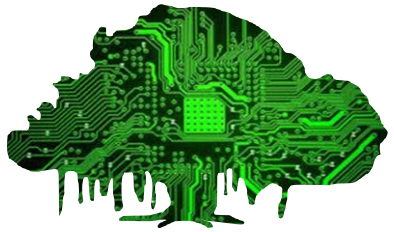


Lecture 3 – Binary Logic

Dr. Aftab M. Hussain,

Assistant Professor, PATRIoT Lab, CVEST

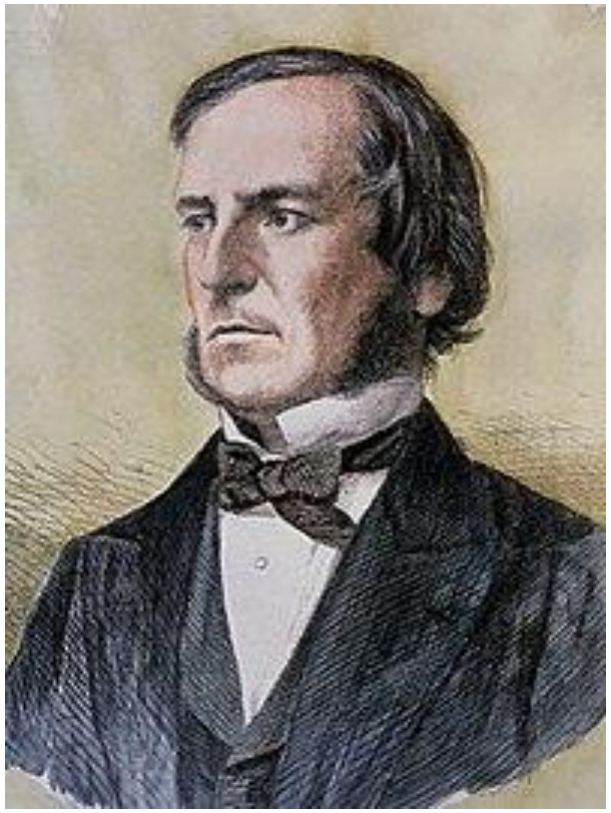
Chapter 2

Binary logic

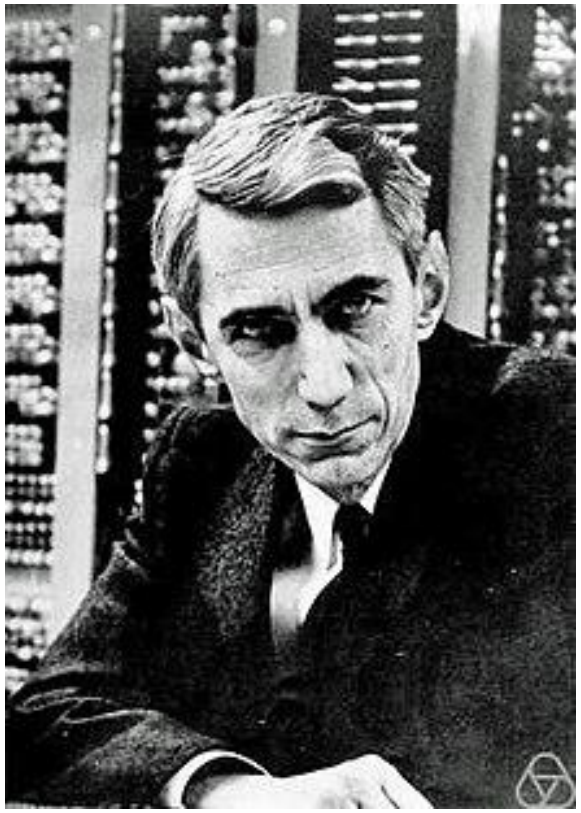
- Binary logic deals with variables that take on two discrete values and with operations that assume logical meaning
- The two values the variables assume may be called by different names (*true* and *false, yes* and *no*, etc.), but for our purpose, it is convenient to think in terms of bits and assign the values 1 and 0
- Binary logic consists of binary variables and a set of logical operations
- The variables are designated by letters of the alphabet, such as *A*, *B*, *C*, *x*, *y*, *z*, etc., with each variable having two and only two distinct possible values: 1 and 0

Boolean algebra

- The system for formalization of binary logic came much before their applications in electronics/computers
- Boolean algebra was introduced by George Boole in his first book The Mathematical Analysis of Logic (1847)
- In the 1930s, while studying switching circuits, Claude Shannon observed that one could also apply the rules of Boole's algebra in this setting, and he introduced switching algebra as a way to analyze and design circuits by algebraic means in terms of logic gates.



George Boole



Claude Shannon

Basic operations

- **NOT:** This operation is represented by a prime (sometimes by an overbar). For example, $x' = z$ (or $\bar{x} = z$); meaning that *z* is what *x* is not
- In other words, if *x* = 1, then *z* = 0, but if *x* = 0, then *z* = 1
- The NOT operation is also referred to as the complement operation, since it changes a 1 to 0 and a 0 to 1, i.e., the result of complementing 1 is 0, and vice versa

- **AND:** This operation is represented by a dot or by the absence of an operator
- For example, $x \cdot y = z$ or $xy = z$
- The logical operation AND is interpreted to mean that *z* = 1 if and only if *x* = 1 and *y* = 1; otherwise *z* = 0

- **OR:** This operation is represented by a plus sign. For example, $x + y = z$, meaning that *z* = 1 if *x* = 1 or if *y* = 1 or if both *x* = 1 and *y* = 1. If both *x* = 0 and *y* = 0, then *z* = 0

Basic operations

- An easy way to remember this is to make a table of all possible values of the variables and the results of these operations

AND

x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

OR

x	y	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

NOT

x	x'
0	1
1	0

Binary logic

- Binary logic is different from binary numbers although it uses some of the same symbols
- In binary logic, we assume that variables can have ONLY two values – no other values are possible
- In binary numbers variables can have higher values or fraction or negative values, however, that is not the case in binary logic
- For example: in binary numbers, $(1+1 = 10)_2$, however, in binary logic, 1+1 = 1 because two trues make a true
- There are formal rules and proofs for many of the statements we make in binary logic
- In modern circuits, logic gates are used to perform binary logic using a variety of complex architectures

Formalization of Boolean algebra

- Boolean algebra, like any other deductive mathematical system, may be defined with a set of elements, a set of operators, and a number of unproved axioms or postulates
- A *binary operator* defined on a set *S* of elements is a rule that assigns, to each pair of elements from *S*, a unique element from *S*
- As an example, consider the relation $a * b = c$. We say that * is a binary operator if it specifies a rule for finding *c* from the pair (*a, b*) and also if $a, b, c \in S$
- However, * is not a binary operator if $a, b \in S$, and if $c \notin S$.

Postulates of Boolean algebra – Closure

- A set *S* is closed with respect to an operator if, for every pair of elements of *S*, the operator specifies a rule for obtaining an element of *S*
- For example, the set of natural numbers $N = \{1, 2, 3, 4, c\}$ is closed with respect to the operator **+** by the rules of arithmetic addition, since, for any $a, b \in N$, there is a unique $c \in N$ such that $a + b = c$
- The set of natural numbers is *not* closed with respect to the operator - by the rules of arithmetic subtraction, because $2 - 3 = -1$ and $2, 3 \in N$, but $(-1) \notin N$
- The Boolean logic structure is closed with respect to NOT, AND and OR logic operations

Postulates of Boolean algebra – Associative law

- The operator * on a set *S* is said to be associative whenever $(x * y) * z = x * (y * z)$ for all $x, y, z, \in S$
- In case of real numbers, the multiplication and addition operations are associative while subtraction and division are not
- Similarly in binary logic, the operators AND and OR are associative
- Thus, x AND (y AND z) is the same as (x AND y) AND z
- Also, x OR (y OR z) is the same as (x OR y) OR z

Postulates of Boolean algebra – Commutative law

- The operator * on a set *S* is said to be commutative whenever $x * y = y * x$ for all $x, y \in S$
- In case of real numbers, the multiplication and addition operations are commutative, while subtraction and division are not
- Similarly in binary logic, the operators AND and OR are commutative
- Thus, x AND y is the same as y AND x
- Also, x OR y is the same as y OR x

Postulates of Boolean algebra – Identity

- A set *S* is said to have an identity element with respect to an operation * on *S* if there exists an element $e \in S$ with the property that $e * x = x * e = x$ for every $x \in S$
- *Example:* The element 0 is an identity element with respect to the operator + on the set of integers $I = \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$, since $x + 0 = 0 + x = x$ for any $x \in I$
- The set of natural numbers, *N*, has no identity element, since 0 is excluded from the set
- In Boolean logic, 0 is the identity element for OR operation and 1 is the identity element for AND operation

Postulates of Boolean algebra – Distributive

- If * and # are two operators on a set *S*, * is said to be distributive over # whenever $x * (y \# z) = (x * y) \# (x * z)$
- In normal algebra, multiplication is distributive over addition: x(y+z) = xy + xz
- In Boolean logic, The operator AND (·) is distributive over OR (+); that is, $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$
- Also, the operator OR (+) is distributive over AND (·); that is, $x + (y \cdot z) = (x + y) \cdot (x + z)$
- This is counter intuitive!
- An easy way to prove the distributive law is the make a table of all possible values of the variables and their results

Postulates of Boolean algebra – Distributive

$$x\,.(y+\ z) = \ (x.y) + \ (x\,.z)$$

x	y	z
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Theorems of Boolean algebra – Duality principle

- The *duality principle* and states that every algebraic expression deducible from the postulates of Boolean algebra remains valid if the operators and identity elements are interchanged, i.e., AND is changed to OR and vice versa
- In a two-valued Boolean algebra, the identity elements and the elements of the set *S* are the same: 1 and 0
- Thus, if the *dual* of an algebraic expression is desired, we simply interchange OR and AND operators and replace 1's by 0's and 0's by 1's

(a)	$x + 0 = x$	(b)	$x \cdot 1 = x$
(a)	$x + x' = 1$	(b)	$x \cdot x' = 0$
(a)	$x + x = x$	(b)	$x \cdot x = x$
(a)	$x + 1 = 1$	(b)	$x \cdot 0 = 0$

Theorems of Boolean algebra

- Theorem: $x + x = x$

$$
\begin{aligned}
x + x &= (x + x) \cdot 1 \\
&= (x + x) \cdot (x + x') \\
&= x + x \cdot x' \\
&= x + 0 \\
&= x
\end{aligned}
$$

- Its dual: $x \cdot x = x$

Theorems of Boolean algebra

- Theorem: $x + 1 = 1$

$$
\begin{aligned}
x + 1 &= 1 \cdot (x + 1) \\
&= (x + x') \cdot (x + 1) \\
&= x + x' \cdot 1 \\
&= x + x' \\
&= 1
\end{aligned}
$$

- Its dual: $x \cdot 0 = 0$

Theorems of Boolean algebra

- Theorem: $x + xy = x$ (absorption theorem)

$$\begin{aligned} x + xy &= x \cdot 1 + xy) \\ &= x \cdot (1 + y) \\ &= x \cdot 1 \\ &= x \end{aligned}$$

- Its dual: $x \cdot (x + y) = x$

Theorems of Boolean algebra

- Theorem: $(x + y)' = x' \cdot y'$(first DeMorgan's theorem)
- A simple way to prove the theorem is to prove that (x+y) and x'y' are complements of each other so that (x+y)' is the same as x'y'
- So, we need to prove that $(x + y) + x'y' = 1$ and $(x + y) \cdot x'y' = 0$. If they are both true, then because of uniqueness of complements, $(x + y)' = x'y'$

$$
\begin{aligned}
(x + y) + x'y' &= x + y + x'y' \\
&= x + (y + x')(y + y') \\
&= x + (y + x') \cdot 1 \\
&= x + y + x' \\
&= y + (x + x') \\
&= y + 1 \\
&= 1
\end{aligned}
$$

Theorems of Boolean algebra

- Theorem: $(x + y)' = x' \cdot y'$(first DeMorgan's theorem)
- A simple way to prove the theorem is to prove that (x+y) and x'y' are complements of each other so that (x+y)' is the same as x'y'
- So, we need to prove that $(x + y) + x'y' = 1$ and $(x + y) \cdot x'y' = 0$. If they are both true, then because of uniqueness of complements, $(x + y)' = x'y'$

$$
\begin{aligned}
(x + y) \cdot x'y' &= x \cdot x' \cdot y' + y \cdot x' \cdot y' \\
&= (x \cdot x') \cdot y' + x' \cdot (y \cdot y') \\
&= 0 \cdot y' + x' \cdot 0 \\
&= 0 + 0 \\
&= 0
\end{aligned}
$$

- Because both are true together, the theorem is proved

Operator precedence

- The operator precedence for evaluating Boolean expressions is (1) parentheses, (2) NOT, (3) AND, and (4) OR. This is similar to BODMAS
- In other words, expressions inside parentheses must be evaluated before all other operations
- The next operation that holds precedence is the complement, and then follows the AND and, finally, the OR
- As an example, consider one of DeMorgan's theorems
- The left side of the expression is $(x + y)'$. Therefore, the expression inside the parentheses is evaluated first and the result then complemented
- The right side of the expression is $x'y'$, so the complement of x and the complement of y are both evaluated first and the result is then ANDed