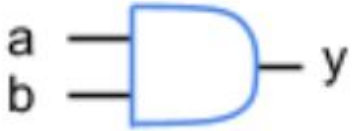
NOT

a	y
0	1
1	0



AND

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

OR

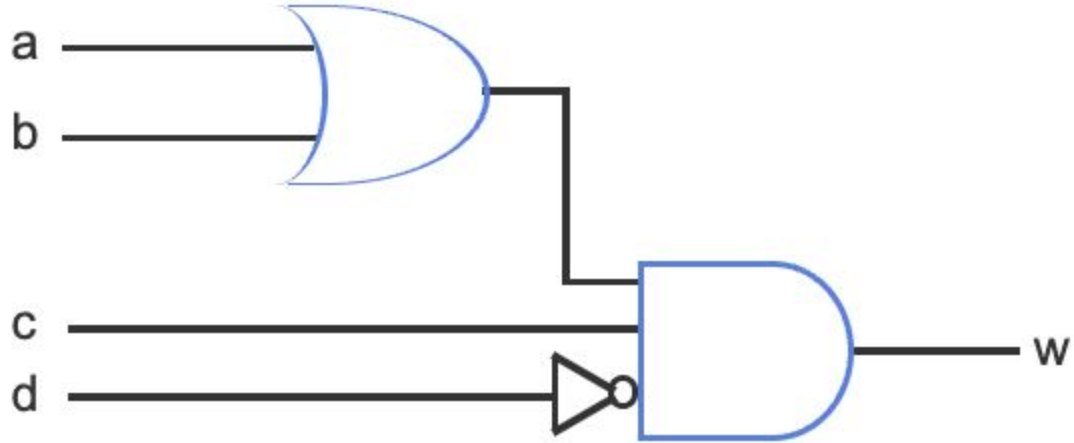
a	b	y
0	0	0
0	1	1
1	0	1
1	1	1

Digital-designer shorthand notation for Boolean operators.

Operation	Shorthand	Notes
a AND b	ab	Intentionally looks like multiplication. Known as *abutment*.
a OR b	a + b	Intentionally looks like addition.
NOT(a)	a'	a' is also called the ***complement*** of a.

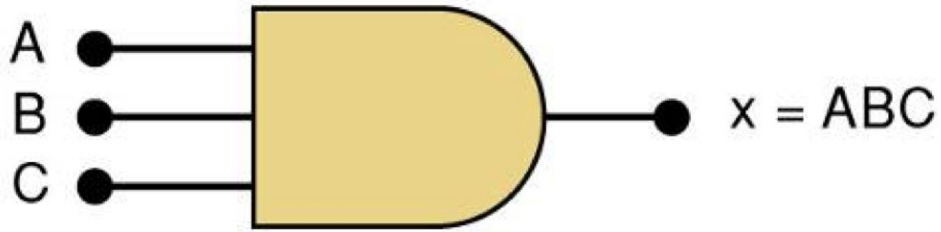
Example: a AND NOT(b) becomes ab'.

w = (a + b)cd'

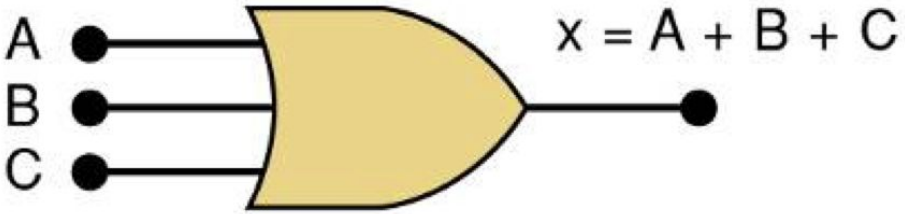


Truth table/circuit symbol for a three input **AND** gate.

A	B	C	x = ABC
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

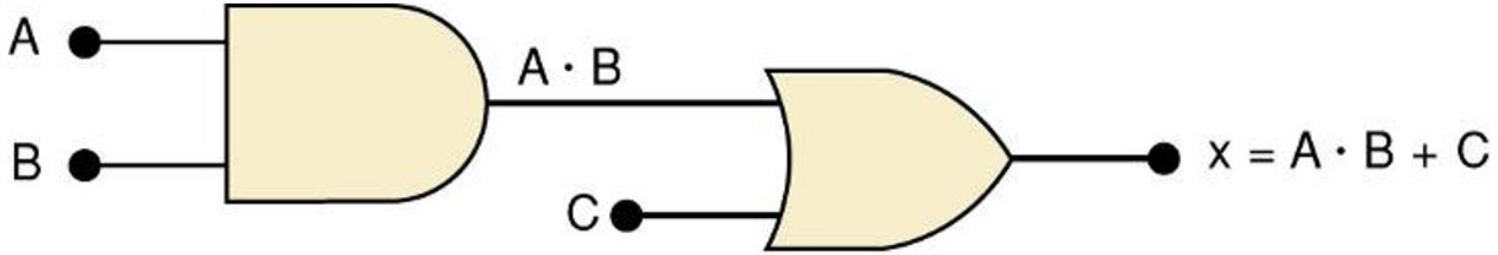


Truth table/circuit symbol for a three input OR gate.

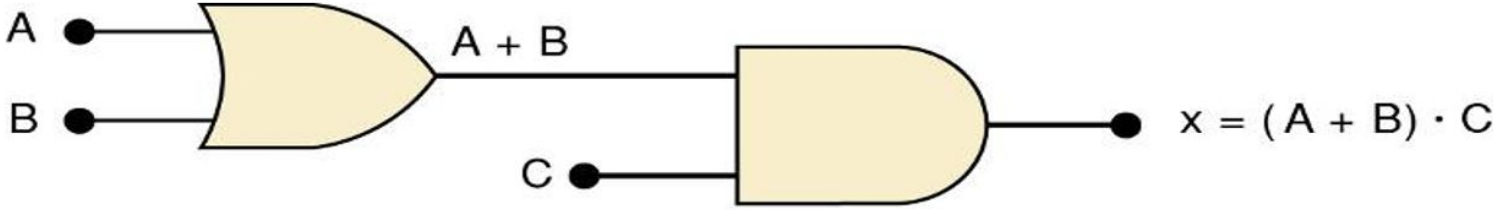


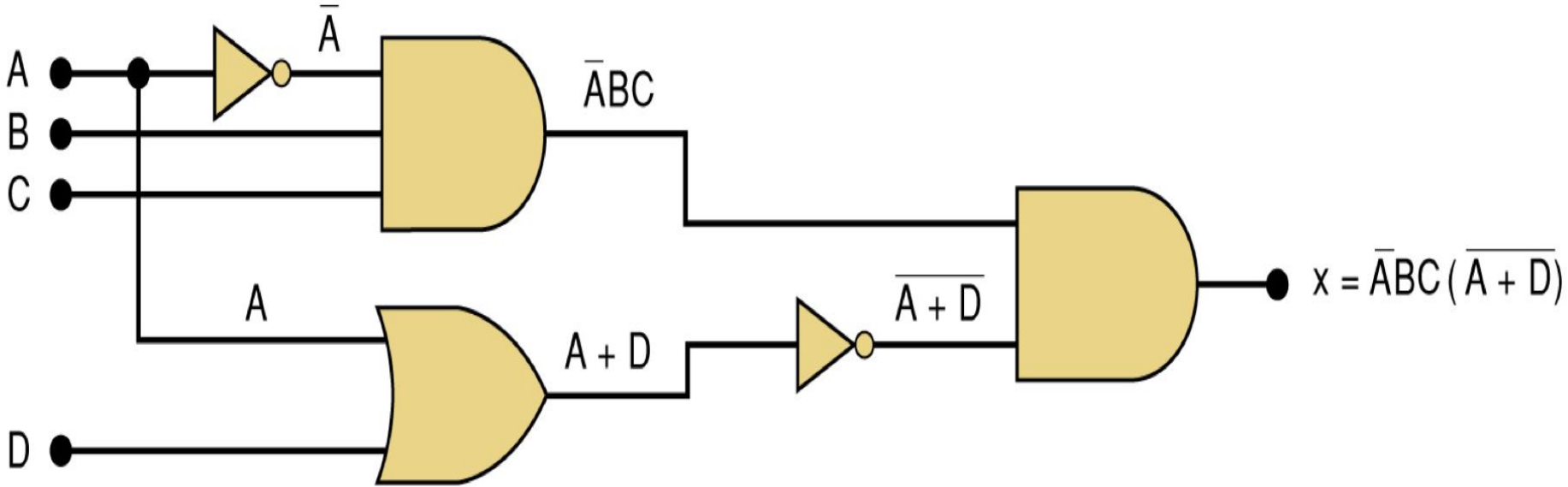
A	B	C	x = A + B + C
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

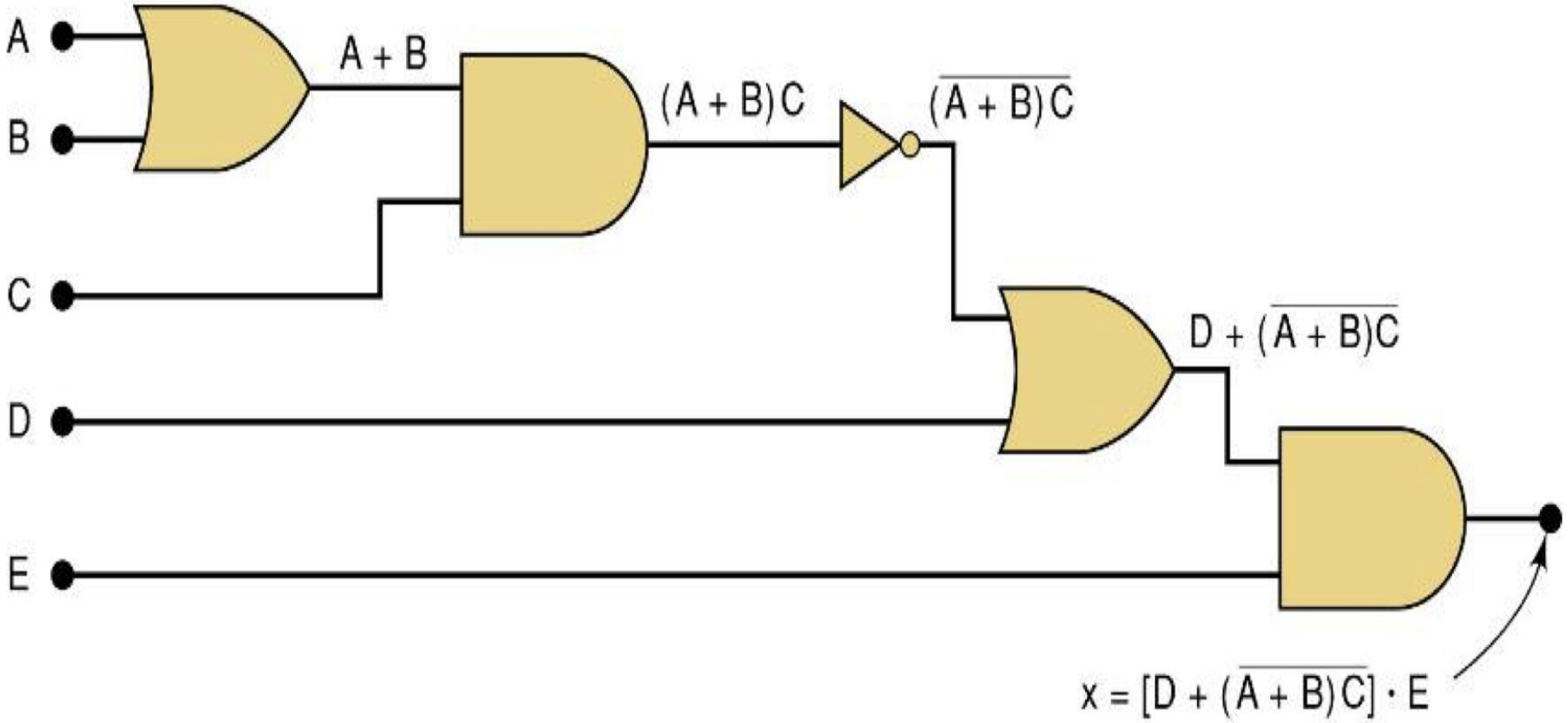
- If an expression contains both AND and OR gates, the AND operation will be performed first.



- Unless there is a parenthesis in the expression.




A
B
C
D
$\bar{A}$
$\bar{A}BC$
A
A + D
$\overline{A + D}$
$x = \bar{A}BC(\overline{A + D})$


A
B
C
D
E
A + B
(A + B)C
$\overline{(A + B)C}$
D + $\overline{(A + B)C}$
x = [D + $\overline{(A + B)C}$] · E

boolean algebra

In 1847, mathematician ***George Boole*** developed an algebra to capture human logic as mathematical equations.

Human logic: If rain is falling and Joe doesn't have an umbrella, Joe will get wet.

r u w

Boolean equation: w = r AND NOT (u)

Boolean equation: w = r AND NOT (u)

Boolean algebra:

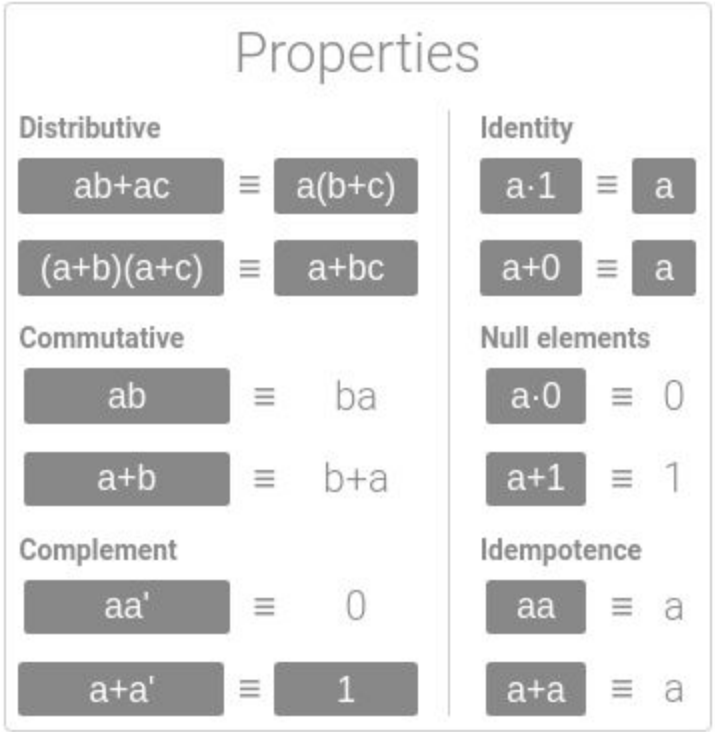
- is an algebra whose only values are true, false.
- the operators are known as **logic operators**: AND, OR, NOT

A Boolean expression is evaluated by evaluating parts and combining. NOT is evaluated first. AND is evaluated before OR.

Given: r = true, u = false

r AND NOT(u)
true AND NOT(false)
true AND true
true

Basic properties of Boolean algebra



Examples:

zy' + zy simplify to z ← Goal

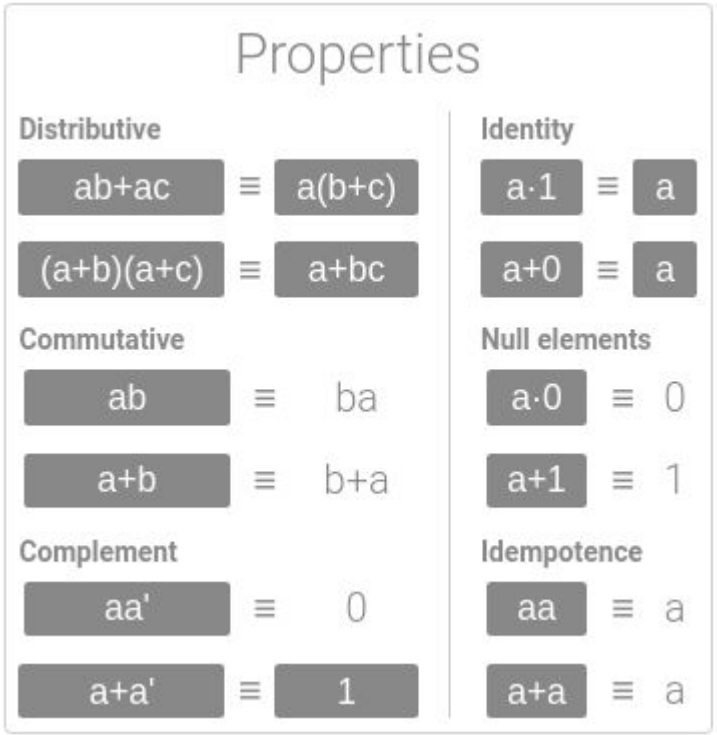
zy' + zy Given

z(y' + y) by distributive

z(y + y') by commutative

z(1) by complement

z ← by identity, answer



(wy' + wy) + wz' simplify to w ← Goal

(wy' + wy) + wz' Given

(w(y' + y)) + wz' by distributive

(w(y + y')) + wz' by commutative

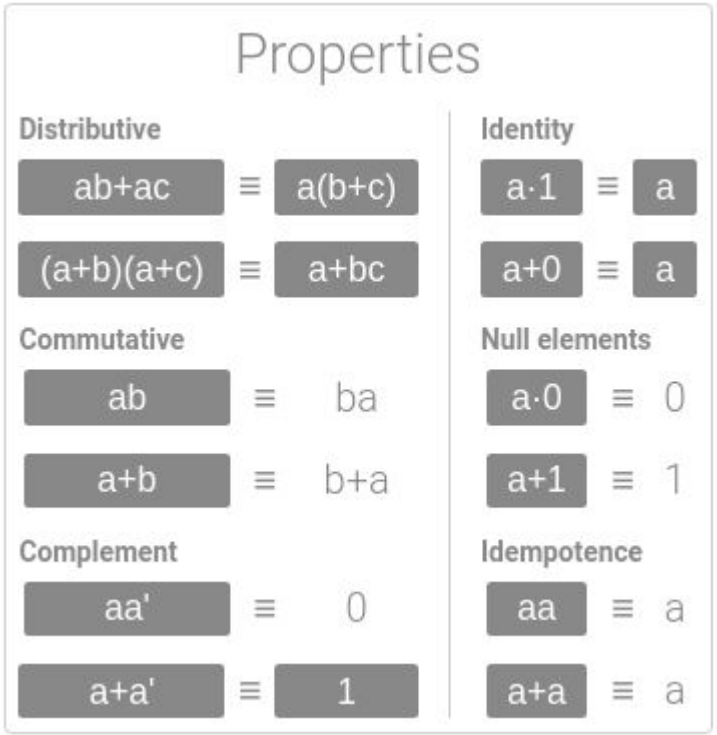
(w(1)) + wz' by complement, w1 + wz'

w(1 + z') by distributive

w(z' + 1) by commutative

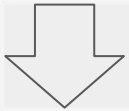
w1 by null element

w ← by identity, answer



Expressions, equations and functions

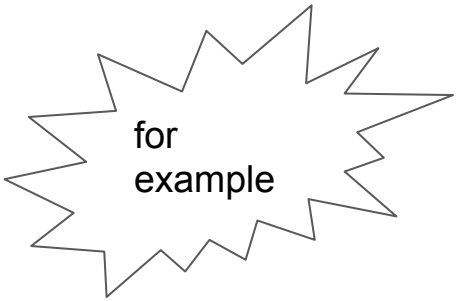
Expressions, equations, and functions.

Item	Notation	Notes
Expression	ab	An expression lacks an equal sign, and involves input variables.
Equation	y = ab	An equation has an =, with expressions of input variables on the right, and an output variable on the left. (In general math, both sides of an equation can be expressions, but in this material, the left side is usually just an output variable.)
Function		A relation of input values to output values. Can be represented in various ways: equation, table, circuit, etc. A function may have more than one input, but has only one output.

In Boolean algebra, a ***function*** is a relation of inputs' values to an output's values. A function can be described in various ways:

- As English: When inputs a, b are both 1's, the output y is 1. Else, y is 0.
- As an equation: y = ab
- As a truth table:

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1



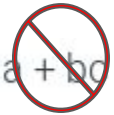
- As a graphical representation: circuit, drawing.

Sum-of-products

A ***product term*** is an ANDing of (one or more) variables, like ab'c.

A product term may involve just one variable.

Valid **product term(s)**: a abc a'b'cd



product term must only involve AND, not OR

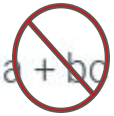
Sum-of-products

Circuits are commonly designed by creating a simplified expression in sum-of-products form, then converting to a simple circuit.

A ***product term*** is an ANDing of (one or more) variables, like ab'c.

A product term may involve just one variable.

Valid **product term(s)**: a abc a'b'cd

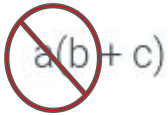


product term must only involve AND, not OR

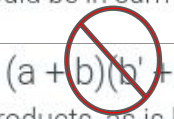
sum-of-products form consists solely of an ORing of product terms, like ab'c + ab.

A single product is still considered a sum of products.

Valid **sum-of-products:** a ab a + c abc' + abc + ab'c



b + c is a sum of products, but then that sum is ANDed with a, so the expression is not just a sum of products. Rewriting as ab + ac would be in sum-of-products form.



a + b is a sum of two products, as is b' + c. But those items are then ANDed, so the entire expression is not in sum-of-products form.

Converting sum-of-products to a circuit

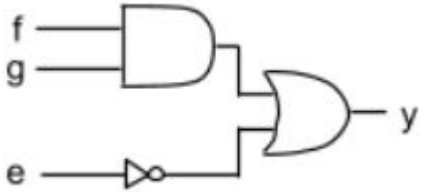
A sum-of-products equation can be easily converted to a circuit

column of AND gates (one gate per product term) followed by an OR gate

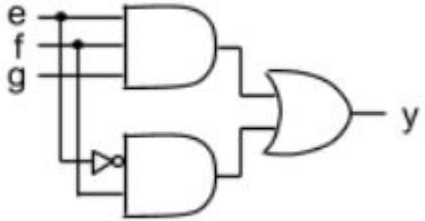
known as a ***two-level circuit***. (The NOT gates preceding the AND gates aren't considered a level)

---

y = f + g



y = fg + e'



y = efg + e'f

Practice

Practice: Convert to sum-of-products form

By convention, to make it easier to see the factors and manage the problem,
we enter terms with variables in alphabetical order.

y = a(b + b'c) ← put it into sum-of-products form

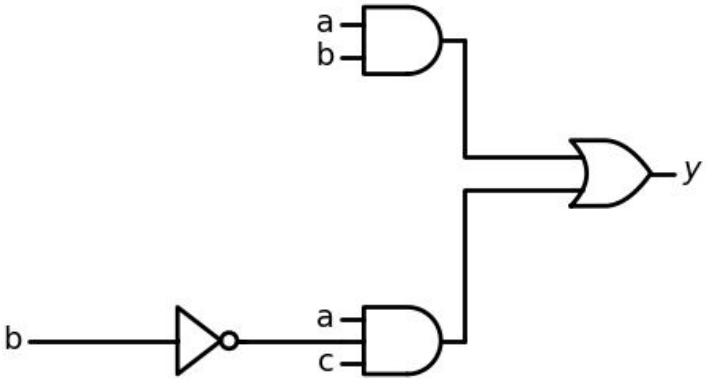
= ab + ab'c

Practice: Convert to a two-level circuit

y = ab + ab'c

How many AND gates?

Do we need an inverter?



Practice: Convert to sum-of-products form

y = c(a + b)

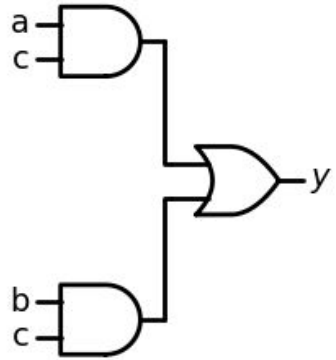
= ac + bc in the sum-of-products form

Practice: Convert to a two-level circuit

y = ac + bc

How many AND gates?

Do we need an inverter?



Practice: Convert to sum-of-products form

y = ab(c + d)

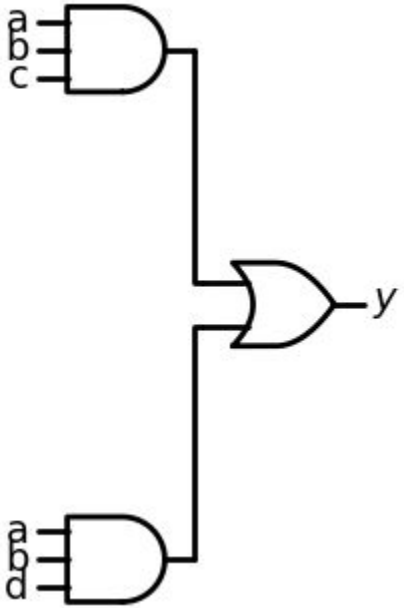
= abc + abd

Practice: Convert to a two-level circuit

y = abc + abd

How many AND gates?

Do we need an inverter?



Practice: Convert to sum-of-products form

y = (a' + b)(c + d)

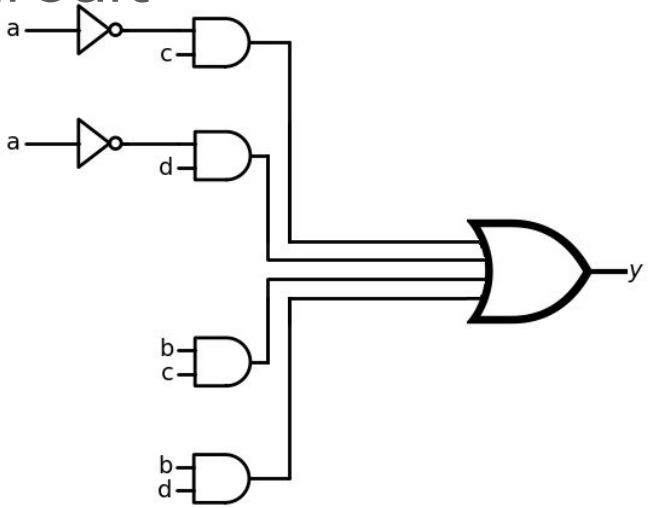
= a’c + a’d + bc + bd

Practice: Convert to a two-level circuit

y = a’c + a’d + bc + bd

How many AND gates?

Do we need an inverter?



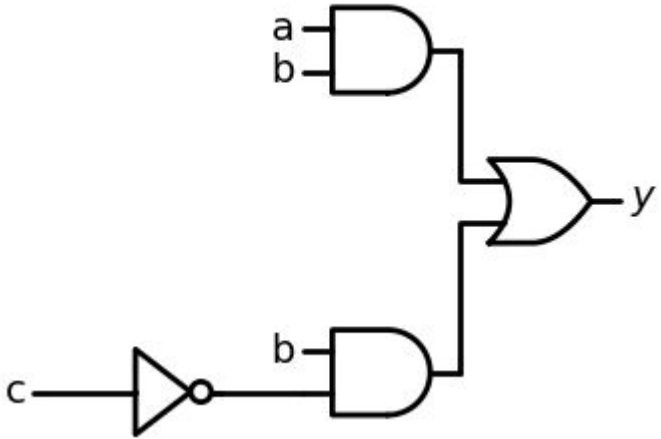
Practice:
Convert from a two-level circuit to sum-of-products

product term(s)

How many AND gates?

How many inputs per AND gates?

Do we have an inverter? Which variable should be written in as complement.



y =

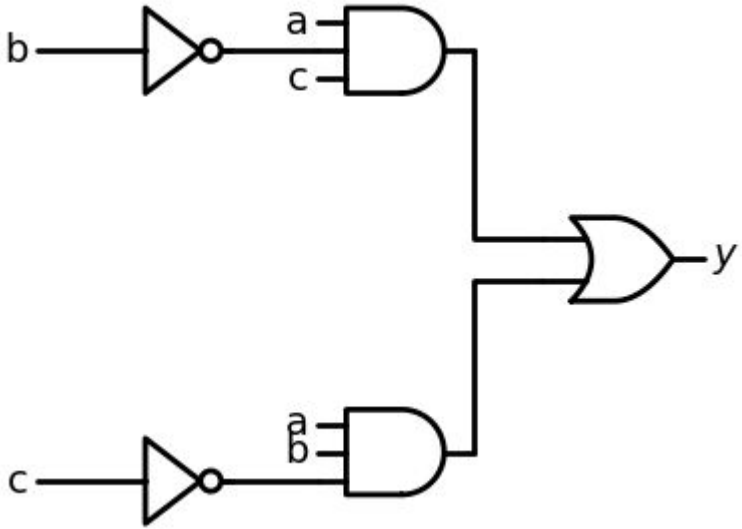
Practice:
Convert <u>from</u> a two-level circuit to sum-of-products

product term(s)

How many AND gates? ?

How many inputs per AND gates? ?

Do we have an inverter? Which variable should be written in as complement. ?



y =