Discrete Structures 2 [1]

A Study Guide for Students of Sorsogon State University - Bulan Campus[2]

Jarrian Vince G. Gojar[3]

January 19, 2025

[1]A course in the Bachelor of Science in Computer Science

[2]This book is a study guide for students of Sorsogon State University - Bulan Campus taking up the course Discrete Structures 2.

[3]https://github.com/godkingjay

Sorsogon State University - Bulan Campus

Contents

List of Figures

List of Tables

Preface

"If people do not believe that mathematics is simple, it is only because they do not realize how complicated life is."

– John von Neumann

Jarrian Vince G. Gojar
https://github.com/godkingjay

1

Boolean Algebra

1.1 Introduction

Circuits in computers are made up of millions of tiny switches that can be in one of two states: on or off. These switches are controlled by electrical signals that represent logical values. The behavior of these switches can be described using a mathematical system called Boolean Algebra. **Boolean Algebra** is a branch of mathematics that deals with logical values and operations on these values. It is widely used in computer science and engineering to design and analyze digital circuits. Computers uses the binary number system, which has only two digits: 0 and 1 which means “low voltage” and “high volt” respectively. These digits correspond to the logical values FALSE and TRUE, respectively.

1.2 History of Boolean Algebra

George Boole was an English mathematician and logician who lived in the 19th century. He was born in 1815 and died in 1864. Boole is best known for his work in the field of logic, which laid the foundation for modern computer science.

Boole’s most famous work is his book *The Laws of Thought*, which was published in 1854. In this book, Boole introduced the concept of Boolean Algebra, which is a mathematical system for dealing with logical values. Boolean Algebra is based on the idea that logical values can be represented as either TRUE or FALSE.

In 1938, **Claude Shannon** showed that the two-valued Boolean Algebra could be used to describe the operation of electrical switches. This discovery laid the foundation for the design of digital circuits and computers.

1.3 Fundamental Operations

The three fundamental operations of Boolean Algebra are:

- **AND** - The AND operation takes two or more inputs and produces a 1 output only if all inputs are 1.
- **OR** - The OR operation takes two or more inputs and produces a 1 output if at least one input is 1.
- **NOT** - The NOT operation takes a single input and produces the opposite value. If the input is 1, the output is 0, and vice versa.

	Formal Logic	**Set Theory**	**Boolean Algebra**
Variables	p, q, r, ...	A, B, C, ...	x, y, z, ...
Operations	$\wedge$, $\vee$, $\neg$	$\cap$, $\cup$, $-$	$\bullet$, $+$, '
Special Elements	F, T	$\emptyset$, U	0, 1

Table 1.1: Comparison of Formal Logic, Set Theory, and Boolean Algebra

Table 1.1 shows a comparison of the notation used in formal logic, set theory, and Boolean Algebra. In formal logic, variables are represented by letters such as p, q, r, etc., and the logical operations are represented by symbols such as $\wedge$, $\vee$, and $\neg$. In set theory, variables are represented by capital letters such as A, B, C, etc., and the set operations are represented by symbols such as $\cap$, $\cup$, and $-$. In Boolean Algebra, variables are represented by letters such as x, y, z, etc., and the Boolean operations are represented by symbols such as $\cdot$, $+$, and $'$. The special elements in each system are F and T in formal logic, $\emptyset$ and U in set theory, and 0 and 1 in Boolean Algebra.

Though the notation used in each system is different, the underlying concepts are the same. For example, the AND operation in Boolean Algebra is similar to the logical conjunction operation in formal logic, where the output is TRUE only if all inputs are TRUE. Similarly, the OR operation in Boolean Algebra is similar to the logical disjunction operation in formal logic, where the output is TRUE if at least one input is TRUE. Compared to set theory, the AND operation in Boolean Algebra is similar to the intersection operation, where the output is the set of elements that are common to all input sets.

1.3.1 AND Operation

The AND operation is denoted by the symbol $\cdot$ or juxtaposition. The output of the AND operation is 1 only if all inputs are 1.

Input 1	**Input 2**	**Output**
0	0	0
0	1	0
1	0	0
1	1	1

Table 1.2: Truth Table for the AND Operation

Table 1.2 shows the truth table for the AND operation. The output is 1 only if both inputs are 1; otherwise, the output is 0.

Suppose we have the variables x and y, and we want to represent the AND operation between them. We can write this as $x \cdot y$ or xy via juxtaposition. The output of this operation is 1 only if both x and y are 1.

Input 1	**Input 2**	**Output**
x	y	xy
0	0	0
0	1	0
1	0	0
1	1	1

Table 1.3: Truth Table for the AND Operation with Variables

Table 1.3 shows the truth table for the AND operation with variables x and y. The output is 1 only if both x and y are 1; otherwise, the output is 0.

Exercise
1. Consider the three input AND operation xyz. Write the truth table for this operation and determine the output for each combination of inputs.

1.3.2 OR Operation

The OR operation is denoted by the symbol $+$. The output of the OR operation is 1 if at least one input is 1.

Input 1	**Input 2**	**Output**
0	0	0
0	1	1
1	0	1
1	1	1

Table 1.4: Truth Table for the OR Operation

Table 1.4 shows the truth table for the OR operation. The output is 1 if at least one input is 1; otherwise, the output is 0.

Suppose we have the variables x and y, and we want to represent the OR operation between them. We can write this as $x + y$. The output of this operation is 1 if at least one of x and y is 1.

Input 1	**Input 2**	**Output**
x	y	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

Table 1.5: Truth Table for the OR Operation with Variables

Table 1.5 shows the truth table for the OR operation with variables x and y. The output is 1 if at least one of x and y is 1; otherwise, the output is 0.

Exercise
Write the truth table for the following OR operations:
1. $x + y + z$
2. $(x + y)z$
3. $x + yz$

1.3.3 NOT Operation

The NOT operation is denoted by the symbol $'$. The output of the NOT operation is the opposite of the input. If the input is 1, the output is 0, and vice versa.

Table 1.6 shows the truth table for the NOT operation. The output is the opposite of the input. If the input is 1, the output is 0, and vice versa.

Input	**Output**
0	1
1	0

Table 1.6: Truth Table for the NOT Operation

Suppose we have the variable x, and we want to represent the NOT operation on it. We can write this as x'. The output of this operation is the opposite of x.

Input	**Output**
x	x'
0	1
1	0

Table 1.7: Truth Table for the NOT Operation with Variables

Table 1.7 shows the truth table for the NOT operation with variable x. The output is the opposite of x. If x is 1, the output is 0; if x is 0, the output is 1.

Exercise
Write the truth table for the following NOT operations:

1. x'
2. $(x')'$
3. $(x+y)'$
4. $(x \cdot y)'$
5. $(x+yz)'$

1.4 Other Operations

In addition to the fundamental operations of AND, OR, and NOT, there are several other operations in Boolean Algebra that are commonly used. These operations include:

- **XOR** - The XOR operation takes two inputs and produces a 1 output if the inputs are different.
- **NAND** - The NAND operation is the complement of the AND operation. The output of the NAND operation is 0 only if all inputs are 1.
- **NOR** - The NOR operation is the complement of the OR operation. The output of the NOR operation is 0 if at least one input is 1.
- **XNOR** - The XNOR operation is the complement of the XOR operation. The output of the XNOR operation is 1 if the inputs are the same.

1.5 Boolean Variables and Expressions

1.6 Boolean Functions

1.7 Laws of Boolean Algebra

2

Logic Gates and Circuits

3

Graph Theory

3.1 Introduction

3.2 Graphs

3.2.1 Terms and Definitions

3.2.2 Paths and Cycles

3.2.3 Hamiltonian Cycles

3.2.4 Shortest Path Algorithms

3.2.5 Representation of Graphs

3.2.6 Isomorphism of Graphs

3.2.7 Planar Graphs

3.3 Trees

3.3.1 Terms and Definitions

3.3.2 Spanning Trees

3.3.3 Binary Trees

3.3.4 Tree Traversals

3.3.5 Decision Trees

3.3.6 Isomorphism of Trees

4

Network Models and Petri Nets

4.1 Network Models

4.2 Maximal Flow Algorithm

4.3 Max Flow, Min Cut Theorem

4.4 Matching

4.5 Petri Nets

5

Automata, Grammars and Languages

5.1 Languages and Grammars

5.2 Finite State Automata

5.3 Regular Expressions

6

Computational Geometry

7

References

A. **Books**

-

B. **Other Sources**

-