A3-Elimination of Immediate Left Recursion using C

Sneha Sriram Kannan 185001157

19-02-2021

1 Code

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

int substr(char str1[], char str2[]);
int checkFunction(char str[]);
int EliminateLeftRecursion();
int Eliminate(char production[],char newFile[10][128],int count,int pos);
int main()
{
  EliminateLeftRecursion();
  return 0;
}

int EliminateLeftRecursion()
{
  char file[10][128];
  char newProduction[10][128];
  int newProductionCount=0;
  int lrcount=0;
  FILE *fd = fopen("input.txt", "r");
  int i = 0;
  //reading code from a file and storing in an array
  while (fgets(file[i], sizeof(file[i]), fd))
    i++;

  printf("============================\n");
  printf("Input Productions:\n");
  printf("============================\n");
  for (int j = 0; j < i; j++)
  {
    printf("%s", file[j]);
  }
  printf("\n\n");
  printf("==========================================\n");
  printf("Result of checking for Left Recursion:\n");
  printf("==========================================\n");
  for (int j = 0; j < i; j++)
  {
    char lhs=file[j][0];
    int start=1;
    int noLeft=1;
    for(int k=3;k<strlen(file[j]);k++){
        if(start==1){
            if(lhs==file[j][k]) //checking if left recursion occurs in the productions
            {
```

```
                noLeft=0;
                lrcount++;  //count of left recursions identified and eliminated
                newProductionCount=Eliminate(file[j],newProduction,newProductionCount,k);
            }
            start=0;
        }
        else if(file[j][k]=='|')
            start=1;
    }
    if(noLeft==1)   //no LR so no change in the production
    {
        strcpy(newProduction[newProductionCount++],file[j]);
    }
  }
  if(lrcount==0)
    printf("NO LEFT RECURSION\n");
  else
  {
    for (int j = 0; j < newProductionCount; j++)
    {
        printf("%s", newProduction[j]);
    }
  }
  printf("\n");
  return 0;
}

int Eliminate(char production[],char newFile[10][128],int count,int pos)
{
    char new[3];
    new[0]=production[0];
    new[1]='\'';
    new[2]='\0';
    char alpha[20];
    int j=0;
    int betacount=0;
    int betapointer=0;
    char betaproduction[30];
    char newProduction[50];
    sprintf(newProduction,"%c-> ",production[0]);
    int k=0;
    int newFlag=0;
    //FINDING A->beta A
    for(int i=3;i<strlen(production);i++){
        if(production[i]=='|' || production[i]=='\n' )  //end of a production
        {
            strcat(newProduction,new);
            betacount++;
            newFlag=1;

        }
        else if(i!=pos)
        {
            if(production[i]!=production[0])    //beta identified
            {
                if(newFlag==1)  //must concatenate |
                {
                    strcat(newProduction,"|");
                    newFlag=0;
                }
                char temp[2];
                temp[0]=production[i];
                temp[1]='\0';
                strcat(newProduction,temp); //Adding character of beta
            }
        }
        else if(i==pos){    //left recursion position so not beta
            while(production[i]!='|')
                i++;
        }
    }
    strcat(newProduction,"\n");
```

```
    strcpy(newFile[count++],newProduction);
    //FINDING A'->epsilon|alphaA
    for(int i=pos+1;i<strlen(production);i++){
        if(production[i]!='|' && production[i]!='\n')
            alpha[j++]=production[i];
        else
            break;
    }
    alpha[j]='\0';
    sprintf(newFile[count++],"%s-> %s%s|  \n",new,alpha,new);
    return count;
}
```

2 Output Screenshot

```
Input Productions:
============================
E->T|E+T|A*B
T->F|T*F
F->id

==========================================
Result of checking for Left Recursion:
==========================================
E-> TE'|A*BE'
E'-> +TE'|ε
T-> FT'
T'-> *FT'|ε
F->id
snehakannan@pop-os:~/Sneha/Semester 6/Compiler Design/Lab/A3$
```

Figure 1: Left Recursion Removed