

Boolean Logic

Introduction

Boolean Logic was invented by George Boole. Boole was the first Professor of Mathematics at Queen's College Cork. During the latter half of the nineteenth century he laid the foundations for a system of mathematical expression, which formed the basis for all modern computer languages.

The basis behind Boolean Logic is pure simplicity. Either something is right or it is wrong. This provides us with two states with which to work, true and false. To represent true and false in numerical fashion Boole decided to use the binary number system and represent false as 0 and true as 1.

Boolean Operators

For those of us who are familiar with programming languages, we will have come across the idea of logical operators such as NOT, AND, OR, etc. These same operators exist in Boolean Logic. NOT is sometimes referred to as negation, AND can be referred to as conjunction and OR is sometimes known as disjunction. We also have more operators than this, which we can use to construct what are known as logic sentences or statements. Below is a table of the operators that we will be dealing with and their corresponding symbols.

\wedge	AND
\vee	OR
\rightarrow	IMPLIES
\neg	NOT
$\dot{\wedge}$	EXCLUSIVE OR
\ll	BICONDITIONAL

AND

The logical AND operator (\wedge) is a binary operator. It needs two inputs a, b. The result of a logical AND is true if both a and b are true and is false otherwise.

OR

The logical OR operator (\vee) is also a binary operator. The result of a logical OR is true if a is true or b is true or a and b are true. If a and b are false then the result is false.

IMPLIES

The logical IMPLIES operator (\rightarrow) is also binary. The result of a logical IMPLIES is true if b is true or both a and b are the same.

NOT

The logical NOT operator (\neg) is a unary operator, it only needs one input. It takes that input and returns the opposite Boolean value. So if a was true then $\neg a$ would be false.

EXCLUSIVE OR

The EXCLUSIVE OR operator (\oplus) is a binary operator. The result of an EXCLUSIVE OR is true if a or b is true but false if both are true.

BICONDITIONAL

The BICONDITIONAL operator (\leftrightarrow) is also a binary operator. The result of a BICONDITIONAL operation is true only when both inputs are the same value.

Truth Tables

We have seen the set of operators that we are going to deal with and now we need some form of representation for these operators, so that we can actually see in Boolean Logic form what each of them means. The representation used for this is the notion of a truth table. A truth table takes the number of inputs used in a logical operation and puts them as column headers. Another column is then added which is the result of the particular operation. In each column there is a number of rows, the number of rows in a truth table is always 2^{inputs} . The rows are filled with logic 1's and 0's. It will become clear when we construct our first example.

AND

The truth table for a logical AND operation which takes two inputs (A, B) is as follows:

A	B	A ^ B
0	0	0
0	1	0
1	0	0
1	1	1

OR

The truth table for a logical OR operation which takes two inputs (A, B) is as follows:

A	B	A v B
0	0	0
0	1	1
1	0	1
1	1	1

IMPLIES

The truth table for a logical IMPLIES operation which takes two inputs (A, B) is as follows:

A	B	A → B
0	0	1
0	1	1
1	0	0
1	1	1

NOT

The truth table for a NOT operation taking only one input (A) is as follows:

A	$\neg A$
0	1
1	0

EXCLUSIVE OR

The truth table for an EXCLUSIVE OR operation taking two inputs (A, B) is as follows:

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

BICONDITIONAL

The truth table for a BICONDITIONAL operation taking two inputs (A, B) is as follows:

A	B	$A \leftrightarrow B$
0	0	1
0	1	0
1	0	0
1	1	1

So now that we have seen how truth tables are constructed and how we read the steps taken to produce an output, we should be able to move onto creating logic sentences from truth tables. We should be able to express a non-obvious truth table (such as IMPLIES) in terms of a combination of other logical operations.

Logical Sentences

First we are going to look at some of the truth tables that we have already created. From these truth tables we are going to create a logical sentence that represents the operation performed in the truth table but never uses that logical operator in the sentence.

To start we will examine the IMPLIES truth table.

A	B	$A \rightarrow B$
0	0	1
0	1	1
1	0	0
1	1	1

By examining the truth table we can see that the result of a logical IMPLIES is true whenever **B** is true or when the negation of A ($\neg A$) is true. Therefore we can now construct the logical sentence which is an alternative representation of the IMPLIES operation.

$$A \rightarrow B = \neg A \vee B$$

So let us prove this using truth tables:

A	$\neg A$	B	$\neg A \vee B$
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1

We can see from this truth table that the result we get is the same as that for a logical IMPLIES therefore we can conclude that our alternative statement is true.

So another example, let us examine the BICONDITIONAL truth table and try to express it in a different form.

A	B	$A \leftrightarrow B$
0	0	1
0	1	0
1	0	0
1	1	1

We can see in this truth table that the result is true whenever both A and B have the same value. Looking at this in terms of logic we can conclude that the result is true whenever the negation of A is true and the negation of B is true or when A is true and B is true. Therefore we can form our new sentence from these facts.

$$A \leftrightarrow B = (\neg A \wedge \neg B) \vee (A \wedge B)$$

Again we will prove this using a truth table.

A	B	$\neg A$	$\neg B$	$\neg A \wedge \neg B$	$A \wedge B$	$(\neg A \wedge \neg B) \vee (A \wedge B)$
0	0	1	1	1	0	1
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	1	0	0	0	1	1

So we can see in a roundabout way that we can arrive at the BICONDITIONAL operation by using a combination of NOT, AND and OR operations.

Similarly if you were presented with a meaningless truth table and asked to construct the logical sentence that represented it, then you would follow the same procedure.

A	B	F
0	0	1
0	1	0
1	0	1
1	1	0

From this table we could conclude the following logical sentence:

$$F = (\neg A \wedge \neg B) \vee (A \wedge \neg B)$$

It's only a matter of examining the truth table and identifying what criteria are in place for a true result.

