Boolean Algebra

Table of contents

Notation conventions 1
Boolean identities 1
De Morgan's law and gate equivalents 2
Simplifying an expression 3
Sum of products 3

Logic functions can be expressed as expressions of **variables** (literals, e.g. A, B, X) and **functions** (the operators $+, \cdot, \oplus, \bar{}$) — i.e. an output written in terms of its inputs, $X = f(A, B)$. Variables and functions can only take the values 0 or 1. **Boolean algebra** is the set of rules governing how such expressions can be manipulated.

Notation conventions

- **Inversion**: overline, e.g. $\mathrm{NOT}(A) = \bar{A}$ (pronounced "A bar").
- **AND**: dot, or implied by adjacency, e.g. $\mathrm{AND}(A, B) = AB = A \cdot B$.
- **OR**: plus sign, e.g. $\mathrm{OR}(A, B, C) = A + B + C$.
- Other examples: $\mathrm{XOR}(A, B) = A \oplus B = \bar{A}B + A\bar{B}$; $\mathrm{NAND}(A, B, C) = \overline{ABC}$; $\mathrm{NOR}(A, B) = \overline{A + B}$.

$+$ and $\cdot$ are **OR** and **AND**, not arithmetic addition and multiplication — $1 + 1 = 1$ here.

Boolean identities

Name	AND form	OR form
Identity law	$1A = A$	$0 + A = A$
Null law	$0A = 0$	$1 + A = 1$
Idempotent law	$AA = A$	$A + A = A$
Inverse law	$A\bar{A} = 0$	$A + \bar{A} = 1$

Name	AND form	OR form
Commutative law	$AB = BA$	$A + B = B + A$
Associative law	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Distributive law	$A + BC = (A + B)(A + C)$	$A(B + C) = AB + AC$
Absorption law	$A(A + B) = A$	$A + AB = A$
De Morgan's law	$\overline{AB} = \bar{A} + \bar{B}$	$\overline{A + B} = \bar{A}\bar{B}$

Two rows in particular do the heavy lifting when simplifying: the **inverse law** ($A\bar{A} = 0$, $A + \bar{A} = 1$) collapses whole terms to a constant, and the **absorption law** ($A + AB = A$) deletes a term outright.

The **distributive law in its AND form is not the arithmetic one** — $A + BC = (A + B)(A + C)$ has no algebraic analogue and is easy to misremember.

De Morgan's law and gate equivalents

De Morgan's law says AND and OR can be **interchanged if you invert both the inputs and the output**. Drawn as gates, that gives four equivalences — every gate has a second symbol, and pushing an inversion through a gate always flips its shape:

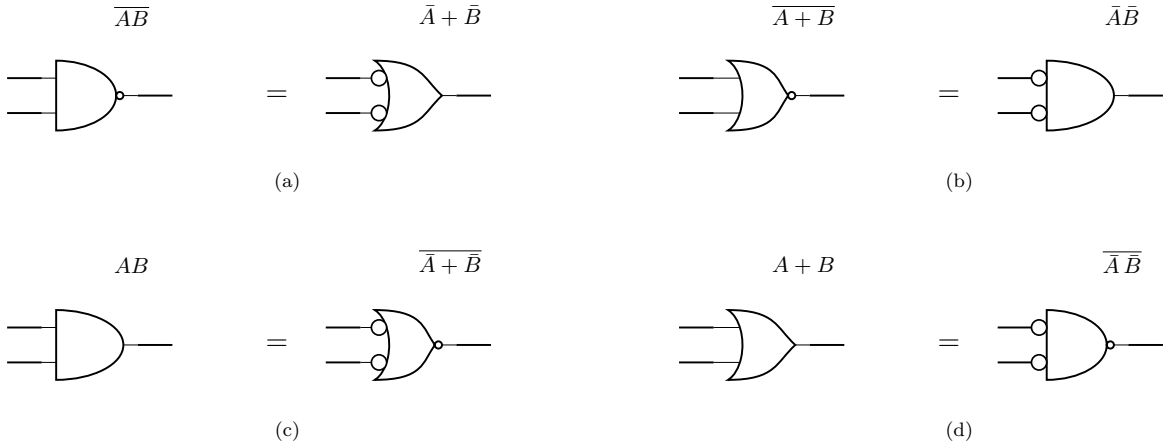

Figure 1: De Morgan's law drawn as gate symbols. (a) a NAND is an OR with inverted inputs; (b) a NOR is an AND with inverted inputs; (c) and (d) are the same two statements read the other way. Each pair is the same function, so either symbol may be used - pick whichever makes the surrounding bubbles cancel.

Exercise: **verify each of the four with a truth table.**

This is also why NAND-only and NOR-only circuits can implement any function (see logic-gates[1]) — an AND feeding an OR can always be redrawn as a NAND feeding a NAND.

Simplifying an expression

Worked example:

$$\text{CODE} = C + CD + \overline{(D + \bar{D})}$$

1. Apply **De Morgan's law** to the last term: $\overline{D + \bar{D}} = \bar{D} \cdot \bar{\bar{D}} = \bar{D} \cdot D = 0$ (the double overline cancels, then the **inverse law** $A\bar{A} = 0$ applies).
2. The **identity law** $0 + A = A$ drops that term: $\text{CODE} = C + CD$.
3. The **absorption law** $A + AB = A$ finishes it: $\text{CODE} = C$.

The whole expression is just C — neither D nor the CD term affects the output at all.

Sum of products

Any logic function can be implemented as the **OR of AND combinations** of its inputs (a *sum of products*):

1. For each row where the truth table output is 1, write the AND term of the inputs (complementing wherever the input is 0) that produces that row.
2. OR all of those terms together.

Because every 1-row gets its own term, and each term is true for exactly that one row, the result is guaranteed to reproduce the table. Worked through in full, with the resulting circuit drawn out, in 2026-07-30-intro-to-logic-gates[2].

Sum of products always works, but does **not** necessarily give the minimum number of gates — the resulting expression can usually be simplified further with the identities above. E.g. $Z = AB + AC = A(B + C)$ (distributive law) needs two gates rather than three.

[1] courses/csse2010/logic-gates.pdf
[2] courses/csse2010/2026-07-30-intro-to-logic-gates.pdf