Department of Computer Science and Engineering

S.G.Shivanirudh , 185001146, Semester VI

1 February 2021

UCS1602 - Compiler Design

Exercise 3: Elimination of Immediate LeftRecursion using C

Objective:

Write a program in C to find whether the given grammar is LeftRecursive or not. If it is found to be left recursive, convert the grammar in such a way that the left recursion is removed.

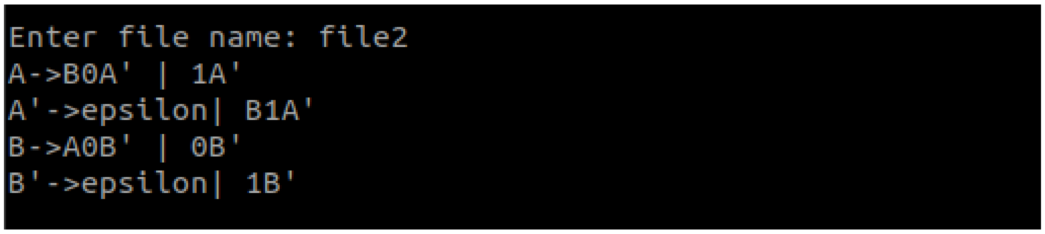
Code:

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

int elim_lr(char* production){

    char* prod = (char*)calloc(100, sizeof(char));
    strcpy(prod, production);

    char* token = strtok(prod, "->");
    char sym = token[0];
    printf("%c->", sym);

    token = strtok(NULL, "->");

    char* tok = strtok(token, "|");
    int flag = 0;
    char *alpha = (char*)calloc(100, sizeof(char));
    char *beta[10];
    int be = 0;
    while(tok){
        if(flag == 0){
            for(int i = 1; tok[i]; i++){
                alpha[i-1] = tok[i];
            }
            flag = 1;
        }
        else{
            beta[be++] = (char*)calloc(100, sizeof(char));
            strcpy(beta[be-1], tok);
        }
        tok = strtok(NULL, "|");
    }

    if(be == 0){
        printf("%s is a Left Recursive production, but cannot
    be reduced", production);
        return 0;
    }

    for(int i = 0;i<be;i++){
        printf("%s%c'", beta[i], sym);
        if(i+1 != be)
            printf(" | ");
    }
    printf("\n");
    printf("%c'->epsilon| %s%c'\n", sym, alpha, sym);
}

int check_lr(char* production){
    char* prod = (char*)calloc(100, sizeof(char));
    strcpy(prod, production);
    char *token = strtok(prod, "->");
    char sym = token[0];
    token = strtok(NULL, "->");
    if(sym == token[0])
        elim_lr(production);
    else
        printf("%s\n", production);
}

int line_count(char *file){
    FILE *fp;
    int count = 0;
    fp = fopen(file, "r");

    if (fp == NULL){
        return 0;
    }
    for(char c = getc(fp); c != EOF; c = getc(fp))
        if (c == '\n')
            count = count + 1;
    fclose(fp);
    return count;
}

int main(){
    char *file_name = (char*)calloc(100, sizeof(char));
    char *production = (char *)calloc(100, sizeof(char));
    printf("\nEnter file name: ");
    scanf(" %[^\n]", file_name);

    FILE *fp;
    fp = fopen(file_name, "r+");
    int ctr = 0;
    fscanf(fp, " %[^\n]", production);
    while (ctr < line_count(file_name))
    {
        check_lr(production);
        fscanf(fp, " %[^\n]", production);
        ctr++;
    }
}
```

Input file:

```
A->AB1|B0|1
B->B1|A0|0
```

Output:



```
Enter file name: file2
A->B0A' | 1A'
A'->epsilon| B1A'
B->A0B' | 0B'
B'->epsilon| 1B'
```