# Practical - 4

2CS701 – Compiler Construction

Harshit Gajipara
19BCE059

**Aim:**

To Implement Left Recursion derivation removal algorithm: Eliminate direct and indirect Left recursion from given grammar for LL(1) parser.

**Code:**

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int n;
    cout << "Enter the number of productions : ";
    cin >> n;

    map<char, vector<string>> map_with_recursions, mp;
    vector<string> productions;
    set<char> start_symbols;
    vector<string> ans;

    cout << "\nEnter the productions (one by one) : \n";
    for (int i = 0; i < n; i++)
    {
        string s;
        cin >> s;
        productions.push_back(s);
        start_symbols.insert(s[0]);

        // For each production rule to be analysed, we will
be storing it in this string
        string s1 = "";

        // i=0,1,2 are for "E->" so we start with i=3 till
all productions for a start symbol
        for (int i = 3; i < s.length(); i++)
        {
```

```
            if (s[i] != '|') // scan until new rule
                s1 += s[i];
            else
            {
                if (s1[0] >= 'A' && s1[0] <= 'Z' && s1[0] ==
s[0]) // string has left recursion
                    map_with_recursions[s[0]].push_back(s1);
                else
                    mp[s[0]].push_back(s1);
                s1 = "";
            }
        }

        // for last production rule to be stored
        if (s1[0] >= 'A' && s1[0] <= 'Z' && s1[0] == s[0])
            map_with_recursions[s[0]].push_back(s1);
        else
            mp[s[0]].push_back(s1);
        s1 = "";
    }

    for (auto it = start_symbols.begin(); it !=
start_symbols.end(); it++)
    {
        if (map_with_recursions[*it].size() == 0)
        {
            string ans1 = "";

            ans1 += (*it);
            ans1 += "-";
            ans1 += ">";
            vector<string> temp = mp[*it];
            for (int i = 0; i < temp.size(); i++)
            {
                if (i != (temp.size() - 1))
                {
                    ans1 += temp[i];
                    ans1 += "|";
```

```
            }
            else
            {
                ans1 += temp[i];
            }
        }
        ans.push_back(ans1);
    }
    else
    {
        string temp = "";
        temp += *it;
        temp += '\'';
        vector<string> temp1 = map_with_recursions[*it];
        vector<string> temp2 = mp[*it];
        string prod1 = "";
        prod1 += *it;
        prod1 += "->";
        for (int i = 0; i < temp2.size(); i++)
        {
            if (i != (temp2.size() - 1))
            {
                string tt = temp2[i];
                tt += temp;
                prod1 += tt;
                prod1 += "|";
            }
            else
            {
                string tt = temp2[i];
                tt += temp;
                prod1 += tt;
            }
        }
        string prod2 = "";
        prod2 += temp;
        prod2 += "->";
        for (int i = 0; i < temp1.size(); i++)
```

```
            {
                string tt = temp1[i].substr(1);
                tt += temp;
                prod2 += tt;
                prod2 += "|";
            }
            prod2 += "~";
            ans.push_back(prod1);
            ans.push_back(prod2);
        }
    }

    cout << "\nGrammar after eliminating left recursion is :
\n";
    for (int i = 0; i < ans.size(); i++)
        cout << ans[i] << endl;
    return 0;
}

/*
4
S->ABC
A->Aa|Ad|b
B->Bb|e
C->Cc|g
*/
```

## Output:

```
PS D:\19BCE059\B.Tech Semester 7\CC\CC Practicals\Practical 4> cd "d:\19BCE
ctical4 } ; if ($?) { .\practical4 }
Enter the number of productions : 4

Enter the productions (one by one) :
S->ABC
A->Aa|Ad|b
B->Bb|f
C->Cc|e

Grammar after eliminating left recursion is :
A->bA'
A'->aA'|dA'|~
B->fB'
B'->bB'|~
C->eC'
C'->cC'|~
S->ABC
PS D:\19BCE059\B.Tech Semester 7\CC\CC Practicals\Practical 4>
```

```
PS D:\19BCE059\B.Tech Semester 7\CC\CC Practicals\Practical 4> cd "d:\
ctical4 } ; if ($?) { .\practical4 }
Enter the number of productions : 3

Enter the productions (one by one) :
E->E(T)|T
T->T(F)|F
F->id

Grammar after eliminating left recursion is :
E->TE'
E'->(T)E'|~
F->id
T->FT'
T'->(F)T'|~
PS D:\19BCE059\B.Tech Semester 7\CC\CC Practicals\Practical 4>
```