Boolean Algebra

Definition: A *Boolean Algebra* is a math construct (B,+, . , ‘, 0,1) where B is a non-empty set, + and . are binary operations in B, ‘ is a unary operation in B, 0 and 1 are special elements of B, such that:

a) + and . are communative: for all x and y in B, x+y=y+x, and x.y=y.x
b) + and . are associative: for all x, y and z in B, x+(y+z)=(x+y)+z, and x.(y.z)=(x.y).z
c) + and . are distributive over one another: x.(y+z)=xy+xz, and x+(y.z)=(x+y).(x+z)
d) Identity laws: 1.x=x.1=x and 0+x=x+0=x for all x in B
e) Complementation laws: x+x’=1 and x.x’=0 for all x in B

Examples:

- (B=set of all propositions, or, and, not, T, F)
- (B=2^A, U, ∩, ^{c}, Φ,A)

Theorem 1: Let (B,+, . , ‘, 0,1) be a Boolean Algebra. Then the following hold:

a) x+x=x and x.x=x for all x in B
b) x+1=1 and 0.x=0 for all x in B
c) x+(xy)=x and x.(x+y)=x for all x and y in B

Proof:

a)	x	= x+0	Identity laws
		= x+xx’	Complementation laws
		= (x+x).(x+x’)	because + is distributive over .
		= (x+x).1	Complementation laws
		= x+x	Identity laws
	x	= x.1	Identity laws
		= x.(x+x’)	Complementation laws
		= x.x +x.x’	because + is distributive over .
		= x.x+0	Identity laws
		= x.x	
b)	x+1	=x+(x+x’)	Complementation laws
		= (x+x)+x’	+ is associative
		= x+x’	using (a)
		= 1	Complementation laws
	0.x	=(x’.x).x	Complementation laws
		= x’.(x.x)	. is associative
		= x’.x	using (a)
		=0	Complementation laws
c)	x+(xy)	= x.1+x.y	Identity laws
		=x.(1+y)	because + is distributive over .
		=x.1	using (b)

$=x$ Identity laws

$x.(x+y) = x.x+x.y$ Distributivity laws
$=x+x.y$ by (a)
$=x$ Just shown above.
Q.E.D.

Definition: An element y in B is called a complement of an element x in B if $x+y=1$ and $xy=0$

Theorem 2: For every element x in B, the complement of x exists and is unique.
Proof:

- Existence. Let x be in B. x' exists because ' is a unary operation. X' is a complement of x because it satisfies the definition of a complement ($x+x'=1$ and $xx'=0$).
- Uniqueness. Let y be a complement of x. We will show that $y=x'$. Since y is a complement of x, we have $x+y=1$ and $xy=yx=0$.
  $y=y.1=y.(x+x')=yx+yx'=0+yx'=xx'+yx'=(x+y)x'=1.x'=x' \Rightarrow y=x'$. QED

Corollary 1: $(x')'=x$.
Proof, since $x'+x=1$ and $x'x=0$, it follows that x is a complement of x'. Since the complement of x' is unique, it follows then that (x')' , which is a complement of x', and x, which is also a complement of x', must be equal. Thus, $(x')'=x$. QED

Theorem 3 (De Morgan's Laws):

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

Proof:

a) Show that $x'y'+(x+y)=1$ and $(x'y')(x+y)=0$.
   $x'y'+(x+y)=(x'y'+x)+y=(x'+x)(y'+x)+y=1.(y'+x)+y=(y'+x)+y=(x+y')+y=x+(y'+y)=x+1=1$
   $(x'y')(x+y)=(x'y')x+(x'y')y=(y'x')x+x'(y'y)=y'(x'x)+x'0=y'0+0=0+0=0$
b) The proof is similar and left as an exercise.

QED.

Definition: Let (B,+, . , ', 0,1) be a Boolean Algebra. Define the following $\leq$ relation in B:

$$x \leq y \text{ if } xy=x$$

Theorem 4: The relation $\leq$ is a partial order relation.

Proof: We need to prove that $\leq$ is reflexive, antisymmetric and transitive

- Reflexivity: since $xx=x$ (by Theorem 1-a), it follows that $x \leq x$
- Antisymmetry: need to show that $x\leq y$ and $y\leq x \Rightarrow x=y$. $x\leq y$ and $y\leq x \Rightarrow xy=y$ and $yx=x \Rightarrow$
  x $=xy$ because $x\leq y$

	=yx	because . is commutative
	=y	because $y \leq x$

Therefore, x=y.

- Transitivity: $x \leq y$ and $y \leq z \Rightarrow x \leq z$?

xz	=(xy)z	because xy=x since $x \leq y$
	=x(yz)	because . is associative
	=xy	because yz=y since $y \leq z$
	=x	because xy=x since $x \leq y$

Therefore, xz=x and hence $x \leq z$.

We conclude that $\leq$ I a partial order relation.

Theorem 5 (without proof): If B is a finite Boolean Algebra, then |B| is a power of 2 and the Hasse Diagram of B with respect to $\leq$ is a hypercube.

Definition: A *Boolean variable* x is a variable (placeholder) where the set from which it takes on its values is a Boolean algebra.

Definition: A *Boolean expression* is any string that can be derived from the following rules and no other rules:

a) 0 and 1 are Boolean expressions
b) Any Boolean variable is a Boolean expression
c) If E and F are Boolean expressions, then (E), (E+F), (E.F), and E' are Boolean expressions.

Note that we can omit the parentheses when no ambiguity arises.

Examples:

- x+y, x'+y, x.y, and x.(y+z') are all Boolean expressions
- xyz+x'yz'+xyz'+(x+y)(x'+z) is a Boolean expression
- x/y is not a Boolean expression
- x^y is not a Boolean expression.

Definition: Let B be a Boolean Algebra. A Boolean function of n variables is a function

$$f: B^n \rightarrow B$$

where f(x1,x2,…,xn) is a Boolean expression in x1,x2,…,xn.

Examples: f(x,y,z)=xy+x'z is a 3-variable Boolean function. The function g(x,y,z,w)=(x+y+z')(x'+y'+w)+xyw' is also a Boolean function.

Definition: Two Boolean expressions are said to be equivalent if their corresponding Boolean functions are the same.

Definition: A *literal* is any Boolean variable x or its complement x'.

Truth Tables of Boolean functions:

- Much like the truth tables for logical propositions
- If f(x,y,z, …) is an n-variable Boolean function, a truth table for f is a table of n+1 columns (one column per variable, and one column for f itself), where the rows represent all the 2^n combinations of 0-1 values of the n variables, and the corresponding value of f for each combination.
- Examples:

f(x,y)=xy+x'y';

x	y	f
1	1	1
1	0	0
0	1	0
0	0	1

g(x,y,z) = xy'z'+x'y'z+x'yz';

x	y	z	g
1	1	1	0
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	1
0	0	1	1
0	0	0	1

h(x,y,z,w) = x'y'w'+xyzw+xz';

x	y	z	w	h
1	1	1	1	1
1	1	1	0	0
1	1	0	1	1
1	1	0	0	1
1	0	1	1	0
1	0	1	0	0
1	0	0	1	1
1	0	0	0	1
0	1	1	1	0
0	1	1	0	0
0	1	0	1	0
0	1	0	0	0
0	0	1	1	1
0	0	1	0	0
0	0	0	1	1
0	0	0	0	0

u(x,y,z,w)= 1 if the string xyzw has an odd number of 1's; otherwise, it is 0.

x	y	z	w	u
1	1	1	1	0
1	1	1	0	1
1	1	0	1	1
1	1	0	0	0
1	0	1	1	1
1	0	1	0	0
1	0	0	1	0
1	0	0	0	1
0	1	1	1	1
0	1	1	0	0
0	1	0	1	0
0	1	0	0	1
0	0	1	1	0
0	0	1	0	1
0	0	0	1	1
0	0	0	0	0

Definitions of Minterms and Maxterms:

- Suppose we're dealing with n Boolean variables. A *minterm* is any product of n literals where each of the n variable appears once in the product.
  - Example, where n=3 and the variables are x, y and z:
    - Then, xyz, xy'z, xy'z' are all miterms.
    - xy is not a minterm because z is missing.
    - Also, xyzy' is not a minterm because y appears multiple times (once as y, and another time as y').
  - For n=2 where the variables are x and y, there are 4 minterms in total: xy, xy', x'y, x'y'.
- A maxterm is any sum of n literals where each of the n variable appears once in the sum.
  - Example, where n=3 and the variables are x, y and z:
    - x+y+z, x+y'+z' are both maxterms (of 3 variables).
    - x+y' is not a maxterm because z is missing.

Definition (Disjunctive Normal Form): A Boolean function/expression is in *Disjunctive Normal Form* (DNF), also called *minterm canonical form*, if the function/expression is a sum of minterms.

Examples:

- f(x,y,z)= xyz+xy'z+x'yz'+x'y'z is in DNF
- g(x,y)=xy+x'y' is in DNF
- But h(x,y,z)=xy+x'y'z is not in DNF because xy is not a minterm of size 3.

Definition (Conjunctive Normal Form): A Boolean function/expression is in *Conjunctive Normal Form* (CNF), also called maxterm canonical form, if the function/expression is a product of maxterms.

Examples:

- f(x,y,z)= (x+y+z)(x+y+z')(x'+y+z')(x'+y'+z) is in CNF
- g(x,y)=(x+y)(x'+y') is in CNF
- But h(x,y,z)=(x+y)(x'+y'+z) is not in CNF because x+y is not a maxterm of size 3.

Observation: Thanks to De Morgan's Laws, if f is in DNF, then f' derived from the DNF using De Morgan's Laws (that is, changing every literal to its complement, and every "." to "+", and every "+" to ".") is in CNF, and vice versa.

Method of Putting a Function in DNF, using Truth Tables:

1. Create the truth table of the given Boolean function f

2. For each row where the value of f is 1, create a minterm as follows: put in the position of every variable x in the minterm either x or x’ according to whether the corresponding value in that combination is 1 or 0.For example:
   - For combination 111, the midterm is xyz.
   - For combination 010, the minterm is x’yz’.
3. The DNF of f is the sum of all the minterms created in step 2.

Examples**:**

For the function f(x,y,z) = xy’z’+y’z+xz’;

x	y	z	f
1	1	1	0
1	1	0	1
1	0	1	1
1	0	0	1
0	1	1	0
0	1	0	0
0	0	1	1
0	0	0	0

The minterms of f (where f is 1) are: xyz’, xy’z, xy’z’, xyz’. Therefore, the DNF of f is: xyz’+ xy’z+xy’z’+xyz’.

Method of Putting a Function in CNF, using Truth Tables:

1. Create the truth table of the given Boolean function *f*
2. Add a column for *f’* to the right of the column of *f*, and fill it with the complements of the column of f (that is, wherever f is 1, put 0 under *f’*, and wherever *f* is 0, put 1 under *f’*)
3. Create the DNF of *f’* by applying steps 2 and 3 of the DNF method.
4. Apply De Morgan’s laws on the DNF of *f’*, we get the CNF of *f*.

For example, for the same function f(x,y,z) = xy’z’+y’z+xz’, we take its truth from the previous example, and the we add the column of f’:

x	y	z	f	f’
1	1	1	0	1
1	1	0	1	0
1	0	1	1	0
1	0	0	1	0
0	1	1	0	1
0	1	0	0	1
0	0	1	1	0
0	0	0	0	1

Then, we obtain the DNF of f’: f’=xyz+x’yz+x’yz’+x’y’z’

Finally, applying De Morgan’s, we get the CNF of f: f=(f’)’= (x’+y’+z’)(x+y’+z’)(x+y’+z)(x+y+z).

Optimization of Boolean functions using Karnaugh Maps:

x’y’z + xz + xy’z’

	y		**y’**	
x	1		1	1
x’				1
	z	**z’**		**z**

Minimized form: xz + xy’ + y’z

xz + yz’ + y’z’

	y		y’	
x	1	1	1	1
x’		1	1	
	z	z’		z

Minimized form: x+z’

xyz’ + xy’z’w’ + x’y’zw + x’yw + y’z’w

	y		y’		
x		1	1		w
		1	1		w’
x’					
	1	1	1	1	w
	z	z’		z	

Minimized form: xz’+x’w

x’zw’ + yz’w’ + x’y’z’ + y’z’w’ + x’yz’

	y		y’		
x					w
		1	1		w’
x’	1	1	1	1	
		1	1		w
	z	z’		z	

Minimized form: x’w’+z’w’+x’z’

General procedure for Karnaugh-map-based minimization of Boolean functions:

1. The Karnaugh map is a table of squares (2^n squares when you have n variables)
2. Divide the map into regions so that each variable “owns” half of the squares, and its complement owns the other half. Each square will end up being owned by n literals (making up a minterm).

For example, for 3 variables, the map (of 8 sqaures) is

	y		y'	
x				
x'				
	z	z'		z

The variable x owns the top 4 yellow squares. Its complement x' owns the bottom yellow row of squares. The variable y owns the left half of the squares, and y' owns the right half. The variavle z owns the left and right column of yellow squares, and its complement z'owns the two middle columns of yellow squares.

For 4 variables, the map (of 16 squares) is:

	y		y'		
x					w
					w'
x'					
					w
	z	z'		z	

- The variable x owns the top two rows of yellow squares. Its complement x' owns the bottom two yellow rows of squares.
- The variable y owns the left half of the squares, and y' owns the right half.
- The variavle z owns the left and right column of yellow squares, and its complement z'owns the two middle columns of yellow squares.
- Finally, the variable w owns the top and bottom row of yellow squares, and its complement owns the two middle rows of the yellow squares.

3. Fill the map for a given function f: for each minterm in f, put "1" inside the square corresponding to (or owned by) that minterm.
4. Grouping the filled squares: Group the 1's into rectangles of totally filled squares such that
   - the length and width of each rectangle are powers of 2
   - no filled square remains ungrouped
   - rectangles can overlap
   - each rectangle must exclusively own at least one filled square.
5. Convert each rectangle to a product of literals: For each rectangle, identify the literals where each literal owns the entire rectangle, then multiply those literals.
6. The minimized form is the sum of the products derived in the previous steps.