

INTRODUCTION

Boolean algebra was defined in 1847 by George Boole (1815-1864), an English physicist. It is an algebra applicable to logical reasoning that deals with functions with binary variables (only two values), i.e. it cannot be applied to systems with more than two states. Boolean algebra can be used to manipulate logical values. Note that a logical value has only two possible states: True (1) or False (0). Several logical values can be combined to give a result, which is also a logical value.

Definitions

State : Logical states are represented by 0 and 1.

Variable: A quantity represented by a symbol, which can take two states (0 or 1).

Function: This represents a group of variables linked by logical operators.

Example

A large number of physical phenomena can be associated with a logical state Example: (stop, run) (open, close) (black, white) (forward, reverse) (on, off).

Logical state 1 is generally associated with the actuated state of the component.

1. COMBINATORIAL LOGIC**1.1 DEFINITION OF COMBINATORIAL LOGIC**

Combinatorial logic uses logic functions to construct a combinatorial system.

A system is said to be combinatorial when it is of the open-loop type, i.e. none of the outputs is looped through as an input.

Each input combination corresponds to a single output. Combinatorial systems are the simplest and can be represented by a truth table indicating the corresponding output state for each input state.

Any logic function can be implemented using a small number of basic logic functions known as **logic operators** or **gates**. There are two types of operator:

- **Basic operators** : NOT, AND, OR
- **Derived operators** : NOT-AND (NAND), NOT-OR (NOR) , XOR, NXOR

1.1 PROPRIETES DE L'ALGEBRE DE BOOLE

	Somme de produits ($\Sigma\Pi$)	Produit de sommes ($\Pi\Sigma$)
<i>AXIOMES</i>		
<i>Associativité</i>	$(a + b) + c = a + (b + c) = a + b + c$	$(a.b).c = a.(b.c) = a.b.c = abc$
<i>Commutativité</i>	$a + b = b + a$	$a.b = b.a$
<i>Distributivité</i>	$a.(b + c) = a.b + a.c$	$a + (b.c) = (a + b).(a + c)$ *
<i>Élément neutre</i>	$a + 0 = a$	$a.1 = a$
<i>Complément</i>	$a + \bar{a} = 1$	$a.\bar{a} = 0$
		*: démontrer au moyen de tables de vérité
<i>THEOREMES</i>		
<i>Idempotence</i>	$a + a = a$	$a.a = a$
<i>Absorption</i>	$a + (a.b) = a$	$a.(a + b) = a$
	$a + 1 = 1$	$a.0 = 0$
<i>Involution</i>	$\bar{\bar{a}} = a$	$\bar{\bar{a}} = a$
<i>Loi de Morgan</i>	$\overline{a + b} = \bar{a}.\bar{b}$	$\overline{a.b} = \bar{a} + \bar{b}$
<i>Concensus de 1re espèce</i>	$a + \bar{a}.b = a + b$	$a.(\bar{a} + b) = a.b$
<i>Concensus de 2e espèce</i>	$ab + \bar{b}c = ab + \bar{b}c + ac$	$(a + b).(\bar{b} + c) = (a + b).(\bar{b} + c).(a + c)$

DUALITY OF BOOLEAN ALGEBRA

Any logical expression remains true if we replace the AND by the OR, the OR by the AND, the 1 by 0, the 0 by 1.

$$A + 1 = 1 \rightarrow A . 0 = 0$$

$$A + \bar{A} = 1 \rightarrow A . \bar{A} = 0$$

MORGAN'S LAW

The logical complemented sum of two variables is equal to the product of the complements of the two variables

$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

The logical product of two variables is equal to the logical sum of the complements of the two variables.

$$\overline{A \cdot B} = \overline{A} + \overline{B}$$

➤ *Generalisation of the MORGANE Theorem to N variables*

$$\overline{A \cdot B \cdot C \dots} = \overline{A} + \overline{B} + \overline{C} + \dots$$

$$\overline{\overline{A} + \overline{B} + \overline{C} + \dots} = \overline{\overline{A} \cdot \overline{B} \cdot \overline{C} \dots}$$

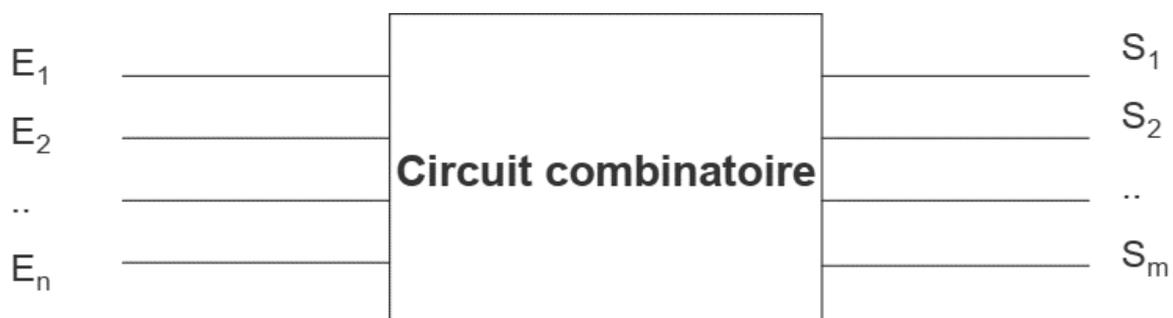
1.3. COMBINATIONAL CIRCUITS

A logic circuit (or combinatorial circuit) is an assembly of logic gates linked together to represent an algebraic expression.

A combinatorial circuit is a digital circuit whose outputs depend solely on the

$$S_i = F(E_i)$$

$$S_i = F(E_1, E_2, \dots, E_n)$$



Combinatorial circuits are used to create other, more complex circuits.

1.4.1 ESTABLISHING THE TRUTH TABLE

Logic functions can be represented by **truth tables**. The truth table shows the output of a logic circuit as a function of the various combinations of input values.

- **The number of columns is the total number of inputs and outputs**
- **The number of lines is 2^N , where "N" is the number of inputs,**

A function with 3 inputs and 1 output is represented by a table with 4 columns and 8 rows.

A	B	C	Résultat
0	0	0	$\bar{A}\bar{B}\bar{C}$
0	0	1	$\bar{A}\bar{B}C$
0	1	0	$\bar{A}B\bar{C}$
0	1	1	$\bar{A}BC$
1	0	0	$A\bar{B}\bar{C}$
1	0	1	$A\bar{B}C$
1	1	0	$AB\bar{C}$
1	1	1	ABC

LOGICAL FUNCTION

This is a function that links N logical variables with a set of basic logical operators.

There are three basic operators in Boolean algebra: NOT , AND , OR .

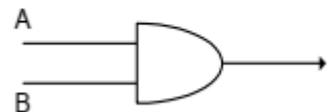
The value of a logic function is equal to 1 or 0, depending on the values of the logic variables.

If a logic function has N logic variables $\rightarrow 2^n$ combinations \rightarrow the function has 2^n values.

TRUTH TABLES FOR BASIC OPERATORS**Logical function (AND)**Representation : $F = A * B$ or $A \cdot B$ or AB

Table de vérité

Entrée		Sortie
B	A	F
0	0	0
0	1	0
1	0	0
1	1	1

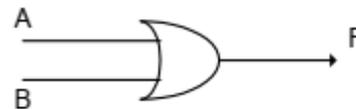


Symbole graphique

Logical function (OR)Representation : $F = A + B$

Table de vérité

Entrée		Sortie
B	A	F
0	0	0
0	1	1
1	0	1
1	1	1

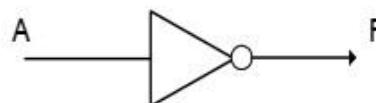


Symbole graphique

Logical function (Not)Representation : $F = \bar{A}$

Table de vérité

Entrée	Sortie
A	F
0	1
1	0



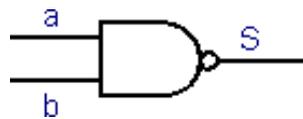
Symbole graphique

FUNCTIONS DERIVED FROM FUNDAMENTAL FUNCTIONS

Logical function NAND*Représentation* : $F = A/B$

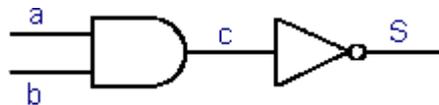
Truth table

A	B	A/B
0	0	1
0	1	1
1	0	1
1	1	0



NAND graphic symbol

A NAND circuit is obtained by connecting an AND gate and an inverter in series, as shown below

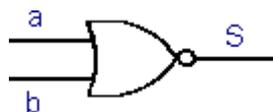


Decomposition of a NAND circuit

Logical function (NOR)*Representation* : $F = A \downarrow B$

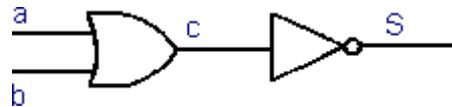
Truth table

A	B	A↓B
0	0	1
0	1	0
1	0	0
1	1	0



NOR graphic symbol

A **NOR** circuit is obtained by connecting an **OR** gate and an inverter in series as shown below



Decomposition of a NOR circuit

XOR Function (EXCLUSIVE OR)

Dans le cas du **OU exclusif**, pour $S = 1$, il faudra que **a OU b** soit à **1** exclusivement, c'est-à-dire que la sortie sera égale à 1 si les deux entrées sont d'états différentes (l'un égale à 1 et l'autre à zéro)

On écrit alors que $F = a \oplus b$ que l'on énonce **F égal a OU exclusif b**.

Le signe \oplus est le symbole du XOR (**OU exclusif**) dans les équations logiques.

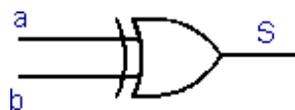
In the case of **exclusive OR**, for $S = 1$, **a OR b** must be **exclusively 1**, i.e. the output will be equal to 1 if the two inputs are in different states (one equal to 1 and the other to zero).

We then write that $F = a \oplus b$ which is expressed as **F equals a OR exclusive b**.

The sign \oplus is the symbol for XOR(**exclusive OR**) in logic equations.

Truth table

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



Symbole graphique du XOR

Note:

AND, OR, NAND, NOR gates can have more than two inputs.

There is no XOR (eXclusive OR) with more than two inputs.

Example: Let F1 and F2 be two logic functions with their respective truth tables

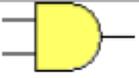
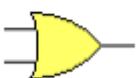
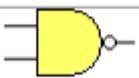
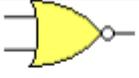
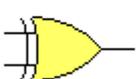
$$F1 = XY + YZ + XZ$$

X	Y	Z	XY	YZ	XZ	F1
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	1
1	0	0	0	0	0	0
1	0	1	0	0	1	1
1	1	0	1	0	0	1
1	1	1	1	1	1	1

$$F2 = X + YZ + \bar{Y}\bar{Z}W$$

X	Y	Z	W	YZ	$\bar{Y}\bar{Z}W$	F2
0	0	0	0	0	0	0
0	0	0	1	0	1	1
0	0	1	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	0	1	0	0	0
0	1	1	0	1	0	1
0	1	1	1	1	0	1
1	0	0	0	0	0	1
1	0	0	1	0	1	1
1	0	1	0	0	0	1
1	0	1	1	0	0	1
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	1	0	1
1	1	1	1	1	0	1

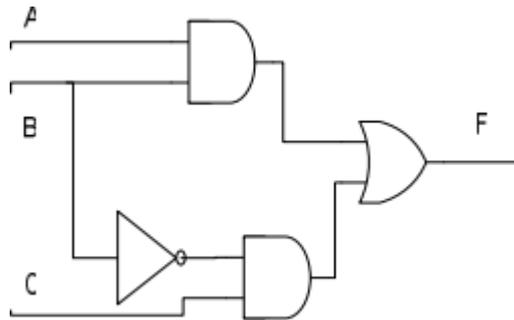
SUMMARY TABLE

Opérateurs de base	NON (NOT) Inversion	-	 /a		<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>A.B</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	A	B	A.B	0	0	0	0	1	0	1	0	0	1	1	1	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>A+B</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	A	B	A+B	0	0	0	0	1	1	1	0	1	1	1	1															
	A	B	A.B																																																
	0	0	0																																																
0	1	0																																																	
1	0	0																																																	
1	1	1																																																	
A	B	A+B																																																	
0	0	0																																																	
0	1	1																																																	
1	0	1																																																	
1	1	1																																																	
ET (AND)	.	 a.b	<table border="1"> <thead> <tr> <th>A</th> <th>/A</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	/A	0	1	1	0	<p>NOT, NON</p>	<p>AND, ET</p>	<p>OR, OU</p>																																							
A	/A																																																		
0	1																																																		
1	0																																																		
OU (OR)	+	 a+b																																																	
Combinaisons d'opérateurs	NAND (NOT-AND)	/	 a/b	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>A/B</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	B	A/B	0	0	1	0	1	1	1	0	1	1	1	0	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>A↓B</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	B	A↓B	0	0	1	0	1	0	1	0	0	1	1	0	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>A ⊕ B</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	B	A ⊕ B	0	0	0	0	1	1	1	0	1	1	1	0
	A	B	A/B																																																
	0	0	1																																																
0	1	1																																																	
1	0	1																																																	
1	1	0																																																	
A	B	A↓B																																																	
0	0	1																																																	
0	1	0																																																	
1	0	0																																																	
1	1	0																																																	
A	B	A ⊕ B																																																	
0	0	0																																																	
0	1	1																																																	
1	0	1																																																	
1	1	0																																																	
NOR (NOT-OR)	↓	 a↓b																																																	
XOR ou exclusif	⊕	 a⊕b	<p>NAND, ET-NON</p>	<p>NOR, OU-NON</p>	<p>XOR, OUX</p>																																														

LOGIC CIRCUIT DIAGRAM (LOGIGRAM)

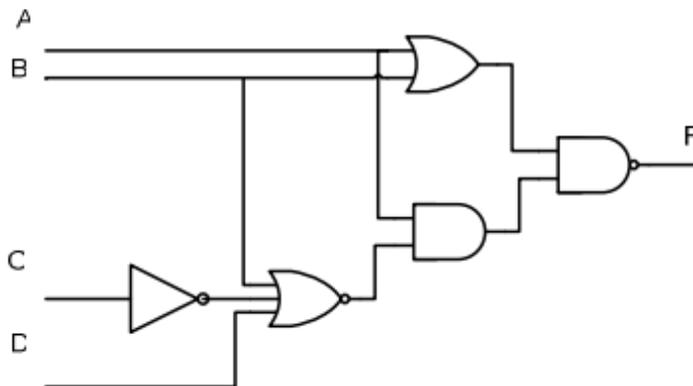
- This is the translation of the logic function into an electronic diagram.
- The principle is to replace each logical operator by the logical gate that corresponds to it.

Example 1: $F(A,B,C) = AB + \bar{B}C$



Example 2

$$F(A,B,C,D) = \overline{(A+B) \cdot (B+\bar{C}+D)} \cdot A$$



CANONICAL FORMS

A	B	C	S	
0	0	0	0	→ $A+B+C$: max terme
0	0	1	0	→ $A+B+\bar{C}$: max terme
0	1	0	0	→ $A+\bar{B}+C$: max terme
0	1	1	1	→ $\bar{A}.B.C$: min terme
1	0	0	0	→ $\bar{A}+B+C$: max terme
1	0	1	1	→ $A.\bar{B}.C$: min terme
1	1	0	1	→ $A.B.\bar{C}$: min terme
1	1	1	1	→ $A.B.C$: min terme

The canonical form of a function is the form where each term of the function includes all the variables.

Example:

$$F(A, B, C) = AB\bar{C} + A\bar{C}B + \bar{A}BC$$

There are several canonical forms: the most commonly used are the first and second forms.

The first canonical form

- First canonical form (disjunctive form): sum of products
- This is the sum of the min terms.
- A disjunction of conjunctions.

Example

$$F(A,B,C) = \bar{A}.B.C + A.\bar{B}.C + A.B.\bar{C} + A.B.C$$

The first canonical form is the most commonly used form

The second canonical form

- Second canonical form (conjunctive): product of sums
- The product of max terms
- Conjunction of disjunctions

Example

$$F(A,B,C) = (A+B+C)(A+B+\bar{C})(A+\bar{B}+C)(\bar{A}+B+C)$$

Note: The first and second canonical forms are equivalent

Note

- Any logical function can always be reduced to one of the canonical forms.
- This means adding the missing variables to the terms that do not contain all the variables (the non-canonical terms).
- This can be done using the rules of Boolean algebra:
- Multiply a term with an expression that is 1
- Add to a term with an expression worth 0 and then do the distribution

Example

$$\begin{aligned}
 1. F(A, B) &= A + B \\
 &= A(B + \bar{B}) + B(A + \bar{A}) \\
 &= AB + A\bar{B} + AB + \bar{A}B \\
 &= AB + A\bar{B} + \bar{A}B
 \end{aligned}$$

$$\begin{aligned}
 2. F(A, B, C) &= AB + C \\
 &= AB(C + \bar{C}) + C(A + \bar{A}) \\
 &= ABC + AB\bar{C} + AC + \bar{A}C \\
 &= ABC + AB\bar{C} + AC(B + \bar{B}) + \bar{A}C(B + \bar{B}) \\
 &= ABC + AB\bar{C} + ABC + A\bar{B}C + \bar{A}BC + \bar{A}\bar{B}C \\
 &= ABC + AB\bar{C} + A\bar{B}C + \bar{A}BC + \bar{A}\bar{B}C
 \end{aligned}$$

Simplifying logic functions

Once we have the expression of a logic circuit, it may be possible to minimise it to obtain an equation with fewer terms or variables per term. This simplification can be done in different ways:

- By using the properties of logic functions
- By using Karnaugh tables
- From a truth table

Exemple 1

$$\begin{aligned}
Z &= (\bar{A}+B). (A+B+D)\bar{D} \\
&= \bar{A}. \overline{\bar{A}. \bar{D}} + \bar{A}. B. \bar{D} + \bar{A}. \bar{D}. \bar{D} + B. A. \bar{D} + B. B. \bar{D} + B. \bar{D}. \bar{D} \\
&\quad \uparrow \qquad \qquad \qquad \uparrow \qquad \qquad \qquad \uparrow \\
&\quad 0 \qquad \qquad \qquad 0 \qquad \qquad \qquad 0 \\
&= \bar{A}. B. \bar{D} + B. A. \bar{D} + B. \bar{D} \\
&= B. \bar{D}. (\bar{A}+A) + B. \bar{D} \\
&= B. \bar{D}. (\bar{A}+A+1) \\
&= B. \bar{D}
\end{aligned}$$

Exemple

$$\begin{aligned}
Z &= A. \bar{B} + \bar{A}. \bar{B} + \bar{A}. B \\
&= \bar{B}. (A + \bar{A}) + \bar{A}. B \\
&= \bar{B} + \bar{A}. B \\
&= \bar{A} + \bar{B} \qquad \text{car } X + \bar{X}. Y = X + Y
\end{aligned}$$

THE KARNAUGH DIGRAM

The Karnaugh digram is a graphical tool used to mathematically simplify a Boolean expression in a logic circuit.

The rules for creating a Karnaugh chart are as follows:

CD AB	00 $\bar{C}\bar{D}$	01 $\bar{C}D$	11 CD	10 $C\bar{D}$
00 $\bar{A}\bar{B}$	$\bar{A}\bar{B}\bar{C}\bar{D}$		$\bar{A}\bar{B}CD$	
01 $\bar{A}B$		$AB\bar{C}D$		
11 AB	$AB\bar{C}D$		$ABCD$	
10 $A\bar{B}$				$A\bar{B}C\bar{D}$

The Karnaugh diagram of a logic function with n variables consists of a rectangle divided into 2^N squares. Each cell corresponds to a combination of the logical states that the output variable can take.

The order of the variables on the abscissa and ordinate is chosen so that when you move from one cell to an adjacent cell, only one variable is modified (the table is coded using the Gray code).

The grouping of squares must be equal to a power of 2, i.e. 2, 4, 8... squares. For the grouping of peripheral cells, we assume that the array is cylindrical, and for the grouping of corner cells, we can assume that the array is spherical.

Simplification steps using the Karnaugh method

1. The number of input variables is determined in order to know the size of the table.
2. Draw up the Karnaugh table using Gray's code
3. Fill in the table according to the terms of the functions to be simplified
4. Group the cells optimally
5. Group adjacent cells.
6. Output the simplified function terms

A. Case of two variables

	\bar{B}	B
\bar{A}	1	0
A	0	1

$X = \bar{A} \cdot \bar{B} + A \cdot B$

	\bar{B}	B
\bar{A}	1	1
A	0	1

$X = \bar{A} + B$

B. Case of three variables

	\bar{C}	C
$\bar{A} \cdot \bar{B}$	0	0
$\bar{A} \cdot B$	1	0
$A \cdot B$	1	0
$A \cdot \bar{B}$	0	0

$X = B \cdot \bar{C}$

	\bar{C}	C
$\bar{A} \cdot \bar{B}$	0	0
$\bar{A} \cdot B$	1	1
$A \cdot B$	0	0
$A \cdot \bar{B}$	0	0

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

Case of four variables

	$\bar{C} \cdot \bar{D}$	$\bar{C} \cdot D$	$C \cdot D$	$C \cdot \bar{D}$
$\bar{A} \cdot \bar{B}$	0	0	0	1
$\bar{A} \cdot B$	0	1	1	0
$A \cdot B$	0	1	1	0
$A \cdot \bar{B}$	0	0	1	0

$X = \bar{A} \cdot \bar{B} \cdot C \cdot \bar{D} + A \cdot C \cdot D + B \cdot D$

	$\bar{C} \cdot \bar{D}$	$\bar{C} \cdot D$	$C \cdot D$	$C \cdot \bar{D}$
$\bar{A} \cdot \bar{B}$	0	1	0	0
$\bar{A} \cdot B$	0	1	1	1
$A \cdot B$	0	0	0	1
$A \cdot \bar{B}$	1	1	0	1

$X = \bar{A} \cdot B \cdot D + B \cdot C \cdot \bar{D} + \bar{B} \cdot \bar{C} \cdot D + A \cdot \bar{B} \cdot \bar{D}$

Special case and indeterminate element

It sometimes happens that a function is indefinite for certain combinations of variables, for various reasons; the most common is that certain combinations of variables are impossible, so it is not considered useful to give a particular value to the function for these combinations.

In the corresponding cells of the Karnaugh table, we place a special sign (\emptyset : indeterminate element).

When the cells are grouped together, we transform the \emptyset into 0 or 1, depending on the convenience or simplifications that may result.

Example

$$F = \{ 4, 7, 6 \} + \emptyset \{ 1,5 \}$$

A \ BC	0 0	0 1	1 1	1 0
0	0 0	X 1	0 3	0 2
1	1 4	X 5	1 7	1 6

Here we obtain the simplest expression of F by transforming the X in square 5 into a '1', which makes it possible to group squares 4, 5, 7 and 6 into a group of 4 and leaving the X in square 1 as it is

We therefore have : $F = A$.

We consider the 'X' to be a '1' if it can give us a better grouping.