Boolean logic (Boolean algebra)

Developed by George Boole in mid 1800's (i.e. long before binary electronic computers).

Should be considered distinct from the binary number system, but as we will see, it can be intimately connected to binary.

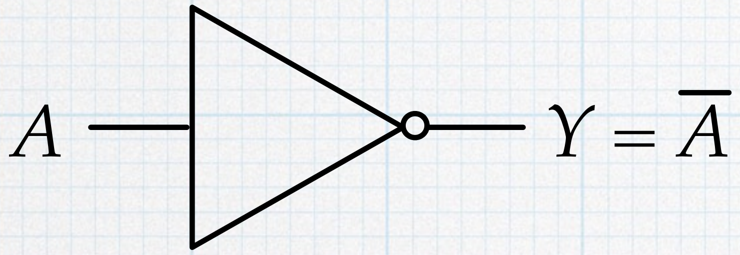
The only values allowed in Boolean logic are true and false (T and F).

The ideas will be developed much more fully in CprE 281, but we will need some of the basic concepts when we write C programs. In our code, we will allow for the program to make decisions based on Boolean algebra.

There are three fundamental operations or (functions or “connectives”). The behavior of each can be described using a *truth table*.

1. NOT (Negation or inversion.)

Input a T or F. The output is the opposite.



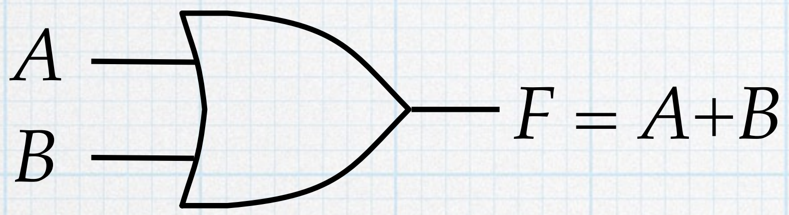
A is a boolean variable. Its value can be either T or F. The output F is the opposite, T or F. The bar over the top indicates the NOT function.

Truth table for NOT

$\boldsymbol{A}$	$\boldsymbol{Y}$
F	T
T	F

2. OR

Take two inputs. Each can be either a T or an F. If *either* of the inputs is T, the output is T. If neither of the inputs is T, the output is F.



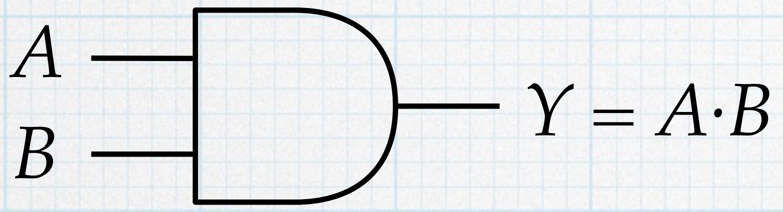
A and B are each boolean variables, and Y is the output. In Boolean algebra, the + symbol is not summation — it denotes the OR function.

Truth table for OR

A	***B***	***Y***
F	F	F
F	T	T
T	F	T
T	T	T

3. AND

Take two inputs. Each can be either a T or an F. If *both* of the inputs are T, the output is T. Otherwise the output is F.



A and B are each boolean variable, and Y is the output. In Boolean algebra, the · symbol is not summation — it denotes the AND function. Can also write it as $Y = AB$.

Truth table for AND

A	B	Y
F	F	F
F	T	F
T	F	F
T	T	T

Connection to binary

The connection to binary number is obvious. Rather than the abstract Boolean variables with value of T or F, we can realize the exact same thing using binary bits, with values of 0 or 1. Usually, we associate 0 with F and 1 with T.

NOT

$Y = \overline{A}$

A	Y
0	1
1	0

OR

$Y = A + B$

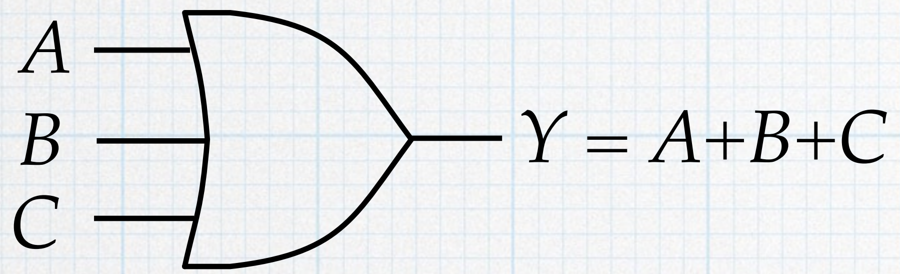
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

AND

$Y = A \cdot B$

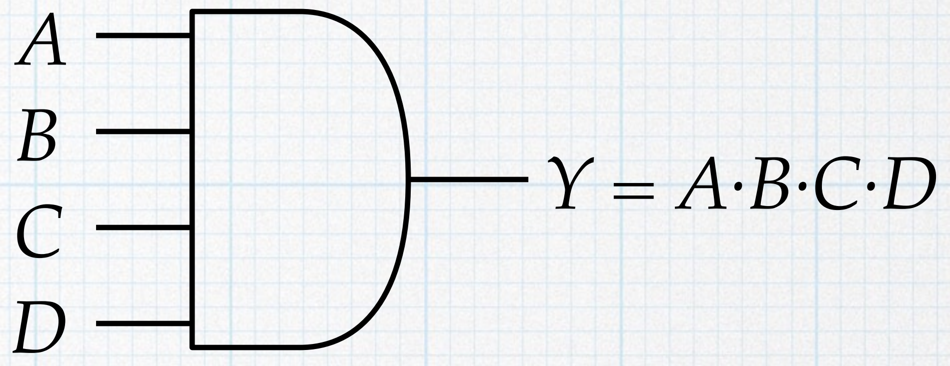
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Boolean operations are not limited to 2 inputs



$Y = A + B + C$

A	B	C	Y
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



$Y = A \cdot B \cdot C \cdot D$

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Lots of algebra rules

Many of them are similar to “regular” algebra. All can be proven using truth tables.

Given Boolean variable, *A*, *B*, and *C*:

$0 + A = A$ $\qquad 0\cdot A = 0$ $\qquad \overline{(\overline{A})} = A$

$1 + A = 1$ $\qquad 1\cdot A = A$

$A + B = B + A$ $\qquad A\cdot B = B\cdot A$

$(A + B) + C = A + (B + C)$ $\qquad (A\cdot B)\cdot C = A\cdot(B\cdot C)$

$A + B\cdot C = (A + B)(A + C)$ $\qquad A(B + C) = A\cdot B + A\cdot C$

$A + A = A$ $\qquad A\cdot A = A$

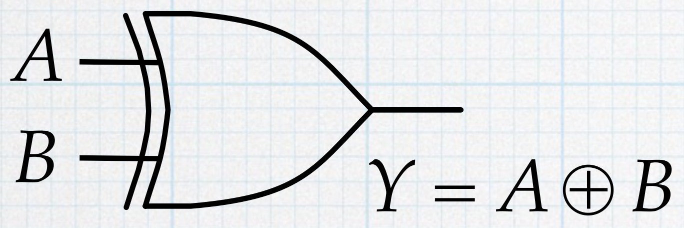
$A + A\cdot B = A$ $\qquad A\cdot(A + B) = A$ $\qquad A + A\cdot B = A + B$

$A + \overline{A} = 1$ $\qquad A\cdot\overline{A} = 0$

$\overline{(A + B)} = \overline{A}\cdot\overline{B}$ $\qquad \overline{(A\cdot B)} = \overline{A} + \overline{B}$

Three other relations

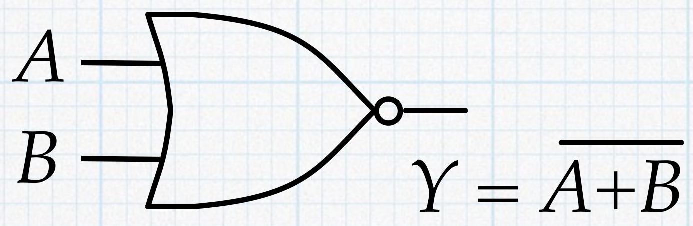
Exclusive - OR



$Y = A \oplus B$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

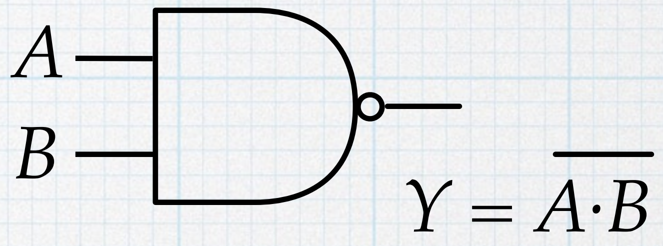
NOR



$Y = A + B$

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

NAND



$Y = A \cdot B$

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0