**LAB # 07 FOLLOW 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.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace follow\_set

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

Hashtable productionRulez = new Hashtable();

Hashtable firstSets = new Hashtable();

Hashtable followSets = new Hashtable();

Char startingvar;

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

{

productionRulez.Clear();

firstSets.Clear();

followSets.Clear();

String temp2 = "";

bool flag = true;

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

startingvar = productionRules[0][0];

richTextBox2.AppendText("Starting var:" + startingvar + "\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("Left side should be non terminals (Capital letters)");

}

}

else

{

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

}

}

//calculating first

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");

}

}

//finding follow set

if (flag)

{

followSets[startingvar] = "$";

foreach(DictionaryEntry production in productionRulez)

{

String calfor = production.Key + "";

calculateFollow(calfor, 0);

}

richTextBox2.AppendText(followsetprint(followSets) + "/n");

}

}

void calculateFollow(String alpha, int index)

{

foreach (DictionaryEntry production in productionRulez)

{

String[] rightside = (production.Value + "").Split(' ');

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

{

if (rightside[i] == alpha)

{

if (i + 1 > rightside.Length - 1)

{

followSets[alpha] += getFollowof(production);

}

else if (!productionRulez.Contains(rightside[i + 1]))

{

followSets[alpha] += rightside[i + 1];

}

else

{

int a = i;

while (true)

{

if (a + 1 > rightside.Length - 1)

{

break;

}

a++;

string firstset = findFirstOf(rightside[a]);

if (firstset[firstset.Length - 1] == '~')

{

firstset = firstset.Replace('~'.ToString(), String.Empty);

}

followSets[alpha] += firstset;

}

}

}

}

}

}

String findFirstOf(String nonterminal)

{

foreach (DictionaryEntry key in firstSets)

{

if (key.Key.ToString() == nonterminal)

{

return key.Value.ToString();

}

}

return "";

}

String getFollowof(DictionaryEntry follows)

{

foreach (DictionaryEntry a in followSets)

{

if (a.Key.ToString() == follows.Key.ToString())

{

return a.Value.ToString();

}

}

return "";

}

private string followsetprint(Hashtable arr)

{

String a = "";

foreach (DictionaryEntry s in arr)

{

a += "Follow(" + s.Key + "}= {" + s.Value + " };\n";

}

return a;

}

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

{

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

{

return alpha[0];

}

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

{

String[] beta = null;

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

if (alpha.Length - 1 < index)

{

return "~";

}

else if (productionRulez.Contains(alpha[index]))

{

var barRule = productionRulez[alpha[index]].ToString().Split('|');

foreach (var rul in barRule)

{

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

}

String term = "";

foreach (var rul in rules)

{

var x = calculateFirst(rul, index);

if (x[x.Length - 1] != '~')

{

term += "," + x;

}

else

{

term += "," + calculateFirst(alpha, index + 1);

}

}

return term;

}

else

{

return alpha[index];

}

}

return "~";

}

}

}

