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:

```
#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];

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

    char* tok = strtok(token, "|");
    char *alpha[10];
    int al = 0;

    char *beta[10];
    int be = 0;
    while(tok){
        if(sym == tok[0]){
            alpha[al] = (char *)calloc(100, sizeof(char));
            for(int i = 1; tok[i]; i++){
                alpha[al][i-1] = tok[i];
            }
            al++;
        }
        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;
    }

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

```
51          if(i+1 != al)
52              printf(" | ");
53      }
54      printf("\n");
55  }
56
57  int check_lr(char* production){
58      char* prod = (char*)calloc(100, sizeof(char));
59      strcpy(prod, production);
60      char *token = strtok(prod, "->");
61      char sym = token[0];
62      token = strtok(NULL, "->");
63      if(sym == token[0])
64          elim_lr(production);
65      else
66          printf("%s\n", production);
67  }
68
69  int line_count(char *file){
70      FILE *fp;
71      int count = 0;
72      fp = fopen(file, "r");
73
74      if (fp == NULL){
75          return 0;
76      }
77      for(char c = getc(fp); c != EOF; c = getc(fp))
78          if (c == '\n')
79              count = count + 1;
80      fclose(fp);
81      return count;
82  }
83
84  int main(){
85      char *file_name = (char*)calloc(100, sizeof(char));
86      char *production = (char *)calloc(100, sizeof(char));
87      printf("\nEnter file name: ");
88      scanf(" %[^\n]", file_name);
89
90      FILE *fp;
91      fp = fopen(file_name, "r+");
92      int ctr = 0;
```

Input file:

```
A->AB1|AB0|1
B->B1|BA0|0
E->E*T
```

Output:

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

Learning Outcomes:

- Understood the basic concept of left recursion and need for its elimination.
- Learnt how to remove left-recursion from specified grammar using C.
