

**Lab Task # 7**

**Submitted By:**

Ramsha Kokab

FA21-BCS-010

**Course Instructor:**

Mr. Syed Bilal Bukhari

**Course:**

Compiler Construction

**Date:**

18th Octuber, 2024

**DEPARTMENT OF COMPUTER SCIENCE**

**COMSATS UNIVERSITY ISLAMABAD, ATTOCK CAMPUS**

**Lab Task :** Write a code for the grammar with at least 4 Non-Terminals and 4 Terminals

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab7\_Task

{

public partial class Form1 : Form

{

private Dictionary<char, List<string>> productions = new Dictionary<char, List<string>>();

private Dictionary<char, HashSet<char>> followSet = new Dictionary<char, HashSet<char>>();

private char startSymbol;

public Form1()

{

InitializeComponent();

}

private void btnAddProduction\_Click(object sender, EventArgs e)

{

string input = txtProductionInput.Text.Trim();

if (string.IsNullOrEmpty(input) || !input.Contains("->"))

{

MessageBox.Show("Please enter a valid production (e.g., S->AB|a).");

return;

}

char nonTerminal = input[0];

string[] rhs = input.Substring(3).Split('|');

if (!productions.ContainsKey(nonTerminal))

{

productions[nonTerminal] = new List<string>();

followSet[nonTerminal] = new HashSet<char>(); // Initialize Follow set for the non-terminal

}

foreach (var production in rhs)

{

productions[nonTerminal].Add(production);

}

lstProductions.Items.Add(input);

// Assume the first production added is the start symbol

if (productions.Count == 1)

startSymbol = nonTerminal;

txtProductionInput.Clear();

}

private void btnFindFollow\_Click(object sender, EventArgs e)

{

char nonTerminal = txtNonTerminal.Text.Trim()[0];

if (!productions.ContainsKey(nonTerminal))

{

MessageBox.Show("Non-terminal not found in the productions.");

return;

}

// Initialize the Follow set of the start symbol with '$'

followSet[startSymbol].Add('$');

// Calculate Follow sets

foreach (var nt in productions.Keys)

{

CalculateFollow(nt);

}

// Display the Follow set of the selected non-terminal

txtFollowResult.Text = $"Follow({nonTerminal}) = {{ {string.Join(", ", followSet[nonTerminal])} }}";

}

private void CalculateFollow(char nonTerminal)

{

foreach (var lhs in productions.Keys)

{

foreach (var production in productions[lhs])

{

for (int i = 0; i < production.Length; i++)

{

if (production[i] == nonTerminal)

{

// If non-terminal is not at the end of the production

if (i + 1 < production.Length)

{

char nextSymbol = production[i + 1];

if (char.IsUpper(nextSymbol))

{

// Add First(nextSymbol) to Follow(nonTerminal) excluding epsilon (~)

foreach (char symbol in GetFirst(nextSymbol))

{

if (symbol != '~')

followSet[nonTerminal].Add(symbol);

}

// If nextSymbol can be epsilon, add Follow(lhs) to Follow(nonTerminal)

if (GetFirst(nextSymbol).Contains('~'))

{

foreach (char symbol in followSet[lhs])

followSet[nonTerminal].Add(symbol);

}

}

else

{

// nextSymbol is a terminal, so add it directly to Follow(nonTerminal)

followSet[nonTerminal].Add(nextSymbol);

}

}

else if (i + 1 == production.Length && lhs != nonTerminal)

{

// If non-terminal is at the end, add Follow(lhs) to Follow(nonTerminal)

foreach (char symbol in followSet[lhs])

followSet[nonTerminal].Add(symbol);

}

}

}

}

}

}

private HashSet<char> GetFirst(char nonTerminal)

{

HashSet<char> first = new HashSet<char>();

foreach (var production in productions[nonTerminal])

{

if (!char.IsUpper(production[0]))

{

first.Add(production[0]);

}

else

{

foreach (char symbol in GetFirst(production[0]))

{

first.Add(symbol);

}

}

}

return first;

}

}

}

**Output:**

