Compiler Design

Elimination of left Recursion

EXPERIMENT – 4A

Roehit Ranganathan | RA1911033010017
15 February 2022

Aim:

Write a program in your preferred language to eliminate the left recursion in the given production rules.

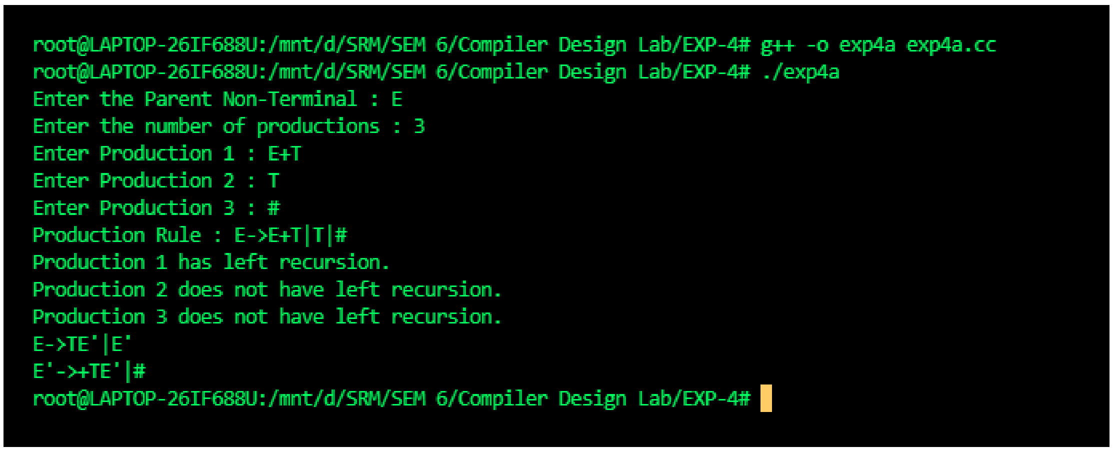
Algorithm:

- Check if the given grammar contains left recursion, if present then separates the production and start working on it.
- Introduce a new nonterminal and write it at the last of every terminal.
- Write newly produced nonterminal in LHS and in RHS it can either produce or it can produce new production in which the terminals or non-terminals which followed the previous LHS will be replaced by new nonterminal at last.

Program:

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string ip, op1, op2, temp;
    int sizes[10] = {};
    char c;
    int n, j, l;
    cout << "Enter the Parent Non-Terminal : ";
    cin >> c;
    ip.push_back(c);
    op1 += ip + "\'->";
    ip += "->";
    op2 += ip;
    cout << "Enter the number of productions : ";
    cin >> n;
    for (int i = 0; i < n; i++)
    {
        cout << "Enter Production " << i + 1 << " : ";
        cin >> temp;
        sizes[i] = temp.size();
        ip += temp;
        if (i != n - 1)
            ip += "|";
    }
    cout << "Production Rule : " << ip << endl;
    for (int i = 0, k = 3; i < n; i++)
    {
        if (ip[0] == ip[k])
        {
            cout << "Production " << i + 1 << " has left recursion." << endl;
            if (ip[k] != '#')
            {
                for (l = k + 1; l < k + sizes[i]; l++)
                    op1.push_back(ip[l]);
                k = l + 1;
                op1.push_back(ip[0]);
                op1 += "\'|";
            }
        }
        else
        {
            cout << "Production " << i + 1 << " does not have left recursion."
<< endl;
            if (ip[k] != '#')
            {
                for (j = k; j < k + sizes[i]; j++)
                    op2.push_back(ip[j]);
                k = j + 1;
                op2.push_back(ip[0]);
                op2 += "\'|";
            }
            else
            {
                op2.push_back(ip[0]);
                op2 += "\'";
            }
        }
    }
    op1 += "#";
    cout << op2 << endl;
    cout << op1 << endl;
    return 0;
}
```

Output:



```
root@LAPTOP-26IF688U:/mnt/d/SRM/SEM 6/Compiler Design Lab/EXP-4# g++ -o exp4a exp4a.cc
root@LAPTOP-26IF688U:/mnt/d/SRM/SEM 6/Compiler Design Lab/EXP-4# ./exp4a
Enter the Parent Non-Terminal : E
Enter the number of productions : 3
Enter Production 1 : E+T
Enter Production 2 : T
Enter Production 3 : #
Production Rule : E->E+T|T|#
Production 1 has left recursion.
Production 2 does not have left recursion.
Production 3 does not have left recursion.
E->TE'|E'
E'->+TE'|#
root@LAPTOP-26IF688U:/mnt/d/SRM/SEM 6/Compiler Design Lab/EXP-4#
```

Result:

The program was implemented.