**МОЛДАВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет Математики и Информатики**

**Департамент Информатики**

Лабораторная работа № 2

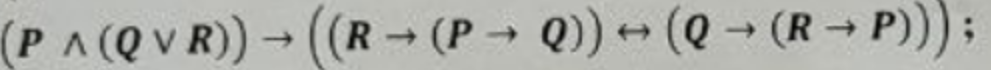
Создание программы для решения и анализа логической функции

Проверил: профессор, др. Cuciuc Natalia

Выполнил: Mamaliga Artur grupa I2302

Кишинев, 2024

Написанная мной программа решает только одну логическую функцию:  
(P ^ (Q V R) -> ((R -> (P -> Q)) <-> (Q -> (R -> P))) )



Программа написана на С++, с использованием библиотеки iostream для упрощения ввода вывода.

Переменные задаются при помощи массивов

bool p[SIZEARR] = {0,0,0,0,1,1,1,1};

  bool q[SIZEARR] = {0,0,1,1,0,0,1,1};

  bool r[SIZEARR] = {0,1,0,1,0,1,0,1};

  bool res[SIZEARR] = {0};

Так как в СИ нет встроенных функций импликаций и эквивалентности я их имплементировал сам.

bool implication(bool a, bool b)

{

    return !a || b;

}

bool equivalence(bool a, bool b)

{

  return a == b;

}

и саму исходную логическую функцию написал в функции

bool func1(bool p, bool q, bool r)

{

  bool res = implication(p && (q || r), equivalence(implication( r , implication(p, q) ), implication(q, implication(r, p) )) );

  return res;

}

Функция просчитывает следуя порядку действий всю логическую функцию и на выходе выдает результат 1 или 0, в зависимости от значения переменных переданных ей.

Для того чтобы посчитать все ответы для всех возможных значений в таблице значений, я циклически вызываю функцию заполняя массив.

Проверка функции на выполнимость:

bool satisfiability(bool\* res)

{

  for (int i = 0; i < SIZEARR; i++){

    if(res[i]==1)

      return 1;

  }

  return 0;

}

Циклически проходят массив результатов функции и если есть хотябы одна 1, значит функция выполнимая.  
Также с опровержимостью, только если есть хотябы один 0

bool unsatisfiability(bool\* res)

{

  for (int i = 0; i < SIZEARR; i++){

    if(res[i]==0)

      return 1;

  }

  return 0;

}

Проверка на тавтологию и противоречия просто вызываю с отрицанием проверку на опровержимость и выполнимость.

bool tautology(bool\* res)

{

  return !(unsatisfiability(res));

}

bool contradiction(bool\* res)

{

  return !(satisfiability(res));

}

Проверка на фиктивные переменные происходит с выбором номера переменной(первая, вторая, третья)

bool is\_dummy(bool\* input, int v\_count)

{

  bool res=1;

  if(v\_count==1)

    goto v1;

  else if(v\_count==2)

    goto v2;

  else if(v\_count==3)

    goto v3;

  else

    return -1;

v1:

  for(int i = 0; i+4<SIZEARR;i++){

    res = res && (input[i]==input[i+4]);

  }

  return res;

v2:

  for (int i = 0; i <= 4; i+=4){

    res = res && (input[i] == input[i+2]);

    res = res && (input[i+1] == input[i+3]);

  }

  return res;

v3:

  for (int i = 0; i < SIZEARR; i+=2){

    res = res && (input[i]==input[i+1]);

  }

  return res;

}

Циклически обхожу массив результатов и соответсвенно сравниваю нужные строки результатов и если значения результата не меняются в строчках, где не меняются значения переменных кроме той которую проверяем, значит переменная фиктивная и не влияет на результат.