Chapter 1

Boolean Algebra

1.1 Boolean Expressions

Definition 1. Let $x_1, x_2, \ldots, x_n$ be a set of n variables (or letters or symbols). A *Boolean Polynomial (Boolean expression, Boolean form or Boolean formula)* $p(x_1, x_2, \ldots, x_n)$ in the variables $x_1, x_2, \ldots, x_n$ is defined recursively as follows:

1. The symbols 0 to 1 are Boolean polynomials

2. $x_1, x_2, \ldots, x_n$ are all Boolean polynomials

3. if $p(x_1, x_2, \ldots, x_n)$ and $q(x_1, x_2, \ldots, x_n)$ are two Boolean polynomials, then so are

$$p(x_1, x_2, \ldots, x_n) \vee q(x_1, x_2, \ldots, x_n)$$

and

$$p(x_1, x_2, \ldots, x_n) \wedge q(x_1, x_2, \ldots, x_n)$$

4. If $p(x_1, x_2, \ldots, x_n)$ is a Boolean polynomial, then so is $(p(x_1, x_2, \ldots, x_n))'$.

5. There are no Boolean polynomials in the variables $x_1, x_2, \ldots, x_n$ other than those obtained in accordance with rules 1 to 4.

Thus, Boolean expression is an expression built from the variables given using Boolean operations $\vee$, $\wedge$ and $'$.

For example, for variable x, y, z, the expression

$$\begin{aligned}
p_1(x, y, z) &= (x \vee y) \wedge z \\
p_2(x, y, z) &= (x \vee y') \wedge (y \wedge 1) \\
p_3(x, y, z) &= (x \vee (y' \wedge z)) \vee (x \wedge (y \wedge 1))
\end{aligned}$$

are Boolean expressions.

Notice that a Boolean expression is n variable may or may not contain all the b variables. Obviously, an infinite number of Boolean expressions may be constructed in n variables.

Definition 2 (Literal)**.** A *literal* is a variable or complemented variable such as x, x', y, y', and so on.

Definition 3 (Fundamental Product)**.** A *fundamental product* is a literal or a product of two or more literal in which no two literal involve same variable.

For example,

$$x \wedge z',\ x \wedge y' \wedge z,\ x,\ y'\ x' \wedge y \wedge z$$

are fundamental products whereas

$$x \wedge y \wedge x' \wedge z \text{ and } x \wedge y \wedge z \wedge y$$

are not fundamental products.

Remark. Fundamental product is also called a *minterm* or *complete product*. In what follows we shall denote $x \wedge y$ by xy.

Any product of literals can be reduced to either 0 or a fundamental product.

For example, consider $x\ y\ x'\ z$. Since, $x \wedge x' = 0$ by complement law, we have $xyx'z = 0$.

Similarly, if we consider $x\ y\ z\ y$ then since $y \wedge y = y$ (idempotent law), we have $xyzy = xyz$, which is a fundamental product.

Definition 4. A fundamental product P_1 is said to be contained in (or included in) another fundamental Product P_2 if the literals of P_1 are also literals of P_2.
For example, $x'z$ is contained in $x'yz$ but $x'z$ is not contained in $xy'z$ since x' is not a literal of $xy'z$.

Observe that if P_1 is contained in P_2, say $P_2 = P_1 \wedge Q$, then, by the absorption law,

$$P_1 \vee P_2 = P_1 \vee (P_1 \wedge Q) = P_1$$

For example,

$$x'z \vee x'yz = x'z$$

Definition 5. A Boolean expression E is called a sum-of-products expression(disjunctive Normal Form or DNF) if E is a fundamental product or the sum (join) of two or more fundamental products none of which is contained in another.

Definition 6. Two Boolean expression $P(x_1, x_2, \ldots, x_n)$ and $Q(x_1, x_2, \ldots, x_n)$ are called equivalent (or equal) if one can be obtained from the other by a finite number of applications of the identities of a Boolean algebra.

Definition 7. Let E be any Boolean expression. A sum of product form of E is an equivalent Boolean sum of products expression.
Example: Consider the expression

$$E_1(x, y, z) = xz' + y'z + xyz'$$

Although the expression E_1 is a sum of products, it is not a sum-of-products expression because, the product xz' is contained in the product xyz'. But, by absorption law, E_1 can be expressed as

$$E_1(x, y, z) = xz' + y'z + xyz' = xz' + xyz' + y'z = xz' + y'z$$

, which is a sum-of-product form for E_1.

1.2 Algorithm for Finding Sum-of-Products Forms

The input is a Boolean expression E. The output is a sum-of-products expression equivalent to E.

Step 1. Use De Morgan's Law and involution to move the complement operation into any parenthesis until finally the complement operation only applies to variables. Then E will consist only sums and products of literals.

Step 2. Use the distributive operation to next transform E into a sum of products.

Step 3. Use the commutative, idempotent, and complement laws to transform each product in E into 0 or a fundamental product.

Step 4. Use the absorption law and identity law to finally transform E into a sum of products expression.

For example, we apply the above Algorithm to the Boolean expression,

$$E = ((xy)'z)'((x'+z)(y'+z'))'$$

Step 1. Using De Morgan's laws and involution, we obtain

$$\begin{aligned} E &= ((xy)'' + z')((x'+z) + (y'+z')') \\ &= (xy + z')(xz' + yz) \end{aligned}$$

Thus E consists only of sum and products of literals.

Step 2. Using the distributive laws, we obtain

$$\begin{aligned} E &= (xy + z')(xz' + yz) \\ &= xyxz' + xyyz + xz'z' + yzz' \end{aligned}$$

Thus E is now a sum of products.

Step 3. Using commutative idempotent and complement law, we obtain

$$E = xyz' + xyz + xz' + 0$$

Each term in E is a fundamental product or 0.

Step 4. The product xz' is contained in xyz'; hence, by the absorption law,

$$xz' + (xz'y) = xz'$$

Thus we may delete xyz' form the sum. Also, by the identity law for 0, we may delete 0 from the sum. Accordingly,

$$E = xyz + xz'$$

E is now represented by a sum-of-products expression.

1.3 Consensus of Fundamental Products

Let P_1 and P_2 be fundamental products such that exactly one variable say x_k appears uncomplemented in one of P_1 and P_2 and complemented in the other. Then the *consensus of* P_1 *and* P_2 is the product (without repetitions) of the literals of P_1 and P_2 after x_k and x_k' are deleted. (we do not define the consensus of $P_1 = x$ and $P_2 = x'$)

Lemma 1. Suppose Q is the consensus of P_1 and P_2. Then $P_1 + P_2 + Q = P_1 + P_2$.

Proof. Since the literals commute, we can assume without loss of generality that

$$P_1 = a_1a_2\dots a_rt \qquad P_2 = b_1b_2\dots b_st' \qquad Q = a_1a_2\dots a_rb_1b_2\dots b_s$$

Now, $Q = Q(t+t') = Qt + Qt'$. Because Qt contains P_1, $P_1 + Qt = P_1$; and because Qt' contains P_2, $P_2 + Qt' = P_2$. Hence,

$$\begin{aligned} P_1 + P_2 + Q &= P_1 + P_2 + Qt + Qt' \\ &= (P_1 + Qt) + (P_2 + Qt') \\ &= P_1 + P_2. \end{aligned}$$

□

Problem 1.3.1. Find the consensus Q of P_1 and P_2, where

(i) $P_1 = xyz's$, $P_2 = xy't$

(ii) $P_1 = xy'$, $P_2 = y$

(iii) $P_1 = x'yz$, $P_2 = x'yt$

(iv) $P_1 = x'yz$, $P_2 = xyz'$.

Solution.

(i) $P_1 = xyz's$, $P_2 = xy't$
Delete y and y' and then multiply the literals of P_1 and P_2 (without repetition) to obtain $Q = xz'st$

(ii) $P_1 = xy'$, $P_2 = y$ Delete y and y' and then multiply the literals of P_1 and P_2 (without repetition) to obtain $Q = x$

(iii) $P_1 = x'yz$, $P_2 = x'yt$ In this case, no variable appears uncomplemented in one of the products and complemented in the other. Hence, P_1 and P_2 have no consensus.

(iv) $P_1 = x'yz$, $P_2 = xyz'$ Each of x and z appear complemented in one of the products and uncomplemented in the other. Hence, P_1 and P_2 have no consensus.

1.4 Consensus Method For Finding Prime Implicants

The following algorithm, known as *consensus Method* is used to find the prime implicants of a Boolean expression.

1.4.1 Algorithm (Consensus Method)

The input is a Boolean expression $E = P_1 + P_2 + \cdots + P_m$, where P_m are fundamental products. The output expresses E as a sum of its prime implicants.

Step 1. Delete any fundamental product P_i which includes any other fundamental product P_j (this is permissible by the absorption law).

Step 2. Add the consensus of any P_i and P_j providing Q does not include any of the P_i (this is permissible by the lemma $P_1 + P_2 + \cdots + P_n + Q = P_1 + \cdots + P_n$).

Step 3. Repeat step 1/or step 2 until neither can be applied.

Example. Let $E(x, y, z) = xyz + x'z' + xyz' + x'y'z + x'yz'$.
Then

$$\begin{aligned}
E &= xyz + x'z' + xyz' + x'y'z && (x'yz' \text{ includes } x'z') \\
&= xyz + x'z' + xyz' + x'y'z + xy && (\text{ consensus of } xyz, \text{ and } xyz' \text{ added}) \\
&= x'z' + x'y'z + xy && (xyz \text{ and } xyz' \text{ include } xy) \\
&= x'z' + x'y'z + xy + x'y' && (\text{ consensus } x'y' \text{ of } x'z' \text{ and } x'y'z \text{ added}) \\
&= x'z' + xy + x'y' && (x'y'z \text{ includes } x'y') \\
&= x'z' + xy + x'y' + yz' && (\text{ consensus of } x'z' \text{ and } xy, \text{ which is } yz', \text{ added})
\end{aligned}$$

After this none of the step in the consensus method will change E. Thus, E is the sum of its prime implicants $x'z'$, xy, $x'y'$, and yz'.

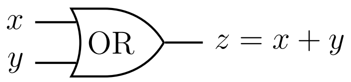
1.5 Logic Gates and Circuits

Definition 8 (Logic Circuit)**.** *Logic circuits* (also called *logic networks*) are structures which are built up from certain elementary circuits called logic gates.

1.5.1 Logic Gates

There are three basic logic gates. The lines (wires) entering the gate symbol from the left are input lines and the single line on the right is the output line.

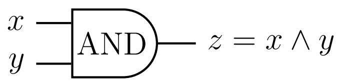
1. *OR Gate*: An OR gate has input x and y and output $z = x \vee y$ or $z = x + y$ where addition (or Join) is defined by the truth table. In this case the output $z = 0$ only when inputs $x = 0$ and $y = 0$.
   The symbol and the truth table for OR gate are shown in the diagram below:



x	y	$x + y$
1	1	1
1	0	1
0	1	1
0	0	0

Table 1.1: Truth table for OR gate

2. *AND gate*: In this gate the inputs are x and y and output is $x \wedge y$ or $x \cdot y$ or xy, where multiplication is defined by the truth table.

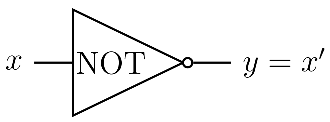


x	y	$x \wedge y$
1	1	1
1	0	0
0	1	0
0	0	0

Table 1.2: Truth table for AND gate

   Thus output is 1 only when $x = 1$, $y = 1$, otherwise it is zero. The AND gate may have more than two inputs. The output in such a case will be 1 if all inputs are 1.

3. *NOT Gate (inverter)*: The diagram below shows NOT gate with input x and output $y = x'$, where inversion, denoted by the prime, is defined by the truth table:



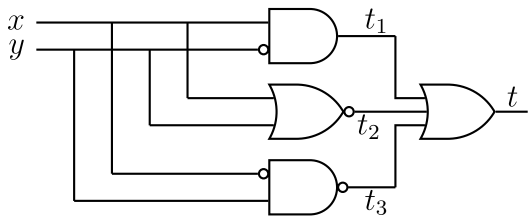
x	$y = x'$
1	0
0	1

Table 1.3: Truth table for NOT gate

   For example, if $x = 10101$, then output x' in NOT gate shall be $x' = 01010$.

Theorem 1.5.1. Logic circuits form a Boolean Algebra.

Problem 1.5.1. Express the output of the logic circuit below as a Boolean expression. (Here small circle represents complement(NOT))



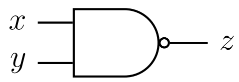
Solution. We note that

$$t_1 = xy'$$
$$t_2 = (x+y)'$$
$$t_3 = (x'y)'$$

and so we have

$$\begin{aligned} t &= t_1 + t_2 + t_3 \\ &= xy' + (x+y)' + (x'y)' \end{aligned}$$

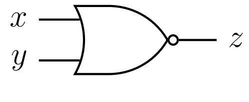
NAND and NOR gates are frequently used in computers.
NAND gate: It is equivalent to AND gate followed by a NOT gate. The symbol and the truth table for NAND gate are shown in the diagram below:



x	y	xy	$z = (xy)'$
1	1	1	0
1	0	0	1
0	1	0	1
0	0	0	1

Table 1.4: Truth table for NAND gate

Thus, the output of a NAND gate is 0 if and only if all the inputs are 1. *NOR gate*: This gate is equivalent to OR gate followed by a NOT gate. The symbol and the truth table for NAND gate are shown in the diagram below:



x	y	$x+y$	$(x+y)'$
1	1	1	0
1	0	1	0
0	1	1	0
0	0	0	1

Table 1.5: Truth table for NOR gate

Thus, the output of a NOR gate is 1 if and only if all the inputs are 0.

1.6 Boolean Function

We know that ordinary polynomials could produce functions by substitution. For example, the polynomial $xy + yz^3$ produces a function $f : \mathbb{R}^3 \to \mathbb{R}$ by letting $f(x, y, z) = xy + yz^3$. Thus, $f(3, 4, 2) = 3 \cdot 4 + 4 \cdot 2^3 = 44$. In a similar way, Boolean polynomials involving n variables produce functions from B_n to B.

Definition 9 (Boolean Function)**.** Let $(B, \cdot, +, ', 0, 1)$ be a Boolean algebra. A function $f : B_n \to B$ which is associated with a Boolean expression (polynomial) in n variables is called a Boolean function.

Thus, a Boolean function is completely determined by the Boolean expression $\alpha(x_1, x_2, \ldots, x_n)$ because it is nothing but the evaluation function of the expression. It may be mentioned here that every function $g : B_n \to B$ needs not be a Boolean function.

If we assume that the Boolean algebra B is of order 2^m for $m \geq 1$, then the number of function from B_n to B is greater than 2^{2n} showing that there are functions from B_n to B which are not Boolean functions. On the other hand, for $m = 1$, that is, for a two element Boolean algebra, the number of function from B_n to B is B^{2n} which is same as the number of distinct Boolean expression in n variable. Hence, every function from B_n to B in this case is a Boolean function.

Problem 1.6.1. Show that the following Boolean expression are equivalent to one-another. Obtain their sum-of-product canonical form.

(a) $(x + y)(x' + z)(y + z)$

(b) $(xz) + (x'y) + (yz)$

(c) $(x + y)(x' + z)$

(d) $xz + x'y$

Solution. The binary valuation of the expression are

x	y	z	$x+y$	$x'+z$	$y+z$	(a)	(c)	xz	$x'y$	yz	(b)	(d)
0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	1	0	1	1	0	0	0	0	0	0	0
0	1	0	1	1	1	1	1	0	1	0	1	1
0	1	1	1	1	1	1	1	0	1	1	1	1
1	0	0	1	0	0	0	0	0	0	0	0	0
1	0	1	1	1	1	1	1	1	0	0	1	1
1	1	0	1	0	1	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	0	1	1	1

Since the values of the given Boolean expression are equal over every triple of the two element Boolean algebra, they are equal.

Two find the sum-of-product canonical (complete) form, we note that (d) is in sum-of-product form. Therefore, to find complete sum-of-product form, we have

$$\begin{aligned}
\text{(d)} &= (xz) + (x'z) \\
&= xz(y + y') + x'z(z + z') \\
&= xzy + xzy' + x'yz + x'yz'
\end{aligned}$$

1.7 Method to Find Truth Table of A Boolean Function

Consider a logic circuit consisting of 3 input devices x, y, z. Each assignment of a set of three bits to the input x, y, z yield an output bit for z. There are $2n = 2^3 = 8$ possible ways to assign bits to the input as follows:

$$000,\ 001,\ 010,\ 011,\ 100,\ 101,\ 110,\ 111.$$

The assumption is that the sequence of first bits is assigned to x, the sequence of second bits to y, and the sequence of third bits to z. Thus, the above set of inputs may be rewritten in the form

$$x = 00001111,\ y = 00110011,\ z = 01010101$$

These three sequences (of 8 bits) contain the eight possible combination of the input bits.
The truth table $T = T(L)$ of the circuit L consists of the output t that corresponds to the input sequences x, y, z.
The truth table is same as we generally have written in vertical columns. The difference is that here we write x, y, z and t horizontally.

Consider a logic circuit L with n input devices. There are many ways to form n input sequences $x_1, x_2, \ldots, x_n$ so that they contain 2^n different possible combinations of the input bits (Each sequence must contain 2^n bits).

The assignment scheme is:

x_1: Assign 2^{n-1} bits which are 0 followed by 2^{n-1} bits which are 1.

x_2: Assign 2^{n-2} bits which are 0 followed by 2^{n-2} bits which are 1.

x_3: Assign 2^{n-3} bits which are 0 followed by 2^{n-3} bits which are 1.

and so on.

The sequence obtained in this way is called "Special Sequence". Replacing 0 by 1 and 1 by 0 in the special sequences yields the complements of the special sequence.

Example. Suppose a logic circuit L has $n = 4$ input devices x, y, z, t. Then $2^n = 2^4 = 16$ bit special sequences for x, y, z, t are
$x = 0000000011111111$ ($2^3 = 8$ zeros followed by 8 ones)
$y = 0000111100001111$ ($2^{n-2} = 2^{4-2} = 4$ zeros followed by 4 ones)
$z = 0011001100110011$ ($2^{n-3} = 2^{4-3} = 2$ zeros followed by 2 ones)
$z = 0101010101010101$ ($2^{n-3} = 2^{4-4} = 2^0 = 1$ zeros followed by 1 ones)

1.7.1 Algorithm For Finding Truth Table For A Logic Circuit L Where Output T is Given by a Boolean Sum-Of-Product Expression in the Inputs

The input is a Boolean sum-of-products expression $t(x_1, x_2, \ldots)$.

Step 1: Write down the special sequences for the inputs $x_1, x_2, \ldots$ and their complements

Step 2: Find each product appearing in $t(x_1, x_2, \ldots)$ keeping in mind that $x_1, x_2, \cdots = 1$ is a position if and only if all $x_1, x_2, \ldots$ have 1 in the position.

Step 3: Find the sum t of the products keeping in mind that $x_1 + x_2 + \cdots = 0$ in a position if and only if all $x_1, x_2, \ldots$ have 0 in the position.

1.8 Representation of Boolean Functions using Karnaugh Map

Karnaugh Map is a graphical procedure to represent Boolean function as an "or" combination of minterms where minterms are represented by squares. This procedure is easy to use with functions $f : B_n \to B$, if n is not greater than 6. We shall discuss this procedure for $n = 2, 3$, and 4.

A Karnaugh map structure is an area which is subdivided into 2^n cells, one for each possible input combination for a Boolean function of n variables. Half of the cells are associated with an input value of 1 for one of the variables and the other half are associated with an input value of 0 for the same variable. This association of cell is done for each variable, with the splitting of the 2^n cells yielding a different pair of halves for each distinct variable.

Case of 1 variables: In this case, the Karnaugh map consists of $2^1 = 2$ squares.

0	1
x'	x

The variable is represented by the right square and its complement x' by the left square.

Case of 2 Variables: For $n = 2$, the Boolean function is of two variable, say x and y. We have $2^2 = 4$ squares, that is, a 2×2 matrix of squares. Each squares contain one possible input from B_2.

The variable x appears in the first row of the matrix as x' whereas x appears in the second row as x. Similarly, y appears in the first column as y' and as y in the second column.

	0	1
0	00	01
1	10	11

	y'	y	
x'	$x'y'$	$x'y$	
	xy'	xy	x

Figure 1.1: 2 variable Karnaugh Map

In this case, x is represented by the points in lower half of the map and y is represented by the points in the right half of the map.

Definition 10. Two fundamental products are said to be *adjacent* if they have the same variables and if they differ in exactly one literal. Thus, there must be an uncomplemented variable in one product which is complemented in the other.

For example, if $P_1 = xyz'$ and $P_2 = xy'z'$, then they are adjacent.
The sum of two such adjacent products will be a fundamental product with one less literal.
For example, in the case of above-mentioned adjacent products,

$$P_1 + P_2 = xyz' + xy'z' = xz'(y + y') = xz'(1) = xz'$$

We note that two squares in Karnaugh map above are adjacent if and only if squares are geometrically adjacent, that is, have a side in common.

We know that a complete sum-of-products Boolean expression $E(x, y)$ is a sum of minterms and hence can be represented in the Karnaugh map by placing checks in the appropriate square. A prime implicant of $E(x, y)$ will be either a pair of adjacent squares in E or an isolated square (a square which is not adjacent to other square of $E(x, y)$). A minimal sum of products for $E(x, y)$ will consists of a minimum number of a prime implicants which cover all the square of $E(x, y)$.

Problem 1.8.1. Find the prime implicants and a minimal sum-of-products form from each of the following complete sum-of-products Boolean expression:

(a) $E_1 = xy + xy'$

(b) $E_2 = xy + x'y + x'y'$

(c) $E_3 = xy + x'y'$

Solution.

(a) The Karnaugh map for E_1 is

	y'	y
x'		
x	✓	✓

Check the squares corresponding to xy and xy'. We note that E_1 consists of one prime implicant, the two adjacent square designated by the loop. The pair of adjacent square represents the variable x. So x is the only prime implicant of E_1. Consequently, $E_1 = x$ is its minimal sum.

(b) The Karnaugh map for E_2 is

y'	y	
✓	✓	x'
	✓	

Check the squares corresponding to xy, $x'y$, $x'y'$. The expression E_2 contains two pairs of adjacent squares (designated by two loops) which include all the squares of E_2. The vertical pair represents y and the horizontal pair x'. Hence, y and x are the prime implicants of E_2. Thus, $E_2(x, y) = x' + y$ is minimal sum.

(c) The Karnaugh map for E_3 is

	y'	y
x'	$x'y'$ ✓	
x		xy ✓

Check (tick) the squares corresponding to xy and $x'y'$. The expression E_3 consists of two isolated squares which represent xy and $x'y'$. Hence xy and $x'y'$ are the prime implicants of E_3 and so $E_3 = xy + x'y'$ is its minimal sum.

Case of 3 variables: We now turn to the case of a function $f : B_3 \to B$ which is function of x, y and z. The Karnaugh map corresponding to Boolean expression $E(x, y, z)$ is shown in the diagram below:

	00	01	11	10
0	000	001	011	010
1	100	101	111	110

	$y'z'$	$y'z$	yz	yz'
x'	$x'y'z'$	$x'y'z$	$x'yz$	$x'yz'$
x	$xy'z'$	$xy'z$	xyz	xyz'

z

z'

Here x, y, z are respectively represented by lower half, right half and middle two quarters of the map.
Similarly, x', y',z' are respectively represented by upper half, left half and left and right quarter of the map.

Definition 11. By a Basic Rectangle in the Karnaugh map with three variables, we mean a square, two adjacent squares or four squares which form a one-by four, or a two by-two rectangle. These basic rectangles correspond to fundamental products of three, two and one literal respectively.

Further, the fundamental product represented by a basic rectangle is the product of just those literals that appear in every square of the rectangle.

Let a complete sum of products Boolean expression $E(x, y, z)$ is represented in the Karnaugh map by placing checks in the appropriate squares. A prime implicant of E will be a maximal basic rectangle of E, i.e., a basic rectangle contained in E which is not contained in any larger basic rectangle in E.

A minimal sum-of-products form for E will consist of a minimal cover of E, i.e., a minimal number of maximal basic rectangles of E which together include all the square of E.

Problem 1.8.2. Find the prime implicants and a minimal sum-of-products form for each of the following complete sum-of-products Boolean expressions:

(a) $E_1 = xyz + xyz' + x'yz' + x'y'z$.

(b) $E_2 = xyz + xyz' + xy'z + x'yz + x'y'z$.

(c) $E_3 = xyz + xyz' + x'yz + x'y'z$.

Solution.

(a) The Karnaugh map for E_1 is

	$y'z'$	$y'z$	yz	yz'
x'		✓		✓
x			✓	✓

We check the four squares corresponding to four summands in E_1. Here E_1 has three prime implicants (maximal basic rectangles) which are encircled. These are xy, yz' and $x'y'z$. All three are needed to cover E_1. Hence, minimal sum for E_1 is $E_1 = xy + yz' + x'y'z$.

(b) The Karnaugh map for E_2 is

	$y'z'$	$y'z$	yz	yz'
x'		✓	✓	
x		✓	✓	✓

Check the squares corresponding to the five summands. E_2 has two prime implicants which are circled. One is the two adjacent squares which represent xy, and the other is the two-by-two square which represents z. Both are needed to cover E_2 so the minimal sum for E_2 is $E_2 = xy + z$.

(c) The Karnaugh map for E_3 is

	$y'z'$	$y'z$	yz	yz'
x'	✓	✓		✓
x			✓	✓

Check the squares corresponding to the five summands. Here E_3 has three prime implicants xy, yz', $x'y'$. All these are needed in a minimal cover of E_3. Hence, E_3 has minimal sum as $E_3 = xy + yz' + x'y'$.

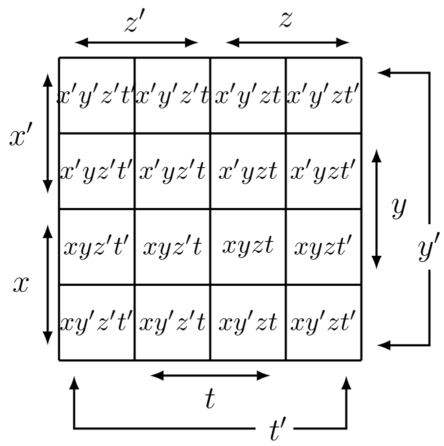
Remark. To find the fundamental product represented by a basic rectangle, find literals which appear in all the squares of the rectangle.

Case of 4 variables: We consider a Boolean function $f : B_4 \to B$, considered as a function of x, y, z and t. Each of the 16 squares (2^4) corresponds to one of the minterms with four variables.

$$xyzt,\ xyzt', ;x'yz't$$

We consider first and last columns to be adjacent, and first and last rows to be adjacent, both by Wrap around, and we look for rectangles with sides of length some power of 2, so the length is 1, 2 or 4. The expression for such rectangles is given by intersecting the large labelled rectangles.

	00	01	11	10
00	0000	0001	0011	0001
01	0100	0101	0111	0110
11	1100	1101	1111	1101
10	1000	1001	1011	1010



A basic rectangle in a four variable Karnaugh map is a square, two adjacent squares, four squares which form one-by-four or two by two rectangle or eight square squares which form a two by four rectangles. These rectangles correspond to fundamental product with four, three, two and one literal respectively. Maximal basic rectangles are prime implicants.

Problem 1.8.3. Find the fundamental product P represented by the basic rectangle in the Karnaugh map given below:

	$y't'$	$z't$	zt	zt'
$x'y'$				
$x'y$				
xy				
xy'	✓	✓		

Solution. We find the literals which appear in all the squares of the basic rectangle. Then P will be the product of such literals.
Here x, y', z appear in both squares. Hence,

$$P = xy'z'$$

is the fundamental product represented by the basic rectangle in question.