

## Exercise 3

### Eliminating Left Recursion

Praveen Kumar R  
312217104114

January 31, 2020

### Aim

To Write a C program that eliminates the left recursion in a grammar and produce left recursion free productions.

### Left recursion

A production is said to have left recursion if the first symbol of the production is the non-terminal on the left hand side of the production.

$$A \rightarrow A\alpha_1|A\alpha_2|A\alpha_3 \dots |A\alpha_m|\beta_1|\beta_2| \dots |\beta_n$$

where  $\alpha_i, \beta_i \in (N \cup T)^*$ . The production has to be rewritten as,

$$\begin{aligned} A &\rightarrow A'\alpha_1|A'\alpha_2|A'\alpha_3 \dots |A'\alpha_m \\ A' &\rightarrow \beta_1A'|\beta_2A'|\beta_3A'| \dots |\beta_nA'|\epsilon \end{aligned}$$

### C Program

```
#include<stdio.h>
#include<string.h>

void slice(char a[],int start,int end,char temp[]){
    int pos = 0;
    temp[pos]='\0';
    int i = start;
    while(i<end){
        temp[pos]=a[i];
```

```

        pos++;
        i++;
    }
    temp[pos]='\0';
}
char getNonTerminal(char p[]){
    return p[0];
}

int getProductionRHS(char p[],char prod[100][100]){
    int i = 0;
    while(p[i]!='>') i++;
    int k = 0,t = 0,j = 0;
    char temp[100];
    while(p[i]!='\0'){
        i++;
        if(p[i]=='|' || p[i]=='\0'){
            temp[j] = '\0';
            strcpy(prod[k++],temp);
            j =0;
            temp[j] = '\0';
        }
        else{
            temp[j++] = p[i];
        }
    }
    return k;
}

int isLeftRecursion(char nt,char p[100][100],int n){
    for(int i = 0;i<n;i++){
        if(p[i][0]==nt) return 1;
    }
    return 0;
}

int getBetaList(char nt,char p[100][100],int n,char beta[100][100]){
    int k = 0;
    for(int i = 0;i<n;i++){
        if(p[i][0]!=nt) {
            strcpy(beta[k++],p[i]);
        }
    }
    return k;
}

```

```

}

int getAlphaList(char nt, char p[100][100], int n, char alpha[100][100]){
    int k = 0;
    char temp[100];
    for(int i = 0; i < n; i++){
        if(p[i][0] == nt) {
            slice(p[i], 1, strlen(p[i]), temp);
            strcpy(alpha[k++], temp);
        }
    }
    return k;
}

void getProduction1(char nt, char alpha[100][100], char prod1[100][100], int n){
    for(int i = 0; i < n ; i++){
        char temp[100];
        temp[0] = nt;
        temp[1] = '\\';
        temp[2] = '\\0';
        strcat(temp, alpha[i]);
        strcpy(prod1[i], temp);
    }
}

void getProduction2(char nt, char beta[100][100], char prod2[100][100], int n){
    int i;
    for( i = 0; i < n ; i++){
        char temp[100] = "";
        strcat(temp, beta[i]);
        int l = strlen(temp);
        temp[l] = nt;
        temp[l+1] = '\\';
        temp[l+2] = '\\0';
        strcpy(prod2[i], temp);
    }
    strcpy(prod2[i], "epsilon");
}

void displayProduction(char nt[], char prod[100][100], int n){
    printf("%s->%s", nt, prod[0]);
    for(int i = 1; i < n; i++){
        printf("|%s", prod[i]);
    }
    printf("\\n");
}

```

```

int main()
{
    char a[100][100];
    int n;
    printf("Enter the number of productions: ");
    scanf("%d",&n);
    printf("Enter %d productions: Example: A->abaA\n",n);
    for(int i = 0; i<n;i++)
    {
        scanf("%s",a[i]);
    }
    printf("\nGrammar after removing all left recursion: \n\n");
    for(int i = 0;i<n;i++){
        char nt[5];
        char prod[100][100],alpha[100][100],beta[100][100];
        char modifiedProduction1[100][100],modifiedProduction2[100][100];
        int at,bt,k;
        nt[0] = getNonTerminal(a[i]);
        nt[1] = '\0';
        k = getProductionRHS(a[i],prod);
        if(isLeftRecursion(nt[0],prod,k)){
            at = getAlphaList(nt[0],prod,k,alpha);
            bt = getBetaList(nt[0],prod,k,beta);
            getProduction1(nt[0],alpha,modifiedProduction1,at);
            getProduction2(nt[0],beta,modifiedProduction2,bt);
            char newnt[5];
            newnt[0] = nt[0];
            newnt[1] = '\ ';
            newnt[2] = '\0';
            displayProduction(nt,modifiedProduction1,at);
            displayProduction(newnt,modifiedProduction2,bt+1);
        }
        else{
            displayProduction(nt,prod,k);
        }
    }
    return 0;
}

```

## Sample Input & Output

```
Enter the number of productions: 4
Enter 4 productions: Example: A->abaA
S->SassA|AbSaS|SSaab|A
A->Abc|AAbc|BBca|Sasa
B->CbaB|ccac|CCAC
C->Baac|CbbA
```

Grammer after removing all left recursion:

```
S->S'assA|S'Saab
S'->AbSaSS'|AS'|epsilon
A->A'bc|A'Abc
A'->BBcaA'|SasaA'|epsilon
B->CbaB|ccac|CCAC
C->C'bbA
C'->BaacC'|epsilon
```