

Compiler Design(18CSC304J)

Experiment 6

ELIMINATION OF LEFT RECURSION

Harsh Goel

RA1811003010185

Aim: To study and implement Elimination of Left Recursion.

Language: C++

Theory:

A production of grammar is said to have left recursion if the leftmost variable of its RHS is same as variable of its LHS.

A grammar containing a production having left recursion is called as Left Recursive Grammar.

Procedure:

1. Start the program.
2. Initialize the arrays for taking input from the user.
3. Prompt the user to input the no. of non-terminals having left recursion and no. of productions for these non-terminals.
4. Prompt the user to input the production for non-terminals.
5. Eliminate left recursion using the following rules:-
 1. $A \rightarrow A\alpha_1 | A\alpha_2 | \dots | A\alpha_m$
 2. $A \rightarrow \beta_1 | \beta_2 | \dots | \beta_n$
 3. Then replace it by
 4. $A \rightarrow \beta_i A' \ i=1,2,3,\dots,m$
 5. $A' \rightarrow \alpha_j \ A' \ j=1,2,3,\dots,n$
 6. $A' \rightarrow \epsilon$
6. After eliminating the left recursion by applying these rules, display the productions without left recursion.
7. Stop

Code Snippet:

```
#include <iostream>
#include <vector>
#include <string>
using namespace std;

int main()
```

```

{
    int n;
    cout<<"\nEnter number of non terminals: ";
    cin>>n;
    cout<<"\nEnter non terminals one by one: ";
    int i;
    vector<string> nonter(n);
    vector<int> lefttreocr(n,0);
    for(i=0;i<n;++i) {
        cout<<"\Non terminal "<<i+1<<" : ";
        cin>>nonter[i];
    }
    vector<vector<string> > prod;
    cout<<"\nEnter 'esp' for null";
    for(i=0;i<n;++i) {
        cout<<"\nNumber of "<<nonter[i]<<" productions: ";
        int k;
        cin>>k;
        int j;
        cout<<"\nOne by one enter all "<<nonter[i]<<" productions";
        vector<string> temp(k);
        for(j=0;j<k;++j) {
            cout<<"\nRHS of production "<<j+1<<": ";
            string abc;
            cin>>abc;
            temp[j]=abc;
            if(nonter[i].length()<=abc.length()&&nonter[i].compare(abc.substr(0,nonter[i].length()))==0)
                lefttreocr[i]=1;
        }
        prod.push_back(temp);
    }
    for(i=0;i<n;++i) {
        cout<<lefttreocr[i];
    }
    for(i=0;i<n;++i) {
        if(lefttreocr[i]==0)
            continue;
        int j;
        nonter.push_back(nonter[i]+"");
        vector<string> temp;
        for(j=0;j<prod[i].size();++j) {
            if(nonter[i].length()<=prod[i][j].length()&&nonter[i].compare(prod[i][j].substr(0,nonter[i].length()))==0) {
                string abc=prod[i][j].substr(nonter[i].length(),prod[i][j].length()-nonter[i].length()+nonter[i]+"");
                temp.push_back(abc);
                prod[i].erase(prod[i].begin()+j);
            }
        }
    }
}

```

```

        --j;
    }
    else {
        prod[i][j]+=nonter[i]+' ';
    }
}
temp.push_back("esp");
prod.push_back(temp);
}
cout<<"\n\n";
cout<<"\nNew set of non-terminals: ";
for(i=0;i<nonter.size();++i)
    cout<<nonter[i]<<" ";
cout<<"\n\nNew set of productions: ";
for(i=0;i<nonter.size();++i) {
    int j;
    for(j=0;j<prod[i].size();++j) {
        cout<<"\n"<<nonter[i]<<" -> "<<prod[i][j];
    }
}
return 0;
}

```

Output Screenshots:

```

leftRec2.exe - exp_5 - Visual Studio Code
3 cppdbg: leftRec2.exe

Enter non terminals one by one:
Non terminal 1 : E

Non terminal 2 : T

Non terminal 3 : F

Enter 'esp' for null
Number of E productions: 2

One by one enter all E productions
RHS of production 1: E+T

RHS of production 2: T

Number of T productions: 2

One by one enter all T productions
RHS of production 1: T+F

RHS of production 2: F

Number of F productions: 2

One by one enter all F productions
RHS of production 1: (E)

RHS of production 2: i

110

New set of non-terminals: E T F E' T'

New set of productions:
E -> TE'
T -> FT'
F -> (E)
F -> i
E' -> +TE'
E' -> esp
T' -> *FT'
T' -> esp

```

```
PS C:\Users\HARSH-PC\Desktop\college\COMPILER_DESIGN\exp_5> 8 "c:\Users\HARSH-PC\.vscode\extensions\ms-vscode.cpptools-1.2.2\debugAdapters\bin\WindowsDebugLauncher.exe" "--stdin=Microsoft-MIEngine-In-utxwvvu0.kyo" "--stdout=Microsoft-MIEngine-Out-vtgoz3z1.itu" "--stderr=Microsoft-MIEngine-Error-wjrl0ed.cw" "--pid=Microsoft-MIEngine-Pid-r123" 8 "c:\Users\HARSH-PC\.vscode\extensions\ms-vscode.cpptools-1.2.2\debugAdapters\bin\WindowsDebugLauncher.exe" "--stdin=Microsoft-MIEngine-In-frfdq0jy.eix" "--stdout=Microsoft-MIEngine-Out-emicfzs.12d" "--stderr=Microsoft-MIEngine-Error-x4kvttbts.1ta" "--pid=Microsoft-MIEngine-Pid-22x2h1vu.543" "--dbgExe=C:\TDM-GCC-64\bin\gdb.exe" "--interpreter=mi"
PS C:\Users\HARSH-PC\Desktop\college\COMPILER_DESIGN\exp_5>
PS C:\Users\HARSH-PC\Desktop\college\COMPILER_DESIGN\exp_5> .\leftRec2.exe

Enter number of non terminals: 3

Enter non terminals one by one:
Non terminal 1 : E

Non terminal 2 : T

Non terminal 3 : F

Enter 'esp' for null
Number of E productions: 2

One by one enter all E productions
RHS of production 1: E+T

RHS of production 2: T

Number of T productions: 2

One by one enter all T productions
RHS of production 1: T*F

RHS of production 2: F

Number of F productions: 2

One by one enter all F productions
RHS of production 1: (E)

RHS of production 2: i

110

New set of non-terminals: E T F E' T'

New set of productions:
E -> TE'
T -> FT'
F -> (E)
F -> i
E' -> +TE'
E' -> esp
```

Result:

The code was successfully implemented in C and output was recorded. Hence, A program for Elimination of Left Recursion was run successfully.