Lab Program - 8

V Pranav - 192311029

Write a C program to find FOLLOW( ) - predictive parser for the given grammar  
S → AaAb / BbBa  
A → ∈  
B → ∈.

**Code:**

#include <stdio.h>

#include <string.h>

#define NT 3 // Number of non-terminals: S, A, B

#define PROD 4 // Number of productions

#define MAX\_RHS 10 // Maximum length for right-hand side

// Structure to hold a production rule

typedef struct {

char lhs;

char rhs[MAX\_RHS];

} Production;

// Grammar productions

Production productions[PROD] = {

{'S', "AaAb"},

{'S', "BbBa"},

{'A', "#"}, // A → ε (represented by '#')

{'B', "#"} // B → ε (represented by '#')

};

// Non-terminals

char nonTerminals[NT] = {'S', 'A', 'B'};

// FIRST and FOLLOW sets

int first[NT][128] = {0};

int follow[NT][128] = {0};

int computedFirst[NT] = {0};

int computedFollow[NT] = {0};

// Check if a character is a non-terminal

int isNonTerminal(char c) {

for (int i = 0; i < NT; i++)

if (nonTerminals[i] == c)

return 1;

return 0;

}

// Get index of a non-terminal

int getIndex(char c) {

for (int i = 0; i < NT; i++)

if (nonTerminals[i] == c)

return i;

return -1;

}

// Compute FIRST set

void computeFirst(int nt) {

if (computedFirst[nt])

return;

computedFirst[nt] = 1;

char A = nonTerminals[nt];

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

if (productions[i].lhs == A) {

int epsilonInAll = 1;

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

char symbol = productions[i].rhs[j];

if (!isNonTerminal(symbol)) {

first[nt][(int)symbol] = 1;

epsilonInAll = 0;

break;

} else {

int idx = getIndex(symbol);

computeFirst(idx);

for (int k = 0; k < 128; k++) {

if (first[idx][k] && k != '#')

first[nt][k] = 1;

}

if (!first[idx]['#']) {

epsilonInAll = 0;

break;

}

}

}

if (epsilonInAll)

first[nt]['#'] = 1;

}

}

}

// Compute FOLLOW set

void computeFollow(int nt) {

if (computedFollow[nt])

return;

computedFollow[nt] = 1;

if (nonTerminals[nt] == 'S')

follow[nt]['$'] = 1; // Add EOF symbol ($) to FOLLOW(S)

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

char \*rhs = productions[i].rhs;

int len = strlen(rhs);

for (int j = 0; j < len; j++) {

if (rhs[j] == nonTerminals[nt]) {

int hasEpsilon = 1;

for (int k = j + 1; k < len; k++) {

char symbol = rhs[k];

if (!isNonTerminal(symbol)) {

follow[nt][(int)symbol] = 1;

hasEpsilon = 0;

break;

} else {

int idx = getIndex(symbol);

computeFirst(idx);

for (int m = 0; m < 128; m++) {

if (first[idx][m] && m != '#')

follow[nt][m] = 1;

}

if (!first[idx]['#']) {

hasEpsilon = 0;

break;

}

}

}

if (hasEpsilon) {

int lhsIdx = getIndex(productions[i].lhs);

computeFollow(lhsIdx);

for (int m = 0; m < 128; m++) {

if (follow[lhsIdx][m])

follow[nt][m] = 1;

}

}

}

}

}

}

int main() {

for (int i = 0; i < NT; i++)

computeFirst(i);

for (int i = 0; i < NT; i++)

computeFollow(i);

// Display FOLLOW sets

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

printf("FOLLOW(%c) = { ", nonTerminals[i]);

for (int c = 0; c < 128; c++) {

if (follow[i][c]) {

if (c == '$')

printf("EOF ");

else

printf("%c ", c);

}

}

printf("}\n");

}

return 0;

}

**Screenshot for I/O:**

