

# 19

There is a more comprehensive treatment of Boolean Algebra near the top of [cargalmathbooks.com](http://cargalmathbooks.com)

## Boolean and Set

## Algebra

It has been known for sometime in mathematics that set algebras and Boolean<sup>1</sup> algebras are different perspectives on the same thing. The treatment of sets here is informal and is known as *naive set theory*. Most of the time naive set theory is sufficient for the purposes of even professional mathematicians. Those familiar with this will want to skip the first part of the chapter. However, later there is some useful material not usually found in texts.

### Set Algebras

A set is a collection of objects. This is not a formal definition but a casual definition. A formal definition of *sets* is deceptively difficult and is unnecessary for our purposes.

### Boolean Algebras

A Boolean algebra is a logic algebra. The variables take on two values corresponding to truth (1 or T) and false (0 or F).

---

<sup>1</sup>After George Boole (1815-1864).

The Universal set can be denoted by U or by 1.

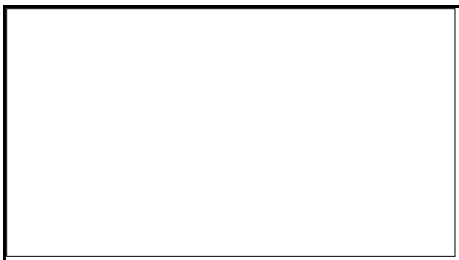
Truth is denoted by T or by the integer 1.



**Figure 1** The Universal Set

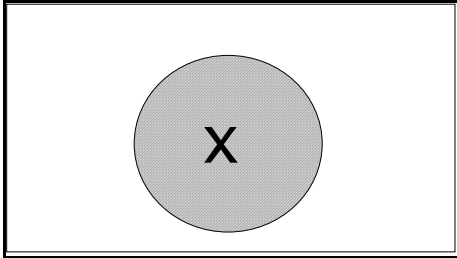
The empty set is denoted by  $\emptyset$  or 0. The empty set is sometimes known as the *null set*. (Think of an empty file cabinet, or equivalently a college administrator's mind.)

False is denoted by F or by the integer 0.



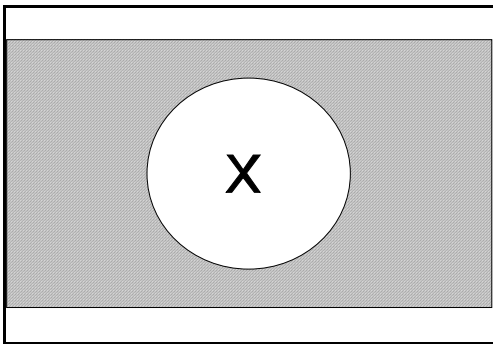
**Figure 2** The Empty Set

A set is a collection of objects. (In this case it is represented by the interior of a circle.)



**Figure 3** A Set X

The **complement** of a set is the collection of all objects not in that set. (If the set is the interior of the circle, the complement is the outside). The complement of set X is denoted by  $\bar{X}$  or by  $X'$ .



**Figure 4** The Complement of the Set X

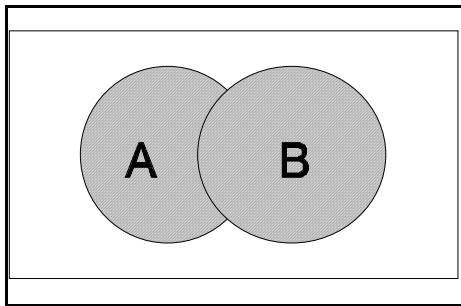
A Boolean variable X can take on either of two values 1 (T) or 0 (F).

**Not X** is denoted by  $\sim X$ .

If  $X = 1$  then  $\sim X = 0$ .

If  $X = 0$  then  $\sim X = 1$ .

The **union** of set A and set B is denoted  $A \cup B$  (sometimes  $A + B$ ).



**Figure 5**  $A \cup B$

**A or B** is denoted by  $A \vee B$  (sometimes  $A + B$ ).

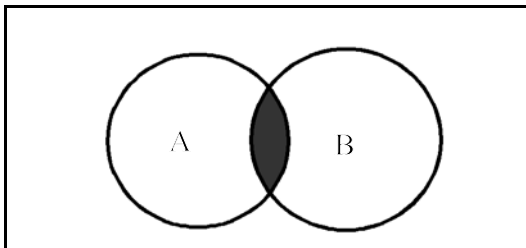
$$A = 1 \text{ and } B = 1 \Rightarrow (\text{implies}) A \vee B = 1.$$

$$A = 1 \text{ and } B = 0 \Rightarrow A \vee B = 1.$$

$$A = 0 \text{ and } B = 1 \Rightarrow A \vee B = 1.$$

$$A = 0 \text{ and } B = 0 \Rightarrow A \vee B = 0.$$

The **intersection** of set A and set B is denoted  $A \cap B$  (sometimes  $A \cdot B$ ).



**Figure 6**  $A \cap B$

**A and B** is denoted by  $A \wedge B$  (sometimes  $A \cdot B$ ).

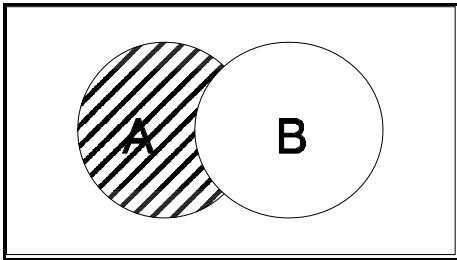
$$A = 1 \text{ and } B = 1 \Rightarrow A \wedge B = 1.$$

$$A = 1 \text{ and } B = 0 \Rightarrow A \wedge B = 0.$$

$$A = 0 \text{ and } B = 1 \Rightarrow A \wedge B = 0.$$

$$A = 0 \text{ and } B = 0 \Rightarrow A \wedge B = 0.$$

The difference of set A and set B is denoted  $A - B$ .



**Figure 7**  $A - B$ .

There is no (common) logic operator corresponding to the set operation  $-$ .

Essentially A minus B is A **and**  $\sim B$ :

$$A \cdot \bar{B}$$

# The Laws of Sets and Logic

## Associativity

$$(A \cup B) \cup C = A \cup (B \cup C)$$

$$(A \cap B) \cap C = A \cap (B \cap C)$$

## Commutativity

$$A \cup B = B \cup A$$

$$A \cap B = B \cap A$$

## Negation

$$\bar{U} = \emptyset$$

$$\bar{\emptyset} = U$$

$$A - B = A \cap \bar{B}$$

## Associativity

$$(A \vee B) \vee C = A \vee (B \vee C)$$

$$(A \wedge B) \wedge C = A \wedge (B \wedge C)$$

## Commutativity

$$A \vee B = B \vee A$$

$$A \wedge B = B \wedge A$$

## Negation

$$\sim 1 = 0 \quad (\sim T = F)$$

$$\sim 0 = 1$$

(Again, there is no common logic relation corresponding to  $A - B$ .)

**Double Negation**

$$\overline{\overline{X}} = X$$

**Double Negation**

$$\sim\sim X = X$$

**Distribution**

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$

**Distribution**

$$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$$

$$A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$$

**Complementarity**

$$X \cup \overline{X} = U$$

$$X \cap \overline{X} = \emptyset$$

**Complementarity**

$$X \vee (\sim X) = 1$$

$$X \wedge (\sim X) = 0$$

**Absorption**

$$X \cup X = X$$

$$X \cap X = X$$

$$X \cup U = U$$

$$X \cap \emptyset = \emptyset$$

**Absorption**

$$X \vee X = X$$

$$X \wedge X = X$$

$$X \vee 1 = 1$$

$$X \wedge 0 = 0$$

**Identity**

$$X \cap U = X$$

$$X \cup \emptyset = X$$

**Identity**

$$X \wedge 1 = X$$

$$X \vee 0 = X$$

**DeMorgan's Laws**

$$\overline{A \cap B} = \overline{A} \cup \overline{B}$$

$$\overline{A \cup B} = \overline{A} \cap \overline{B}$$

**DeMorgan's Laws**

$$\sim(A \wedge B) = (\sim A) \vee (\sim B)$$

$$\sim(A \vee B) = (\sim A) \wedge (\sim B)$$

Of the above laws the distributive laws and the DeMorgan laws are the least intuitive and perhaps the most useful. It is a good idea to memorize both of these laws.

Earlier, to define the logic operator **or** I used:

$$A = 1 \text{ and } B = 1 \Rightarrow A \vee B = 1.$$

$$A = 1 \text{ and } B = 0 \Rightarrow A \vee B = 1.$$

$$A = 0 \text{ and } B = 1 \Rightarrow A \vee B = 1.$$

$$A = 0 \text{ and } B = 0 \Rightarrow A \vee B = 0.$$

Frequently a *truth table* is used instead. The truth table for **or** is:

<b>A or B</b>	1	0
1	1	1
0	1	0

The 1 and the 0 at the left are the possible values for A; the 1 and the 0 at the top are the possible values for B.

Recapitulating *and* and *not* we get:

<b>Not A</b>	1
1	0
0	1

<b>A and B</b>	1	0
1	1	0
0	0	0

Three other important logical relations are XOR, Equiv, and Implies.

XOR represents the exclusive-or relation and is represented by  $\oplus$ .  $A \oplus B$  means A or B but not both.

<b>A XOR B</b>	1	0
1	0	1
0	1	0

- **Exercise 1** Draw the set diagram for A XOR B. By a set diagram I mean a diagram like the diagrams for  $A \cup B$  and for  $A \cap B$ . These types of diagrams are known as *Venn diagrams*.

EQUIV represents the equivalence relation and is represented by  $\equiv$ .  $A \equiv B$  means A and B, or not A and not B. Note that  $A \equiv B$  is true if and only if  $A \oplus B$  is false.

<b>A EQUIV B</b>	1	0
1	1	0
0	0	1

Implication is symbolized by  $\Rightarrow$  or  $\rightarrow$ . This is the definition that always bothers students:  $A \Rightarrow B$  is false if and only if A is true but B is false. In ordinary discourse we usually avoid the case that A is (known to be) false. Formally though, if A is false then  $A \Rightarrow B$  is true.

<b>A IMPLIES B</b>	<b>1</b>	<b>0</b>
<b>1</b>	<b>1</b>	<b>0</b>
<b>0</b>	<b>1</b>	<b>1</b>

- **Exercise 2** Rewrite  $A \oplus B$ ,  $A \equiv B$ ,  $A \Rightarrow B$  each using just  $\vee$ ,  $\wedge$ , and  $\sim$ . Hint: the answer is not unique. Note also, that due to DeMorgan's laws one can use just  $\vee$  and  $\sim$  or one can use  $\wedge$ , and  $\sim$ . Lastly, if you are new to logic, don't lose sleep over this exercise; just try to see if you can understand the answer given in the back of the book.

## A Useful Programming Trick (Indicator Functions)

In programming, there are two methods used for handling Boolean logic. The first method is to use Boolean logic that is built into the language itself. Another method is to define Boolean functions. A definition for **or** might look like this:

```
Define Or(X,Y)
  Begin
    Comment: X and Y should only have values 0 and 1.
    If X = 1 then Return 1
      Else If Y = 1 then Return 1
      Else Return 0
  End
```

This works okay but it is too much (computational) time and code. Another method is simply to define the following function:

$$\text{OR}(X,Y) = X + Y - X \cdot Y$$

Note that if either X or Y is 1, this function returns 1 otherwise it returns 0. It behaves as the *or* is supposed to behave, that is according to the truth table in the previous section. All the logical functions can be treated analogously:

$$\text{NOT}(X) = 1 - X$$

$$\text{AND}(X,Y) = X \cdot Y$$

$$\text{XOR}(X,Y) = X + Y - 2 \cdot X \cdot Y$$

$$\text{EQUIV}(X,Y) = 1 - X - Y + 2 \cdot X \cdot Y$$

$$\text{IMPLIES}(X,Y) = 1 - X + X \cdot Y$$

These functions are sometimes referred to as indicator functions. ( $\text{AND}(X,Y) = 1$  indicates that X and Y is true.

- **Exercise 3** Show that the functions just defined give the answers that are required by their respective truth tables.

## Some Unusual Boolean Algebra<sup>1</sup>

(This section *extremely optional*)

It so happens that there are many sets of logical operations that are sufficient for describing the rest.<sup>2</sup> Ordinarily most of us feel comfortable with *and*, *or* and *not*. Again, we can use DeMorgan's laws to dispense with either *and* or *or*. However, it is convenient to use these three operators. This is largely because they obey the laws given above. In particular both *and* and *or* are associative and commutative.

However, the two operations  $\oplus$  and  $\equiv$  are also commutative and associative and they can be extremely useful. In particular, all of Boolean algebra can be easily described with  $\wedge$ , and  $\oplus$ . Again,  $\oplus$  is the exclusive-or relation and is sometimes call the *ring-sum* relation.<sup>3</sup> Also, it is convenient to write the and relation as multiplication. That is, instead of  $A \wedge B$ , we write  $A \cdot B$  or just  $AB$ .

In the rest of this section, I will work in the language of logic. However, everything can be reinterpreted in terms of sets. I am going to state the laws of Boolean rings, that is the logic based upon  $\wedge$ , and  $\oplus$ . The laws of Boolean rings are easier to use than the usual Boolean algebra that is described in the first section of this chapter. Boolean rings require fewer laws than Boolean algebras. The greatest advantage of Boolean rings is that given two expressions  $E_1$  and  $E_2$  in a Boolean ring, it is easy to see if they are equivalent, that is whether  $E_1 = E_2$ . The corresponding problem in Boolean algebras can be difficult to solve. The drawback to Boolean rings is that Boolean ring expressions require far more terms than Boolean algebra expressions.

---

<sup>1</sup>This material is definitely not required later.

<sup>2</sup>In particular all of the logic operations can be described in terms of just a single operation. One operation that will do this trick is NOT BOTH. The other operation that will do the trick is NEITHER-NOR. Logic based upon using just one of these two operations is quite unpleasant. The operation IMPLIES will do everything as well if one is willing to use  $F \text{ IMPLIES } x$ , to indicate NOT  $x$ .

<sup>3</sup>(For advanced students) the set algebra with operators  $\wedge$ , and  $\oplus$  is an algebraic ring. It is known as a *Boolean ring*. The ring has elements 1 (the universal set) and 0 (the empty set). A Boolean ring can be characterized as a ring such that for any element  $x$ ,  $x \cdot x = x$ .

However, for small problems that occur in computer programming, Boolean rings can be much more convenient than Boolean algebras.

It is not difficult to prove any of the following statements but I am going to simply list the basic laws of Boolean rings:

- ▶  $(XY)Z = X(YZ)$ ,  $(X \oplus Y) \oplus Z = X \oplus (Y \oplus Z)$  (associativity)
- ▶  $XY = YX$ ,  $X \oplus Y = Y \oplus X$  (commutivity)
- ▶  $X(Y \oplus Z) = (XY) \oplus (XZ)$  (distribution)
- ▶  $X \oplus X = 0$  (cancellation)
- ▶  $XX = X$
- ▶  $X \text{ or } Y = X \oplus Y \oplus XY$ . In general,  $A \text{ or } B \text{ or } \dots N$  is the ring-sum of all products of terms of  $A, B, \dots, N$ . For example,  $A \text{ or } B \text{ or } C = A \oplus B \oplus C \oplus AB \oplus AC \oplus BC \oplus ABC$ .
- ▶  $A \text{ xor } B = A \oplus B$ .  $XOR(A, B, C)$  by which we mean  $A \text{ or } B \text{ or } C$  but no more than one of those is  $A \oplus B \oplus C \oplus ABC$ . In general,  $XOR(A, B, \dots N)$  is the ring-sum of all *odd* products of terms of  $A, B, \dots, N$ .
- ▶ Not  $X$  is written  $1 \oplus X$ .

If we extend our algebra to include  $\equiv$  we get the following very interesting laws:<sup>1</sup>

- ▶  $(X \equiv Y) \equiv Z = X \equiv (Y \equiv Z)$  (associativity)
- ▶  $X \equiv Y = Y \equiv X$  (commutivity)
- ▶  $X \equiv Y = 1 \oplus (X \oplus Y)$
- ▶  $X \oplus Y = 0 \equiv (X \equiv Y)$
- ▶  $X \oplus Y \oplus Z = X \equiv Y \equiv Z$
- ▶  $X \oplus (Y \equiv Z) = (X \oplus Y) \equiv Z$ ,  $X \equiv (Y \oplus Z) = (X \equiv Y) \oplus Z$
- ▶  $X \oplus (Y \equiv Z) = X \equiv (Y \oplus Z)$

---

<sup>1</sup>These laws can be extended to include the **or** relation,  $\vee$ . Obviously if we use all the relations  $\sim, \equiv, \oplus, \wedge, \vee$  we get one hell of a syntax, but it is not as difficult to learn as one might think, and there can be a great pay-off in efficiency.

## A Simple Application of Boolean Rings

We will use the laws of Boolean rings to prove DeMorgan's laws:

$$\sim(AB) = (\sim A) \vee (\sim B)$$

$$\sim(A \vee B) = (\sim A)(\sim B)$$

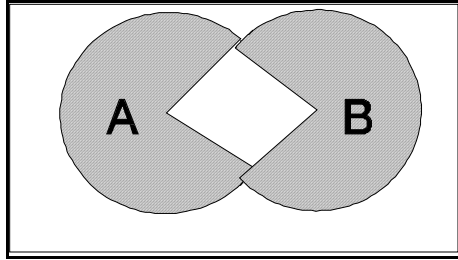
To prove the first law, we write  $(\sim A) \vee (\sim B)$  as

$$(1 \oplus A) \oplus (1 \oplus B) \oplus (1 \oplus A)(1 \oplus B).$$

Using the distributive law we get  $1 \oplus A \oplus 1 \oplus B \oplus 1 \oplus A \oplus B \oplus AB$ . Using cancellation we get  $1 \oplus AB$  and we are done.

To prove the second law we write  $\sim(A \vee B)$  as  $1 \oplus (A \oplus B \oplus AB)$ . However, this can be seen to be equal to  $(1 \oplus A)(1 \oplus B)$  and we have proven the second law.

1.



A cubist rendition of A **XOR** B.

2.

**XOR:**  $A \oplus B = (A \wedge \sim B) \vee ((\sim A) \wedge B)$

**EQUIV:**  $A \equiv B = (A \wedge B) \vee ((\sim A) \wedge (\sim B))$

**IMPLIES:**  $A \Rightarrow B = (\sim A) \vee B$  also worth mentioning is  $\sim(A \wedge (\sim B))$

3.

This exercise requires that in each function you substitute all four cases ( $X = 0$  or  $1$  and  $Y = 0$  or  $1$ ) and that the function gives the same value as required in the truth table for that logical operation. There is one logical operation that only has two cases to investigate:  $\text{NOT}(X) = 1 - X$ . If  $X$  is  $0$  then  $\text{NOT}(X) = 1$ . Similarly, if  $X = 1$  then  $\text{NOT}(X) = 0$ . This is precisely as is required by the truth table for NOT.