Lab Program - 9

V Pranav - 192311029

Implement a C program to eliminate left recursion from a given CFG.

**Code:**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#define MAX\_PRODUCTIONS 10

#define MAX\_SYMBOLS 10

typedef struct {

char lhs;

char rhs[MAX\_SYMBOLS][MAX\_SYMBOLS]; // Store multiple right-hand sides

int rhsCount;

} Production;

Production productions[MAX\_PRODUCTIONS];

int prodCount = 0;

// Function to check if a rule has left recursion

int hasLeftRecursion(Production p) {

for (int i = 0; i < p.rhsCount; i++) {

if (p.rhs[i][0] == p.lhs) // Leftmost symbol matches LHS

return 1;

}

return 0;

}

// Function to eliminate immediate left recursion

void eliminateLeftRecursion(Production p) {

char newNonTerminal = p.lhs + '\''; // Create a new non-terminal (A' or B')

Production noRecursion, recursion;

noRecursion.lhs = p.lhs;

recursion.lhs = newNonTerminal;

noRecursion.rhsCount = 0;

recursion.rhsCount = 0;

for (int i = 0; i < p.rhsCount; i++) {

if (p.rhs[i][0] == p.lhs) { // Left-recursive case: A → Aα

strcpy(recursion.rhs[recursion.rhsCount], p.rhs[i] + 1);

strcat(recursion.rhs[recursion.rhsCount], &newNonTerminal);

recursion.rhsCount++;

} else { // Non-left-recursive case: A → β

strcpy(noRecursion.rhs[noRecursion.rhsCount], p.rhs[i]);

strcat(noRecursion.rhs[noRecursion.rhsCount], &newNonTerminal);

noRecursion.rhsCount++;

}

}

// A' → ε (if left recursion was found)

strcpy(recursion.rhs[recursion.rhsCount], "#");

recursion.rhsCount++;

// Print transformed productions

printf("New Productions:\n");

printf("%c → ", noRecursion.lhs);

for (int i = 0; i < noRecursion.rhsCount; i++) {

printf("%s ", noRecursion.rhs[i]);

if (i < noRecursion.rhsCount - 1)

printf("| ");

}

printf("\n");

printf("%c → ", recursion.lhs);

for (int i = 0; i < recursion.rhsCount; i++) {

printf("%s ", recursion.rhs[i]);

if (i < recursion.rhsCount - 1)

printf("| ");

}

printf("\n");

}

int main() {

// Input Grammar

printf("Enter number of productions: ");

scanf("%d", &prodCount);

getchar(); // Consume newline

for (int i = 0; i < prodCount; i++) {

printf("Enter production (e.g., A->Aa|b): ");

char input[50];

fgets(input, sizeof(input), stdin);

char \*token = strtok(input, "->|");

productions[i].lhs = token[0];

productions[i].rhsCount = 0;

while ((token = strtok(NULL, "->|")) != NULL) {

strcpy(productions[i].rhs[productions[i].rhsCount++], token);

}

}

// Process Productions

printf("\nAfter Left Recursion Removal:\n");

for (int i = 0; i < prodCount; i++) {

if (hasLeftRecursion(productions[i])) {

eliminateLeftRecursion(productions[i]);

} else {

printf("%c → ", productions[i].lhs);

for (int j = 0; j < productions[i].rhsCount; j++) {

printf("%s ", productions[i].rhs[j]);

if (j < productions[i].rhsCount - 1)

printf("| ");

}

printf("\n");

}

}

return 0;

}

**Screenshot for I/O:**

