**LAB # 06 FIRST SET**

Hammad Tufail (SP20-BCS-028)

Kulsoom Khurshid (SP20-BCS-044)

using System;

using System.Collections;

using System.Collections.Generic;

using System.ComponentModel;

using System.Text.RegularExpressions;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.TextBox;

using System.Security.Policy;

namespace First\_Set

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

//Non-terminals are used as keys to save the production rule and first set

Hashtable productionRulez = new Hashtable();

Hashtable firstSets = new Hashtable();

private void button1\_Click(object sender, EventArgs e)

{

productionRulez.Clear();

firstSets.Clear();

richTextBox2.Clear();

String temp2 = "";

bool flag = true;

var productionRules = richTextBox1.Text.Split('\n');

foreach (var productionRule in productionRules)

{

var temp = productionRule.Split('>');

if (!productionRulez.Contains(temp[0]))

{

productionRulez.Add(temp[0], temp[1]);

var te = temp[0].ToCharArray()[0];

if (!(new Regex(@"^[A-Z]$")).Match(te + "").Success)

{

flag = false;

MessageBox.Show("Non terminals cant be small letters");

}

}

else

{

productionRulez[temp[0]] += "|" + temp[1];

}

}

if (flag)

{

foreach (DictionaryEntry rule in productionRulez)

{

List<String[]> rules = new List<String[]>();

var alpha = rule.Value.ToString().Split('|');

foreach (var rul in alpha)

{

rules.Add(rul.Split(' '));

}

foreach (var rul in rules)

{

if (!firstSets.Contains(rule.Key))

{

firstSets.Add(rule.Key, calculateFirst(rul, 0));

}

else

{

firstSets[rule.Key] += "," + calculateFirst(rul, 0);

}

}

}

foreach (DictionaryEntry x in firstSets)

{

richTextBox2.AppendText("First(" + x.Key.ToString() + ") = " + "{" + x.Value.ToString() + "}\n");

}

}

}

private string calculateFirst(String[] alpha, int index)

{

if (!productionRulez.Contains(alpha[0]) && alpha[0] != "~")

{

return alpha[0];

}

else if (alpha[0] != "~" && alpha.Length >= 1 && index < alpha.Length)

{

List<String[]> rules = new List<String[]>();

foreach (String rul in productionRulez[alpha[index]].ToString().Split('|'))

{

var arr = rul.Split(' ');

var temp1 = new List<string>();

foreach (string s in arr)

{

if (!string.IsNullOrEmpty(s))

{

temp1.Add(s);

}

}

rules.Add(temp1.ToArray());

}

string temp = "";

foreach (String[] rul in rules)

{

if (rul[0] != alpha[index])

{

string x = calculateFirst(rul, 0);

temp += x + ", ";

}

}

string y = temp;

if (y.Contains("~"))

{

return y + calculateFirst(alpha, index + 1);

}

else

{

return y;

}

}

return "~";

}

}

}

Graphical user interface

Description automatically generated