Boolean algebra

Boolean algebra is algebra of logic. It deals with variables that can have two discrete values, 0 (False) and 1 (True); and operations that have logical significance. The earliest method of manipulating symbolic logic was invented by George Boole and subsequently came to be known as Boolean algebra. Boolean algebra has now become an indispensable tool in computer science for its wide applicability in switching theory, building basic electronic circuits and design of digital computers.

Boolean Functions

A Boolean function is a special kind of mathematical function f: Xn → X of degree n, where X = {0, 1} is a Boolean domain and n is a non-negative integer. It describes the way how to derive Boolean output from Boolean inputs.

Example: Let, F(A, B) = A'B'. This is a function of degree 2 from the set of ordered pairs of Boolean variables to the set {0, 1} where F(0, 0) = 1, F(0, 1) = 0, F(1, 0) = 0 and F(1, 1) = 0

Boolean Expressions

A Boolean expression always produces a Boolean value. A Boolean expression is composed of a combination of the Boolean constants (True or False), Boolean variables and logical connectives. Each Boolean expression represents a Boolean function.

Example: AB'C is a Boolean expression.

Boolean Identities

- **Double Complement Law**

  ~(~A) = A

- **Complement Law**

  A + ~A = 1 (OR Form)

  A · ~A = 0 (AND Form)

- **Idempotent Law**

  A + A = A (OR Form)

$A \cdot A = A$ (AND Form)

- **Identity Law**

  $A + 0 = A$ (OR Form)

  $A \cdot 1 = A$ (AND Form)

- **Dominance Law**

  $A + 1 = 1$ (OR Form)

  $A \cdot 0 = 0$ (AND Form)

- **Commutative Law**

  $A + B = B + A$ (OR Form)

  $A \cdot B = B \cdot A$ (AND Form)

- **Associative Law**

  $A + (B + C) = (A + B) + C$ (OR Form)

  $A \cdot (B \cdot C) = (A \cdot B) \cdot C$ (AND Form)

- **Absorption Law**

  $A \cdot (A + B) = A$

  $A + (A \cdot B) = A$

- **Simplification Law**

  $A \cdot (\sim A + B) = A \cdot B$

  $A + (\sim A \cdot B) = A + B$

  $A \cdot B + A \cdot C + \sim B \cdot C = A \cdot B + \sim B \cdot C$

- **Distributive Law**

  $A + (B \cdot C) = (A + B) \cdot (A + C)$

  $A \cdot (B + C) = (A \cdot B) + (A \cdot C)$

- **De-Morgan's Law**

  $\sim(A \cdot B) = \sim A + \sim B$

~(A+ B) = ~A · ~B

Canonical Forms

For a Boolean expression there are two kinds of canonical forms:

1. The sum of minterms (SOM) form

2. The product of maxterms (POM) form

➢ The Sum of Minterms (SOM) or Sum of Products (SOP) form

A minterm is a product of all variables taken either in their direct or complemented form. Any Boolean function can be expressed as a sum of its 1-minterms and the inverse of the function can be expressed as a sum of its 0-minterms. Hence,

F (list of variables) = Σ (list of 1-minterm indices)

and

F' (list of variables) = Σ (list of 0-minterm indices)

A	B	C	Minterm
0	0	0	m_0
0	0	1	m_1
0	1	0	m_2
0	1	1	m_3
1	0	0	m_4
1	0	1	m_5
1	1	0	m_6
1	1	1	m_7

Example

Let, F(x, y, z) = x' y' z' + x y' z + x y z' + x y z

Or, F(x, y, z) = m0 + m5 + m6 + m7

Hence,

F(x, y, z) = Σ (0, 5, 6, 7)

Now we will find the complement of F(x, y, z)

F’ (x, y, z) = x’ y z + x’ y’ z + x’ y z’ + x y’ z’

Or, F’(x, y, z) = m3 + m1 + m2 + m4

Hence,

F’(x, y, z) = Σ (3, 1, 2, 4) = Σ (1, 2, 3, 4)

- **<u>The Product of Maxterms (POM) or Product of Sums (POS) form</u>**

A maxterm is addition of all variables taken either in their direct or complemented form. Any Boolean function can be expressed as a product of its 0-maxterms and the inverse of the function can be expressed as a product of its 1-maxterms. Hence,

F (list of variables) = π (list of 0-maxterm indices)

and

F’(list of variables) = π (list of 1-maxterm indices).

A	B	C	Maxterm
0	0	0	M_0
0	0	1	M_1
0	1	0	M_2
0	1	1	M_3
1	0	0	M_4
1	0	1	M_5
1	1	0	M_6
1	1	1	M_7

Example

Let, F(x, y, z) = (x+y+z) • (x+y+z’) • (x+y’+z) • (x’+y+z)

Or, F(x, y, z) = M0 • M1 • M2 • M4

Hence,

F (x, y, z) = Π(0, 1, 2, 4)

F’'(x, y, z) = (x+y’+z’) • (x’+y+z’) • (x’+y’+z) • (x’+y’+z’)

Or, F(x, y, z) = M3 • M5 • M6 • M7

Hence,

F ' (x, y, z) = Π(3, 5, 6, 7)

Simplification Using Algebraic Functions

In this approach, one Boolean expression is minimized into an equivalent expression by applying Boolean identities.

Problem 1

Minimize the following Boolean expression using Boolean identities:

F (A, B, C) = A’B+ BC’+ BC+ AB’C’

Solution

Given, F (A, B, C) = A’B+ BC’+ BC+ AB’C’

Or, F (A, B, C) = A’B+ (BC’+ BC’) + BC+ AB’C’

[By adding BC’, as it does not change F]

Or, F (A, B, C) = A’B+ (BC’+ BC) + (BC’+ AB’C’)

Or, F (A, B, C) = A’B+ B(C’+ C) + C’(B+ AB’)

Or, F (A, B, C) = A’B+ B.1+ C’(B+ A)

[(C’+ C) =1 and (B+ AB’)= (B+ A)]

Or, F (A, B, C) = A’B+ B+ C’(B+ A)

[B.1 =B]

Or, F (A, B, C) = B(A’+ 1)+ C’(B+ A)

Or, F (A, B, C) = B.1+ C’(B+ A)

[(A’+ 1) =1]

Or, F (A, B, C) = B+ C’(B+ A)

[As, B.1 =B]

Or, F (A, B, C) = B+ BC’ + AC’

Or, F (A, B, C) = B(1+ C’) + AC’

Or, F (A, B, C) = B.1 + AC’

[As, (1+ C’) =1]

Or, F (A, B, C) = B + AC’

[As, B.1 =B]

So, F (A, B, C) = B + AC’ is the minimized form.

Problem 2

Minimize the following Boolean expression using Boolean identities:

F (A, B, C) = (A+B) (B+ C)

Solution

Given, F (A, B, C) = (A+B) (A+ C)

Or, F (A, B, C) = A.A+A.C+B.A+ B.C [Applying distributive Rule]

Or, F (A, B, C) = A+A.C+B.A+ B.C [Applying Idempotent Law]

Or, F (A, B, C) = A(1+C)+B.A+ B.C [Applying distributive Law]

Or, F (A, B, C) = A+B.A+ B.C [Applying dominance Law]

Or, F (A, B, C) = (A+1).A+ B.C [Applying distributive Law]

Or, F (A, B, C) = 1.A+ B.C [Applying dominance Law]

Or, F (A, B, C) = A+ B.C [Applying dominance Law]

So, F (A, B, C) = A+ BC is the minimized form.