The first topic of this book deals with logical symbols and their manipulations. We will also introduce various other kinds of logical operators as well. The study of such manipulations has great connections with the notion of **mathematical proof**, which is an essential theme of this book (from Chapter 4 on, most of the concepts and problems will be of things that are *proven*, not computed). These operators will be useful in later parts of the book. To introduce these concepts we will motivate them by first discussing some physical analogues of these concepts that appear frequently in daily life.

1 Truth and Falsity

Consider what it means for a statement to be true (and respectively, to be false). One common idea is that statements that are true represent a fact. For example, it is true that upon first writing this book, the author was still living with his parents. The statement that the author is filthy rich and living the high life in Las Vegas is a false statement. In general, to assert the validity or invalidity of a statement, one must consider the objects in question and verify that whatever is asserted in the statement holds for them. For example, in order to confirm that the author is not filthy rich and living the high life in Las Vegas, one would check that the author does not make much money by checking their tax statements and check that he does not indeed live or frequent Las Vegas to have fun in morally dubious ways.

For the rest of this book we will not be concerned with any sort of epistemological justification for why certain statements are true or false. We will simply work with objects that are simple enough that we will be satisfied that certain statements can be taken by default to be true or false if necessary, or we will be able to determine the truth or falsity of these claims. We will also simplify our notion of truth to be *absolute*: to simplify our assumptions we will assume that any statement is either true, false, or ambiguous. In general we need to distinguish ambiguous statements. For example, consider the following statement:

$$2 \oplus 2 = 4.$$

Colloquially, one would read this as "2 oplus 2 equals 4". But the truth of this statement is ambiguous, for one, because it depends on how the symbol $\oplus$ is defined. Suppose I defined it in the following way:

$$a \oplus b = \begin{cases} a + b & b < a \\ a - b & b \geq a \end{cases}.$$

Then if we are working with integers or any number system containing the integers this statement is false. If instead we defined it a little differently as

$$a \oplus b = \begin{cases} a + b & b \leq a \\ a - b & b > a \end{cases}$$

then the previous statement would be true.

Now that we have motivated the notion of truth, in this section we will develop means of abstractly manipulating truth and falsity. The most abstract notion of this is denoted

by the term **boolean algebra**. To make this notion more precise, we will introduce some terminology. There are only two values that we will work with explictly. These are true, denoted by T, and false, denoted by F. Some people prefer to work instead with the value 1 and 0, respectively. Whichever we choose, these two values are called boolean values. When refering to an expression that is one of T or F, we will call such an expression a boolean expression. We will indicate examples of these later in this section.

In general we will be concerned with manipulation of expressions where one can change an F in the expression to T, and vice versa. This is because we are interested in when certain patterns of boolean expressions are always equivalent to one another (that is, always both T or both F). For this purpose we will develop the notion of a boolean variable.

Definition 1. A boolean variable p represents a boolean value T or F. Usually this is ambiguous and not explicit. We denote an explicit assignment of a boolean value to a boolean variable using the $\equiv$ symbol. For example,

$$p \equiv T$$

denotes the explicit assignment from the boolean variable p to the boolean value T. Usually we will use the letters p, q, r, and so on to represent boolean variables.

If p is a boolean variable assigned to a boolean value, we define $\sim p$ by the other boolean value. So if $p \equiv T$, then $\sim p \equiv F$, and vice versa.

The full study of boolean algebra is the manipulation of boolean variables and boolean values using what are known as boolean operators. We will give an example of such an example below before introducing the definition.

Example 1. Given two boolean variables p and q, the expression $p \wedge q$ is defined as follows:

$$p \wedge q = \begin{cases} T & p \equiv T, q \equiv T \\ F & \text{otherwise.} \end{cases}$$

The symbol $\wedge$ is known as the **logical and operator**. It is a binary operator (that is, it acts on two variables p and q).

Why would we introduce such an operator in the first place? This is related to the colloquial use of "and" in everyday language. Intuitively, a statement of the form "P and Q" is only true if P and Q are both true. The $\wedge$ operator simply reflects this statement.

Definition 2. A binary logical operator $\oplus$ is an operation that takes two boolean values p and q and outputs depending on the values p and q a boolean value denoted $p \oplus q$.

A unary operator Δ is the same as a binary logical operator, but instead of two boolean values it takes one boolean value p and outputs depending on the value p a boolean value Δp.

As we can verify above, the logical and operator is an example of a binary logical operator. The negation operation $\sim$ is an example of a unary logical operator. We will give more examples of logical operators we will study in depth below.

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

p	q	r	$p \wedge q$	$(p \wedge q) \wedge r$
T	T	T	T	T
T	F	T	F	F
F	T	T	F	F
F	F	T	F	F
T	T	F	T	F
T	F	F	F	F
F	T	F	F	F
F	F	F	F	F

Given two boolean variables p and q, there are only finitely many combinations of boolean values that p and q can evaluate to. This means that we can tabulate the values of $p \oplus q$ in a finite table which completely describes the logical operator $\oplus$. Such a table is called a **truth table**. Below is the truth table for $\wedge$.

Here is how to interpret this table: each row's last entry denotes the value of $p \wedge q$ given the values for columns p and q. We tabulate the value of $p \wedge q$ given every combination of T and F that can be assigned to p and q together.

In general, truth tables can be extended to tabulate expressions of different variables. For example, here is a truth table for the boolean expression $(p \wedge q) \wedge r$.

2 Evaluation of Boolean Expressions as Functions.

In this section we highlight the notion of a boolean expression as a function. Using this perspective it will be clear what it means for two expressions to be logically equivalent.

> A function will take a value (a number, an animal, anything) and output another value (a letter of the alphabet, the number of the animal's legs, anything). Later on we will learn a more rigorous definition than that described above.

Then a boolean expression is really a function. Each variable takes in either true or false, at which point we can *evaluate* the expression and we get some boolean value (either true or false). Consider the expression

$$(p \wedge q) \vee r.$$

Then we can feed in any combination of trues or falses. For example, if $p \equiv T$, $q \equiv F$, and $r \equiv T$, then the expression above will evaluate these values to

$$(T \wedge F) \vee T \equiv F \vee T \equiv T,$$

or true.

A **truth table** is then really just a way of recording the output values of this function. We summarize our findings here:

> Two logical expressions are called **logically equivalent** if they have the same truth value given a set of truth values for each variable.
> Equivalently, two logical expressions are logically equivalent if they have the same truth table.

1. Create truth tables for the following expressions.
   - $p \wedge (\sim p)$ (This formula is called *unsatisfiable*. Why is this?)
   - $p \wedge T$ (Here, T represents true. Similarly, F represents false).
   - $p \wedge F$
   - $p \vee T$ and $p \vee F$.
2. Let's consider the logical expressions in problem 1 again. Figure out simpler expressions which are equivalent to these.
3. Consider the following chunk of code. Here, `var1`, `var2` are boolean variables (ie, they are true or false).

   ```
   if (var1 && (var1 || var2)) { printf("She loves me!"); }
   else { printf("She loves me not..."); }
   ```

   Simplify the code.
4. In logical expressions, parentheses matter!

   Build the truth tables for $(\sim a) \wedge b$ and $\sim (a \wedge b)$. Compare it to the viral facebook math problem below:
5. This problem mostly deals with the implies ($\implies$) logical operator. The motivation behind this operator is that eventually we want to do **proofs**. That is, given a set of initial hypotheses, we want to make deductions and logically deduce that a conclusion is true. For example, eventually all of you will be able to prove, roughly stated:

   $$(n \text{ is even}) \implies (n+1 \text{ is odd}).$$

   Construct the truth tables for all the logical expressions below.
   - $p \implies (p \wedge q)$ and $p \implies (p \vee q)$.
   - $(p \wedge q) \implies (p \vee q)$. This formula is an example of a *tautology*.
   - $(p \vee q) \implies (p \wedge q)$.
   - $(p \implies q) \implies (p \iff q)$.
   - $(p \iff q) \implies (p \implies q)$.
   - $p \implies q \implies r$ and $p \implies r$.

Figure 1: Viral Facebook Math Meme

3 Boolean Satisfiability

In this worksheet, we will briefly discuss a property of boolean formulas called **satisfiability**. This was briefly discussed on the last worksheet, with respect to the boolean formula $p \wedge (\neg p)$.

> A boolean formula is **satisfiable** if there exists an assignment of true or false to each of its variables which makes the formula evaluate to true. Note that we only need *one* such assignment of variables. A boolean formula is **unsatisfiable** if it isn't satisfiable. In particular, all assignments of variables will evaluate to false.

For example, the formula $p \wedge q \wedge r$ is satisfiable (with satisfying assignment $p \equiv q \equiv r \equiv T$. And this is the ONLY satisfying assignment).

1. Determine whether or not the following Boolean formulae are satisfiable:

   - $x_1 \wedge x_2 \wedge \cdots \wedge x_n$, and $x_1 \vee x_2 \vee \cdots \vee x_n$, where n is a natural number.
   - $(x_1 \wedge (\neg x_2)) \vee (x_2 \wedge (\neg x_3)) \vee \cdots \vee (x_{n-1} \wedge (\neg x_n))$.
   - $(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)$.
   - $(x_1 \vee (\neg x_2) \vee x_3) \wedge (x_1 \vee (\neg x_2) \vee (\neg x_3)) \wedge ((\neg x_1) \vee x_2 \vee (\neg x_3))$.

2. Explain why a boolean formula with n variables has 2^n different assignments.

   (a) Suppose I was to tell you that a certain boolean formula was satisfiable. What would I need to tell/demonstrate to you in order to convince you of this fact? How much time would it take?

(b) Suppose on the other hand I was to tell you that a certain boolean formula was *unsatisfiable*. What would I need to demonstrate to you in order to convince you of this fact? How much time would it take?

3. Sometimes, despite the computational time it takes to find a satisfying assignment, some unfortunate circumstance may require that you do so. This problem will outline one or two heuristics in order to possibly simplify your job a little further.

   Consider the boolean formula

$$(x_1 \wedge x_5 \wedge x_2) \vee (x_3 \wedge (\neg x_3) \wedge x_5) \vee (x_2 \wedge (\neg x_7) \wedge x_6 \wedge x_1 \wedge (\neg x_4)).$$

   (a) Notice that this formula is in *disjunctive normal form*. That is, it can be represented as an OR of ANDS. Explain how this might simplify our problem a little bit.

   (b) Simplify the formula by considering the variable x_3.

   (c) Discuss (with your neighbors) the theoretical or practical use of such simplifications.