Heavens' Light is Our Guide


# Rajshahi University of Engineering and Technology

CSE-2102

Lab-1

---

# Discrete Mathematics Sessional

---

*Submitted To:*
Suhrid Shakhar Ghosh
Asst. Professor
Dept. of Computer Science & Engineering

*Submitted By :*
Kaif Ahmed Khan
ID: 2103163
Dept. of Computer Science & Engineering

December 10, 2023

# 1 Truth Table Generation

Generate truth table for the following propositions:

$$p,\ q,\ p \wedge q,\ p \vee q,\ p \rightarrow q,\ p \leftrightarrow q,\ p \oplus q$$

## 1.1 Source Code

```cpp
#include <iomanip>
#include <iostream>

using namespace std;

#define T cout << "T\t"
#define F cout << "F\t"

int main() {
  int p, q;
  cout << "p\t"
       << "q\t"
       << "p^q\t"
       << "p\\/q\t"
       << "p->q\t"
       << "p<->q\t"
       << "p(+)q\t" << endl;
  cout << "-------------------------------------------------------" << endl;
  for (p = 1; p >= 0; p--) {
    for (q = 1; q >= 0; q--) {
      p ? T : F;
      q ? T : F;
      (p && q) ? T : F;
      (p || q) ? T : F;
      (!p || q) ? T : F;
      ((!p || q) && (!q || p)) ? T : F;
      (p ^ q) ? T : F;
      cout << endl;
    }
  }
}
```

## 1.2 Output

```
 .\truth-table-1.exe
p       q       p^q     p\/q    p->q    p<->q   p(+)q
-------------------------------------------------------
T       T       T       T       T       T       F
T       F       F       T       F       F       T
F       T       F       T       T       F       T
F       F       F       F       T       T       F
```

## 1.3 Analysis

From the output, it is clear that the code successfully generates the truth tables for the given logic. The code assumes that logic 1 is True (T) and logic 0 is False (F).

To implement the $p \wedge q$ I have used the logical AND (&&) operator in C++. Hence the result is only true when both $p$ and $q$ is true; and false otherwise.
$p \vee q$ is implemented using the logical OR (||) operator.
For $p \rightarrow q$, it is easier to use the equivalent logic $\neg p \vee q$.
In case of $p \leftrightarrow q$, the equivalent logic $(p \rightarrow q) \wedge (q \rightarrow p)$ is applied.
Finally, $p \oplus q$ is implemented using the XOR (^) operator in C++.

Using nested for loop, starting the value of $p$ from 1 to 0, in each iteration $q$ has two value 1 and 0. Thus, the truth values for $p$ and $q$ is generated dynamically.

# 2 Logical Equivalence Checking

Generate a truth table to find whether the following two are equivalent or not:

$$p \rightarrow (q \wedge r)$$

$$(p \rightarrow q) \wedge r$$

## 2.1 Source Code

```
#include <iomanip>
#include <iostream>
using namespace std;

#define T cout << "T\t"
#define F cout << "F\t"

int main() {
  int p, q, r;
  cout << "p\t"
       << "q\t"
       << "r\t"
       << "p->(q^r)\t"
       << "(p->q)^r\t" << endl;
  cout << "-------------------------------------------------" << endl;
  int logic1, logic2;
  int isEquivalent = 1;
  for (p = 0; p <= 1; p++) {
    for (q = 0; q <= 1; q++) {
      for (r = 0; r <= 1; r++) {
        logic1 = (!p || (q && r));
        logic2 = ((!p || q) && r);
        p ? T : F;
        q ? T : F;
        r ? T : F;
        cout << setw(5);
        logic1 ? T : F;
        cout << setw(14);
        logic2 ? T : F;
        if (logic1 != logic2)
          isEquivalent = 0;
        cout << endl;
      }
```

```
34      }
35    }
36    cout << endl;
37    if (!isEquivalent) {
38      cout << "The two logics are not equivalent\n";
39    } else {
40      cout << "The two logics are equivalent\n\n";
41    }
42  }
```

## 2.2 Output

```
 .\truth-table-2.exe
p       q       r       p->(q^r)        (p->q)^r
-----------------------------------------------
T       T       T           T               T
T       T       F           F               F
T       F       T           F               F
T       F       F           F               F
F       T       T           T               T
F       T       F           T               F
F       F       T           T               T
F       F       F           T               F

The two logic are not equivalent
```

## 2.3 Analysis

The truth table shows that, $p \to (q \wedge r)$ is not equivalent to $(p \to q) \wedge r$.
In the code, implication is implemented using the equivalent logic of implication $\neg p \vee q$, and the conjunction is implemented using the logical AND (&&) operator in C++.

To generate the truth values of $p, q, r$, nested for loop is used.

To check whether the two logic are equivalent or not, initially `isEquivalent` variable was set to true. In each iteration both logic is checked whether they are equal. If in any case, the two logic output doesn't match the `isEquivalent` is set to false or 0.

Finally, the `isEquivalent` variable is checked and relevant output,in this case "not equivalent" is printed in the console.