CISP 440

Austin Smothers

Homework 10.42

/\*

LogicEq.cpp - Logical Equivalence

Homework 10.42

Austin Smothers

Professor Ross

CISP 440

March 2017

This program will test logical equivalence functions

\*/

/\*

Header File Declarations

\*/

#pragma warning(disable:4800)

#include <iostream>

#include <iomanip> //included for setw()

using namespace std;

/\*

Global Variable Declaration

\*/

//none needed

/\*

Global Function Declaration

\*/

//Organized by Precedence for coder's ease of memory

/\*1\*/ bool not(bool a); //!a

/\*2\*/ bool and(bool a, bool b); //a & b

/\*3\*/ bool xor(bool a, bool b); //a ^ b (a or b, but not both)

/\*4\*/ bool or(bool a, bool b); //a v b

/\*5\*/ bool implication(bool a, bool b); //a -> b

/\*6\*/ bool biconditional(bool a, bool b); //a <-> b

/\*7\*/ bool maybe(bool a, bool b); //a ? b

/\*8\*/ bool because(bool a, bool b); //a @ b

void printsamples();

void testA(bool p, bool q); //p | q = ~(~p & ~q)

void testB(bool p, bool q, bool r); //p & (q -> r) = (p & q) ^ r

void testC(bool p, bool q); //p <-> q = ~(p ^ q)

void testD(bool p, bool q, bool r); //(p <-> q) -> r = p & (~q | r)

void testE(bool p, bool q, bool r); //~(p -> (q & r)) = p & ~(q & r)

void testF(bool p, bool q); //p | (p ? q) = p & (p @ q)

/\*

The Main Function

\*/

int main()

{

bool p = false;

bool q = false; //T/F variables that will be input into equivalence functions

bool r = false;

printsamples();

testC(p, q);

testD(p, q, r);

testE(p, q, r);

testF(p, q);

return 0;

}

/\*

Boolean Expression Function Definitions

\*/

//not (!)

bool not(bool a)

{

return !a;

}

//and (&)

bool and(bool a, bool b)

{

return (a && b);

}

//xor (^)

bool xor(bool a, bool b)

{

return (!(a == b));

}

//or (v)

bool or(bool a, bool b)

{

return (a || b);

}

//Implication (a -> b)

bool implication(bool a, bool b)

{

return ((!a) || (b));

}

//biconditional (<->)

bool biconditional(bool a, bool b)

{

return (a == b);

}

//maybe (?)

bool maybe(bool a, bool b)

{

return ((b) && (!a));

}

//because (@)

bool because(bool a, bool b)

{

return a;

}

/\*

Logical Equivalence Function Definitions

\*/

/\*Tests A & B are covered in the printsamples() function

which prints the sample output required by the professor\*/

void printsamples()

{

bool p = false;

bool q = false;

bool r = false;

int counter = 0;

//the first sample output

cout << "Equivalence Test A:" << endl;

cout << setw(1) << "p" << setw(3) << "q" << setw(3) << "|"

<< setw(9) << "p or q" << setw(3) << "|"

<< setw(14) << "~(~p and ~q)" << endl;

while (!(p && q))

{

cout << setw(1) << p << setw(3) << q << setw(3) << "|"

<< setw(6) << or(p, q) << setw(6) << "|"

<< setw(9) << not(and(not(p), not(q))) << endl;

counter++;

p = counter & 1;

q = counter & 2;

}

cout << setw(1) << p << setw(3) << q << setw(3) << "|"

<< setw(6) << or(p, q) << setw(6) << "|"

<< setw(9) << not(and(not(p), not(q))) << endl

<< "These statements are equivalent" << endl << endl;

counter = 0;

p = false;

q = false;

//the second sample output

cout << "Equivalence Test B:" << endl;

cout << setw(1) << "p" << setw(3) << "q" << setw(3) << "r" << setw(3) << "|"

<< setw(17) << "p and (q -> r)" << setw(3) << "|"

<< setw(18) << "(p and q) xor r" << endl;

while (!(p && (q && r)))

{

cout << setw(1) << p << setw(3) << q << setw(3) << r << setw(3) << "|"

<< setw(10) << and(p, implication(q, r)) << setw(10) << "|"

<< setw(10) << xor(and(p, q), r) << endl;

counter++;

r = counter & 1;

q = counter & 2;

p = counter & 4;

}

cout << setw(1) << p << setw(3) << q << setw(3) << r << setw(3) << "|"

<< setw(10) << and(p, implication(q, r)) << setw(10) << "|"

<< setw(10) << xor(and(p, q), r) << endl

<< "These statements are not equivalent" << endl << endl;

}

//p <-> q = ~(p ^ q)

void testC(bool p, bool q)

{

int counter = 0;

cout << "Equivalence Test C:" << endl;

cout << setw(1) << "p" << setw(3) << "q" << setw(3) << "|"

<< setw(10) << "p <-> q" << setw(3) << "|"

<< setw(11) << "~(p ^ q)" << endl;

while (!(p && q))

{

cout << setw(1) << p << setw(3) << q << setw(3) << "|"

<< setw(7) << biconditional(p, q) << setw(6) << "|"

<< setw(8) << not(xor(p, q)) << endl;

counter++;

p = counter & 1;

q = counter & 2;

}

cout << setw(1) << p << setw(3) << q << setw(3) << "|"

<< setw(7) << biconditional(p, q) << setw(6) << "|"

<< setw(8) << not(xor(p, q)) << endl

<< "These statements are equivalent" << endl << endl;

}

//(p <-> q) -> r = p & (~q | r)

void testD(bool p, bool q, bool r)

{

int counter = 0;

cout << "Equivalence Test D:" << endl;

cout << setw(1) << "p" << setw(3) << "q" << setw(3) << "r" << setw(3) << "|"

<< setw(17) << "(p <-> q) -> r" << setw(3) << "|"

<< setw(15) << "p & (~q | r)" << endl;

while (!(p && (q && r)))

{

cout << setw(1) << p << setw(3) << q << setw(3) << r << setw(3) << "|"

<< setw(10) << implication(biconditional(p, q), r) << setw(10) << "|"

<< setw(8) << and(p, or(not(q), r)) << endl;

counter++;

r = counter & 1;

q = counter & 2;

p = counter & 4;

}

cout << setw(1) << p << setw(3) << q << setw(3) << r << setw(3) << "|"

<< setw(10) << implication(biconditional(p, q), r) << setw(10) << "|"

<< setw(8) << and(p, or(not(q), r)) << endl

<< "These statements are not equivalent" << endl << endl;

}

//~(p -> (q & r)) = p & ~(q & r)

void testE(bool p, bool q, bool r)

{

int counter = 0;

cout << "Equivalence Test E:" << endl;

cout << setw(1) << "p" << setw(3) << "q" << setw(3) << "r" << setw(3) << "|"

<< setw(18) << "~(p -> (q & r))" << setw(3) << "|"

<< setw(15) << "p & ~(q & r)" << endl;

while (!(p && (q && r)))

{

cout << setw(1) << p << setw(3) << q << setw(3) << r << setw(3) << "|"

<< setw(11) << not(implication(p, and(q, r))) << setw(10) << "|"

<< setw(8) << and(p, not(and(q, r))) << endl;

counter++;

r = counter & 1;

q = counter & 2;

p = counter & 4;

}

cout << setw(1) << p << setw(3) << q << setw(3) << r << setw(3) << "|"

<< setw(11) << not(implication(p, and(q, r))) << setw(10) << "|"

<< setw(8) << and(p, not(and(q, r))) << endl

<< "These statements are equivalent" << endl << endl;

}

//p | (p ? q) = p & (p @ q)

void testF(bool p, bool q)

{

int counter = 0;

cout << "Equivalence Test F:" << endl;

cout << setw(1) << "p" << setw(3) << "q" << setw(3) << "|"

<< setw(14) << "p | (p ? q)" << setw(3) << "|"

<< setw(14) << "p & (p @ q)" << endl;

while (!(p && q))

{

cout << setw(1) << p << setw(3) << q << setw(3) << "|"

<< setw(9) << or(p, maybe(p, q)) << setw(8) << "|"

<< setw(9) << and(p, because(p, q)) << endl;

counter++;

p = counter & 1;

q = counter & 2;

}

cout << setw(1) << p << setw(3) << q << setw(3) << "|"

<< setw(9) << or(p, maybe(p, q)) << setw(8) << "|"

<< setw(9) << and(p, because(p, q)) << endl

<< "These statements are not equivalent" << endl << endl;

}

