Boolean Algebra – Logical Functions

Boolean Algebra (named after mathematician, George Boole) is the mathematics associated with the binary number system and the theory of logic. It is similar to the formal algebra in the sense that it has independent variables that can produce dependent variables through functional relations established by the operations of addition and multiplication. However, the Boolean variables are binary (i.e. their domain is 1, 0) and the Boolean operations of addition and multiplication are defined differently.

Boolean Variable X: value can only be a 1, or a 0. Sometimes these values are referred to as *true* or *false* respectively. The convention of associating 1 with *true* and 0 with *false* is known as **positive logic**. The convention of associating 0 with true and 1 with false is known as **negative logic**. This presentation adopts positive logic.

Boolean Addition (+)
Boolean addition is generally known as the **OR** operation (logical addition)
If X, Y, Z are Boolean variables, then X + Y = Z implies the truth of X or Y produces the truth of Z. In other words, if either X or Y is true (1), then Z is true (1).

Boolean Multiplication (.)
Boolean multiplication is generally known as the **AND** operation (logical multiplication)
If X, Y, Z are Boolean variables, then X . Y = Z implies the truth of X and the truth of Y produces the the truth of Z. In other words, If both X and Y are true (1), then Z is true (1).

The result of Boolean operations on Boolean variables are often represented in tables called **truth tables**. The following are the truth tables for the **OR** and **AND** operations:

Truth Table - OR operation (Logical addition)

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

Truth Table - AND operation (Logical multiplication)

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

The **OR** and **AND** operations in association with the **NOT** operation (the compliment of X = X**’**) form the basis of digital logic circuits constituted by **gates**.

Boolean Algebra – Logical Functions

De Morgan's Theorem

The importance of the **OR**, **AND** and **NOT** operations is revealed in De Morgan's Theorem:

(X + Y)' = X' . Y' ----------(1)

The compliment of the result of an **OR** operation on X, Y = the result of the **AND** operation on their respective compliment.

(X . Y)' = X' + X' ----------(2)

The compliment of the result of an **AND** operation on X, Y = the result of the **OR** operation on their respective compliment.

The implication of De Morgan's Theorem is that any logical function can be obtained by either the **OR** operation and the **NOT** (compliment) operation, or by the **AND** operation and the **NOT** operation. Consequently, there are two families of logical functions: **sums of products** and **product of sums**.

Some Important Rules Of Boolean Algebra

Rule 1:	0 + X = X	
Rule 2:	1 + X = 1	
Rule 3:	X + X = X	
Rule 4:	X + X' = 1	where X' is the compliment of X
Rule 5:	0 . X = 0	
Rule 6:	1 . X = X	
Rule 7:	X . X = X	
Rule 8:	X . X' = 0	
Rule 9:	X'' . X = X	where X'' is the double compliment of X
Rule 10:	X + Y = Y + X	Commutative Law
Rule 11:	X . Y = Y . X	Commutative Law
Rule 12:	X + (Y+ Z) = (X + Y) + Z	Associative Law
Rule 13:	X . (Y . Z) = (X . Y) . Z	Associative Law
Rule 14:	X . (Y + Z) = X . Y + X . Z	Distributive Law
Rule 15:	X + X . Z = X	Absorption Law
Rule 16:	X . (X + Y) = X	
Rule 17:	(X + Y) . (X + Z) = X + Y . Z	
Rule 18:	X + X' . Y = X + Y	
Rule 19:	X . Y + Y . Z + X' . Z = X . Y + X' . Z	

Boolean Algebra – Logical Functions

(1) Use truth tables to prove that BC + BC' + B'A = A + B

Ans (1)

A	B	C	BC	BC'	B'A	BC + BC' + B'A	A + B
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	1	0	0	1	0	1	1
0	1	1	1	0	0	1	1
1	0	0	0	0	1	1	1
1	0	1	0	0	1	1	1
1	1	0	0	1	0	1	1
1	1	1	1	0	0	1	1

(2) Simplify the expression: f(A, B, C, D) = ABC + A'CD + B'CD.

Ans (2) f(A, B, C, D) = ABC + CD(A' + B').

(3) F is a Boolean function that is the result of a **AND** operation on f_1 (A,B), f_2 (C, D), f_3 (E).
f_1 (A, B) is the result of a **NAND** operation on A, B
f_2 (C, D) is the result of a **NAND** operation on C, D
f_3 (E) is the result of a **NOT** operation on E
Determine the expression for F.

Ans (3) f_1 (A, B) = (A . B)' The compliment of A . B
f_2 (C, D) = (C . D)' The compliment of C . D
f_3 (E) = E' The compliment of E
So, F = (A B)' . (C D)' . E' = (AB + CD + E)'

(4) Three commissioners constitute the board of a small county. Each commissioner votes on measures presented to the board by pressing a button indicating whether the commissioner votes for or against a measure. A measures passes if two or more commissioners vote for it. Design a logic circuit that takes the three votes as inputs and lights either a green or a red light to indicate whether a measure passes.

Ans (4) Let the vote of each of the three commissioners be Boolean variables X, Y, Z. Where X = 1 implies that commissioner X votes for measure; and X= 0, commissioner X votes against measure. Similarly for variables Y, Z. Then the logic circuit consists of three **AND** operations on each of the variable pairs (X, Y), (X, Z), (Y, Z); and one **OR** operation on the results XY, XZ, and YZ from the AND operations. In other words, XY, XZ, YZ are inputs for the **OR** operation. So, the Boolean logic function describing the circuit = F(X, Y, Z) = XY + XZ + YZ. Logic gates can be used to illustrate the Boolean logic function, however they are left out deliberately in this presentation because their absence necessitates more abstract thinking.

Boolean Algebra – Logical Functions

(5) State the sum of products expression for the Boolean variables, X,Y, Z, W. What basic Boolean operations produce the expression?

Ans (5) Sum of products expression: (X.Y) + (Z.W). Two **AND** operations and one **OR** operation produce the sum of products expression.

(6) State the product of sums expression for the Boolean variables, A, B, C, D. What basic Boolean operations produce the expression?

Ans (6) Product of sums expression: (A + B).(C + D). Two **OR** operations and one **AND** operation produce the product of sums expression.
Its been mentioned earlier, that any logical expression can be reduced to either sum of products expression or the product of sums expression. The two forms are equivalent, one may have a simplier implementation over the other in a given logic circuit design.

(7) Consider a *fail-safe autopilot system* in a commercial aircraft. The *system check* must pass before takeoff or landing maneuver. The *system check* passes when the following conditions are satisfied: two of the three (pilot, copilot and autopilot) possible pilots must be available; there is a signal test that confirms the pilot and copilot are seated properly in their respective control seat; the autopilot system can *self-check* to verify that its in proper operational state. Let the Boolean variable X denotes the pilot state (1 if the pilot is sitting at the controls), Y denote the same condition for the copilot, and Z denote the state of the autopilot (1 indicates the autopilot is functioning properly). Determine the logic function that establishes *system ready* (i.e. positive check - system check passes).

Ans (7) Required function f (X, Y, Z) = X.Y + X.Z + Y.Z
In order to determine the logic function for *system fails*, we apply De Morgan's Theorem to f, which is in *sum of products* form. So, logic function g, for *system fails* (negative check) is:
g = f' = (X.Y + X.Z + Y.Z)' = (X' + Y') . (X' + Z') . (Y' + Z'), g is *product of sums*.

Peter Oye Simate Sagay
Simate was my mother
Sagay was my father.