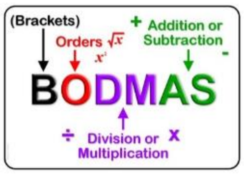
Laws of Boolean Algebra

These are the set of rules or Laws of Boolean Algebra expressions have been invented to help **reduce the number of logic gates needed** to perform a particular logic operation, resulting in a list of functions or theorems.

Operator Precedence, etc.

Just as different arithmetic operators have different properties, so do logical operators.

They have an order of precedence (like BODMAS for arithmetic), and associative, commutative and transitive properties. They are quite simple to understand, though.



The OR operator behaves in the same way as **addition**

you sometimes see *p OR q* written as *p + q*

The AND operator behaves like **multiplication**

you sometimes see *p AND q* written as *p.q*

This even applies when parentheses are used, so

a AND (b OR c) is the same as *(a AND b) OR (a AND c)*

because *a.(b + c)* is the same as *a.b + a.c*

1) $X \cdot 0 = 0$	10A) $X \cdot Y = Y \cdot X$
2) $X \cdot 1 = X$	10B) $X + Y = Y + X$
	Commutative Law
3) $X \cdot X = X$	11A) $X(YZ) = (XY)Z$
4) $X \cdot \overline{X} = 0$	11B) $X + (Y + Z) = (X + Y) + Z$
	Associative Law
5) $X + 0 = X$	12A) $X(Y + Z) = XY + XZ$
6) $X + 1 = 1$	12B) $(X + Y)(W + Z) = XW + XZ + YW + YZ$
	Distributive Law
7) $X + X = X$	13A) $X + \overline{X}Y = X + Y$
8) $X + \overline{X} = 1$	13B) $\overline{X} + XY = \overline{X} + Y$
9) $\overline{\overline{X}} = X$	13C) $X + \overline{X}\,\overline{Y} = X + \overline{Y}$
	13D) $\overline{X} + X\overline{Y} = \overline{X} + \overline{Y}$
	Consensus Theorem
	14A) $\overline{X\,Y} = \overline{X} + \overline{Y}$
	14B) $\overline{X + Y} = \overline{X}\ \overline{Y}$
	DeMorgan's

Figure 1The Laws of Boolean Algebra

Laws of Boolean Algebra Applied Example 1

(A + B)(A + C)

$\downarrow$ Distributing terms

AA + AC + AB + BC

$\downarrow$ Applying identity **AA = A**

A + AC + AB + BC

$\downarrow$ Applying rule **A + AB = A** to the A + AC term

A + AB + BC

$\downarrow$ Applying rule **A + AB = A** to the A + AB term

A + BC

TIP: it might be worthwhile re-writing these examples and applying the Laws to them yourself to get a deeper understanding of each step given here

Laws of Boolean Algebra Applied Example 2

$$\begin{tikzcd}
AB + BC(B + C) \arrow[d, "\text{Distributing terms}"] \\
AB + BBC + BCC \arrow[d, "\text{Applying identity } AA = A \text{ to 2nd and 3rd terms}"] \\
AB + BC + BC \arrow[d, "\text{Applying identity } A + A = A \text{ to 2nd and 3rd terms}"] \\
AB + BC \arrow[d, "\text{Factoring } B \text{ out of terms}"] \\
B(A + C)
\end{tikzcd}$$

All Boolean operators can be combined to create complex Boolean expressions. No matter how complicated they become, and how many operators they use, they will always evaluate to either **True** or **False**.

De Morgan's Laws

A mathematician named DeMorgan developed a pair of important rules regarding group complementation in Boolean algebra.

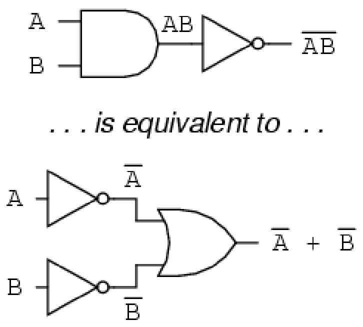
There are circumstances when you might want to ***turn your ANDs into ORs, or vice-versa***.

One situation where you might want to do this is in the construction of electronic logic circuits, where an AND can be represented using transistors. De Morgans laws allow you to do this.

Inverting all inputs to a gate reverses that gate's essential function from AND to OR, or vice versa, and also inverts the output.

- An OR gate with all inputs inverted (a Negative-OR gate) behaves the same as a NAND gate
- an AND gate with all inputs inverted (a Negative-AND gate) behaves the same as a NOR gate

DeMorgan's theorems state the same equivalence in "backward" form: that inverting the output of any gate results in the same function as the opposite type of gate (AND vs. OR) with inverted inputs



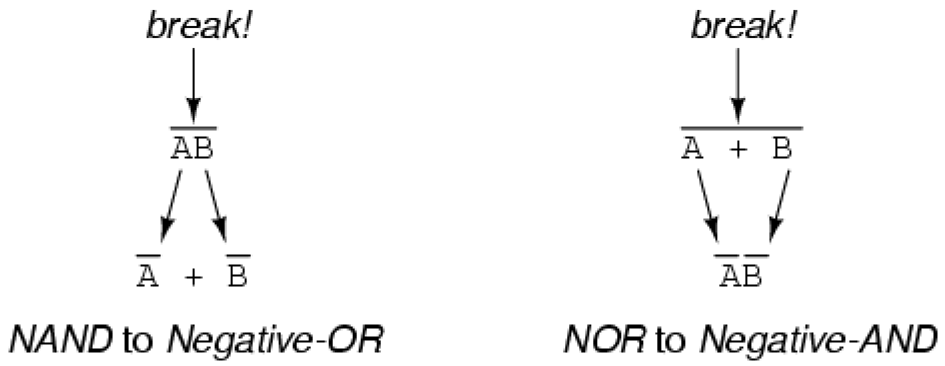
DeMorgan's theorem: *breaking* a long bar symbol

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

DeMorgan's theorem may be thought of in terms of *breaking* a long bar symbol.

When a long bar is broken, the operation directly underneath the break changes from addition to multiplication, or vice versa, and the broken bar pieces remain over the individual variables.

DeMorgan's Theorems

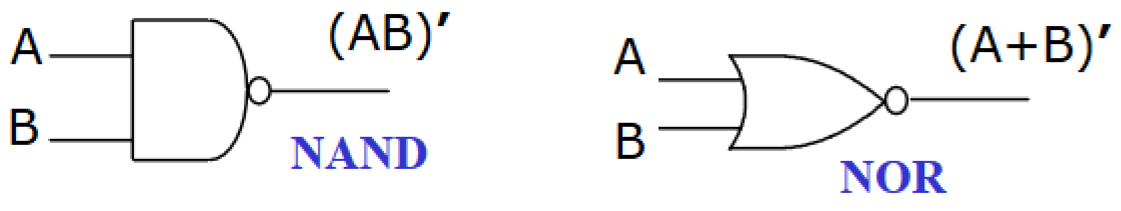


For example:

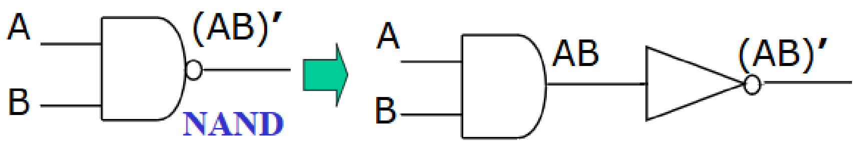
$$\begin{aligned}(AB + C'D)' &= (AB)'\,(C'D)' \\ &= (A'+B')(C + D')\end{aligned}$$

Note that De Morgan's Law was applied twice

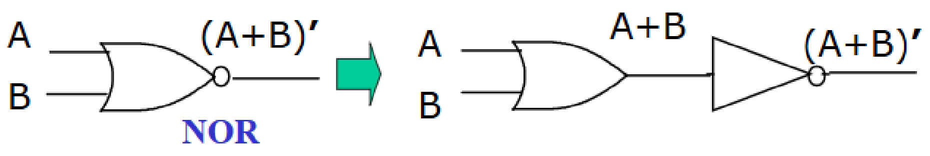
Simply put, a NAND gate is equivalent to a Negative-OR gate, and a NOR gate is equivalent to a Negative-AND gate.



NAND (NOT AND) - can be thought of as an AND gate followed by an inverter.



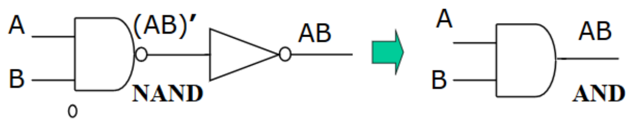
NOR (NOT OR) - can be thought as an OR gate followed by an inverter.

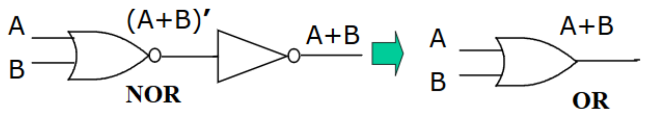


In the real world:

An **AND** gate is made from a **NAND** gate followed by an **inverter**

An **OR** gate is made from a **NOR** gate followed by an **inverter**





Law/Theorem	Law of Addition	Law of Multiplication
Identity Law	$x + 0 = x$	$x \cdot 1 = x$
Complement Law	$x + x' = 1$	$x \cdot x' = 0$
Idempotent Law	$x + x = x$	$x \cdot x = x$
Dominant Law	$x + 1 = 1$	$x \cdot 0 = 0$
Involution Law	$(x')' = x$	
Commutative Law	$x + y = y + x$	$x \cdot y = y \cdot x$
Associative Law	$x+(y+z) = (x+y)+z$	$x \cdot (y \cdot z) = (x \cdot y) \cdot z$
Distributive Law	$x \cdot (y+z) = x \cdot y+x \cdot z$	$x+y\cdot z = (x+y) \cdot (x+z)$
Demorgan's Law	$(x+y)' = x' \cdot y'$	$(x \cdot y)' = x' + y'$
Absorption Law	$x + (x \cdot y) = x$	$x \cdot (x + y) = x$

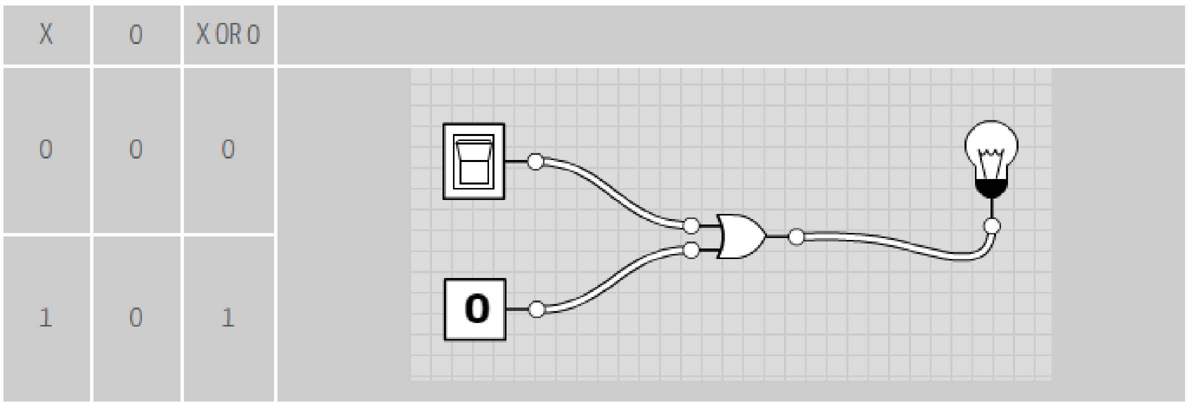
Boolean algebra finds its most practical use in the **simplification of logic circuits**.

If we translate a logic circuit's function into symbolic (Boolean) form, and apply certain algebraic rules to the resulting equation to reduce the number of terms and/or arithmetic operations, the simplified equation may be translated back into circuit form for a logic circuit performing the same function with **fewer components**.

If equivalent function may be achieved with fewer components, the result will be

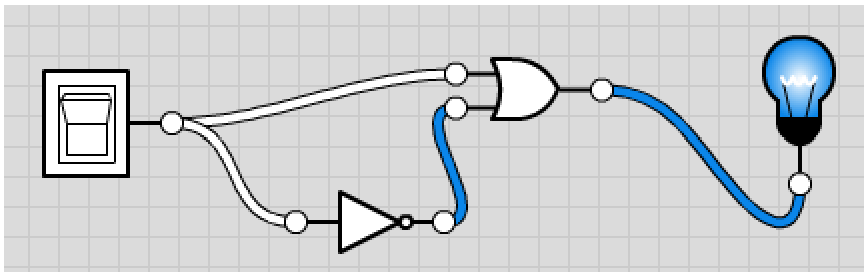
- increased reliability and
- decreased cost of manufacture.

Exercise 1. Here's a worked example of how you might demonstrate the **Identity Law**

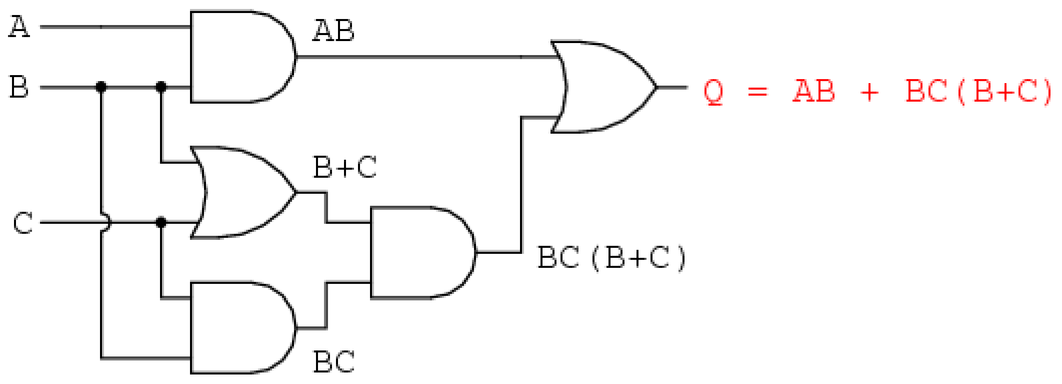


X	0	X OR 0
0	0	0
1	0	1

What does the following circuitry demonstrate?



Simplifying Circuits using Laws of Logic:



Reduce the expression to its simplest form means that the fewest gates will be needed to implement

AB + BC(B + C)

↓ Distributing terms

AB + BBC + BCC

↓ Applying identity **AA = A** to 2nd and 3rd terms

AB + BC + BC

↓ Applying identity **A + A = A** to 2nd and 3rd terms

AB + BC

↓ Factoring **B** out of terms

B(A + C)

The final expression, B(A + C), is much simpler than the original, yet **performs the same function**.

Exercise 2. If you would like to verify this, you may generate a truth table for both

d	M	s	L
0	0	0	0
0	0	1	0
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

expressions and determine Q's status (the circuits' output) for all eight logic-state combinations of A, B, and C, for both circuits. Are the two truth tables identical??

How to derive a truth table and Boolean Expression from simple problem statements

PROBLEM STATEMENT: If it is dark and a motion sensor detects a person on the front porch, or if the switch is on, then turn on the porch light

POSSIBLE SOLUTION: Here, there are 3 INPUT variables:

INPUTS:

1. A sensor that detects dark (d)
2. A sensor that detects motion (m)
3. If the switch is on (**s**)

OUTPUT:

1. A switch to turn on/off the porch light (**L**)

Complete the truth table

Learning Boolean Algebra is like learning to ride a bike, or to juggle. You may read all the reference material you like but you won't understand it until you get in and start doing.