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 программных систем  
  
Дисциплина  
**«Системы искусственного интеллекта»  
  
  
  
ОТЧЁТ**по лабораторной работе №1

Вариант № 3

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

Самара 2020

**Краткая теория**

*Высказывание* - утвердительное предложение, о котором в данной ситуации можно сказать, что оно истинно или ложно, но не то и другое одновременно.

Логическое значение сложного высказывания можно описать с помощью таблицы, называемой *таблицей истинности* (верхняя строка содержит обозначения высказываний, последующие строки – логическое значение высказываний).

***Операции над высказываниями****:*

* *Конъюнкция* - это сложное логическое выражение, которое считается истинным в том и только том случае, когда оба простых выражения являются истинными, во всех остальных случаях данное сложенное выражение ложно.
* *Дизъюнкция* - это сложное логическое выражение, которое истинно, если хотя бы одно из простых логических выражений истинно и ложно тогда и только тогда, когда оба простых логических выражения ложны.
* *Импликация* - это сложное логическое выражение, которое истинно во всех случаях, кроме как из истины следует ложь. То есть данная логическая операция связывает два простых логических выражения, из которых первое является условием (А), а второе (В) является следствием
* *Отрицание* - унарная операция над суждениями, результатом которой является суждение (в известном смысле) «противоположное» исходному.
* *Инверсия* - это сложное логическое выражение, если исходное логическое выражение истинно, то результат отрицания будет ложным, и наоборот, если исходное логическое выражение ложно, то результат отрицания будет истинным.
* *Эквивалентность* - это сложное логическое выражение, которое является истинным тогда и только тогда, когда оба простых логических выражения имеют одинаковую истинность.

***Основные свойства логических операций:***

* Коммутативность: (А∨B) ⇔ (B∨A) и (А∧B) ⇔ (B∧A).
* Ассоциативность: ((А∨В)∨C) ⇔ (А∨(В ∨С)) и ((А∧В)∧C) ⇔ (А∧(В ∧С)).
* Идемпотентность: А∨А ⇔А и А∧А ⇔А.
* Законы Моргана: ((А∨B)) ⇔ ( А∧B) и ( (А∧B)) ⇔ ( А∨B).
* Поглощение: А∨(А ∧В) ⇔А и А∧ (А ∨В) ⇔А.

**Задание:** Формализовать условие логической задачи. Разработать программу автоматического преобразования логического выражения.

Вариант 3: Семья из отца А матери B и трёх дочерей C, D и E купила телевизор. Условились, что в первый вечер смотреть передачи будут в таком порядке

1. Когда отец А смотрит передачу, то мать B делает то же.
2. Дочери D и B, обе или одна из них, смотрят передачу.
3. Из двух членов семьи – мать B и дочь C – смотрят передачу одна и только одна.
4. Дочери C и D или обе смотрят передачу, или обе не смотрят.
5. Если дочь E смотрит передачу, то отец A и дочь D делают то же.

Кто из членов семьи в этот вечер смотрит телевизор?

**Пример работы программы:**

Пусть высказывание X означает, что человек с именем X смотрит телевизор, а высказывание ^Х – не смотрит. Тогда 5 условий задачи можно представить как конъюнкцию импликаций:

(^A+B)(D+E)(B^C+^BC)(CD+^C^D)(^E+AD)=1

Подаём данную формулу на вход программе, которая поочерёдно перемножает все скобки и получает длинное выражение. Если у нас не может быть X и ^X вместе, то такие части нашего выражения программа обнуляет. В итоге все ненулевые ответы есть решения.

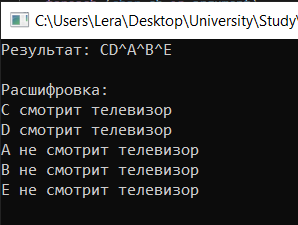


Рисунок 1. Пример работы программы

**Листинг программы:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data;

namespace lab1

{

class Program

{

static void Main(string[] args)

{

var inputString = "(^A+B)(D+E)(B^C+^BC)(CD+^C^D)(^E+AD)";

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

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";

}

}

}

}

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());

}

}

Console.WriteLine("Результат: " + argument + "\n\n" + "Расшифровка:");

foreach (char el in argument)

{

if (el == '^') continue;

else

{

string otricanie = "^" + el;

if (argument.Contains(otricanie))

{

Console.WriteLine(el + " не смотрит телевизор");

}

else Console.WriteLine(el + " смотрит телевизор");

}

}

}

}

Console.ReadLine();

}

}

}