Logic, Sets, and Function

John M. Morrison

August 21, 2023

Contents

0 Introduction **2**

1 Predicates and Boolean Operations **2**

2 Logical Connectives **3**

3 Tautologies **4**

4 Associating and Distributing **5**

5 Quantifiers **7**

6 Sets **8**

7 Relations **10**

7.1 Equivalences and Equivalence Classes 10

7.2 Ordering Relations 11

8 Functions **11**

9 Composition of Functions **13**

0 Introduction

In this chapter we will lay out certain ideas that are absolutely fundamental to the study of mathematics or computer science. Logically enough, we will begin with the study of propositional calculus and elementary set theory. This is the semantic foundation of mathematics.

1 Predicates and Boolean Operations

We begin with a definition. A *predicate* or *proposition* is a statement that evaluates to `true` or `false`. For example, the proposition $4 * 5 = 17$ evaluates to `false`. The proposition stating that a US dollar has a vale of 100 cents to `true`.

Things get interesting when we begin to combine propositions. The most basic operation you can do on a proposition is to negate it, i.e. reverse its truth value. This is accomplished using the prefix unary operator $\neg$. For example, if we take $P = (4 * 5 = 17)$, then $\neg P = (4 * 5! = 17)$, which is `true`. This operator is defined in a truth-table as follows.

P	$\neg P$
T	T
T	F

If P and Q are propositions, then the proposition $P \vee Q$ is defined to be `true` if at least one of P or Q is true. The infix binary operator $\vee$ is called the *Boolean or operator*. Here is the truth-table for it. Note that the table must hve four lines, one for each possible state of the two propositions.

P	Q	$P \vee Q$
T	T	T
T	F	T
F	T	T
F	F	F

If P and Q are propositions, then the proposition $P \wedge Q$ is defined to be `true` if both of P and Q are true. The infix binary operator $\wedge$ is called the *Boolean and operator*. Here is its truth-table.

P	Q	$P \wedge Q$
T	T	T
T	F	F
F	T	F
F	F	F

If P and Q are propositions, then the proposition $P \oplus Q$ is defined to be `true` if exactly one of P or Q are true. The infix binary operator $\oplus$ is called the *Boolean xor (exclusive or) operator*. Here is its truth-table.

P	Q	$P \oplus Q$
T	T	F
T	F	T
F	T	T
F	F	F

There is an order of precedence for and, or and not is not, and, or. So int expression $\neg P \vee Q$, the $\neg P$ binds to the P before $\vee$ can act. The order of operations can be overridden, just as in Miss Wormood's Algebra class, using parentheses.

2 Logical Connectives

This is all very interesting, but we are not equipped with a means for connecting propositons. We shall remedy that deficiency presently.

If P nd Q are propositons, we define the *implies* operator $\implies$ by $P \implies Q$ when $\neg P \vee Q$. Here is a truth-table

P	Q	$P \implies Q$
T	T	T
T	F	F
F	T	T
F	F	T

A consequence of this definition is that a false statement implies anything. This is how we reperesent the if-then turn of logc. If P is true, Q must be true. But if P is false, all bets are off. The $\implies$ operator behaves much like $\leq$ for numbers.

There is a second logical connector, $\iff$. We show its truth-table here.

P	Q	$P \iff Q$
T	T	T
T	F	F
F	T	F
F	F	T

The proposition $P \iff Q$ is true precisely when P and Q have the same truth-values.

If we have an implcation $P \implies Q$ we say that $Q \implies P$ is the *converse* of $P \implies Q$. Because an implication is true, there is no guarentee its converse is true. Consider these predicates, $P(x) =$ "x is a cat, and $Q(x)$, "x is a tiger." We have $Q(x) \implies P(x)$ becasue every tiger is a cat. On the other hand, a lion is not a tiger, so we do not have $Q(x) \implies P(x)$. This is a common source of faulty logic.

In the order of operations, logical connectors have the lower precedence than not, and, or, and exclusive or.

3 Tautologies

A *tautology* is a proposition that is always true. You can verify tautologies by using truth-tables. It is the most primitive form of mathematical truth.

Theorem 1. *For a proposition P, $\neg\neg P \iff P$.*

Proof. To establish this, we build out a truth-table as follows. First let's negate P

P	$\neg P$
T	F
T	F

Now add a column for $\neg\neg P$ We populate it by negating the items in the $\neg P$ columnn.

P	$\neg P$	$\neg P$
T	F	T
T	F	T

Now we add a $P \iff \neg P$ column, populating it by checking if the P and $\neg P$ columns have the same truth-values

P	$\neg P$	$\neg\neg P$	$P \iff \neg\neg P$
T	F	T	T
T	F	T	T

□

The not operator $\neg$ is an *involution*, i.e. it is its own inverse. This is has a familiar look, as the prefix unary change-sign opeator - in arithmetic is its own inverse. We will now establish a very important tautology that relates implication to logical equivalance.

Theorem 2. *If P and Q are propositions, then*

$$(P \iff Q) \iff (P \implies Q \wedge Q \implies P).$$

Proof. We begin by obtaining the truth values for $P \implies Q \wedge Q \implies P$.

P	Q	$P \implies Q$	$Q \implies P$	$P \implies Q \wedge Q \implies P$
T	T	T	T	T
T	F	F	T	F
F	T	T	F	F
F	F	T	T	T

Now compare this with the truth-table for $P \iff Q$.

P	Q	$P \iff Q$
T	T	T
T	F	F
F	T	F
F	F	T

Since the truth values in the last columns agree, the tautology follows. So $\implies$ behaves much like $\leq$ does for numbers. □

Exercises

1. Verify the tautology $P \vee Q \iff Q \vee P$. The operator $\vee$ is commutative. So is $\wedge$, and the proof of that is boringly similar to the case of $\vee$.
2. Verify the tautology $(P \implies Q) \iff (\neg Q \implies \neg P)$. This is called the *contrapositive*. If it's not a cat, it's not a tiger.
3. Verify the De Morgan law $\neg(P \vee Q) \iff \neg P \wedge \neg Q$.
4. Verify the De Morgan law $\neg(P \wedge Q) \iff \neg P \vee \neg Q$. Can you obtain this DeMorgan law from the other?

4 Associating and Distributing

In general, if you want to verify a tautology with n propositions in it, you must make a table with 2^n rows, so you get all of the joint states of the truth-values of the propositions. Here is Here is a case study

Theorem 3. *For any propositions P, Q and R, $P \vee (Q \vee R) \iff (P \vee Q) \vee R$.*

Proof. Here are the first three columns. Note the "alternating scheme" for listing all of the joint true/false possiblities.

P	Q	R
T	T	T
T	T	F
T	F	T
T	F	F
F	T	T
F	T	F
F	F	T
F	F	F

Now let us get the truth-values for $(P \vee Q) \vee R$.

P	Q	R	$P \vee Q$	$(P \vee Q) \vee R$
T	T	T	T	T
T	T	F	T	T
T	F	T	T	T
T	F	F	T	T
F	T	T	T	T
F	T	F	T	T
F	F	T	F	T
F	F	F	F	F

This shows us that $(P \vee Q) \vee R$ is true exactly when at least one of P, Q, or R is true. Since $(P \vee Q) \vee R \iff R \vee (P \vee Q)$, we see that $(P \vee Q) \vee R$ is true when at least one of R, P or Q is true. Our result follows immediately. □

Exercise Write a similar proof to show that $\wedge$ is associative.

Theorem 4. *If P, Q, and R are propositions, then*

$$P \vee (Q \wedge R) \iff (P \vee Q) \wedge (P \vee R).$$

Proof. Begin by constructing a a table for $P \vee (Q \wedge R)$

P	Q	R	$Q \wedge R$	$P \vee (Q \wedge R)$
T	T	T	T	T
T	T	F	F	T
T	F	T	F	T
T	F	F	F	T
F	T	T	T	T
F	T	F	F	F
F	F	T	F	F
F	F	F	F	F

Now let us construct a table for $(P \vee Q) \wedge (P \vee R)$.

P	Q	R	$P \vee Q$	$P \vee R$	$(P \vee Q) \wedge (P \vee R)$
T	T	T	T	T	T
T	T	F	T	T	T
T	F	T	T	T	T
T	F	F	T	T	T
F	T	T	T	T	T
F	T	F	T	F	F
F	F	T	F	T	F
F	F	F	F	F	F

The last columns in the two tables match, so our result follows. □

5 Quantifiers

Two quantifiers exist in logic, the *universal quantifier* $\forall$ and the *existential quantifer* $\exists$. You read $\forall$ as "For all..." and $\exists$ as "There exists... "' or "There is... "

Mathematical conditions are rife with quantifiers. Here is a simple example.

$$\forall x \in \mathbb{Z}, x < x + 1.$$

This proposition is true becuase if you add 1 to any integer, that integer becomes larger. Quantifiers can be used to establish context in a discussion.

Context, in fact, is vital because no meaningful discussion can occur until a context for it exists. This fact is a consequence of Russel's paradox.

Notice that the universal quantifier is an extension of the notion of "or," and that the existential quantifier is a generalization of "and."

Suppose we have the proposition $\forall x P(x)$, where P is some boolean-valued expression in x. How do we negate it? All we need is one x for which $P(x)$ is false and we are done. And that proposition is $\exists x \neg P(x)$. Informally, youm can move the negation inside to the predicate if you "flip" the quantifier from universal to existential. We have this tautology.

$$\neg \forall x P(x) \iff \exists x \neg P(x).$$

A similar sort of thing occurs with the existential quantifier. We have

$$\neg \exists x P(x) \iff \forall x \neg P(x).$$

Think about this for a moment and you will see why it works.

6 Sets

In mathematics, you often deal with collections of objects. The most fundamental collection in mathematics is the set. You can go into the tall weeds and get lost in the vagaries of axiomatic set theory. This gets complicated and it's probably a wee bit abstruse for this leel. We will approach set theory naively.

Every meaningful discussion requires a context in which it can occur. In set theory we do this by specifying a *universe of discourse*, which is customarily represented by the Greek letter Ω. Even in the finite world of computer science, such a universe can be infinite. For example, the set of all characters sequences (strings) is infinite. What you can hold in a computer's memory is not.

Sets are defined by the primitive notion of *membership*. We write $x \in A$ to indicate that the item x is a member of the set A. To negate this statement, we write $x \notin A$. By convention a universe of discourse Ω will be present. Often Ω will be specifed explicitly.

For now let us take Ω to be the integers. One way to specify a set is to make an explicit list like so $A = \{1, 5, -3, 9, 14\}$. We see that $1 \in A$ but $42 \notin A$. Another ways is to use a predicate as we see here,

$$B = \{x \in \mathbb{Z} | 2 \mid x\},$$

where $2 \mid x$ is the predicate that asserts that 2 divides into x evenly. This is the set of all even numbers. Some authors call this *set builder notation*.

For a set A in a universe of discourse Ω we define the *complement* of A by

$$A^c = \{x \in \Omega | x \notin A\}.$$

The complement operator is a unary operator. You can easily see that $x \in A^c \iff \neg x \in A$.

If A and B are sets in a universe of discourse Ω, we say that A is a *subset* of B if $\forall x \in A, x \in B$. Equiv:alently we can say $\forall x \in \Omega, x \in A \implies x \in B$. For this situation, we write $A \subseteq B$.

The sets A and B are said to be `equal` if $A \subseteq B$ and $B \subseteq A$. Two sets are equal if they contain the same elements.

Note! If you make a set with a list, and put a duplicate element in it, that duplicate is ignored. For exaple, you have

$$\{1, 2, 3\} = \{1, 2, 2, 3, 3, 3\},$$

because both sets contain 1, 2, and 3 and no other elements.

There is a variant on set called *multiset*, which allows for duplicate entries. In that case, two multisets are equal if they are equal as sets and all items in the two sets have the same multiplicities.

A set with no elements called an *empty set*. In fact,there is only one empty set. Here is why. Supposei there are two empty sets E an F and that $x \in E$. Since an empty set is devoid of elements, this statment is false. A fals statement implies anything so this statment implies $x \in F$. We have $E \subseteq F$. By symmetry, $F \subseteq E$, so $E = F$.

We shall denote *the* empty set by $\emptyset$. By this same turn of logic, the empty set is a subset of every set. So if A is a set in a universe of discourse Ω we are guaranteed that $\emptyset \subseteq A \subseteq \Omega$.

Exercises

1. Show that if A and B are sets in a universe of discourse Ω,
$$A \subseteq B \iff B^c \subseteq A^c.$$
2. If A is a set in a unverse of discourse Ω, show $A^{cc} = A$.

There are two infix binary operators on sets. The *union* of sets A and B is defined to be
$$A \cup B = \{x \in \Omega | x \in A \vee x \in B\}.$$
This is the set of all elements of the two sets combined. The *intersection* of two sets is the set of all elements common to both. We define it as follows
$$A \cap B = \{x \in \Omega | x \in A \wedge x \in B\}.$$
We say A and B are *disjoint* if $A \cap B = \emptyset$.

Exercises

1. Show that if A and B are sets that A and B are disjoint if and only if $A \subseteq B^c$.
2. Show that union and intersection are commutative.
3. Show that union and intersection are associative.
4. Show that $(A \cup B)^c = A^c \cap B^c$.
5. Show that $(A \cap B)^c = A^c \cup B^c$.
6. Show that $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$.
7. Show that $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$.

The *relative complement* or *difference* of two sets A and B, $A - B$ is defined to be $A \cap B^c$. Note that we have the equality
$$A - B = \{x \in A | x \notin B\}.$$

7 Relations

Relations on sets allow us to do two imporant things. They give us an apparatus for "lumping together" parts of sets and declaring them equivalent. They are also imporant for creating orders on objects, so they become sortable. You will see that this is a fundamental tool in the mathematican's bag o' tricks.

If A and B are sets we define the *cartesian product* of A and B by

$$A \times B = \{(a, b), a \in A, B \in B\}.$$

The cartesian product is just the set of ordered pairs from the two factor-sets. A *relation* $\sim$ on a set S is a subset of $S \times S$.

It is common convention to use infix notation for relations. We write $x \sim y$ for $(x, y) \in\sim$, and we say "x is related to y." Now let us introduce a little taxonomy.

- A relation is *reflexive* if every element is related to itself.
- A relation is *irreflexive* if no element is related to itself.
- The relation R on the set S is *symmetric* if $\forall x, y \in S, x \sim y \implies y \sim x$
- The relation R on the set S is *asymmetric* if $\forall x, y \in S, x \sim y \implies \neg y \sim x$
- The relation R on the set S is *asymmetric* if $\forall x, y \in Sx \sim y \wedge y \sim x \implies x = y$
- The relation R on the set S is *transitive* if $\forall x, y, z \in S, x \sim y \wedge y \sim z \implies x \sim z$

7.1 Equivalences and Equivalence Classes

A relation on a set is an *equivalence relation* if it is reflexive, symmetric, and transitive.

Suppose that S is a set and that $\approx$ is and equivalence relation on S. We define the *equivalence class* of x to be

$$[x] = \{y \in S | x \approx y\}.$$

Theorem 5. *Any two equivalence classes on a set are either equal or disjoint.*

Proof. Suppose that S is a set and that $\approx$ is and equivalence relation on S. Choose $a, b \in S$. Suppose that $[a] \cap [b]$ is nonempty; choose an element c from it. Since $c \in [a]$, $c \approx a$. Since $c \in [b]$, $c \approx b$. By symmetry, we have $a \approx c$. Now $c \approx b$ so we can invoke transitivity to see that $a \approx b$. By transitivity, any element equivalent to a is also equivalent to b, so $[a] \subseteq [b]$.

By an identical argument, we can see that $[b] \subseteq [a]$. □

As a result, the equivalence classes on a set partition the set into disjoint pieces. A converse also holds: if you partition a set into disjoint pieces and declare two elements related if they reside on the same piece, then the result is an equivalence relation If $\sim$ is an equivalence on a set S, we denote by $X/\sim$ the set of all equivalence classes on X.

7.2 Ordering Relations

These come in two flavors, non-strict and strict. Both types are transitive. A parial order is *strict* if it is asymmetric, transitive, and irreflexive. An example of this is the usual $<$ on numbers. A partial order is *non-strict* if it is anti-symmetric, reflexive, and transtive. Or old friend $\leq$ on numbers is a non-strict partial order.

A partial order $\sim$ on a set $S$is a *linear order* if

$$\forall x, y \in S, (x \sim y) \vee (y \sim x) \vee (x = y).$$

to wit, a partial order is a linear order if every pair of nonequal elements is comparable in the order.

Sets that are linearly ordered are sortable. This notion is very useful in computing.

Exercises

1. If $\sim$ is a strict partial order, define a new relation $\approx$ by $x \approx y \iff (x \sim y) \vee (x = y)$. Show this is a non-strict partial order.
2. If $\approx$ is a non-strict partial order, define a relation $\sim$ by

   $$x \sim y \iff (x \approx y) \wedge (x \neq y).$$

   Show this is a strict partial order.
3. Suppose an order is transitive and symmetric. Find the flaw in this argument puporting to show it is an equivalence. Choose any $x \in S$ and a $y \in S$ so $y \sim x$ Since the order is symmetric, we have $x \sim y$. So, with $x \sim y$ and $y \sim x$, we conclude that $x \sim x$. The order is reflexive, and is therefore an equivalence.

8 Functions

Let A and B be sets. A *function* $f : A \to B$ is a rule that associates with eah $a \in A$ some $f(a) \in B$. The set A is called the *domain* of the function and the

set B is called the *codomain*. Hence, to completely specify a function, you must specify three things: domain, codomain, and rule.

In mathematical parlance, it is common to speak of "the function $f(x) = x^2$." Tactly, the math people are taking the domain and codomain of f to be the real numbers.

Here is an example of a function commonly seen in computing. Let A denote the set of all character strings. We can define a function len : $A \to \mathbb{N}_0$ by taking len(s) to be the number of characters in s.

Note to Computer Scientists A function in a program is only a function in this sense if it has no side-effects and if it is consistent; i.e. the same input produces the same output very time. For example, a function that generates a pseudorandom number does not meet the criterion of consistency. Programming functions that are also mathematical functions are often referred to as *pure functions*. The string-length example we just gave is a pure function.

The use of pure functions is very desirable in writing programs, since these function are both easy to test and they do not interfere with each others' internal mechanisms.

There are two important properties of function we are going to focus on. A function $f : A \to B$ is said to be 1-1 or *injective* if $\forall x, y \in A, f(x) = f(y) \implies x = y$. By the contrapositive, this is equivalent to saying that distinct domain values are associated via f with distinct values in the codomain.

A function $f : A \to B$ is said to be 1-1 or *surjective* or *onto* if $\forall y \in B \exists x \in A$ with $f(x) = y$. This says that every value in B is associated with some value in A.

A function that is 1-1 and onto is said to be *bijective*.

These notions of injectivity and surjectivity depend upon the domain and codomain specified for the function. Suppose you have the rule $f(x) = x^2$. If both the domain and range are real numbers, then f is not 1-1 because $f(1) = f(-1) = 1$. It is not onto because there is no real number whose square is -1.

However, we cut the domain and codomain down to the nonnegative numbers, f is both 1-1 and onto.

The function len we discussed earlier is onto because you can create a character string of any length you wish. For example len("aaaaaaaaaa") = 10. However, it is clear that it is not injective since len("a") = len("b") = 1.

Exercises

1. Consider the function that has as domain and range all character strings

and which associates each string with a an upper cased verison of itself ("abc123" $\mapsto$ "ABC123"). Is it 1-1? Onto?

2. What domain and codomain can you give the tangent function so it is 1-1 and onto?
3. Consider the folowoing definition. For $n \in \mathbb{N}$ define $f(n) = 2n+1+f(n-1)$ for $n > 1$ and $f(0) = 0$. Is this function defined for all integers $n \geq 0$? Compute a short table of values and see if you can write an explicit formula in n for it.

9 Composition of Functions

Suppose that A, B and C are sets and that $f : A \to B$ and $g \to C$ are functions. We define the function $g \circ f : A \to C$ by $g \circ f(a) = g(f(a))$, $a \in \mathbb{A}$. This operation $\circ$ is called *functional compsition*.

This operation is *not* commutative. To see this define for a real number x, $f(x) = x^2$ and $g(x) = x + 1$. Then $f \circ g(x) = (x + 1)^2$ and $g \circ f(x) = x^2 + 1$ for all real x. They fail to be equal for "nearly all" values of x. (Where are they equal?)

It is, however, associative. The expressions $(h \circ g) \circ f$ and $h \circ (g \circ f)$ just amount to applying f, then g then h in succession.

Exercises Suppose that A, B and C are sets and that $f : A \to B$ and $g : B \to C$. are functions.

1. Show that if f and g are 1-1 then $g \circ f$ is 1-1.
2. Show that if f and g are onto then $g \circ f$ is onto.
3. Define a relation $\sim_f$ on A by $a_0 \sim_f a_1$ if $f(a_0) = f(a_1)$. Show this is an equivalence relation on A.
4. We define the *canonical projection* $\pi_A(a, b) = a$ for $a \in A$. Show that this is onto. Under what condition is it 1-1?