МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева (Самарский университет)»  
  
Факультет информатики  
Кафедра информационных систем и технологий  
  
Дисциплина  
**«Системы искусственного интеллекта»  
  
  
  
ОТЧЕТ**по лабораторной работе №2

Вариант № 3

Студент: Гижевская В.Д.  
Группа: 6313-020302D  
  
Преподаватель: Додонова Н.Л.

Самара 2020

**Краткая теория:** Метод резолюций – это метод автоматического доказательства теорем – основы логического программирования. Это алгоритм, проверяющий отношение выводимости Г├А. В общем случае алгоритм автоматического доказательства теорем не существует, но для формальных теорий с несложной структурой (таких как исчисление высказываний, исчисление предикатов с одним одноместным предикатом) подобные алгоритмы известны.

Проверка выводимости формулы состоит в проверке того, является ли формула тавтологией или нет. Это можно легко установить по таблицам истинности. Но этот метод не обеспечивает построения вывода формулы.

Метод резолюций – классический алгоритм автоматического доказательства теорем. Для простоты изложения рассмотрим его для исчисления высказываний. Для любого множества формул Г и любой формулы А метод дает утвердительный ответ, если Г├ А, и дает отрицательный ответ, если неверно, что Г├ А.

Правило резолюций: , ├.

**Задание:** Разработать и программно реализовать вывод формул исчисления высказываний методом резолюций.

Протестировать разработанную систему на примере

**Пример работы программы:**

Строим отрицание формулы и формируем набор дизъюнктов. Вводим их в программу. Затем программа на основе правил резолюций строит логический вывод. После этого производится проверка на равенство нулю: если равно нулю, то программа пишет, что предложение доказуемо.

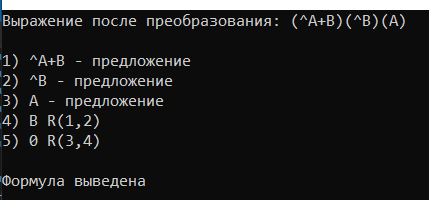


Рисунок 1. Пример работы программы

**Листинг программы:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data;

namespace lab2

{

class Program

{

static void Main(string[] args)

{

var pr3 = "(^A+B)(^B)(A)";

var pr5 = "(A)(^A^B)";

var inputString = pr3;

int skobki = 3;

Console.WriteLine("Выражение после преобразования: " + inputString + "\n");

var terms = inputString.Split(new[] { '(', ')' }, StringSplitOptions.RemoveEmptyEntries).ToList();

int a = 1;

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

{

Console.WriteLine(a + ") " + terms[i] + " - предложение");

a++;

}

var resultArguments = terms[1].Split(new[] { '+', ' ' }, StringSplitOptions.RemoveEmptyEntries).ToList();

int counter = 0;

while (true)

{

if (terms.Count == 0)

break;

var nexTermArguments = terms[0].Split(new[] { '+', ' ' }, StringSplitOptions.RemoveEmptyEntries);

var newResultArguments = new List<string>();

foreach (var argument1 in resultArguments)

{

foreach (var argument2 in nexTermArguments)

{

var newArgument = argument1 + argument2;

newResultArguments.Add(newArgument);

}

}

resultArguments = newResultArguments;

terms.Remove(terms[0]);

counter++;

if (counter == 1)

{

if (terms.Count > 0)

terms.Remove(terms[0]);

}

}

var sb = new StringBuilder();

for (var i = 0; i < resultArguments.Count; i++)

{

var argument = resultArguments[i];

sb.Append(" " + argument);

if (i + 1 != resultArguments.Count) sb.Append(" +");

}

//Console.WriteLine(sb.ToString());

var str = sb.ToString();

var expression = str.Split(new[] { '+', ' ' }, StringSplitOptions.RemoveEmptyEntries).ToList();

for (int i = 0; i < expression.Count; i++)

{

string buff = expression[i];

foreach (char el in buff) //буковки

{

string otricanie = "^" + el;

if (buff.Contains(otricanie))

{

buff = buff.Remove(buff.IndexOf('^'), 2);

if (buff.Contains(el))

{

expression[i] = "0";

}

}

}

}

int res = 0;

for (int i = 0; i < expression.Count; i++)

{

if (expression[i] != "0")

{

var argument = expression[i];

sb.Append(argument);

for (int k = 0; k < sb.Length; k++)

{

int first = -1;

int last = -1;

foreach (char ch in argument)

{

if (ch == '^') continue;

first = argument.IndexOf(ch);

last = argument.LastIndexOf(ch);

if (first != -1 && last != -1 && first != last)

argument = argument.Replace(ch.ToString(), "").Insert(k, ch.ToString());

}

}

res++;

Console.WriteLine("Результат: " + argument);

}

else

{

res = res + Convert.ToInt32(expression[i]);

}

}

int number = 1;

if (res == 0)

{

if (a > 3)

{

Console.WriteLine(a + ")" + resolution + " R(" + number + "," + (number+1) + ")");

a++;

number = number + 2;

}

Console.WriteLine(a + ") 0" + " R(" + number + "," + (number + 1) + ")");

Console.WriteLine("\nФормула выведена");

}

else Console.WriteLine("\nФормула не выводима");

Console.ReadLine();

}

}

}