Boolean Algebra-1

This chapter is divided into two parts. We first give some definitions to introduce Boolean Algebra ; we define logic gates which are electronic circuits that can be used to implement the logic expressions also known as Boolean expressions. There are three basic logic gates, namely the **OR** gate, the **AND** gate and the **NOT** gate, the **EXCLUSIVE-OR** gate and the **EXCLUSIVE-NOR** gate. In the second part we present postulates for the definition of Boolean algebra. Boolean algebra is mathematics of logic, developped in 1854 by George Boole to treat the logic functions, it is used for simplification of complex logic expressions. Other simple techniques of simplification are Karnaugh maps and the tabular method given by Quine-McCluskey. We first give some basic definitions.

Definition 1. *Let S be a set of finite and denumerable elements, a binary operator defined on S is a rule that assigns, to each pair of elements from S, a unique element from S.*

Example 1. Take a look at the equation $x * y = z$. * is a binary operator if and only if it specifies a rule for finding z from the pair (x, y) and also if x, y and z are in S. But, * is not a binary operator if $x, y \in S$ and $z \notin S$.

Definition 2. *A set is closed according to a binary operator if the binary operator provides a rule for obtaining a unique element of S for every pair of elements of S. For example, the set of natural numbers $\mathbb{N} = \{0, 1, 2, 3, 4, ...\}$is said to be closed according to the binary operator plus (+) under arithmetic addition rules, because $\forall x, y \in \mathbb{N}$ the operation $x + y$ gives a unique element $z \in \mathbb{N}$. However, the set of natural numbers is not closed according to the binary operator minus (-) under arithmetic subtraction rules since $4 - 7 = -3$, where -3 is not a natural number.*

Definition 3. *A binary operator * on a set S is associative if and only if*

$$\forall x, y, z \in S; \ \ (x * y) * z = x * (y * z).$$

Example 2. Addition in $\mathbb{N}$ is associative.

Definition 4. *A binary operator * on a set S is commutative if and only if*

$$\forall x, y \in S; \ \ x * y = y * x.$$

Example 3. Addition in $\mathbb{N}$ is commutative.

Definition 5. *A set S is to have an identity element with respect to a binary operation * on S, if and only if there exist a unique element $e \in S$ such that $e * x = x * e = x, \ \forall x \in S$.*

Example 4. — 0 is an identity element with respect to the binary operator $+$ on $\mathbb{Z}$.

— 1 is an identity element with respect to the binary operator $\times$ on $\mathbb{Z}$.

Definition 6. *If a set S has the identity elemnt e with respect to a binary operator $*$, for every $x \in S$, there exists an element $x' \in S$ such that $x * x' = x' * x = e$. l x' is called the inverse of x*

Example 5. — In the set of integers $\mathbb{Z}$, 0 is an identity element with respect to the binary operator $+$, the inverse of element x is $-x$ since $x + (-x) = 0$.

— There is no inverse of a element in $\mathbb{N}$.

Definition 7. *If $*$ and $\perp$are two binary operators on a set S, $*$ is said to be distributive over $\perp$, whenever :*

$$\forall x, y, z \qquad x * (y \perp z) = (x * y) \perp (x * z).$$

Example 6. Multiplication (.) in $\mathbb{R}$ is distributive over addition $(+)$, since :

$$\forall x, y, z \in \mathbb{R} \quad x.(y + z) = (x.y) + (x.z).$$

Definition 8. *The binary variables (boolean variable), as we know can have either of the two states, i.e. the logic $'0'$ state or the logic $'1'$ state. A variable in state 1 is referred to as active and the symbols 0 and 1 represent logical states but do not have a numerical value.*

Example 7. A suitable K can only have two states : open $'0'$ or closed $'1'$.

Definition 9. *A logical (binary or Boolean) function of binary variables is a function whose values can be either of the two states, $'0'$ or $'1'$.*

Definition 10. (Truth Table) *A truth table of a logical function is a table who lists all possible combinations of input binary variables and the corresponding outputs of a logic system. Note that a truth table of logical function with n binary variables is a table with $n+1$ columns and 2^n lines.*

Example 8. If the number of input binary variables is 2, then there are $2^2 = 4$ possible input combinations.

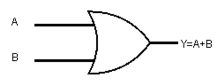
Example 9. Let f be a logical function with two binary variables a and b, defined by :

$$f(a, b) = \begin{cases} 1; \text{ if } a = b = 1 \\ 0 \text{ otherwise.} \end{cases}$$

a	b	$f(a,b)$
0	0	0
0	1	0
1	0	0
1	1	1

0.1 Logic Gates

We define a set of components, known as logic gates, each of one correspods to a logical function. The three basic logic gates are the **OR** gate, the **AND** gate and the **NOT** gate.



A	B	$A.B$
0	0	0
0	1	1
1	0	1
1	1	1

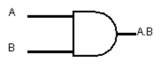
FIGURE 1 – Two-input OR Gate

0.1.1 OR Gate

An OR gate is a logic circuit with two ore more inputs and one output. The output of an OR gate is LOW (0) if and only if all of its inputs are LOW, for all other possible input combinations, the output is HIGH. Figure0.1.1 shows the circuit symbol and the truth table of a two-input OR gate $Y = A + B$.

0.1.2 AND Gate

An AND gate is a logic circuit with two ore more inputs and one output. The output of an AND gate is HIGH (1) if and only if all of its inputs are HIGH, for all other possible input combinations, the output is LOW (0). Figure0.1.2 shows the circuit symbol and the truth table of a two-input AND gate $Y = A\dot{B}$.

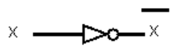


A	B	$A.B$
0	0	0
0	1	0
1	0	0
1	1	1

FIGURE 2 – Two-input AND Gate

0.1.3 NOT Gate

An NOT gate is a logic circuit with one input and one output. The output is always the complement of the input, i.e a LOW input produces a HIGH output, and vice versa. Figure0.1.3 shows the circuit symbol and the truth table of a NOT gate ($\overline{X}$ is the complement of X).



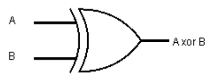
X	$\overline{X}$
0	1
1	0

FIGURE 3 – NOT Gate

0.1.4 EXCLUSIVE-OR Gate

The EXCLUSIVE-OR gate, written as EX-OR gate, with two-input and one-output. Figure0.1.4 shows the logic symbol and truth table of a two-input EX-OR gate. We see from the truth table, that the output of an EX-OR gate is a logic $'1'$ when the inputs are unlike and a logic $'0'$ when the inputs are like. The output of a two-input EX-OR gate is expressed by

$$Y = (A \oplus B) = \overline{A}.B + A.\overline{B}.$$



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

FIGURE 4 – EX-OR Gate

Example 10. We establish the truth table of EX-OR gate of four inputs and one output $Y = A \oplus B \oplus C \oplus D$.

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

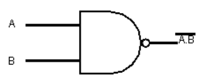
Remark. We remark that the output of a multiple-input EX-OR logic function is a logic $'1'$ when the number of 1s in the input sequence is odd and a logic $'0'$ when the number of 1s in the input sequence is even, including zero.

0.1.5 NAND Gate

NAND is an abbreviation of NOT AND, the truth table of a NAND gate is obtained from the truth table of an AND gate by complementing the output entries. Figure 0.1.5 shows the circuit symbol of a two-input NAND gate. The little invert bubble (small circle) on the right end of the symbol means to invert the output of AND.

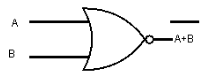
0.1.6 NOR Gate

NOR is an abbreviation of NOT OR, the truth table of a NOR gate is obtained from the truth table of an OR gate by complementing the output entries. Figure NOR shows the circuit symbol of a two-input NOR gate.



A	B	$\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

FIGURE 5 – NAND Gate

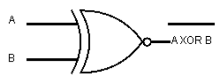


A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

FIGURE 6 – NOR Gate

0.1.7 EXCLUSIVE-NOR Gate

EXCLUSIVE-NOR written as EX-NOR, the truth table of a NOR gate is obtained from the truth table of an EX-OR gate by complementing the output entries. Figure NXOR shows the circuit symbol of a two-input NOR gate.

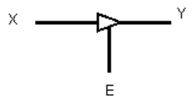


A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	1

FIGURE 7 – EX-NOR Gate

0.1.8 Three-inputs Gate

A three-state buffer functions as a signal-controlled switch. An enable signal controls whether the input signal is sent to the output or isolated from the output, which remains in a high-impedance state. Figure0.1.8 shows circuit and the truth table of the three-state buffer ; (A : High impedance state)



E	X	Y
0	x	Z
1	0	0
1	1	1

FIGURE 8 – Three-state buffer

0.1.9 Different representations of a logical function

There are many different ways to represent a logical function

Electric representation

The diagram is created by interaction.

Algebric representation

For the algebric representation, we use the logical operations such that $+, ., \oplus$.

Example 11. f is a logical function with algebric representation.

$$f(a,b) = \underbrace{a.b}_{\text{logical term}} + \overline{a}.b$$

Arithmetic representation

The arithmetic representation means representation by the truth table.

Example 12. The arithmetic representation of $f(A,B) = A.B + A + C.D$ is given by the following truth table, we have $f(A,B,C,D) = 1 \iff \begin{cases} A.B = 1 \\ \text{or} \\ A = 1 \\ \text{or} \\ C.D = 1 \end{cases} \iff \begin{cases} A = 1 \text{ and } B = 1 \\ \text{or} \\ A = 1 \\ \text{or} \\ C = 1 \text{ and } D = 1 \end{cases}$

A	B	C	D	$f(A,B,C,D)$
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Time representation or Timing diagram-Chronogram

The timing diagram is a graphical representation that illustrates the timing relationships between various signals or events in a system ; often employed in electronics, digital design, and other fields to study and understand system behaviour across time.

Example 13. Complete the timing diagram of the logical function $S = A + B$.

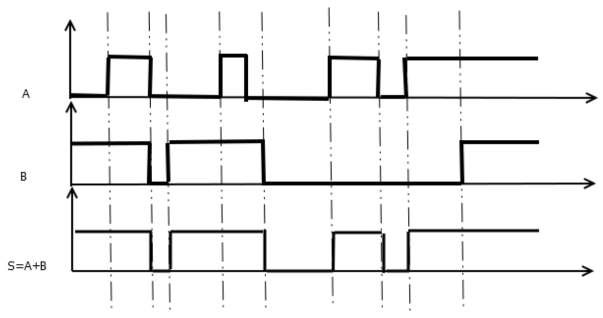


FIGURE 9 – Chronogram of S

Graphic representation-diagram

We can define a logical function by representing it using logic gates, then read it from left to right. We call this representation Diagram.

Example 14. The following diagram represents the logical function $f(a,b) = a.b + \overline{a}.b$.

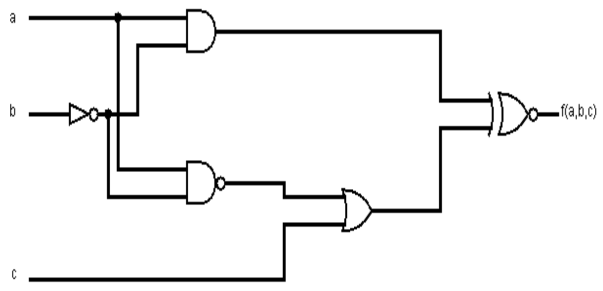


FIGURE 10 – Diagram of f

0.1.10 Relation between various representations

All the representations of a logical function are equivalent. It is possible to switch different representations.

Example 15. Let us represent the logical function f; where $a, b,$ and c are binary variables and

$f(a,b,c) = a.\overline{b}.c + \overline{a}.\overline{b}.c + a.b.c$, by his truth table.

$$\begin{aligned} f(a,b,c)=1 &\iff a.\overline{b}.c=1 \text{ or } \overline{a}.\overline{b}.c=1 \text{ or } a.b.c=1 \\ &\iff \begin{cases} a=1 \text{ and } \overline{b}=1 \text{ and } c=1 \\ \text{or} \\ \overline{a}=1 \text{ and } \overline{b}=1 \text{ and } c=1 \\ \text{or} \\ a=1 \text{ and } b=1 \text{ and } c=1 \end{cases} \\ &\iff \begin{cases} a=1 \text{ and } b=0 \text{ and } c=1 \\ \text{or} \\ a=0 \text{ and } b=0 \text{ and } c=1 \\ \text{or} \\ a=1 \text{ and } b=1 \text{ and } c=1. \end{cases} \end{aligned}$$

Then the truth table is given by

a	b	c	$f(a,b,c)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Example 16. Let us give the algebric representation of the logical function represented by the following truth table.

a	b	$f(a,b)$
0	0	0
0	1	1
1	0	1
1	1	1

$f(a,b) = \overline{a}.b + a.\overline{b} + a.b$

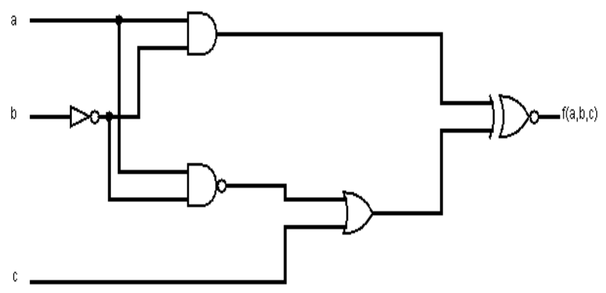
Remark. To represent a logical function from his truth table, we proceed as follows.

— We only consider logical terms for which f equals 1.

— On each logical term, we replace ones by the variables and zeros by the complement of variables. As a result, each logical term corresponds to the logical product of its variables or complements.

— The logical function will be the logical sum of all logical products found previously.

Example 17. Consider the following circuit and find the algebric representation of the output.



$$f(a,b,c) = \overline{a.\bar{b} \oplus (\overline{a.\bar{b}} + c)}$$

0.2 Boolean Algebra

Boolean algebra, is simpler than ordinary algebra, it is also composed of a set of symbols and a set of rules to manipulate these symbols.

Definition 11. *Let B be a set of logical variables supplied with two binary operations* **AND** *noted by ". ",* **OR** *noted by "+ " and* **NOT** *noted by " ", $(B, .,+,\bar{\ })$ is a Boolean algebra if and only if the following postulates (axioms) are verified.*

(A) Commutativity : $\forall\ a, b \in B : a + b = b + a$ *and* $a.b = b.a$.

(B) associativity : $\forall\ a, b, c \in B : (a + b) + c = a + (b + c)$ *and* $(a.b).c = a.(b.c)$.

(C) distrubitivity : $\forall\ a, b, c \in B : a.(b + c) = a.b + a.c$ *and* $a + b.c = (a + b).(a + c)$.

(D) $\forall a \in B : a + 0 = a$ *and* $a.1 = a$.

(E) For all element a in B, there exist a unique complement element noted by $\overline{a}$ such that $a + \overline{a} = 1$ *and* $a.\overline{a} = 0$.

Example 18. 1. $(\mathcal{P}(E), \cap, \cup, , \mathcal{C})$ is a Boolean algebra.

2. $(P, \wedge, \vee, \text{-})$ is a Boolean algebra ; where P is a set of propositions.

0.2.1 Principle of duality

The dual of a Boolean expression is obtained by replacing all "." operations by "+" operations, all "+"operations by "."operations, all 0s by 1s and all 1s by 0s and living all literals unchanged.

Example 19. The coresponding dual of the Boolean expression $(a + b).(\overline{a} + \overline{b})$ is $(a.b) + (\overline{a}.\overline{b})$

Remark. Duals of Boolean expressions are mainly of interest in the study of Boolean postulates and theorems.

0.2.2 Theorems of Boolean Algebra

We apply theorems of Boolean algebra to simplify Boolean expressions and transform them into a more useful and meaningful expressions. We note that if a given expression is valid, its dual will also be valid.

Theorem 1. *(Idempotent or Identity Laws)*

$$\forall a \in B;\ \ a + a = a \text{ and } a.a = a.$$

Démonstration. We have $\forall a \in B$;

$$\begin{aligned} a &= a + 0 \ (D) \\ &= a + a.\overline{a} \ (E) \\ &= (a + a).(a + \overline{a}) \ (C) \\ &= (a + a).1 \ (E) \\ &= a + a. \ (D) \end{aligned}$$

To prove the dual expression, we use the same method. □

Remark. We can prove by mathematics induction that

$$\forall a \in B,\ \forall n \in \mathbb{N} :\ a + a + \cdots + a = a \ \ \textit{and}\ a.a.\cdots.a = a.$$

Theorem 2.

$$\overline{0} = 1 \ \textit{and}\ \overline{1} = 0.$$

Démonstration. We have from (D) $0 + 1 = 1$ and $0.1 = 0$, so $\overline{0} = 1$ is the unique complement of 0 and by duality $\overline{1} = 0$. □

Theorem 3. *(Operations with '0' and '1')*

$$\forall a \in B;\ a.0 = 0 \ \textit{and}\ a + 1 = 1.$$

Démonstration. We have $\forall a \in B$

$$\begin{aligned} a + 1 &= a + (a + \overline{a}) \ (E) \\ &= (a + a) + \overline{a} \ (B) \\ &= a + \overline{a} \text{ by Theorem1} \\ &= 1 \ (E). \end{aligned}$$

By duality, we can prove that $a.0 = 0$. □

Theorem 4. *(Involution law)*

$$\forall a \in B;\ \overline{\overline{a}} = a.$$

Démonstration. From (E) we have $a+\overline{a} = 1$ and $a.\overline{a} = 0$, by commutativity $\overline{a}+a = 1$ and $\overline{a}.a = 0$, this implies that a is the complement of $\overline{a}$ then $\overline{a}$ admits a complement $\overline{\overline{a}}$ and by unicity of complement, we deduce that $\overline{\overline{a}} = a$. □

Theorem 5. *(Absorption Law1)*

$$\forall a, b \in B;\ a + a.b = a \ \textit{and}\ a.(a + b) = a$$

Démonstration. $\forall a, b \in B$;

$$\begin{aligned} a + a.b &= a.1 + a.b \ (D) \\ &= a.(1 + b) \ (C) \\ &= a.1 \text{ by theorem3} \\ &= a \ (D) \end{aligned}$$

and by duality we obtain $a.(a + b) = a$. □

Theorem 6. *(Absorption Law2)*

$$\forall a,b \in B;\ a+\overline{a}.b = a+b \textit{ and } a.(\overline{a}+b) = a.b.$$

Démonstration. $\forall a,b \in B$;

$$\begin{aligned} a+\overline{a}.b &= (a+\overline{a}).(a+b)\ (C)\\ &= 1.(a+b)\ (E)\\ &= a+b\ (D) \end{aligned}$$

and by duality, we obtain $a.(\overline{a}+b) = a.b$. □

Theorem 7. *(Consensus Theorem)*

$$\forall a,b,c \in B;\ a.b+b.c+\overline{a}.c = a.b+\overline{a}.c \textit{ and } (a+b).(b+c).(\overline{a}+c) = (a+b).(\overline{a}+c)$$

Démonstration. $\forall a,b,c \in B$;

$$\begin{aligned} a.b+\overline{a}.c &= a.b+a.b.c+\overline{a}.c.b \text{ by Theorem5}\\ &= a.b+\overline{a}.c+(a+\overline{a}).b.c\ (A),(C)\\ &= a.b+\overline{a}.c+b.c\ (E) \end{aligned}$$

and by duality, we obtain $(a+b).(b+c).(\overline{a}+c) = (a+b).(\overline{a}+c)$. □

Theorem 8.

$$\forall a,b,c \in B;\ (a+c).(\overline{a}+b) = a.b+\overline{a}.c \textit{ and } a.c+\overline{a}.b = (a+b).(\overline{a}+c)$$

Démonstration. $\forall a,b,c \in B$;

$$\begin{aligned} (a+c).(\overline{a}+b) &= a.\overline{a}+a.b+c.\overline{a}+c.b\ (C)\\ &= 0+a.b+c.\overline{a} \text{ by (E) and Theorem7}\\ &= a.b+c.\overline{a}, \end{aligned}$$

and by duality, we obtain $a.c+\overline{a}.b = (a+b).(\overline{a}+c)$. □

Theorem 9. *(DeMorgan's Theorem)*

$$\forall a,b \in B;\ \overline{a+b} = \overline{a}.\overline{b} \textit{ and } \overline{a.b} = \overline{a}+\overline{b}.$$

Démonstration. $\forall a,b,c \in B$;

$$\begin{aligned} (\overline{a}.\overline{b}).(a+b) &= \overline{a}.\overline{b}.a+\overline{a}.\overline{b}.b\ (C)\\ &= 0.\overline{b}+\overline{a}.0\ (A),(E)\\ &= 0. \end{aligned}$$

In the other side, we have

$$\begin{aligned} \overline{a}.\overline{b}+a+b &= a+\overline{b}+b \text{ by Theorem6}\\ &= a+1\ (E)\\ &= 1 \text{ by Theorem3}, \end{aligned}$$

then $\overline{a}.\overline{b}$ is the complement of $a+b$, so $\overline{a+b} = \overline{a}.\overline{b}$. □

The above theorems and postulats are used to simplify boolean expressions, we call this method the algebric simplification and the theorem of involution law is the basis of finding the equivalent product-of-sums expression for a given sum-of-products expression, and vice versa.

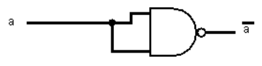
Example 20. Apply boolean laws and theorems to modify a two-input OR gate into two-input NAND gates only.

A two-input OR gate is represented by the boolean expression.$Y = a + b$, where a, b are the input logics variables and Y is the output.

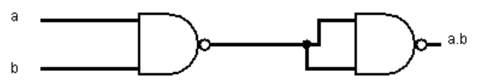
$$\begin{aligned} a + b &= \overline{\overline{a}} + \overline{\overline{b}} \qquad \text{(Involution law)} \\ &= \overline{\overline{a}.\overline{b}} \qquad \text{(DeMorgan's theorem)} \\ &= \overline{\overline{a.a}.\overline{b.b}} \qquad \text{(Idempotent law)} \end{aligned}$$

Remark. The two-input NAND gate (NOR resp.) is a complete gate because we can use it to implement AND, OR, NOT, NAND, NOR and NOT with many inputs.

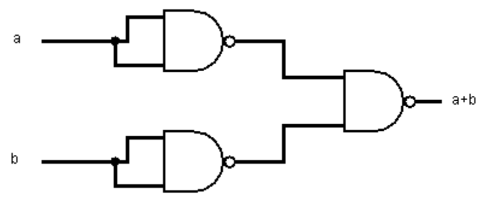
$\overline{a} = \overline{a.a}$ Idempotent law.



$a.b = \overline{\overline{a.b}}$,

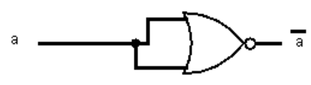


$a + b = \overline{\overline{a+b}} = \overline{\overline{a}.\overline{b}}$,

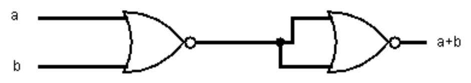


Similary,

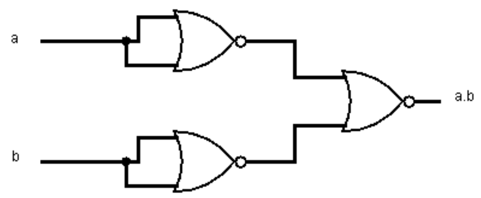
$a = \overline{a+a}$ Idemptent law,



$a + b = \overline{\overline{a+b}}$,



$a.b = \overline{\overline{a.b}} = \overline{\overline{a} + \overline{b}}$,



0.2.3 Examples of algebric simplification

Show that.

1. $(a.b) \oplus (a+b) = a \oplus b$.
2. $a.b + \overline{c} + c(\overline{a} + \overline{b}) = 1$.

—

$$\begin{aligned}(a.b) \oplus (a+b) &= a.b(\overline{a+b}) + \overline{a.b}(a+b) \text{ by definition}\\ &= a.b(\overline{a}.\overline{b}) + (\overline{a} + \overline{b}).(a+b) \text{ DeMorgan's theorem}\\ &= \underbrace{a.\overline{a}}_{0}.\underbrace{b.\overline{b}}_{0} + \underbrace{a.\overline{a}}_{0} + \overline{a}.b + a.\overline{b} + b.\overline{b} \text{ (C),(A)}\\ &= \overline{a}.b + a.\overline{b} = a \oplus b \text{ (E).}\end{aligned}$$

—

$$\begin{aligned}a.b + \overline{c} + c(\overline{a} + \overline{b}) &= a.b + \overline{c} + \overline{\overline{c} + (a.b)} \text{ Theorem 9}\\ &= 1\ (E)\end{aligned}$$

Example 21. Simplify the following expression.

$$S = \overline{a}.b.c + a.\overline{b}.\overline{c} + \overline{a}.\overline{b}.\overline{c} + a.\overline{b}.c + a.b.c.$$

$$\begin{aligned}\overline{a}.b.c + a.\overline{b}.\overline{c} + \overline{a}.\overline{b}.\overline{c} + a.\overline{b}.c + a.b.c &= \overline{a}.b.c + (a + \overline{a}).\overline{b}.\overline{c} + a.c(\overline{b} + b) \text{ by (A), (B) and (C)}\\ &= \overline{a}.b.c + 1.\overline{b}.\overline{c} + a.c.1 \text{ by (E)}\\ &= \overline{a}.b.c + \overline{b}.\overline{c} + a.c \text{ by (D)}\\ &= (a + \overline{a}.b).c + \overline{b}.\overline{c} \text{ by (C)}\\ &= (a+b).c + \overline{b}.\overline{c} \text{ by Absorption Law2}\\ &= a.c + b.c + \overline{b}.\overline{c}.\end{aligned}$$

0.3 Simplification techniques

The primary objective of simplification is to obtain an expression that has the minimum number of logical terms. There are other methods of simplification than the application of laws and theorems of Boolean algebra such as the Karnaugh map method and the Quine-Mc Cluskey tabular method.

We first describe sum-of products and product-of sums Boolean expressions, to will be able to minimize expressions in the same or the other form.

0.3.1 Sum-of-Products Boolean Expressions or First canonical form (disjonctif canonical form)

A sum-of-products expression contains the sum of different terms (product of all logical variables). It can be obtained directly from the truth table by considering those input combinations that produce $'1'$ at the output.

Different terms are given by the product of the inputs, where $'0'$ means complemented variable and $'1'$ means uncomplemented variable, and the sum of all such terms gives the expression.

A	B	C	$f(A,B,C)$
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

TABLE 1 – Truth table of boolean expression of the example 22

Example 22. The truth table bellow can be represented by the boolean expression

$$f(A,B,C) = \overline{A}.\overline{B}.\overline{C} + \overline{A}.B.\overline{C} + \overline{A}.B.C + A.B.\overline{C}.$$

A sum-of-product expression is also known us a minterm expression.

Definition 12. *We mean by minterm the term m_i, where i represents the decimal equivalent of the logical values product of the inputs from the truth table, m_i is the product of logical variables complemented or not. With n logical variables, we construct 2^n minterms.*

Example 23. For two inputs a, b, we have four minterms $m_0 = \overline{a}.\overline{b}, m_1 = \overline{a}.b, m_2 = a.\overline{b}$ and $m_3 = a.b$ which correspond, respecively to $00, 01, 10$ and 11.

In example 22 we can write $f(A,B,C) = m_0 + m_2 + m_3 + m_6$ or by the decimal form $f(A,B,C) = \sum(0,2,3,6)$.

0.3.2 Product-of-Sums Boolean Expressions or second canonical form (conjunctif canonical form)

A product-of-sums expression contains the product of different terms (sum of all logical variables). It can be obtained directly from the truth table by considering those input combinations that produce $'0'$ at the output.

Different terms are given by the sum of the inputs, where $'0'$ means uncomplemented variable and $'1'$ means complemented variable, the product of such terms gives the expression.

An OR gate produces a logic $'0'$ only when all its inputs are in the logic $'0'$ state.

Example 24. Keep the example 22; the product-of-sums Boolean expression is given by

$$f(A,B,C) = (A + B + \overline{C}).(\overline{A} + B + C).(\overline{A} + B + \overline{C}).(\overline{A} + \overline{B} + \overline{C}).$$

If we multiply the given expression and apply the obvious simplification, we obtain the equivalent sum-of-products expression.

Remark. A sum-of-products can be transformed into an equivalent product-of-sums expression by

1. taking the dual of the expression,
2. multiplying out different terms to get the sum-of-products form
3. removing redundancy
4. taking a dual to get the equivalent product-of-sums.

A product-of-sums expression is also known us a maxterm expression.

Definition 13. *We mean by maxterm the term M_i, where i represents the decimal equivalent of the logical values sum of the inputs from the truth table, M_i is the sum of logical variables complemented or not. With n logical variables, we construct 2^n maxterms.*

Example 25. For two inputs a, b, we have four maxterms $M_0 = a + b, m_1 = a + \overline{b}, m_2 = \overline{a} + b$ and $m_3 = \overline{a} + \overline{b}$ which correspond, respecively to $00, 01, 10$ and 11.

In example 22 we can write $f(A, B, C) = M_1.M_4.M_5.M_7$ or by the decimal form $f(A, B, C) = \prod(1, 4, 5, 7)$.

Remark.

a	b	minterms	maxterms	Binary form	Decimal form
0	0	$m_0 = \overline{a}.\overline{b}$	$M_0 = a + b$	00	0
0	1	$m_1 = \overline{a}.b$	$M_1 = a + \overline{b}$	01	1
1	0	$m_2 = a.\overline{b}$	$M_2 = \overline{a} + b$	10	2
1	1	$m_3 = a.b$	$M_3 = \overline{a} + \overline{b}$	11	3

$\overline{a} + b = 0 \iff a = 1 \mathrm{and} b = 0$ then the binary form of this term is 10.

Example 26. We consider the logical function f defined by

$$f(a, b, c) = a + \overline{b}.c.$$

1. Let us give the SOP form.

$$\begin{aligned}
f(a, b, c) &= a + \overline{b}.c \\
&= a.1.1 + 1.\overline{b}.c \\
&= a.(b + \overline{b}).(c + \overline{c}) + (a + \overline{a}).\overline{b}.c \\
&= a.b.c + a.b.\overline{c} + \underbrace{a.\overline{b}.c} + a.\overline{b}.\overline{c} + \underbrace{a.\overline{b}.c} + a.\overline{b}.\overline{c} + \overline{a}.\overline{b}.c \\
&= \underbrace{a.b.c}_{111} + \underbrace{a.b.\overline{c}}_{110} + \underbrace{a.\overline{b}.c}_{101} + \underbrace{a.\overline{b}.\overline{c}}_{100} + \underbrace{\overline{a}.\overline{b}.c}_{001} \\
&= \sum(1, 4, 5, 6, 7) \\
&= m_1 + m_4 + m_5 + m_6 + m_7.
\end{aligned}$$

2. Let us give the POS form.

$$\begin{aligned}
f(a, b, c) &= a + \overline{b}.c \\
&= (a + \overline{b}).(a + c) \\
&= (a + \overline{b} + 0).(a + 0 + c) \\
&= (a + \overline{b} + c.\overline{c}).(a + b.\overline{b} + c) \\
&= (a + \overline{b} + c).(a + \overline{b} + \overline{c}).(a + b + c).(a + \overline{b} + c) \\
&= (a + \overline{b} + c).(a + \overline{b} + \overline{c}).(a + b + c) \\
&= M_0.M_2.M_3. \\
&= \prod(0, 2, 3).
\end{aligned}$$

3. We will note the complement of $f(a, b, c)$ by $f'(a, b, c)$, then from SOP form

$$\begin{aligned}
f'(a, b, c) &= (\overline{a} + \overline{b} + \overline{c}).(\overline{a} + \overline{b} + c).(\overline{a} + b + \overline{c}).(\overline{a} + b + v).(a + b + \overline{c}) \\
&= \prod(1, 4, 5, 6, 7),
\end{aligned}$$

4. and from the POS form, we obtain

$$\begin{aligned} f'(a,b,c) &= \overline{a}.b.\overline{c} + \overline{a}.b.c + \overline{a}.\bar{b}.\overline{c} \\ &= \sum(0,2,3). \end{aligned}$$

Corollary 1. *Let $I = \{0, \cdots, 2^n - 1\}$ and I_1, I_2 are two subsets of I such that $I_1 \cap I_2 = \varnothing$ and $I_1 \cup I_2 = I$, f is a boolean function of n logical variables.*

If $f(a_0, \cdots, a_{n-1}) = \sum_{i \in I_1} m_i$ is the SOP form, then the POS is defined by $f(a_0, \cdots, a_{n-1}) = \prod_{j \in I_2} M_j$ and the complement of f is given by $f'(a_0, \cdots, a_{n-1}) = \prod_{i \in I_1} m_i = \sum_{j \in I_2} M_j$.