EXPERIMENT - 9

Date 11/05/23

COMPUTATION OF LR(0) ITEMS

AIM: A program to implement LR(0) items.

ALGORITHM:

1. Start
2. Create structure for production with LHS and RHS
3. Open file and read input from file
4. Build state 0 from extra grammar rule $S' \to S\$$ that is all start symbol of grammar and one DOT (·) before S symbol.
5. If Dot symbol is before a non-terminal, add grammar rules that this non-terminal is in left hand of rule and set Dot in before first part of RHS
6. If state exists use that instead.
7. Now find set of terminals and non-terminals in which Dot exists in before.
8. If step 7 is non empty, go to 9 else 10.
9. For each terminal / non terminal in set step 7, Create new state by using all grammar rule that Dot position is before of that terminal / non terminal in ref state by increasing Dot point to next part in RHS of that rules.
10. Go to step 5
11. End of state building
12. Display output end.

$E \to E+T$

$T \to T*F$

$F \to (M)$

$M \to i$

$A \to E$

↓

augumented grammar.

$A \to .E$

$E \to .E+T$

$T \to .T*F$

$F \to .(M)$

$M \to .i$

→ augumented grammar set I_0

I_0

$A \to .E$

$E \to .E+T$

$T \to .T*F$

$F \to .(M)$

$M \to .i$

I_1

$A \to E.$

$E \to E.+T$

I_2

$T \to T.*F$

I_3

$F \to (.M)$

$M \to .i$

I_4

$M \to i.$

I_5

$E \to E+.T$

$T \to .T*F$

I_6

$T \to T*.F$

$F \to .(M)$

I_7

$F \to (M.)$

I_8

$E \to E+T.$

$T \to T.*F$

I_9

$T \to T*F.$

I_{10}

$F \to (M).$

o/p verified

PROGRAM:

```
#include <iostream.h>
#include <conio.h>
#include <string.h>
char prod[20][20], listofvar[26] = "ABCDEFGHIJKLMNOPQR";
int novar = 1, i=0, j=0, k=0, n=0, m=0, arr[30];
int noitem = 0;
struct Grammar
{   char lhs;
    Char rhs[8]; }
g[20], item[20], clos[20][10];
int isvariable(char variable)
{   for (int i=0; i<novar; i++)
            if (g[i].lhs == variable)
                    return i+1;
        return 0;
  }
void findclosure(int z, char a)
{  int n=0, i=0, j=0, k=0, l=0;
   for (i=0; i<arr[z]; i++)
{ for (j=0; j<strlen(clos[z][i].rhs); j++)
  { if(clos[z][i].rhs[j] == '.' && clos[z][i].rhs[j+1]
                                    == a)
{ clos[noitem][n].lhs = clos[z][i].lhs;
strcpy(clos[noitem][n].rhs, clos[z][i].rhs);
char temp = clos[noitem][n].rhs[j];
clos[noitem][n].rhs[j] = clos[noitem][n].rhs[j+1];
clos[noitem][n].rhs[j+1] = temp;
  n=n+1; }}}
for (i=0; i<n; i++) {
```

```
for (j=0; j< strlen ( clos [noitem][i].rhs); j++)
{ if (clos [noitem].[i].rhs[j] == '.' && isvariable(clos
                                    [noitem][i].rhs[j+1]))
{ for (k=0; k< novar ; k++)
  { if (clos [noitem][i].rhs[j+1] == clos[0][k].lhs)

{ for( l=0; l<n ; l++)
        if (clos [noitem [i].lhs == clos [0][k].lhs &&
 strcmp( clos[noitem [i].rhs, clos [0][k].rhs) == 0)
  break;
 if ( l==n)
 { clos [noitem ][n].lhs = clos[0] [k].lhs;
 strcpy( clos [noitem][n].rhs, clos[0][k].rhs);
   n = n+1;
  } } } }

void main()
 { closer();
  cout << "ENTER THE PRODUCTIONS OF GRAMMAR
       (add 0 at END) :\n";
do { cin >> prod [i++]; }
while { strcmp ( prod [i-1], "0") != 0);
 for (n=0; n< i+1; n++)
{  m=0;
   j=novar;
   g [novar ++].lhs = prod [n][0];
  for ( k=3; k< strlen (prod [n]); k++)
```

```
{ if (prod [n][k] != '|')
  g [j].rhs [m++] = prod [n][k];
  if (prod [n][k] == '|')
  {   g [j].rhs [m] = '\0';
      m = 0;
      j = novar;
      g [novar ++].lhs = prod [n][0];
    } } }

for (i=0; i< 26; i++)
   if ( !is variable ( listofvar [i]))
      break;
g[0].lhs = listofvar[i];
char temp [2] = { g [1].lhs ; '\0'};
strcat ( g [0].rhs , temp);
cout <<"\n\n augmented grammar \n";
for (i=0; i<novar ; i++)
   cout <<endl << g[i].lhs <<"->"
   << g[i].rhs <<" ";
 getch ();

{ char list [10];
 int  l=0;
 for (j=0; j<arr[2] ;j++)
 {  for (k=0; k<strlen (clos [z][j].rhs)-1; k++)
    {  if ( clos [z][j].rhs [k] == '.')
       { if ( clos[z] [j
```

```
for (m= 0; m< l; m++)
if (list[m] == clos [z][j].rhs[k+1])
    break;
if(m==1)
    list[l++] = clos [z][j].rhs [k+1];
}}}
for (int x =0; x<l; x++)
    findclosure (z, list[x]);
}
cout<<"\n The set of items are \n";
for (z=0; z<noitem; z++)
{
    cout <<"\n I" <<z<<"\n"
    for (j=0; j <arr [z]; j++)
        cout << clos [z][j].lhs<<"->"
        << clos[z][j].rhs <<"\n";
    getch();
}
getch();
}
```

RESULT: Hence the LR(0) computation is successfuly executed in cpp with the above program.