2.1 Boolean Algebra

Boolean algebra (for our purposes) operates on variables that can take on 1 of 2 values.

Ex: \( X = \text{ON or OFF} \)
\[ = \text{TRUE or FALSE} \]
\[ = \text{full or empty} \]
\[ = 1 \text{ or } 0 \]
\[ = +5 \text{V or } 0 \text{V} \]

It’s arbitrary how we assign values as long as we’re consistent.

\[ \begin{array}{c}
\text{"fullness" } x=1 \\
\text{"emptiness" } x=0
\end{array} \]

2.2 Basic Operations

Invert/NOT/Complement Operator

Denote inversion/complementation by \( \overline{X} \) or \( X' \) or \( \neg X \)

If \( X = \text{ON} \), then \( \overline{X} = \text{OFF} \)
If \( Y = 0 \), then \( Y' = 1 \)

Symbol: \( \overline{X} \quad \text{Inverter} \)

Sometimes denoted as just a “bubble”

\[ f \quad B = f(A) \]

If we want to invert \( B \) or \( A \), add an inverter or bubble to the output or input, respectively.

\[ f \quad \overline{B} = B' \]

\[ f \quad C = f(A) \]

Boolean variables have a (very!) limited number of possible values so we can enumerate all possibilities in a truth table.

<table>
<thead>
<tr>
<th>( X )</th>
<th>( X' )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Note: \( N \) input Boolean variables have \( 2^N \) possible input combinations, so we can build truth tables only for a small number of inputs.
AND Operator
AND must have two or more inputs:
X AND Y AND Z or X•Y•Z or XYZ or "Product of X, Y, Z"

Def: Output is 1 or TRUE only when all inputs are 1 or TRUE.

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>X•Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>X•Y•Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Symbol: \( X \quad \bullet \quad Y \quad \rightarrow \quad X \cdot Y \)

Ex: Switches in series

OR Operator
Also has two or more inputs: X OR Y OR Z or X+Y+Z or "Sum of X, Y, Z"

Def: Output is 1 or TRUE if any input is 1 or TRUE.

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>X+Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>X+Y+Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Symbol: \( X \quad + \quad Y \quad \rightarrow \quad X+Y \)

Ex: Switches in parallel
NAND (NOT AND) Operator

\[ X \bar{Y} = (X \cdot Y)' \equiv X \bar{Y} = (X \cdot Y)' \]

NAND is easier to build (much more common) in hardware.

NOR (NOT OR) Operator

\[ X \bar{Y} = (X + Y)' \equiv X \bar{Y} = (X + Y)' \]

XOR (eXclusive OR) Operator

XOR has two or more inputs: \( X \) XOR \( Y \) or \( X \oplus Y \)

Def: Output is 1 or TRUE if one or the other inputs (but not both) are 1 or TRUE, i.e. \( X=1 \) or \( Y=1 \), exclusively.

\[ \begin{array}{c|c|c}
X & Y & X \oplus Y \\
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
\end{array} \]

Symbol:

Ex: 

Useful in addition: \( \text{Sum} = X \) PLUS \( Y \equiv \text{Sum} = X \oplus Y \)

XNOR Operator

Complement of XOR Symbol: \( \bar{X} \oplus Y \equiv \bar{X} \oplus Y \)

Def: Output is 1 or TRUE if both inputs are the same.

\[ \begin{array}{c|c|c}
X & Y & \bar{X} \oplus Y \\
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 0 \\
1 & 1 & 1 \\
\end{array} \]

XNOR is sometimes called the "equivalence" operator.
More complex expressions are built out of these fundamental blocks:

\[ W \\
X \\
Y \\
\downarrow \\
WXY \\
\downarrow \\
WX + Y \\
\downarrow \\
Z \\
\frac{(WX + Y) \cdot Z}{'} \]

**Literal**  
**Def:** A variable or its complement in an expression.  
**Ex:** \[
(WXY + Y) \cdot Z \]
has 5 literals (\(W, X, Y\) twice, and \(Z\)).

### 2.4 Basic Theorems

**Operations with 0 or 1**

\[
\begin{align*}
X + 0 &= X \\
X + 1 &= 1 \quad \text{(set)} \\
X \cdot 0 &= 0 \quad \text{(clear)} \\
X \cdot 1 &= X
\end{align*}
\]

We frequently need to set or clear part of a larger binary word (number).

**Ex:** Set bit 1  
\[
\begin{array}{c}
XXX \\
or \quad 0 \quad 0 \quad 1 \\
XXX \quad 1
\end{array}
\]

**Ex:** Clear bits 2 and 3  
\[
\begin{array}{c}
YYY \\
\text{AND} \quad 0 \quad 0 \quad 1 \\
0 \quad 0 \quad Y
\end{array}
\]

**Idempotent Laws**

**OR:**  
\[ X + X = X \]  
Verify using truth table:

\[
\begin{array}{c|c|c}
X & Y & X + Y \\
1 & 1 & 1
\end{array}
\]

Symbolically,  
\[ X \quad \downarrow \\
\quad \Rightarrow \quad X = X \quad \text{no bubble} \]

**AND:**  
\[ X \cdot X = X \]  
Verify using truth table:

\[
\begin{array}{c|c|c}
X & Y & X \cdot Y \\
1 & 1 & 1
\end{array}
\]

Symbolically,  
\[ X \quad \downarrow \\
\quad \Rightarrow \quad X = X \]
Involution Law "Double flip" \[ (X')' = \bar{X} = X \]

Laws of Complementarity
\[ X + X' = 1 \quad \text{(Either } X \text{ or } X' \text{ is always 1)} \]
\[ X \cdot X' = 0 \quad \text{(Either } X \text{ or } X' \text{ is always 0)} \]

Truth table:
\[
\begin{array}{c|c|c|c}
X & X' & X + X' & X \cdot X' \\
0 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 \\
\end{array}
\]

Ex: \[(x + y) \cdot (x + y) = x + y \quad (x' + y \cdot z') \cdot (x' + y \cdot z')' = 0 \]

Commutative Law AND: \[ ABC = CBA = BAC \]
OR: \[ A + B + C = C + B + A = C + A + B \]

Symbolically,

\[
\begin{array}{c|c|c}
A & B & C \\
\hline
\text{AND} & \equiv & \text{OR} \\
\end{array}
\]

Associative Law AND: \[(AB)C = A(BC) = ABC \]
OR: \[(A + B) + C = A + (B + C) = A + B + C \]

Symbolically,

\[
\begin{array}{c|c|c}
A & B & C \\
\hline
\text{AND} & \equiv & \text{OR} \\
\end{array}
\]

Distributive Law
a) \[ A(B + C) = AB + AC \]
b) \[ A + BC = (A + B)(A + C) \]

Helpful Theorems (Roth, p. 41)
Ex: \[ X + XY = X \]
Help simplify complicated expressions:
\[
DC(KL' + M)(G + H) + C = C
\]

Looks like multiplication and addition, but meaning is very different!