Compiler Design

Left factoring

EXPERIMENT – 4B

Roehit Ranganathan | RA1911033010017
15 February 2022

Aim:

Write a program in your preferred language to perform left factoring in the given production rules.

Algorithm:

1. For all A ∈ non-terminal, find the longest prefix a that occurs in two or more right-hand sides of A.

2. If a ¹ ∈ then replace all of the A productions, A → a bI | a b2 | - - - | a bn | r with

3. A → a A | r A → bI | b2| - - - | bn | ∈ Where, A is a new element of non-terminal.

4. Repeat until no common prefixes remain.

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 + "\'->";
    op2 += ip + "\'\'->";
    ;
    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;
```

```
    char x = ip[3];
    for (int i = 0, k = 3; i < n; i++)
    {
        if (x == ip[k])
        {
            if (ip[k + 1] == '|')
            {
                op1 += "#";
                ip.insert(k + 1, 1, ip[0]);
                ip.insert(k + 2, 1, '\'');
                k += 4;
            }
            else
            {
                op1 += "|" + ip.substr(k + 1, sizes[i] - 1);
                ip.erase(k - 1, sizes[i] + 1);
            }
        }
        else
        {
            while (ip[k++] != '|')
                ;
        }
    }
    char y = op1[6];
    for (int i = 0, k = 6; i < n - 1; i++)
    {
        if (y == op1[k])
        {
            if (op1[k + 1] == '|')
            {
                op2 += "#";
                op1.insert(k + 1, 1, op1[0]);
                op1.insert(k + 2, 2, '\'');
                k += 5;
            }
            else
            {
                temp.clear();
                for (int s = k + 1; s < op1.length(); s++)
                    temp.push_back(op1[s]);
                op2 += "|" + temp;
                op1.erase(k - 1, temp.length() + 2);
            }
        }
    }
    op2.erase(op2.size() - 1);
    cout << "After Left Factoring : " << endl;
```

```
    cout << ip << endl;
    cout << op1 << endl;
    cout << op2 << endl;
    return 0;
}
```

Output:

```
root@LAPTOP-26IF688U:/mnt/d/SRM/SEM 6/Compiler Design Lab/EXP-4# g++ -o exp4b exp4b.cc
root@LAPTOP-26IF688U:/mnt/d/SRM/SEM 6/Compiler Design Lab/EXP-4# ./exp4b
Enter the Parent Non-Terminal : L
Enter the number of productions : 4
Enter Production 1 : i
Enter Production 2 : iL
Enter Production 3 : (L)
Enter Production 4 : iL+L
Production Rule : L->i|iL|(L)|iL+L
After Left Factoring :
L->iL'|(L)
L'->#|LL''
L''->#|+L
root@LAPTOP-26IF688U:/mnt/d/SRM/SEM 6/Compiler Design Lab/EXP-4#
```

Result:

The program was implemented.