RA1911026010033

Compiler Design

Week - 9

Topic :

Implementation of LR(0)

AIM:

TO FIND LR(0) ITEMS OF GIVEN GRAMMAR IN C++.

ALGORITHM:

- Start with C0 by including all marked productions [S→.α]
- Compute the closure of the item set C0 by If [A→. X α2] where X €Vn, we include in C0 all items of the form [X→.β]
- Perform a read operation on items in an item set,

e.g. if [A →α.Xβ] is in the item set, then read X yielding a new item set whose initial member is [A →AX.β].

- Compute the closure of the new item set.

If [A →α1 .Xα 2] where X €Vn, we include in C0 all items of the form [X →.β].

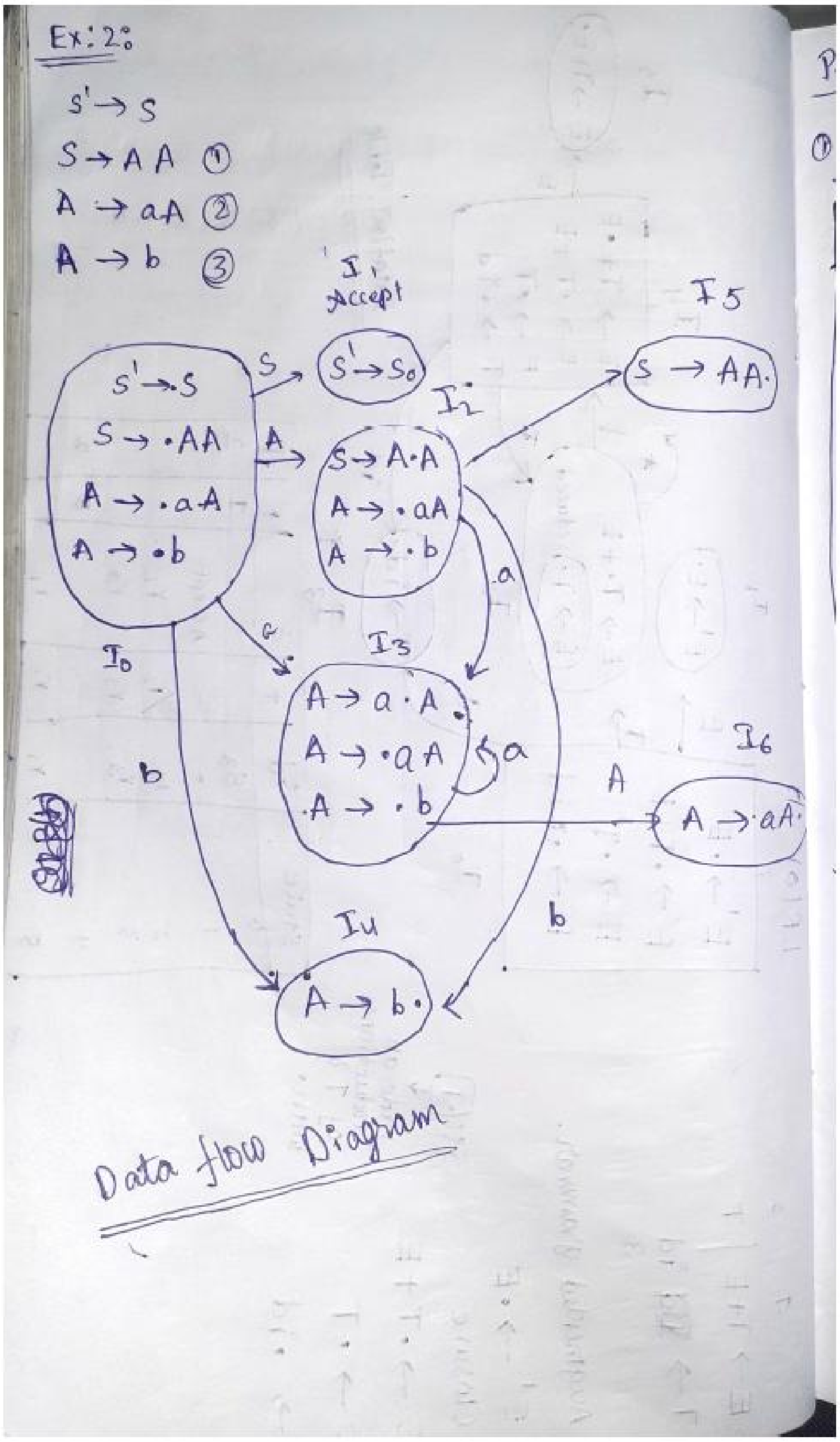
- Continue reading until all s have traveled through all item sets.

Code:

```
#include<iostream>
#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,"e")==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";

    }
    return 0;
}
```

Manual Calculation :



Ex: 2:

$s' \to s$

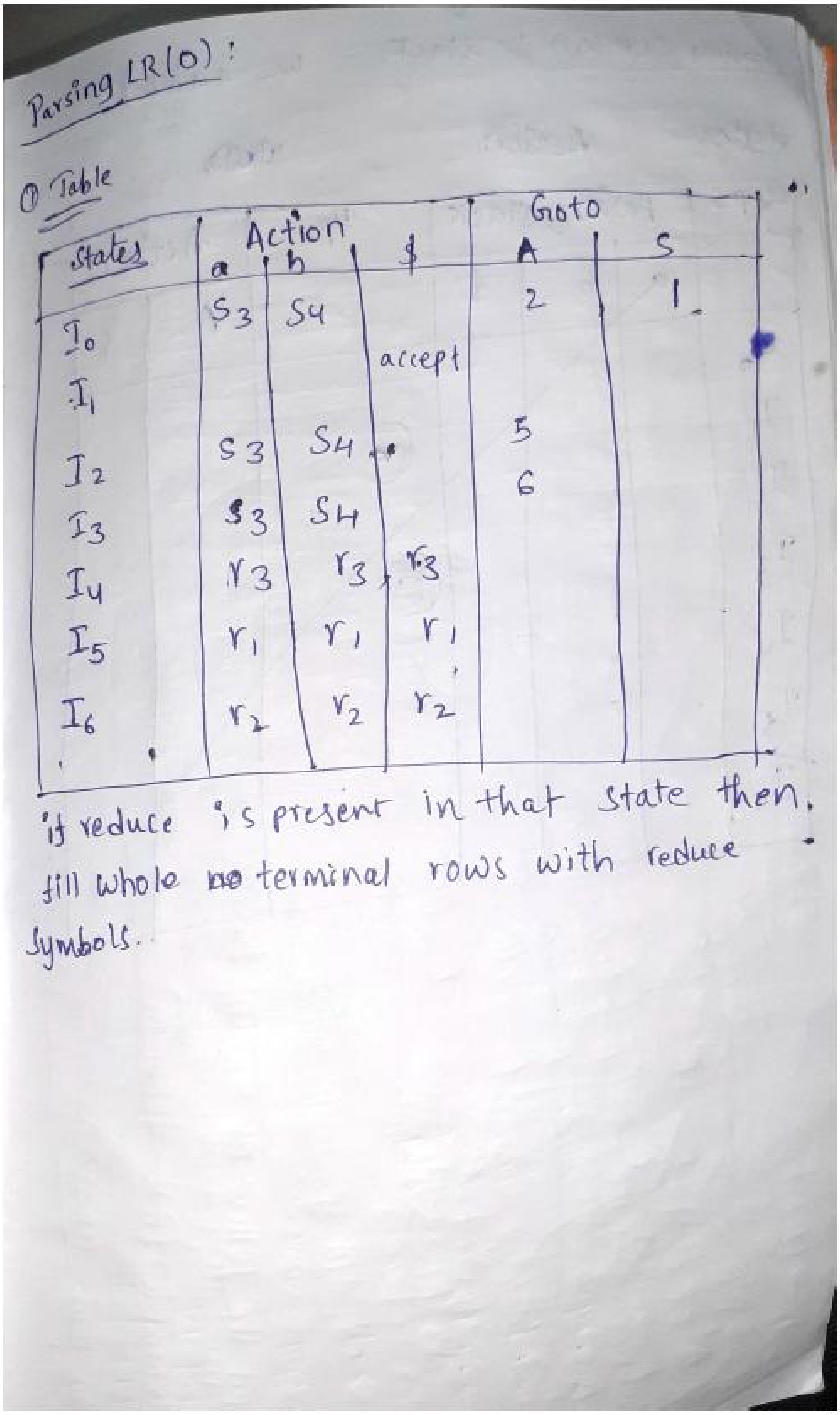
$S \to AA$ ①

$A \to aA$ ②

$A \to b$ ③

Data flow Diagram

RA1911026010033



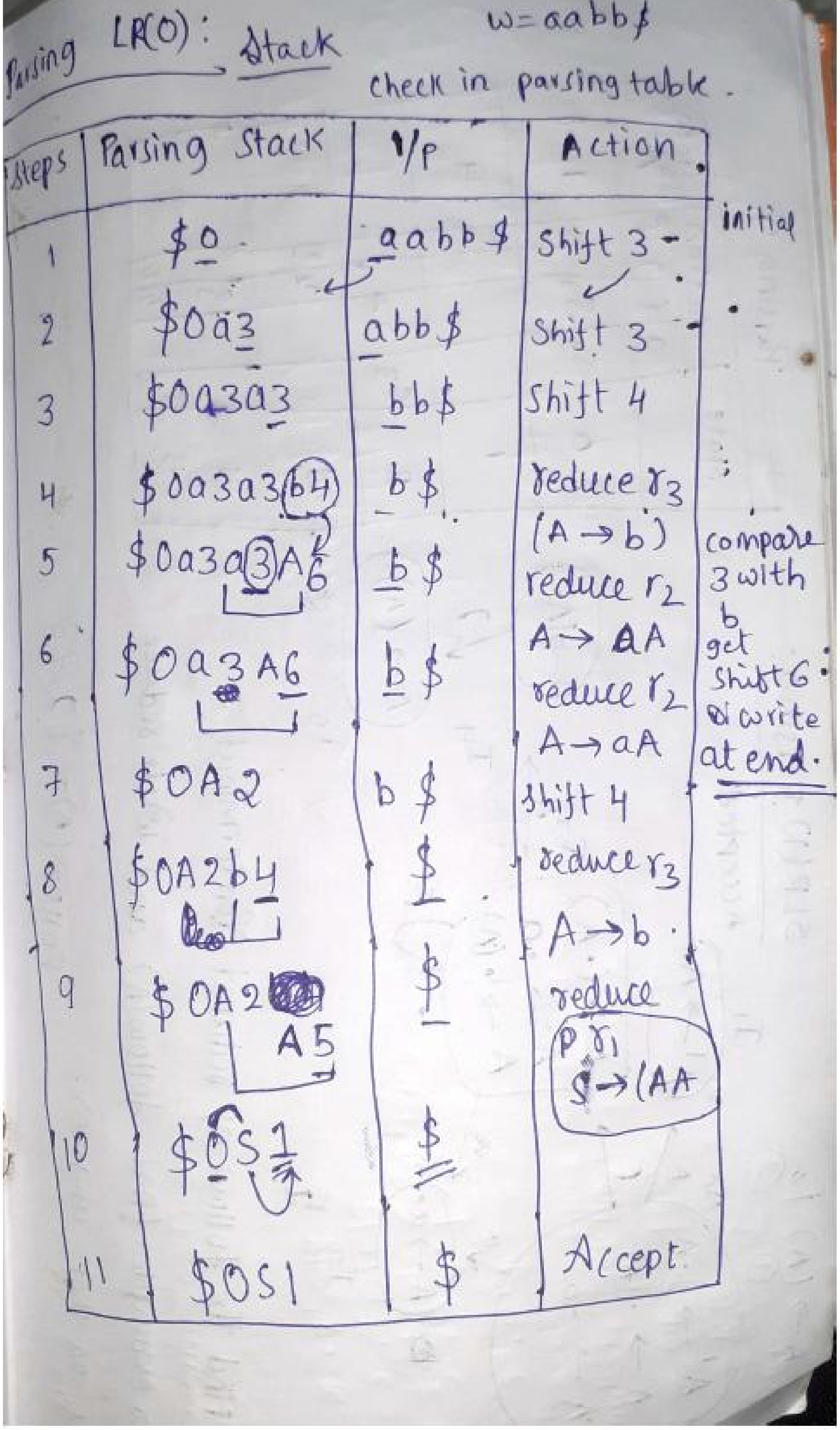
Parsing LR(0):

① Table

States	Action			Goto	
	a	b	$	A	S
I_0	S_3	S_4		2	1
I_1			accept		
I_2	S_3	S_4		5	
I_3	S_3	S_4		6	
I_4	r_3	r_3	r_3		
I_5	r_1	r_1	r_1		
I_6	r_2	r_2	r_2		

if reduce is present in that state then fill whole terminal rows with reduce symbols.

RA1911026010033



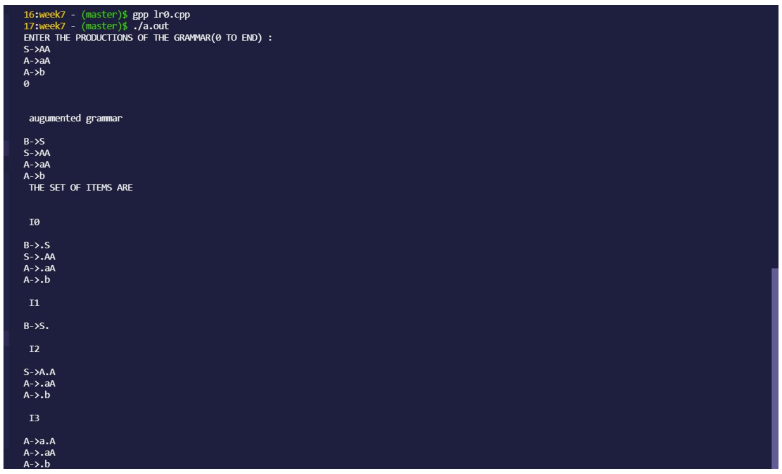
Parsing LR(0): → Stack   w = aabb$

check in parsing table.

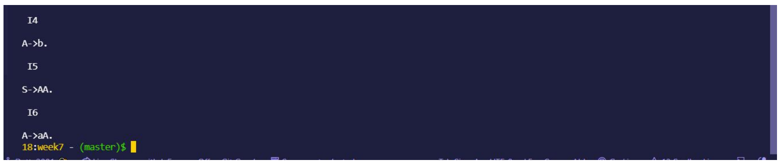
Steps	Parsing Stack	I/P	Action
1	$0	aabb$	Shift 3
2	$0a3	abb$	Shift 3
3	$0a3a3	bb$	Shift 4
4	$0a3a3b4	b$	reduce r_3 (A → b)
5	$0a3a3A6	b$	reduce r_2 A → aA
6	$0a3A6	b$	reduce r_2 A → aA
7	$0A2	b$	shift 4
8	$0A2b4	$	reduce r_3 A → b
9	$0A2A5	$	reduce P r_1 S → AA
10	$0S1	$	
11	$0S1	$	Accept

initial

compare 3 with b get shift 6 & write at end.



```
16:week7 - (master)$ gpp lr0.cpp
17:week7 - (master)$ ./a.out
ENTER THE PRODUCTIONS OF THE GRAMMAR(0 TO END) :
S->AA
A->aA
A->b
0


 augumented grammar

B->S
S->AA
A->aA
A->b
 THE SET OF ITEMS ARE


 I0

B->.S
S->.AA
A->.aA
A->.b

 I1

B->S.

 I2

S->A.A
A->.aA
A->.b

 I3

A->a.A
A->.aA
A->.b
```



```
 I4

A->b.

 I5

S->AA.

 I6

A->aA.
18:week7 - (master)$ 
```

Result: LR(0) Succefully implemented.