Ex 9: LR(0) computation

RA1911030010138
Dipali Singh
01/04/2022

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

Algorithm:

```
1. Start.
2. Create a structure for production with LHS and RHS.
3. Open the file and read input from the file.
4. Build state 0 from extra grammar Law S' -> S $ that is all start symbol of grammar and one
Dot ( . ) before S symbol.
5. If the Dot symbol is before a non-terminal, add grammar laws that this non-terminal is in the
Left-Hand Side of that Law and set Dot in before of first part of Right Hand Side.
6. If a state exists (a state with these Laws and the same Dot position), use that instead.
7. Now find the set of terminals and non-terminals in which Dot exist before.
8. If step 7 Set is non-empty go to 9, else go to 10.
9. For each terminal/non-terminal in step 7 create a new state by using all grammar laws that
Dot position is before that terminal/non-terminal in reference state by increasing Dot point
to next part in Right Hand Side of that laws.
10. Go to step 5.
11. End of state-building.
12. Display the output.
13. End.
```

Code:

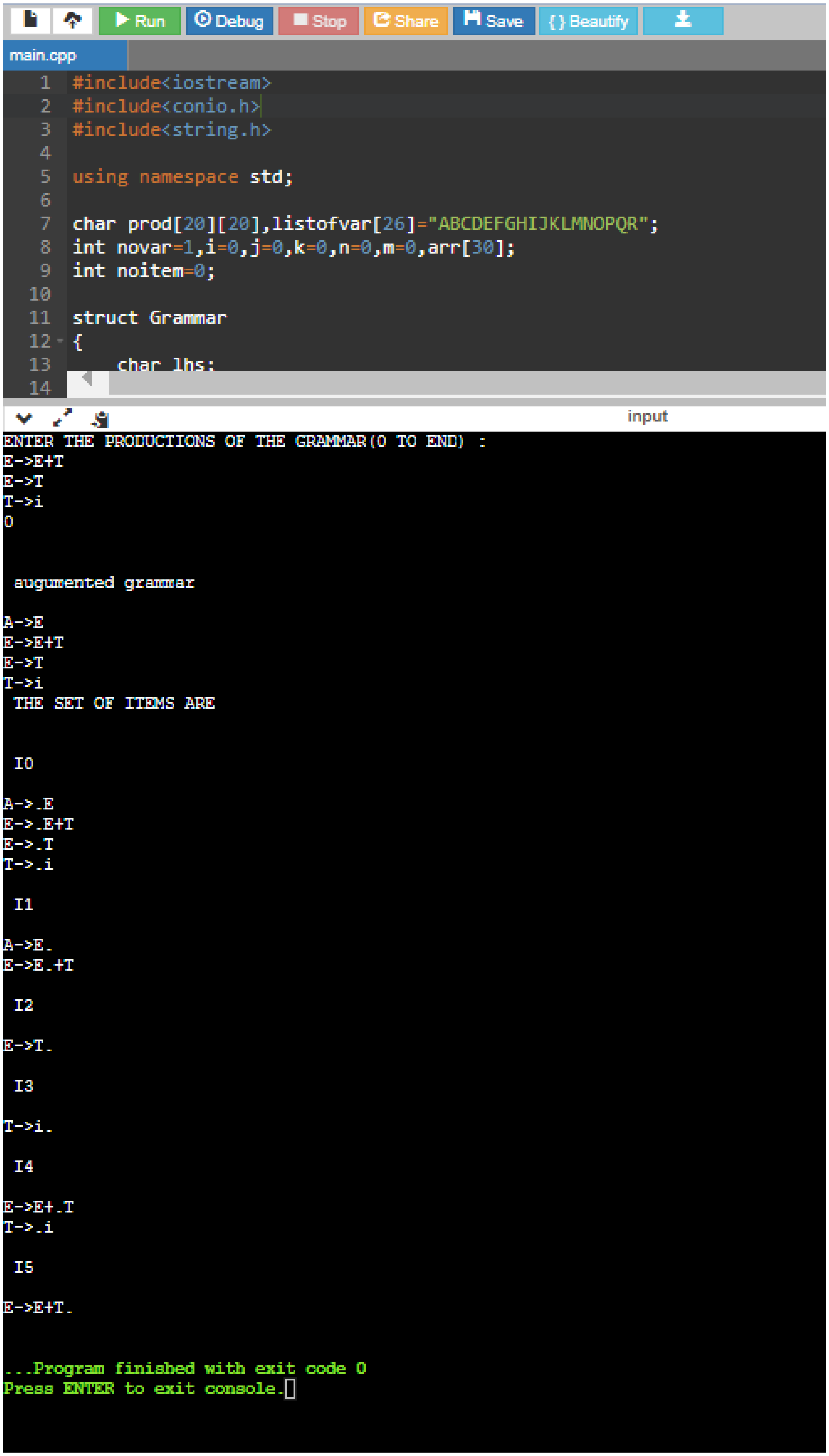
```
#include<iostream>
#include<conio.h>
#include<string.h>

using namespace std;

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])>0)
            {
                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][l].lhs==clos[0][k].lhs &&
strcmp(clos[noitem][l].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;
                        }
                    }
                }
            }
        }
    }
    arr[noitem]=n;
    int flag=0;
    for(i=0;i<noitem;i++)
    {
        if(arr[i]==n)
        {
            for(j=0;j<arr[i];j++)
            {
                int c=0;
                for(k=0;k<arr[i];k++)
                    if(clos[noitem][k].lhs==clos[i][k].lhs &&
strcmp(clos[noitem][k].rhs,clos[i][k].rhs)==0)
                        c=c+1;
                if(c==arr[i])
                {
                    flag=1;
                    goto exit;
                }
            }
        }
    }
    exit:;
    if(flag==0)
        arr[noitem++]=n;
}

int main()
{
    cout<<"ENTER THE PRODUCTIONS OF THE GRAMMAR(0 TO 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(!isvariable(listofvar[i]))
        break;
g[0].lhs=listofvar[i];
char temp[2]={g[1].lhs,'\0'};
strcat(g[0].rhs,temp);
cout<<"\n\n augumented grammar \n";
for(i=0;i<novar;i++)
    cout<<endl<<g[i].lhs<<"->"<<g[i].rhs<<" ";

for(i=0;i<novar;i++)
{
    clos[noitem][i].lhs=g[i].lhs;
    strcpy(clos[noitem][i].rhs,g[i].rhs);
    if(strcmp(clos[noitem][i].rhs,"ε")==0)
        strcpy(clos[noitem][i].rhs,".");
    else
    {
        for(int j=strlen(clos[noitem][i].rhs)+1;j>=0;j--)
```

```
                clos[noitem][i].rhs[j]=clos[noitem][i].rhs[j-1];
            clos[noitem][i].rhs[0]='.';
        }
    }
    arr[noitem++]=novar;
    for(int z=0;z<noitem;z++)
    {
        char list[10];
        int l=0;
        for(j=0;j<arr[z];j++)
        {
            for(k=0;k<strlen(clos[z][j].rhs)-1;k++)
            {
                if(clos[z][j].rhs[k]=='.')
                {
                    for(m=0;m<l;m++)
                        if(list[m]==clos[z][j].rhs[k+1])
                            break;
                    if(m==l)
                        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\n";
    for(int z=0; z<noitem; z++)
    {
        cout<<"\n I"<<z<<"\n\n";
        for(j=0;j<arr[z];j++)
            cout<<clos[z][j].lhs<<"->"<<clos[z][j].rhs<<"\n";

    }

}
```

Output:



```cpp
#include<iostream>
#include<conio.h>
#include<string.h>

using namespace std;

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;
```

```
ENTER THE PRODUCTIONS OF THE GRAMMAR(0 TO END) :
E->E+T
E->T
T->i
0


 augumented grammar

A->E
E->E+T
E->T
T->i
 THE SET OF ITEMS ARE


 I0

A->.E
E->.E+T
E->.T
T->.i

 I1

A->E.
E->E.+T

 I2

E->T.

 I3

T->i.

 I4

E->E+.T
T->.i

 I5

E->E+T.


...Program finished with exit code 0
Press ENTER to exit console.
```

Result:

The program was successfully compiled and run.