S | a

1. S 🡪 i S e S
2. S🡪iS
3. S🡪a

Ans :

1. Construct LR(0) Items

Step 1 :

Construct I0

Augumented production for the above grammar is

S’ 🡪S

Augumented grammar is

S’ 🡪S

S 🡪 i S e S

S🡪iS

S🡪a

Step2 : apply dot at right sentential of the production

i.e

S’ 🡪.S

S 🡪 .i S e S I0

S🡪.iS

S🡪.a

Step3: apply goto() and closure()

Goto(I0,S)

S’ 🡪S.

Closure(goto(I0,S))

S’ 🡪S. -------------------- **I1**

Step 4:

Apply goto() and closure() on I0

Goto(I0,i)

S 🡪 i.S e S

S🡪i.S

Closure(Goto(I0,i))

S 🡪 i.S e S

S🡪i.S

S 🡪 .i S e S **I2**

S🡪.iS

S🡪.a

Step5 : goto(I0,a)

S🡪a.

Closure(goto(I0,a))

S🡪a. **I3**

Step6 : goto(I2,S)

S 🡪 iS. e S

S🡪iS.

Closure(goto(I2,S))

S 🡪 iS. e S

S🡪iS. **I4**

Step7 : Goto(I2,i)

S 🡪i. S e S

S🡪i.S

Closure(Goto(I2,i))

S 🡪 i.S e S

S🡪i.S

S 🡪 .i S e S I2

S🡪.iS

S🡪.a

Step 8: goto(I2,a)

S🡪a. I3

Step 9: goto(I4,e)

S 🡪 iSe.S

Closure( goto(I4,e))

S 🡪 iSe.S

S 🡪 .i S e S

S🡪.iS **I5**

S🡪.a

Step 10 : goto(I5,S)

S🡪iSeS.

Closure(goto(I5,S))

S🡪iSeS. **I6**

Step 11: goto(I5,i)

S 🡪 i. S e S

S🡪i.S

Closure(goto(I5,i))

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ I2

Step 12 : goto(I5, a)

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ I3

Goto(I0,S) = I1

Goto(I0,i) = I2

Goto(I0,a) = I3

goto(I2,S) = I4

goto(I2,i) = I2

goto(I2,a) =I3

goto(I4,e) = I5

goto(I5,S)= I6

goto(I5,i) = I2

goto(I5,a)=I3

shift,reduce,accept,error

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| states | **i** | **a** | **e** | **$** | **S** |
| 0 | S2 | S3 |  |  | 1 |
| 1 |  |  |  | ACC |  |
| 2 | S2 | S3 |  |  | 4 |
| 3 |  |  | R3 | R3 |  |
| 4 |  |  | **S5 /R2** | R2 |  |
| 5 | S2 | S3 |  |  | 6 |
| 6 |  |  | R1 | R1 |  |

For reduce action we need to find FIRST() and FOLLOW() of given grammar.

1. S 🡪 i S e S
2. S🡪iS
3. S🡪a

FIRST(S)={i, a}

FOLLOW(S)={e,$}

Example : iiaea

**Stack Symbols Input Action**

0 iiaea$ shift

02 i iaea$ shift

022 ii aea$ shift

0223 iia ea$ reduce S🡪a

0224 iiS ea$ shift

02245 iiSe a$ shift

022453 iiSea $ reduce S🡪a

022456 iiSeS $ reduce S🡪iSeS

024 iS $ reduce S🡪iS

01 S $ ACC