**МОЛДАВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет Математики и Информатики**

**Департамент Информатики**

Лабораторная работа № 3

Создание программы для решения и анализа логической функции

Проверил: профессор, др. Cuciuc Natalia

Выполнил: Mamaliga Artur grupa I2302

Кишинев, 2024

Написанная мной программа решает 4 логические функции:

1. x || (y && !y) || z
2. (x ^ (y V z) -> ((z -> (x -> y)) <-> (y -> (z -> x))) )
3. ((x | y)+(y|z))->(x v (!(y pirs z)))
4. (x -> y) pirs z) | (!(x -> y))

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

Переменные задаются при помощи массивов, а заполняются массивы при помощи трех вложенных циклов, функция также принимает колбэк функцию которая является логической функцией

void fill\_table(int \*x\_arr, int \*y\_arr, int \*z\_arr, int \*f\_arr, int size, int(\*f)(int x, int y, int z))

{

int values[2] = {0, 1};

int i = 0;

for (int x : values){

for (int y : values){

for (int z : values){

x\_arr[i] = x;

y\_arr[i] = y;

z\_arr[i] = z;

f\_arr[i] = f(x, y, z);

i++;

}

}

}

}

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;

}

логические функции вынес в отдельные функции

int f1(int x, int y, int z)

{

return (x || (y && !y) || z);

}

int f2(int x, int y, int z)

{

return (equivalence(x && (implication(y,!z) || z),implication(equivalence(x,z || !y),y && !y)));

}

int f3(int x, int y, int z)

{

return implication(!(x && y) ^ !(y && z), (x || !(!(y || z))));

}

int f4(int x, int y, int z)

{

return (!((implication(x,y) || z) && (!(implication(x,y)))));

}

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

Функция для вывода таблицы в консоль:

void print\_table(int \*x\_arr, int \*y\_arr, int \*z\_arr, int \*f\_arr, int size)

{

  cout << "Table of function's values:" << endl;

  cout << "x | y | z | f" << endl;

  cout << "-------------" << endl;

  for (int i = 0; i < size; i++){

    cout << x\_arr[i] << " | " << y\_arr[i] << " | " << z\_arr[i] << " | " << f\_arr[i] << endl;

  }

  cout << endl;

}

Проверка на фиктивные переменные во вложенном цикле, который проверяет в if чтобы изменилась только проверяемая переменная а остальные остались теми же и тогда проверит значение функции в данных двух позициях

void is\_dummy(int \*x\_arr, int \*y\_arr, int \*z\_arr, int \*f\_arr, int size)

{

  int xdummy = 1;

  int ydummy = 1;

  int zdummy = 1;

  for (int i = 0; i < size-1; i++){

    int x = x\_arr[i];

    int y = y\_arr[i];

    int z = z\_arr[i];

    int f = f\_arr[i];

    for(int j = i+1; j < size; j++){

      if (x != x\_arr[j] && y == y\_arr[j] && z == z\_arr[j]){

        xdummy = xdummy && (f == f\_arr[j]);

      }

      if (y != y\_arr[j] && x == x\_arr[j] && z == z\_arr[j]){

        ydummy = ydummy && (f == f\_arr[j]);

      }

      if (z != z\_arr[j] && y == y\_arr[j] && x == x\_arr[j]){

        zdummy = zdummy && (f == f\_arr[j]);

      }

    }

  }

На принадлежность классу функций T0 T1, просто проверяю первое значение на равенство 0 и последнее 1

int t1(int \*f\_arr, int size){

int res = f\_arr[7];

cout<<"t1: "<<res<<endl;

return res;

}

int t0(int \*f\_arr, int size){

int res = !f\_arr[0];

cout<<"t0: "<<res<<endl;

return res;

}

На монотонность проверяю в несколько этапов. Проверяю условия для монотонности. Во первых сравниваю наборы значений по типу ai <= ai+1, если соответствует данному условию следую дальше, а иначе перехожу к следующему набору. Далее если выполнилось предыдущее условие проверяю если выполняется условие f(ai)<=f(ai+1), если данное условие выполняется для всех наборов удовлетворяющих предыдущему условию это значит что функция монотонна.

void is\_monotonic(int \*x\_arr, int \*y\_arr, int \*z\_arr, int \*f\_arr, int size){

int is\_monotonic = 1;

for (int i = 0; i < size-1; i++){

if (!(x\_arr[i]<=x\_arr[i+1] && y\_arr[i]<=y\_arr[i+1] && z\_arr[i]<=z\_arr[i+1])){

continue;

}

is\_monotonic = is\_monotonic && (f\_arr[i]<=f\_arr[i+1]);

if (is\_monotonic==0){

cout<<"not monotonic"<<endl;

return;

}

}

cout<<"monotonic"<<endl;

return;

}

для анализа каждой функции создал отдельную функцию которая принимает параметрами ссылку на массивы значений переменных и результатов функции для разных наборов и указатель на функцию логической функции.

void analyze\_func(int \*x\_arr, int \*y\_arr, int \*z\_arr, int \*f\_arr, int size, int(\*f)(int x, int y, int z))

{

fill\_table(x\_arr, y\_arr, z\_arr, f\_arr, 8,f);  
 print\_table(x\_arr, y\_arr, z\_arr, f\_arr, 8);

is\_dummy(x\_arr, y\_arr, z\_arr, f\_arr, 8);

t1(f\_arr, 8);

t0(f\_arr, 8);

is\_monotonic(x\_arr, y\_arr, z\_arr, f\_arr, 8);

}

Вывод программы:

x || (y && !y) || z

Table of function's values:

x | y | z | f

-------------

0 | 0 | 0 | 0

0 | 0 | 1 | 1

0 | 1 | 0 | 0

0 | 1 | 1 | 1

1 | 0 | 0 | 1

1 | 0 | 1 | 1

1 | 1 | 0 | 1

1 | 1 | 1 | 1

find dummy variables

x is not dummy

y is dummy

z is not dummy

t1: 1

t0: 1

monotonic

(P ^ (Q V R) -> ((R -> (P -> Q)) <-> (Q -> (R -> P))) )

Table of function's values:

x | y | z | f

-------------

0 | 0 | 0 | 0

0 | 0 | 1 | 0

0 | 1 | 0 | 1

0 | 1 | 1 | 0

1 | 0 | 0 | 0

1 | 0 | 1 | 0

1 | 1 | 0 | 1

1 | 1 | 1 | 0

find dummy variables

x is dummy

y is not dummy

z is not dummy

t1: 0

t0: 1

not monotonic

((x | y)+(y|z))->(x v (!(y pirs z)))

Table of function's values:

x | y | z | f

-------------

0 | 0 | 0 | 1

0 | 0 | 1 | 1

0 | 1 | 0 | 1

0 | 1 | 1 | 1

1 | 0 | 0 | 1

1 | 0 | 1 | 1

1 | 1 | 0 | 1

1 | 1 | 1 | 1

find dummy variables

x is dummy

y is dummy

z is dummy

t1: 1

t0: 0

monotonic

((x -> y) pirs z) | (!(x -> y))

Table of function's values:

x | y | z | f

-------------

0 | 0 | 0 | 1

0 | 0 | 1 | 1

0 | 1 | 0 | 1

0 | 1 | 1 | 1

1 | 0 | 0 | 1

1 | 0 | 1 | 0

1 | 1 | 0 | 1

1 | 1 | 1 | 1

find dummy variables

x is not dummy

y is not dummy

z is not dummy

t1: 1

t0: 0

not monotonic