8 - Leading and Trailing

Tuesday, 12 March 2024 10:11 AM

Ashima Fatima Seik Mugibur Raghman, 21BAI1830

AIM: Write a program to complete leading and trailing of a given grammar production

Sample I/p :

$E \rightarrow E+E \mid E*E \mid id$

$S \rightarrow (L) \mid a$

$L \rightarrow L,S \mid S$

Sample O/p :

Leading (E) = { +, *, id }

Trailing (E) = { +, *, id }

Leading (S) = { (, a }

Leading (L) = { , , (, a }

Trailing (S) = {), a }

Trailing (L) = { , ,), a }

Procedure :

1. input the number of productions and the productions themselves.
2. iterate over the productions to identify the terminals(T) and non-terminals.(NT)
3. for each production, check the first character to determine if its a new non terminal. if yes, add it to the 'NT' array
4. for each character in the productions (from the 4th character), if it's not already present in the NT array, and it is not a non-terminal, add it to the terminals array.
5. initialize the leading 'l' and trailing 'tr' arrays with 'f' to indicate all entries are initially 'false'
6. iterate over the non-terminals & productions to compute the trailing sets.
   a. if the last character of a production is a non-terminal, add the last terminal symbol preceding it to its training set.
   b. if the last character is a terminal, add it to the trailing set.
   c. update the trailing set based on the symbols preceding the non-terminal in each production.
7. print the computed leading and trailing sets for each non-terminal.

CODE :

```
#include<bits/stdc++.h>
using namespace std;
#include <cstring>
int nt, t, top = 0;
char s[50], NT[10], T[10], st[50], l[10][10], tr[50][50];
int searchnt(char a)
{
    int count = -1, i;
    for (i = 0; i < nt; i++)
    {
        if (NT[i] == a)
            return i;
    }
    return count;
}
int searchter(char a)
{
    int count = -1, i;
    for (i = 0; i < t; i++)
    {
        if (T[i] == a)
            return i;
    }
    return count;
}
void push(char a)
{
    s[top] = a;
    top++;
}
char pop()
{
    top--;
    return s[top];
}
void installl(int a, int b)

{
    if (l[a][b] == 'f')
    {
        l[a][b] = 't';
        push(T[b]);
        push(NT[a]);
    }
}
void installt(int a, int b)
{
    if (tr[a][b] == 'f')
    {
        tr[a][b] = 't';
        push(T[b]);
        push(NT[a]);
    }
}

int main()
{
    int i, s, k, j, n;
    char pr[30][30], b, c;
    cout<< "Enter the no of productions:";
    cin>> n;
```

```
cout << "Enter the productions one by one\n";
for (i = 0; i < n; i++)
    cin >> pr[i];
nt = 0;
t = 0;
for (i = 0; i < n; i++)
{
    if ((searchnt(pr[i][0])) == -1)
        NT[nt++] = pr[i][0];
}
for (i = 0; i < n; i++)
{
    for (j = 3; j < strlen(pr[i]); j++)
    {
        if (searchnt(pr[i][j]) == -1)
        {
            if (searchter(pr[i][j]) == -1)
                T[t++] = pr[i][j];
        }
    }
}
for (i = 0; i < nt; i++)
{
    for (j = 0; j < t; j++)
        l[i][j] = 'f';
}
for (i = 0; i < nt; i++)
{
    for (j = 0; j < t; j++)

        tr[i][j] = 'f';
}
for (i = 0; i < nt; i++)
{
    for (j = 0; j < n; j++)
    {
        if (NT[(searchnt(pr[j][0]))] == NT[i])
        {
            if (searchter(pr[j][3]) != -1)
                installl(searchnt(pr[j][0]), searchter(pr[j][3]));
            else
            {
                for (k = 3; k < strlen(pr[j]); k++)
                {
                    if (searchnt(pr[j][k]) == -1)
                    {
                        installl(searchnt(pr[j][0]), searchter(pr[j][k]));
                        break;
                    }
                }
            }
        }
    }
}
while (top != 0)
{
    b = pop();
    c = pop();
    for (s = 0; s < n; s++)
    {
        if (pr[s][3] == b)
```

```
            installl(searchnt(pr[s][0]), searchter(c));
        }
    }
    for (i = 0; i < nt; i++)
    {
        cout << "Leading[" << NT[i] << "]"
             << "\t{";
        for (j = 0; j < t; j++)
        {
            if (l[i][j] == 't')
                cout << T[j] << ",";
        }
        cout << "}\n";
    }

    top = 0;
    for (i = 0; i < nt; i++)
    {
        for (j = 0; j < n; j++)
        {
            if (NT[searchnt(pr[j][0])] == NT[i])
            {
                if (searchter(pr[j][strlen(pr[j]) - 1]) != -1)
                    installt(searchnt(pr[j][0]), searchter(pr[j][strlen(pr[j]) - 1]));
                else
                {
                    for (k = (strlen(pr[j]) - 1); k >= 3; k--)
                    {
                        if (searchnt(pr[j][k]) == -1)
                        {
                            installt(searchnt(pr[j][0]), searchter(pr[j][k]));
                            break;
                        }
                    }
                }
            }
        }
    }
    while (top != 0)
    {
        b = pop();
        c = pop();
        for (s = 0; s < n; s++)
        {
            if (pr[s][3] == b)
                installt(searchnt(pr[s][0]), searchter(c));
        }
    }
    for (i = 0; i < nt; i++)
    {
        cout << "Trailing[" << NT[i] << "]"
             << "\t{";
        for (j = 0; j < t; j++)
        {
            if (tr[i][j] == 't')
                cout << T[j] << ",";
        }
        cout << "}\n";
    }
    return 0;
}
```