Notes on Discrete Mathematics

Justin Goodman

Updated November 27, 2019

Mom, I never thought you were bad at math

Justin

Contents

Chapter 0

Introduction

> In the beginning the Universe was
> created. This has made a lot of
> people very angry and been
> widely regarded as a bad move.
>
> Douglas Adams

Welcome! This book is essentially my own typed-up notes/explanations from the *Discrete Structures* (CMSC250) course at UMD. There may be typos – my apologies. My hope is to improve your overall understanding of the course content. If you're willing to put forth the effort to *understand* the course material, and think mathematically, then you will succeed.

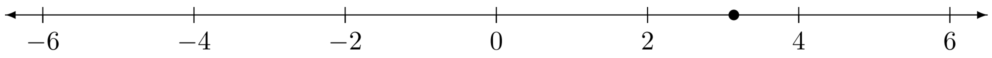
0.1 What is Discrete Mathematics?

Sometimes the actual name of the course is never addressed in-class. The 'mathematics' part hopefully should be familiar – numbers and stuff. What does 'discrete' mean? Well, Merriam-Webster provides two good definitions:

Definition 0.1.1 Discrete

- consisting of distinct or unconnected elements
- taking on or having a finite or countably infinite number of values

Consider the real number line:



We can point to any position on the line and find some real number to represent that point. It may have some long string of decimal digits, however we can do it. In the example above, the point is $\pi \approx 3.14159$. We refer to this real line as *continuous*. Broken up in-between all of these real numbers is a set of discrete points $\{\cdots, -2, -1, 0, 1, 2, \cdots\}$. These points are not continuous, and hence discrete, since we must make a jump (on the real line) from number to number. Our goal in this class is to examine these discrete numbers (which we call the *Integers* $\mathbb{Z}$) and structures.

0.2 Why Should You Care?

It seems somewhat pointless just to learn this whole proof crap (that you figured you never had to do again – see high school Geometry) when you will probably never use it again. So, here (in no particular order) is a list of reasons for and against learning this material.

FOR

- You become a free-thinker, and you become more skeptical
- You begin to question your world and your own perceptions
- The mathematical mindset will help you throughout your life, and will be especially useful while coding
- Your creativity comes back, and you may end up enjoying the course
- Computer science is just applied mathematics

AGAINST

- Proofs are boring and math sucks
- Other people can do the proofs and math for me – I am willing to accept their work
- Discrete math is just a weed-out class
- I do not need math to know how to code
- Discrete math will not improve my programming

If you disagree with any of the FOR reasons, then you should strongly consider why you chose computer science. We now provide counter-arguments to each of the above AGAINST points.

First, proofs and math are very akin to problem-solving. Programming is entirely problem-solving. If you do not enjoy problem-solving, then you are going to hate any programming job you get and you should really consider why you chose computer science.

Second, having other people do the leg-work will only get you so far in life. Eventually, someone is going to ask you to do the hard part too. Plus, if you blindly accept other people's work without checking it for yourself, you become vulnerable to mistakes. Unchecked work leads to mistakes, which leads to wasted money and lost revenue, which leads to you getting fired from your job.

Third, discrete math is a weed-out class in the sense of the previous point. It is a basis for computer science, and if you do poorly in it then that may be an indicator of your future success in the field.

Fourth, yeah right.

Fifth, you keep telling yourself that while the top programmers at Google use mathematical reasoning to create super-fast mapping algorithms.

0.3 Thinking Mathematically

Well, hopefully at the end of the course you will be able to answer this question for yourself. Our goal is to teach you how to think mathematically. We want you to ask and answer questions based on logic. We want you to have sound reasoning. We want you to be able to back-up your answers with solid proof. We want your explanations to make sense, and we want to be convinced. In the real-world, no one is going to just *accept* things you say unless you have good *evidence* to back your claims. Our goal is to teach you how to (1) formally write your claims, and (2) provide proof for your claims. Mathematical thinking comes naturally for some, however for others it may take a whole semester. Keep working at it until you succeed.

0.4 How to Succeed in this Course

This course will provide you a toolkit of concepts from which you can pull to solve problems. Unlike methodological/algorithm-based courses (e.g. Calculus III), this course highly depends on your solid *understanding* of the material. To succeed in this course, you will need to immerse yourself in the material. You will need to struggle through it until you understand. For some of you, this will be quick and easy. For others, not so much. That is okay. You will succeed so long as you put in the effort.

0.5 Summary

In order to succeed, you must:

- Immerse yourself in the course content
- Struggle until you understand

- Think mathematically

Chapter 1

Logic

All opinions are not equal. Some are a very great deal more robust, sophisticated and well supported in logic and argument than others.

Douglas Adams

Contents

1.1 Introduction

Let's spell out two logic puzzles:

Sam has 1 cow. If Sam has at least 2 cows, then Sam can breed the cows to make one more cow. Assuming Sam has access to infinite resources and time, how many cows can Sam make?

Two givens: knights always tell the truth, and knaves always lie. On the island of knights and knaves, you are approached by two people. The first one says to you, "we are both knaves." What are they actually? (from Popular Mechanic's Riddle of the Week #43: Knights and Knaves, Part 1)

Thinking logically about these puzzles will help you – think about what can and cannot happen; what can and cannot be true. Solutions to the above puzzles appear at the end of the chapter. There are only two possibilities for Boolean statements – True or False. Here's a formal definition of logic, from Merriam-Webster:

Definition 1.1.1 Logic
A science that deals with the principles and criteria of validity of inference and demonstration : the science of the formal principles of reasoning

This chapter includes a wealth of topics. We will touch on propositional logic and its corollaries, as well as predicate logic and it's applications to the rest of this course. Thinking logically should be a natural process, so we hope this section is relatively straightforward.

1.2 Propositional Logic

Propositional logic is a branch of logic that deals with simple propositions and logical connectives. Sometimes propositional logic is referred to as **zeroth-order logic**, as it lays the foundations for *predicate logic*, also known as *first-order logic.*

Definition 1.2.1 Proposition
A statement that is exclusively either true or false. A proposition must *have* a true or false value, and it cannot have *both.* Propositions are usually represented as variables. These variables can represent a single statement, or large compound statements. We will see examples later

Definition 1.2.2 Logical Connective
An operation that connects two propositions. We study these below

The motivation behind propositional logic is that we want to represent basic logical statements as an expression of variables and operators. Propositional logic also lays the groundwork for higher-order logic.

1.2.1 Truth Tables and Logical Connectives

Before we dive into the logical connectives, let's study the notion of a truth table. This will help us fully understand the logical connectives.

Definition 1.2.3 Truth Table
A table that shows us all truth-value possibilities. For example, with two propositions p and q:

p	q	*some compound proposition*
F	F	T/F
F	T	T/F
T	F	T/F
T	T	T/F

Now we can begin our study of the logical connectives. The following definitions explain the intuition behind the logical connectives, and present their associated truth tables.

Definition 1.2.4 And $\wedge$
Also known as the *conjunction*. Logical connective that evaluates to true when the propositions that it connects are both true. If either proposition is false, then *and* evaluates to false. To remember: *prop 1* **and** *prop 2* must *both* be true. Truth table:

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

Notice the only row that evaluates to true is when both propositions are true

Definition 1.2.5 Or $\vee$
Also known as the *disjunction*. Logical connective that evaluates to true when either of the propositions that it connects are true (at least 1 of the connected propositions is true). If both propositions are false, then *or* evaluates to false. **Note**: if both propositions are true, then *or* still evaluates to true. To remember: either *prop 1* **or** *prop 2* must be true. Truth table:

p	q	$p \vee q$
F	F	F
F	T	T
T	F	T
T	T	T

Notice the only row that evaluates to false is when both propositions are false

Definition 1.2.6 Not $\neg$, $\sim$
Also known as the *negation*. Logical connective that flips the truth value of

the proposition to which it is connected. Unlike *and* and *or*, *not* only affects 1 proposition. Truth table:

p	$\neg p$
F	T
T	F

Definition 1.2.7 Implication/Conditional $\Rightarrow$, $\rightarrow$
Logical connective that reads as an *if-then* statement. The implication must be false if the first proposition is true and the implied (connected/second) proposition is false. Otherwise it is true. Truth table:

p	q	$p \Rightarrow q$
F	F	T
F	T	T
T	F	F
T	T	T

Note: the direction of an implication can be flipped: $p \Leftarrow q$ is the same as $q \Rightarrow p$

Let's try to understand the truth table for the implication statement before we continue. We present two examples that attempt to form an intuitive analogy to the implication.

Example: Think of the implication as a vending machine. p is the statement *we put money into the vending machine*, and q is the statement *we received a snack from the vending machine.* Notice that the statements do not necessarily depend on each other. We examine the four cases and see when we are *unhappy*:

1. p is **false** and q is **false** – we did not put in money, and we did not get a snack, so we remain happy (normal operations)
2. p is **false** and q is **true** – we did not put in money, and we did get a snack, so we are very very happy (free snack!)
3. p is **true** and q is **false** – we did put in money, and we did not get a snack, so we are very very unhappy (we got robbed!)
4. p is **true** and q is **true** – we did put in money, and we did get a snack, so we are happy (normal operations)

When we are unhappy, then the implication statement is false. Otherwise it is true (we are not *unhappy*).

Example: Think of the implication in the lens of a program. You want to evaluate whether your program *makes sense.* Here is the example program from the statement $p \Rightarrow q$:

```
(...)
if (p is true) {
        <run body code if q is true>
        }
        ( . . . )
```

The body code is run only if q is true. Now let's examine the 4 cases and see whether the program makes sense:

1. p is **false** and q is **false** – the program does not go into the body of the if-statement and hence makes sense
2. p is **false** and q is **true** – the program again does not go into the body of the if-statement and hence makes sense (regardless of the value of q)
3. p is **true** and q is **false** – the program goes into the body of the if-statement but since q is false the program does not evaluate the body code. This does not make sense
4. p is **true** and q is **true** – the program goes into the body of the if-statement and evaluates the body code. This makes sense

When the code evaluator makes sense, then the implication statement is true.

Definition 1.2.8 Bi-conditional $\Leftrightarrow$, $\leftrightarrow$
Logical connective that reads as an *if and only if* statement. This means that both propositions must imply each other. For the bi-conditional to be true, both propositions must either be true or false. Truth table:

p	q	$p \Leftrightarrow q$
F	F	T
F	T	F
T	F	F
T	T	T

Some more complicated ones:

Definition 1.2.9 Exclusive Or (Xor) $\oplus$
Logical connective that evaluates to true when *only* one of the two propositions that it connects is true. If both propositions are true or false, then *xor* evaluates to false. The exclusive part means we *exclude* the *or* case when both propositions are true. Truth table:

p	q	$p \oplus q$
F	F	F
F	T	T
T	F	T
T	T	F

Definition 1.2.10 Exclusive Nor ($Xnor$) $\otimes$
Logical connective that negates the *xor*. It is just an *xor* connective appended with a *not* connective. Truth table:

p	q	$\neg(p \oplus q) \equiv (p \otimes q)$
F	F	T
F	T	F
T	F	F
T	T	T

Now an example:

Example: Translate the following statement into a propositional logic statement: *exclusively either the weather rains or students wear rain jackets.*

Solution: Let r be the proposition *the weather rains* and j be the proposition *students wear rain jackets.* Then the statement becomes $r \oplus j$

1.2.2 Boolean Algebra

The motivation behind Boolean algebra is that we want to take complicated compound propositional statements and simplify them. If we notice that a variable does not affect the final output, then getting rid of that variable cuts the amount of truth-value possibilities (truth-table rows) in half.

Definition 1.2.11 Boolean Statement
A statement that is exclusively either true or false

Some handy notations:

Definition 1.2.12 Equivalence $\equiv$
Logical equivalence says that the two connected statements are logically the same. You can think of this notation as the *equals* sign. Equality is poorly defined for Boolean expressions, so we use the equivalence notation instead

Definition 1.2.13 Tautology t *or* **T**
A proposition that is always true

Definition 1.2.14 Contradiction c *or* **F**
A proposition that is always false

We provide a handful of helpful theorems to aid in your Boolean algebra simplifications. You do not need to memorize these theorems – they will be given to you as a table.

Theorem 1.2.1 Commutativity
For any propositions p and q the ***and*** *and* ***or*** *operations are commutative:*

$$p \vee q \equiv q \vee p$$

$$p \wedge q \equiv q \wedge p$$

Theorem 1.2.2 Associativity
For any propositions p, q, and r the ***and*** *and* ***or*** *operations are associative:*

$$(p \vee q) \vee r \equiv p \vee (q \vee r)$$

$$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$$

Theorem 1.2.3 Distributivity
For any propositions p, q, and r the ***and*** *and* ***or*** *operations are distributive:*

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

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

Theorem 1.2.4 Identity
For any proposition p the following hold:

$$p \vee c \equiv p$$

$$p \wedge t \equiv p$$

Theorem 1.2.5 Negation
For any proposition p the following hold:

$$p \vee \neg p \equiv t$$

$$p \wedge \neg p \equiv c$$

Theorem 1.2.6 Double Negation
For any proposition p the following holds:

$$\neg(\neg p) \equiv p$$

Theorem 1.2.7 Idempotence
For any proposition p the following hold:

$$p \vee p = p$$

$$p \wedge p = p$$

Theorem 1.2.8 De Morgan's
For any propositions p and q the following hold:

$$\neg(p \vee q) = \neg p \wedge \neg q$$

$$\neg(p \wedge q) = \neg p \vee \neg q$$

Theorem 1.2.9 Universal Bound
For any proposition p the following hold:

$$p \vee t \equiv t$$

$$p \wedge c \equiv c$$

Theorem 1.2.10 Absorption
For any propositions p and q the following hold:

$$p \vee (p \wedge q) \equiv p$$

$$p \wedge (p \vee q) \equiv p$$

Theorem 1.2.11 Negation of Tautology and Contradiction
The following hold:

$$\neg t \equiv c$$

$$\neg c \equiv t$$

Here are two important theorems that you will use throughout your proofs in this course. The first theorem says that for a bi-conditional both propositions must imply each other. The second theorem gives an equivalence between the implication and *or* connective. The proofs of these theorems are left as an exercise to the reader.

Theorem 1.2.12 Bi-conditional to Implication
For any propositions p and q the following hold:

$$p \Leftrightarrow q \equiv (p \Rightarrow q) \wedge (q \Rightarrow p)$$

Theorem 1.2.13 Implication to Disjunction
For any propositions p and q the following holds:

$$p \Rightarrow q \equiv \neg p \vee q$$

Now some examples of Boolean statement simplification.

Example: Simplify the following expression: $(p \Rightarrow q) \Rightarrow r$

Solution:

$$\begin{aligned}
(p \Rightarrow q) \Rightarrow r &\equiv \big(\neg(p \Rightarrow q)\big) \vee r && \text{Implication to Disjunction}\\
&\equiv \big(\neg((\neg p) \vee q)\big) \vee r && \text{Implication to Disjunction}\\
&\equiv ((\neg(\neg p)) \wedge (\neg q)) \vee r && \text{De Morgan's}\\
&\equiv (p \wedge (\neg q)) \vee r && \text{Double Negation}
\end{aligned}$$

Notice how we reference a theorem in each step. This allows us to fully explain our equivalence, keeps us from making mistakes, and ensures our equivalence is valid.

Example: Simplify the following expression: $(p \otimes q) \wedge p$

Solution:

$$\begin{aligned}
(p \otimes q) \wedge p &\equiv \neg(p \oplus q) \wedge p && \text{XNOR equivalence}\\
&\equiv \neg\big((p \wedge (\neg q)) \vee ((\neg p) \wedge q)\big) \wedge p && \text{XOR equivalence}\\
&\equiv \big(\neg(p \wedge (\neg q)) \wedge \neg((\neg p) \wedge q)\big) \wedge p && \text{De Morgan's}\\
&\equiv \big(((\neg p) \vee \neg(\neg q)) \wedge (\neg(\neg p) \vee (\neg q))\big) \wedge p && \text{De Morgan's}\\
&\equiv \big(((\neg p) \vee q) \wedge (p \vee (\neg q))\big) \wedge p && \text{Double Negation}\\
&\equiv ((\neg p) \vee q) \wedge \big((p \vee (\neg q)) \wedge p\big) && \text{Associativity}\\
&\equiv ((\neg p) \vee q) \wedge \big(p \wedge (p \vee (\neg q))\big) && \text{Commutativity}\\
&\equiv ((\neg p) \vee q) \wedge p && \text{Absorption}\\
&\equiv p \wedge ((\neg p) \vee q) && \text{Commutativity}\\
&\equiv (p \wedge (\neg p)) \vee (p \wedge q) && \text{Distributivity}\\
&\equiv c \vee (p \wedge q) && \text{Negation}\\
&\equiv p \wedge q && \text{Identity}
\end{aligned}$$

Note in the preceding example that we used equivalences between the *xnor* to the *xor*, and the *xor* to the *or*. We will discuss later exactly how we deduced these equivalences.

1.2.3 Circuits

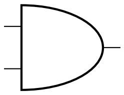
Propositions have two possible values: true and false. If we set true to mean *on* and false to mean *off*, then we can translate our Boolean statements into logical circuits. To do this, think of each logical connective as a *gate*. Similar to the logical connectives, a gate takes 2 (or more) inputs and returns some output. The inputs and outputs are all 1s and 0s (*on*s and *off* s – true and false). Circuits are the bare-bones to computers, so it is necessary you understand the basics. For computer engineers, you must know circuits by heart.

Definition 1.2.15 Circuit
Representations of Boolean statements into electronic components

Boolean variables can be exclusively either true or false; this is analogous to electric wires being exclusively either on or off. We let a tautology be equivalent to a *power source*, which is a wire that is always **on**. Similarly we let a contradiction be equivalent to **off**, or a wire receiving no power.

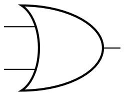
The following gates are exactly equivalent to their logic counterparts. We thus only include the gate picture.

Definition 1.2.16 And Gate – *conjunction*
Picture:

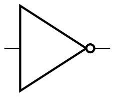


To remember the *and* gate, note that the picture looks like a **D**, which corresponds to the D in AN**D**

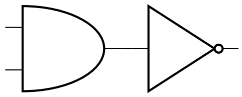
Definition 1.2.17 Or Gate – *Disjunction*
Picture:



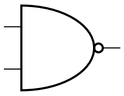
Definition 1.2.18 Not Gate – *Negation*
Picture:



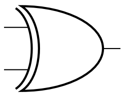
Remark 1.2.19
When we have a gate that has a *not* after it, we can simplify the gate by just adding a circle to the output. Example (the *nand* gate is the negation of the *and* gate):



becomes



Definition 1.2.20 Xor Gate (*eXclusive Or gate*)
Picture:



We can think of truth tables in an equivalent fashion, where 1 is true and 0 is false.

Example: Write the truth table for the *nand* gate.

Solution:

p	q	$\neg(p \wedge q)$
0	0	1
0	1	1
1	0	1
1	1	0

You can use 1/0 or T/F, whichever you prefer. If an assignment specifies you use a specific key, then use the one specified. Later when we discuss different number systems, you will notice a correlation between binary numbers and truth tables with 1/0s. This makes it easy to construct a truth table very quickly with a given amount of inputs.

We can also create circuits that add numbers. We will come back to this after our discussion of different number bases in section 3.2.8.

1.2.4 Translations

You must know how to translate between circuits, Boolean statements, and truth tables.

We start by motivating the translation between truth tables and Boolean statements. First, notice that we have already discussed how to make a truth table from a Boolean statement – simply draw the truth table and fill in each row. Now we can focus on the reverse.

We attempt to first motivate a process for this translation by showing a few examples.

Example: First recall the truth table for the conjunction:

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

Recall that we can extend the conjunction to take multiple inputs – in this case, *each* input *must* be **True** for the output to be **True**.
Now consider the following unknown table:

p	q	unknown
F	F	F
F	T	T
T	F	F
T	T	F

This table is similar to the conjunction table in that it has *one row* that outputs **True**. In the conjunction case, the assignments to p and q were both **True**. What are the assignments to the previous truth table? Well, the p input is **False** and the q input is **True**, as per the table. If we apply the conjunction to this row we have, we get the following table:

p	$\neg p$	q	$(\neg p) \land q$
F	T	F	F
F	T	T	T
T	F	F	F
T	F	T	F

In the previous table, if we look at the middle two columns, we recover the conjunction with the p variable negated.

This is only half the story though.

Example: First recall the truth table for the disjunction:

p	q	$p \lor q$
F	F	F
F	T	T
T	F	T
T	T	T

For this example, we only care about the middle two rows where *either* variable is **True**.

These bring light to a simple 3-step process to translate from truth tables to Boolean statements:

1. Collect the rows whose *output* is **True**
2. For each of those rows
   (a) Look at the truth value assigned to the *input*
   (b) Construct a Boolean statement where, for each input
      - If the assignment is **True** then use the variable itself
      - If the assignment is **False** then use the *negation* of the variable
   (c) Chain each input with **And** ($\land$) connectives
3. Chain each row's statement with **Or** ($\lor$) connectives

Example: Find the unknown Boolean formula corresponding to the following truth table:

p	q	unknown
F	F	T
F	T	F
T	F	T
T	T	F

Solution: We follow the algorithm as before. The first and third rows return true in the output. In the first row, we see neither input is true, so we AND the negation of each input: $(\neg p) \wedge (\neg q)$. In the third row, we see the first input is true, while the second input is false, thus we get: $p \wedge (\neg q)$. OR-ing each statement together thus yields our unknown formula:

$$((\neg p) \wedge (\neg q)) \vee (p \wedge (\neg q))$$

Remark 1.2.21

In our Boolean formula algorithm, we only focus on the true rows. If we wanted to also include the false rows, then we would need the statement in each false row to return false. We know how to get a statement that returns true, so we can simply take that statement and negate it! What happens, though, if we include the false rows then?

Example: Include the false rows in the Boolean formula from the previous example.

Solution: We negate the second and fourth row returned by the algorithm: $\neg((\neg p) \wedge q)$ and $\neg(p \wedge q)$

So our entire statement is now

$$((\neg p) \wedge (\neg q)) \vee (p \wedge (\neg q)) \vee (\neg((\neg p) \wedge q)) \vee (\neg(p \wedge q))$$

Let's simplify this statement (rules omitted):

$$\begin{aligned}
&\equiv ((\neg p) \wedge (\neg q)) \vee (p \wedge (\neg q)) \vee (\neg((\neg p) \wedge q)) \vee (\neg(p \wedge q)) \\
&\equiv ((\neg p) \wedge (\neg q)) \vee (p \wedge (\neg q)) \vee (p \vee (\neg q)) \vee ((\neg p) \vee q) \\
&\equiv ((\neg p) \wedge (\neg q)) \vee (p \wedge (\neg q)) \vee (p \vee (\neg p)) \vee ((\neg q) \vee q) \\
&\equiv ((\neg p) \wedge (\neg q)) \vee (p \wedge (\neg q)) \vee t \vee t \\
&\equiv ((\neg p) \wedge (\neg q)) \vee (p \wedge (\neg q))
\end{aligned}$$

Nice, we have recovered the original statement!

Remark 1.2.22
Including each false row in the Boolean statement generated by the algorithm – so long as their true versions are negated – does not logically change the Boolean statement

We have a name for the type of statement generated from our algorithm above:

Definition 1.2.23 Disjunctive Normal Form
Describes a Boolean statement that is a conjunction of disjunctions – abbreviated DNF

If we flip each gate (AND goes to OR, OR goes to AND), then we get another important type of statement:

Definition 1.2.24 Conjunctive Normal Form
Describes a Boolean statement that is a disjunction of conjunctions – abbreviated CNF

Interestingly, any Boolean statement can be translated to an equivalent statement in CNF. Namely, statements in DNF, which are easy to generate, can be translated into CNF. This is important for computational complexity theory – specifically NP-completeness. There exists a problem in computer science which entails finding a set of truth-value assignments for n different Boolean variables which makes a Boolean statement in CNF return true (or, become *satisfiable*).

1.2.5 Reasoning/Deductions

Propositional logic also allows us to *reason* about things.

Definition 1.2.25 Knowledge Base
A group of information that you know is true

Definition 1.2.26 Reasoning
The process of deriving new information from a given knowledge base

See the introduction of this chapter for an example.

Classical Rules of Deduction

We may refer to *deductions* as *inferences.* They are the same.

Definition 1.2.27 Deductions
Using previously-known knowledge in your knowledge base to obtain/create new knowledge

We have a whole list of useful deductions that are provably valid. As with our Boolean algebra theorems, you do not need to memorize these theorems – they will be given to you as a table.

Theorem 1.2.14 Modus Ponens

$$\begin{array}{rl} & p \\ & p \Rightarrow q \\ \hline \therefore & q \end{array}$$

Theorem 1.2.15 Modus Tollens

$$\begin{array}{rl} & \neg q \\ & p \Rightarrow q \\ \hline \therefore & \neg p \end{array}$$

Theorem 1.2.16 Disjunctive Addition

$$\begin{array}{rl} & p \\ \hline \therefore & p \vee q \end{array}$$

Theorem 1.2.17 Conjunctive Addition

$$\begin{array}{rl} & p, q \\ \hline \therefore & p \wedge q \end{array}$$

Theorem 1.2.18 Conjunctive Simplification

$$\begin{array}{rl} & p \wedge q \\ \hline \therefore & p, q \end{array}$$

Theorem 1.2.19 Disjunctive Syllogism

$$\begin{array}{rl} & p \vee q \\ & \neg p \\ \hline \therefore & q \end{array}$$

Theorem 1.2.20 Hypothetical Syllogism

$$\begin{array}{rl} & p \Rightarrow q \\ & q \Rightarrow r \\ \hline \therefore & p \Rightarrow r \end{array}$$

Theorem 1.2.21 Resolution

$$\begin{array}{rl} & p \vee q \\ & (\neg q) \vee r \\ \hline \therefore & p \vee r \end{array}$$

Theorem 1.2.22 Division Into Cases

$$\begin{array}{rl} & p \vee q \\ & p \Rightarrow r \\ & \underline{q \Rightarrow r} \\ \therefore & r \end{array}$$

Theorem 1.2.23 Law of Contradiction

$$\begin{array}{rl} & \underline{(\neg p) \Rightarrow \mathbf{c}} \\ \therefore & p \end{array}$$

You may be wondering how to prove these deductions are *valid.* We have two equivalent methods:

1. Tautological implication
2. Critical-row identification

Consider an arbitrary deduction:

$$\begin{array}{rl} & P_1 \\ & P_2 \\ & \cdots\cdots\cdots \\ & \underline{P_n \qquad\quad} \\ \therefore & Q \end{array}$$

To prove it valid,

Tautological implication

1. Construct a new proposition $A \Rightarrow B$ where A is a **conjunction of the premises** and B is the conclusion. For our arbitrary deduction, we would have
$$\big(P_1 \wedge P_2 \wedge \cdots \wedge P_n\big) \Rightarrow Q$$
2. Inspect this proposition in a truth-table. If the proposition is a tautology, then the deduction is valid. Otherwise, the deduction is invalid. So, to be valid we must have
$$(A \Rightarrow B) \equiv \big(P_1 \wedge P_2 \wedge \cdots \wedge P_n\big) \Rightarrow Q \equiv t$$

Critical-row identification

1. Construct a truth-table with columns for each proposition and for the conclusion. *You may need extra columns, that is fine.* For our arbitrary deduction, we would have

P_1	P_2	$\cdots$	P_n	Q
F/T	F/T	$\cdots$	F/T	F/T
		$\vdots$		
T/T	F/T	$\cdots$	F/T	F/T

2. Identify the rows in which each **premise** is true. We call these rows **critical-rows**.

3. For each critical-row, inspect the conclusion. If the conclusion is true in **every** critical-row, then the deduction is valid. Otherwise, the deduction is invalid.

Examples following the above steps.

Example: Show that Modus Ponens is a valid rule of inference.

Solution: *method 1 – tautological implication*
Our proposition we care about is $(p \wedge (p \Rightarrow q)) \Rightarrow q$, so we build the following truth-table with extraneous rows:

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

Inspect the last column.

p	q	p	$p \Rightarrow q$	$p \wedge (p \Rightarrow q)$	q	$(p \wedge (p \Rightarrow q)) \Rightarrow q$
F	F	F	T	F	F	**T**
F	T	F	T	F	T	**T**
T	F	T	F	F	F	**T**
T	T	T	T	T	T	**T**

In the case of Modus Ponens, the final column is a tautology, hence the deduction is valid.

Example: Show that Modus Ponens is a valid rule of inference.

Solution: *method 2 – critical-row identification*
Construct a truth table with each premise and conclusion:

p	q	p	$p \Rightarrow q$	q
F	F	F	T	F
F	T	F	T	T
T	F	T	F	F
T	T	T	T	T

Identify the critical-rows.

p	q	p	$p \Rightarrow q$	q
F	F	F	T	F
F	T	F	T	T
T	F	T	F	F
T	T	**T**	**T**	T

Inspect the conclusion in each critical-row.

p	q	p	$p \Rightarrow q$	q
F	F	F	T	F
F	T	F	T	T
T	F	T	F	F
T	T	**T**	**T**	T

In the case of Modus Ponens, the conclusion is true in each critical row, hence the deduction is valid.

We leave it to the reader to understand why the two methods are equivalent.

Deducing Things

In a later section, we will see that this order of logic is not powerful enough to prove mathematical statements. For now, we can still do interesting things with a given knowledge base.

Example: Given the following knowledge base, deduce as much new information as possible using the following rules of inference: Modus Ponens, Modus Tollens, Hypothetical Syllogism, and Disjunctive Syllogism.

$a \Rightarrow b \qquad b \Rightarrow (\neg d) \qquad e \qquad d \vee (\neg e)$

Solution:

By Hypothetical Syllogism $a \Rightarrow b$, $b \Rightarrow (\neg d)$, $\therefore a \Rightarrow (\neg d)$

By Disjunctive Syllogism $d \vee (\neg e)$, e, $\therefore d$

By Modus Tollens d, $a \Rightarrow (\neg d)$, $\therefore \neg a$

By Modus Tollens d, $b \Rightarrow (\neg d)$, $\therefore \neg b$

Remark 1.2.28
From the above example, we restricted the rules you could have used. We did this mainly because Disjunctive Addition can allow you to generate any new knowledge you like – so long as you have one thing that is true, then you can add in a disjunction infinitely-many times.

Remark 1.2.29
In the above example, we could have translated $d \vee (\neg e) \equiv e \Rightarrow d$ and concluded d by Modus Ponens. This somewhat tells you that Disjunctive Syllogism and Modus Ponens are equivalent.

Remark 1.2.30
From an inconsistent database, anything follows.

This is due to the law of contradiction. An *inconsistent database* is one that contains a contradiction. Recall from the law of contradiction that $(\neg p) \Rightarrow c \equiv (\neg(\neg p)) \vee c \equiv p$ using the Identity Boolean algebra theorem. Using Disjunctive Addition, we have the contradiction c, $\therefore A \vee c$, and by Identity, $\therefore A$. A can be *Anything*.

In Artificial Intelligence, there exists an algorithm called **The Resolution Algorithm**. Essentially, it says to take a given knowledge base, translate each statement into disjunctive normal form, then apply the Resolution rule of inference as many times as possible.

1.3 Predicate Logic

Sometimes basic propositions are not enough to do what you want. In programming we can have functions that return true or false. We can do the same thing with logic – we call this *first-order* logic, or *predicate* logic. Predicate logic includes all of propositional logic, however it adds predicates and quantifiers.

Definition 1.3.1 Predicate
A property that the subject of a statement can have. In logic, we represent this sort-of like a function. A predicate takes, as input, some element, and returns whether the inputted element has the specific property.

Example: We could use the predicate $EVEN(x)$ to mean x is an even number. In this case, the predicate is $EVEN(\cdot)$

Example: We could use the predicate $P(y)$ to mean y is an integer multiple of 3. In this case, the predicate is $P(\cdot)$

Remark 1.3.2
Predicates take **elements**. They do **not** take in other predicates. This is because predicates say whether the input element *has* the property specified by the predicate – true and false cannot have properties.

In terms of programming, you can think of a predicate as a program method. For example, the $EVEN(x)$ predicate might be implemented as follows:

```
func EVEN(Entity x) -> bool {
        if IS_INTEGER(x) {
                return x % 2 == 0
        }
        return false
}
```

In this case, entities are *objects* and true/false are *Boolean primitives* (or, propositional statements, which can only be true or false). In contrast to, say, Java, a compiler for this code would not allow *true/false* to be an object.

A better example,

```
class Foo extends Entity {
        bool isInteger
        bool isOdd

        Foo(Integer i) {
                this.isInteger = true
                this.isOdd = i % 2 == 1
        }
}

func ODD(Entity x) -> bool {
        if x has type Foo {
                return x.isOdd
        }
        if IS_INTEGER(x) {
                return x % 2 == 1
        }
        return false
}
```

Definition 1.3.3 Quantifier
A way to select a specific range of elements that get inputted to a predicate. We have two quantifiers:

- The Universal quantifier ∀
- The Existential quantifier ∃

The universal quantifier says to select **all** elements, and the existential quantifier says to select **at least one** element.

Definition 1.3.4 Quantified Statement
A logical statement involving predicates and quantifiers. Syntax:

$$(\text{quantifier } var \in D)[\text{statement involving predicates}]$$

And now we can define:

Definition 1.3.5 Predicate Logic
Also called *first-order logic*, is a logic made up of quantified statements.

Example: Translate the following statements to predicate logic:

1. All people are mortal
2. Even integers exist
3. If an integer is prime then it is not even

Solution:

1. Denote P as the domain of people, and the predicate $M(x)$ to mean x is mortal. Then the statement translates to
$$(\forall p \in P)[M(p)]$$
2. Denote $\mathbb{Z}$ as the domain of integers, and the predicate $EVEN(x)$ to mean x is even. Then the statement translates to
$$(\exists x \in \mathbb{Z})[EVEN(x)]$$
3. Denote the predicate $PRIME(y)$ to mean y is prime. Then the statement translates to
$$(\forall a \in \mathbb{Z})[PRIME(a) \Rightarrow \neg EVEN(a)]$$

1.3.1 Negating Quantified Statements

One may find useful to negate a given quantified statement. We present here how to do this, first with an English example, followed by a quantified example, followed by an algorithm.

Example: The following statement

There is no student who has taken calculus.

is equivalent to

All students have not taken calculus.

Example: The following statement

Not all students have taken calculus.

is equivalent to

There is a student who has not taken calculus.

Example: The following statement

$$\neg(\exists x \in D)[C(x)]$$

is equivalent to

$$(\forall x \in D)[\neg C(x)]$$

Example: The following statement

$$\neg(\forall x \in D)[C(x)]$$

is equivalent to

$$(\exists x \in D)[\neg C(x)]$$

The generic algorithm for pushing the negation into a quantified statement:

1. Flip each quantifier $\forall \to \exists$ and $\exists \to \forall$
2. Apply the negation to the propositional part of the quantified statement, and simplify
   (a) If the inside contains another quantified statement, then recursively apply this algorithm

Remark 1.3.6
The domain and variable attached to any quantifier are **not** changed.

Example: Push the negation in as far as possible:

$$\neg(\forall x, y \in \mathbb{Z})[(x < y) \Rightarrow (\exists m \in \mathbb{Q})[x < m < y]]$$

Solution:

$$\begin{aligned}
&\neg(\forall x, y \in \mathbb{Z})[(x < y) \Rightarrow (\exists m \in \mathbb{Q})[x < m < y]]\\
\equiv&(\exists x, y \in \mathbb{Z})\neg[(x < y) \Rightarrow (\exists m \in \mathbb{Q})[x < m < y]]\\
\equiv&(\exists x, y \in \mathbb{Z})\neg[\neg(x < y) \vee (\exists m \in \mathbb{Q})[x < m < y]]\\
\equiv&(\exists x, y \in \mathbb{Z})[\neg\neg(x < y) \wedge \neg(\exists m \in \mathbb{Q})[x < m < y]]\\
\equiv&(\exists x, y \in \mathbb{Z})[(x < y) \wedge (\forall m \in \mathbb{Q})\neg[x < m < y]]\\
\equiv&(\exists x, y \in \mathbb{Z})[(x < y) \wedge (\forall m \in \mathbb{Q})\neg[x < m \wedge m < y]]\\
\equiv&(\exists x, y \in \mathbb{Z})[(x < y) \wedge (\forall m \in \mathbb{Q})[x \geq m \vee m \geq y]]
\end{aligned}$$

Remark 1.3.7
We typically expect the final statement to contain no $\neg$ operators.

1.3.2 Quantified Rules of Inference

Again, *deductions* and *inferences* are the same. We present a handful of important *quantified* rules of inference.

Theorem 1.3.1 Universal Instantiation
For a predicate $P(\cdot)$ and some domain D with $c \in D$,

$$\begin{array}{rl} & (\forall x \in D)[P(x)] \\ \hline \therefore & P(c) \end{array}$$

As an example, if our domain consists of all dogs and Fido is a dog, then the above rule can be read as

"All dogs are cuddly"

"Therefore Fido is cuddly"

Theorem 1.3.2 Universal Generalization
For a predicate $P(\cdot)$ and some domain D for an arbitrary $c \in D$,

$$\begin{array}{rl} & P(c) \\ \hline \therefore & (\forall x \in D)[P(x)] \end{array}$$

This is most-often used in mathematics. As an example, if our domain consists of all dogs, then the above rule can be read as

"An arbitrary dog is cuddly" (which in-turn applies to all dogs)

"Therefore all dogs are cuddly"

Theorem 1.3.3 Existential Instantiation
For a predicate $P(\cdot)$ and some domain D for some element $c \in D$,

$$\begin{array}{rl} & (\exists x \in D)[P(x)] \\ \hline \therefore & P(c) \end{array}$$

As an example, if our domain consists of all dogs, then the above rule can be read as

"There is a dog who is cuddly"

"Let's call that dog c, and so c is cuddly"

Theorem 1.3.4 Existential Generalization
For a predicate $P(\cdot)$ and some domain D for some element $c \in D$,

$$\begin{array}{rl} & P(c) \\ \hline \therefore & (\exists x \in D)[P(x)] \end{array}$$

As an example, if our domain consists of all dogs and Fido is a dog, then the above rule can be read as

"Fido is cuddly"

"Therefore there is a dog who is cuddly"

Theorem 1.3.5 Universal Modus Ponens
For two predicates $P(\cdot)$ and $Q(\cdot)$, and some domain D with $a \in D$,

$$\begin{array}{rl} & P(a) \\ & (\forall x \in D)[P(x) \Rightarrow Q(x)] \\ \hline \therefore & Q(a) \end{array}$$

Theorem 1.3.6 Universal Modus Tollens
For two predicates $P(\cdot)$ and $Q(\cdot)$, and some domain D with $a \in D$,

$$\begin{array}{rl} & \neg Q(a) \\ & (\forall x \in D)[P(x) \Rightarrow Q(x)] \\ \hline \therefore & \neg P(a) \end{array}$$

1.3.3 Proving Things

Our familiar rules of inference are not strong enough to prove abstract mathematical statements. Typically we want our proof to apply to a whole *set* of things (numbers). Now that we know about *predicate logic*, we can apply our more powerful *quantified* rules of inference to prove real mathematical statements.

Example: Using Universal Modus Ponens, verify the validity of the following proof:

Proof. Let $m, n \in \mathbb{Z}$, and let m be even. Then $m = 2p$ for some integer p.(1) Now,

$$\begin{aligned} m \cdot n &= (2p)n && \text{by substitution} \\ &= 2(pn)^{(2)} && \text{by associativity} \end{aligned}$$

Now, $pn \in \mathbb{Z}$,(3) so by definition of even $2(pn)$ is even.(4) Thus mn is even. □

Solution:

(1) If an integer is even, then it equals twice some integer.
m is a particular integer, and it is even.
$\therefore$ m equals twice some integer p.

(2) If a quantity is an integer, then it is a real number.
p and n are both particular integers.
$\therefore$ p and n are both real numbers.

For all a, b, c, if $a, b, c \in \mathbb{R}$ then $(ab)c = a(bc)$.
2, p, and n are all particular real numbers.
$\therefore$ $(2p)n = 2(pn)$.

(3) For all u, v, if $u, v \in \mathbb{Z}$ then $uv \in \mathbb{Z}$.
p and n are both particular integers.
$\therefore$ $pn \in \mathbb{Z}$.

(4) If a number equals twice some integer, then that number is even.
$2(pn)$ equals twice the integer pn.
$\therefore$ $2(pn)$ is even.

Of course, we would never do a mathematical proof like this. In reality, you do this in your head automatically. Seeing this form, however, allows you to easily verify the **validity** of the proof.

1.4 Summary

- Propositional logic contains the entirety of Boolean algebra and logic connectives, with True and False as the only inputs/outputs
- Predicate logic contains the entirety of propositional logic and uses func-

tions along with entities

- Deriving knowledge from familiar rules entails mathematical proof

1.5 Practice

1. Answer the two logic puzzles presented in the introduction of this chapter.
2. Translate the following statement into propositional logic: *turn right then turn left*.
3. Translate the following statement into propositional logic: *if it is raining then everyone has an umbrella*.
4. How can you quickly construct a truth table with all row possibilities? Use your technique to construct a truth table with 4 variables.
5. How many rows does a truth table with n variables have?
6. Prove theorem 1.2.12.
7. Prove theorem 1.2.13.
8. Draw the circuit representation of theorem 1.2.12.
9. Prove the following rule valid or invalid:

$$\begin{array}{rl} & (a \wedge d) \Rightarrow b \\ & e \\ & b \Rightarrow (\neg e) \\ & (\neg a) \Rightarrow f \\ & (\neg d) \Rightarrow f \\ \hline \therefore & f \end{array}$$

10. Prove the following rule valid or invalid:

$$\begin{array}{rl} & (a \wedge d) \Rightarrow b \\ & e \\ & b \Rightarrow (\neg e) \\ & (\neg a) \Rightarrow f \\ & (\neg d) \Rightarrow f \\ \hline \therefore & b \Rightarrow e \end{array}$$

11. Push the negation inside the following statement as far as possible:

$$\neg(\forall x \in \mathbb{R})(\exists m \in \mathbb{Z})[(0 \leq x - m < 1) \Leftrightarrow (m = \lfloor x \rfloor)]$$

Chapter 2

Set Theory

Sets are wild fam.

Justin Goodman

Contents

2.1 Introduction

Sets were introduced by Georg Cantor in the late 1800s. Cantor is the grandfather of set theory and continuity. We will see continuity topics later, including Cantor's famous diagonal argument. We can define all of discrete mathematics using sets. What is a set though?

Definition 2.1.1 Set
An unordered collection of unique objects. We denote sets using curly braces $\{\}$ with objects appearing in them – e.g. $\{\circ, 3, \pi, \blacktriangle\}$

Each part of this definition is important – we do not have a set unless it satisfies the entire definition. We now examine the definition.

- A set is a **collection** of stuff. Think of a set like a box. You can put things in your box, and you can take them out. Your box is special – it can expand/contract to fit anything you like.
- A set is **unordered**. This simply means that any different ordering we give to a set does not change the equality property of the set – $\{1, 2\} = \{2, 1\}$
- A set contains **objects**. A set can contain anything you want.
- A set contains **unique** objects. For any two distinct objects in a set, the objects cannot be equal. Sometimes we see books describe the sets $\{1, 1, 2, 3\}$ and $\{1, 2, 3\}$ as equal, however we argue that the first set is not even a set![1] We recommend you ask your instructor about this distinction, and follow what they prefer.

This chapter includes an overview of set theory – sets, operations, binary relations, and theorems.

2.2 Building Sets

Before we dive into set theory concepts, we introduce set-builder notation. As the name implies, this is how we will build our sets.

Definition 2.2.1 Set-Builder Notation
A way of defining sets. Syntax: $\{\,\text{element} \mid \text{condition(s)}\,\}$, read as, “element *such that* condition(s) is (are) satisfied.” For example, the set of even integers (which we will learn about soon) can be written as: $\{e \mid (\exists k \in \mathbb{Z})[e = 2k]\}$

It is pivotal that you know how to read and create sets this way. The syntax is very flexible because the conditions can be almost anything you like. The conditions should, however, relate to the set itself (otherwise the set would be trivially pointless).

Example: Build a set that contains all square roots of even integers (for now, use the definition of even integers given previously).

Solution:

$$S = \{x \mid (\exists k \in \mathbb{Z})[x^2 = 2k]\}$$

[1] We would classify the set $\{1, 1, 2, 3\}$ as a *multi-set*

Example: Build a set that contains everything except for the object $\star$.

Solution:

$$S = \{y \mid y \neq \star\}$$

2.3 Definitions

We include a handful of definitions and notations we use in our study of set theory.

Binary relations:

Definition 2.3.1 Member/Element
An object that is part of a set. Symbol: $\in$. Example: $5 \in \{1, 2, 3, 4, 5\}$

Definition 2.3.2 Subset
A set of elements that are all members of another set. A subset CAN be equal to its parent set. Symbol: $\subseteq$. Example: for sets S and T, $S \subseteq T \Leftrightarrow (\forall s \in S)[s \in T]$

Definition 2.3.3 Proper Subset
A set of elements that are all members of another set, but the subset is NOT equal to the parent set. Symbol: $\subset$. Example: for sets S and T, $S \subset T \Leftrightarrow (\forall s \in S)[(s \in T) \wedge (S \neq T)]$

Definition 2.3.4 Superset
A flipped version of subset. Symbol: $\supseteq$. Example: for sets S and T, $S \supseteq T \Leftrightarrow (\forall t \in T)[t \in S]$

Definition 2.3.5 Proper Superset
A flipped version of proper subset. Symbol: $\supset$. Example: for sets S and T, $S \supset T \Leftrightarrow (\forall t \in T)[(t \in S) \wedge (S \neq T)]$

Definition 2.3.6 Equality
Two sets are equal if and only if both sets are subsets of each other. Notation: for sets S and T, $S = T \Leftrightarrow (S \subseteq T) \wedge (T \subseteq S)$. To prove this, you can use set-builder notation & theorems (given later in this chapter) to show equivalence, or you can prove $(e \in S \Rightarrow e \in T) \wedge (e \in T \Rightarrow e \in S)$

Things:

Definition 2.3.7 Cardinality
The number of elements in a set. Notation: for a set S, cardinality is denoted by $|S|$. For example, $|\{2, 3, 4, 5, 6\}| = 5$

Definition 2.3.8 Empty/Null Set
The set containing zero elements. Notation: $\emptyset$ or $\{\}$ – these symbols are *interchangable*. **Note**: $|\emptyset| = 0$

Definition 2.3.9 Universal Set
The set of all possible sets. Notation: U is the Universal Set

Definition 2.3.10 Power Set
The set of all possible subsets of a set. Notation: of a set S, the power set is denoted $\mathcal{P}(S)$. For example, $\mathcal{P}(\{1,2,3\}) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$. **Note**: for any set S, $|\mathcal{P}(S)| = 2^{|S|}$

Definition 2.3.11 Disjoint
Two sets are disjoint if and only if both sets have no members in common. For two sets S and T, this is equivalent to $S \cap T = \emptyset$

Definition 2.3.12 Partition
(of a set) A set of sets T where the union of every element in T equals the original set, and all elements in T are disjoint. For example, $\{\{1,2\}, \{3\}, \{4,5,6\}\}$ is a partition of $\{1,2,3,4,5,6\}$

Operations:

Definition 2.3.13 Union
(of two sets) A set that includes all elements from both sets (discounting duplicates, since a set must contain unique elements). Notation: for sets S and T, $S \cup T = \{e \mid (e \in S) \vee (e \in T)\}$. For example, $\{1,2,3\} \cup \{3,4,5\} = \{1,2,3,4,5\}$

Definition 2.3.14 Intersection
(of two sets) A set that includes only elements from both sets. Notation: for sets S and T, $S \cap T = \{e \mid (e \in S) \wedge (e \in T)\}$. For example, $\{1,2,3\} \cap \{3,4,5\} = \{3\}$

Definition 2.3.15 Subtraction
(of two sets) A set representing the elements of the second set taken out of the first set. **Note**: subtraction order *matters* (i.e. subtraction is not commutative). Notation: for sets S and T, $S - T = \{e \mid (e \in S) \wedge (e \notin T)\}$. You may also see set subtraction represented as $S \setminus T$. For example, $\{1,2,3\} \setminus \{3,4,5\} = \{1,2\}$

Definition 2.3.16 Compliment
(of a set) A set containing every element from the universal set that is NOT contained in the original set. **Note**: you can take the compliment with respect to different universes so long as you specify which one – by default the universal set is assumed. Notation: for a set S, the complement is noted as S^{c}, or $S^{'}$, or $\overline{S}$; with the universe U, $S^{'} = \{e \mid e \in (U - S)\}$

Definition 2.3.17 Cross Product
(of two sets) The set of all ordered pairings of two sets. Notation: for sets S and T, $S \times T = \{(s,t) \mid s \in S \wedge t \in T\}$.

Note: you can take the cross product of multiple sets. For sets $A_1 \cdots A_n$, the cross product $A_1 \times A_2 \times \cdots \times A_n = \{(a_1, a_2, \cdots, a_n) \mid a_i \in A_i\}$. When $A_1 = \cdots = A_n = A$ then we let $A^n = A_1 \times A_2 \times \cdots \times A_n$

2.4 Theorems

We will not go into depth on the axioms of Zermelo–Fraenkel set theory. We do include a handful of nice theorems that will aid in proving statements about sets. Sometimes these are referred to as axioms, however we argue that the following statements can be derived from ZF set theory axioms and should hence be called theorems. It does not really matter though.

You are not required to memorize these theorems – they will be given to you as a table.

Theorem 2.4.1 Commutativity
For any sets A and B the union and intersection operations are commutative:

$$A \cup B = B \cup A$$

$$A \cap B = B \cap A$$

Theorem 2.4.2 Associativity
For any sets A, B, and C the union and intersection operations are associative:

$$(A \cup B) \cup C = A \cup (B \cup C)$$

$$(A \cap B) \cap C = A \cap (B \cap C)$$

Theorem 2.4.3 Distributivity
For any sets A, B, and C the union and intersection operations are distributive:

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$

Theorem 2.4.4 Identity
For any set A and universe U the following hold:

$$A \cup \emptyset = A$$

$$A \cap U = A$$

Theorem 2.4.5 Inverse
For any set A and universe U the following hold:

$$A \cup A^{\mathsf{c}} = U$$

$$A \cap A^{\mathsf{c}} = \emptyset$$

Theorem 2.4.6 Double Compliment
For any set A the following holds:

$$(A^{\mathsf{c}})^{\mathsf{c}} = A$$

Theorem 2.4.7 Idempotence
For any set A the following hold:

$$A \cup A = A$$

$$A \cap A = A$$

Theorem 2.4.8 De Morgan's
For any sets A and B the following hold:

$$(A \cup B)^{\mathsf{c}} = A^{\mathsf{c}} \cap B^{\mathsf{c}}$$

$$(A \cap B)^{\mathsf{c}} = A^{\mathsf{c}} \cup B^{\mathsf{c}}$$

Theorem 2.4.9 Universal Bound (Domination)
For any set A and universe U the following hold:

$$A \cup U = U$$

$$A \cap \emptyset = \emptyset$$

Theorem 2.4.10 Absorption
For any sets A and B the following hold:

$$A \cup (A \cap B) = A$$

$$A \cap (A \cup B) = A$$

Theorem 2.4.11 Absolute Compliment
For a given universe U the following hold:

$$\emptyset^{\mathsf{c}} = U$$

$$U^{\mathsf{c}} = \emptyset$$

Theorem 2.4.12 Set Subtraction Equality
For any sets A and B the set subtraction operation satisfies the following:

$$A - B = A \cap B^{\mathsf{c}}$$

This establishes a relationship between the relative and absolute compliment

The aforementioned theorems are helpful for simplifying complicated sets.

Example: Simplify the following expression:

$$((A \cup B) \cap C) \cup ((A^{\mathsf{c}} \cap B^{\mathsf{c}}) \cup D^{\mathsf{c}})^{\mathsf{c}}$$

Solution: Lots of compliments is a good indication for using De Morgan's.

$$\begin{aligned}
&((A \cup B) \cap C) \cup ((A^{\mathsf{c}} \cap B^{\mathsf{c}}) \cup D^{\mathsf{c}})^{\mathsf{c}} & \\
&= ((A \cup B) \cap C) \cup ((A \cup B)^{\mathsf{c}} \cup D^{\mathsf{c}})^{\mathsf{c}} & \text{De Morgan's} \\
&= ((A \cup B) \cap C) \cup ((A \cup B) \cap D) & \text{De Morgan's} \\
&= (A \cup B) \cap (C \cup D) & \text{Distributivity}
\end{aligned}$$

Example: Show the following equivalence:

$$A \cup (B \cup (A \cap C)) = A \cup B$$

Solution: Sometimes just trying random things works out in your favor.

$$\begin{aligned}
A \cup (B \cup (A \cap C)) &= A \cup ((A \cap C) \cup B) & \text{Commutativity} \\
&= (A \cup (A \cap C)) \cup B & \text{Associativity} \\
&= A \cup B & \text{Absorption}
\end{aligned}$$

Remark 2.4.1
There is a stark similarity between set and Boolean simplification.

2.5 Important Sets

We introduce some notation for a handful of important sets.

Definition 2.5.1 The Natural Numbers – $\mathbb{N}$
The standard discrete numbers with which you count. Some math classes start the naturals at 1, however in computer science we start the naturals at 0. Just remember simply that arrays utilize 0-indexing, so we do the same. The set looks like so: $\{0, 1, 2, 3, 4, 5, \cdots\}$

In mathematics, we define the natural numbers *inductively* – we will discuss induction in a few chapters. We introduce the inductive definition here. You do not need to know this, however we think it is interesting.

Proposition 2.5.1 Inductive Definition of $\mathbb{N}$
Define a set $S \subseteq \mathbb{R}$ to be inductive if and only if the following conditions hold:

- $0 \in S$

- *if* $x \in S$ *then* $x + 1 \in S$

Define $\mathbb{N}$ *as the intersection of all inductive sets.*

Understandably you may be confused on the notation of $\mathbb{R}$ – we will come back to this in a moment. For now, we continue to the integers.

Definition 2.5.2 The Integers – $\mathbb{Z}$
The standard *signed* discrete numbers with which you count. The integers include all of the natural numbers as well as all of their negatives[2]. The set looks like so: $\{\cdots, -3, -2, -1, 0, 1, 2, 3, \cdots\}$

The next set, which you may be familiar with, is the rationals.

Definition 2.5.3 The Rationals – $\mathbb{Q}$
The set of numbers that can be written as a ratio ($\mathbb{Q}$uotient) of integers. $\mathbb{Q} = \{x = \frac{a}{b} \mid a \in \mathbb{Z} \wedge b \in \mathbb{Z}^{\neq 0}\}$

Naturally we can define somewhat of an 'opposite' to the rationals.

Definition 2.5.4 The Irrationals – $\mathbb{R} \setminus \mathbb{Q}$
The set of real numbers that do not satisfy the rational property. A provable example is that $\sqrt{2} \in \mathbb{R} \setminus \mathbb{Q}$

The real numbers are not the main focus in discrete mathematics, however we still provide a definition.

Definition 2.5.5 The Reals – $\mathbb{R}$
The continuous interval $(-\infty, \infty)$. Any number that does not have the form $a + bi$, where $i = \sqrt{-1}$

In an analytical mathematics course, the reals and irrationals are more strongly defined. The above definitions are enough for this course.

2.6 Sets to Logic

We can use set-builder notation along with our familiar logic rules to prove things about sets. This proof technique can be used to prove the theorems we showed earlier.

Example: Prove theorem 2.4.12:

$$A - B = A \cap B^{\mathsf{c}}$$

[2] 0 is neither positive nor negative, so we cannot take its negation, however we let $0 \in \mathbb{Z}$

Proof.

$$\begin{aligned} A - B &= \{x \mid x \in A \land x \notin B\} && \text{defn of subtraction} \\ &= \{x \mid x \in A \land x \in B^{\mathsf{c}}\} && \text{defn of compliment} \\ &= A \cap B^{\mathsf{c}} && \text{defn of intersection} \end{aligned}$$

□

Example: Prove example 2.4

$$A \cup (B \cup (A \cap C)) = A \cup B$$

Proof.

$$\begin{aligned} &A \cup (B \cup (A \cap C)) \\ &= \{x \mid x \in A \lor (x \in B \lor (x \in A \land x \in C))\} && \text{defn of } \cup / \cap \\ &= \{x \mid x \in A \lor ((x \in A \land x \in C) \lor x \in B)\} && \text{Commutativity (logic)} \\ &= \{x \mid (x \in A \lor (x \in A \land x \in C)) \lor x \in B\} && \text{Associativity (logic)} \\ &= \{x \mid x \in A \lor x \in B\} && \text{Absorption (logic)} \\ &= A \cup B && \text{defn of } \cup \end{aligned}$$

□

2.7 Summary

- Sets are unordered collections of unique objects
- Many operations and theorems exist in set theory
- $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R} \setminus \mathbb{Q}, \mathbb{R}$ are all important sets

2.8 Practice

1. Provide an example of a valid set, and an invalid set.
2. Build a set of natural numbers that are all multiples of 3.
3. Explain why the cardinality of a finite set is a natural number.

4. For two sets A and B, is it necessarily the case that $|A \cup B| = |A| + |B|$?
5. Explain intuitively why $|\mathcal{P}(A)| = 2^{|A|}$.
6. Is $\{\mathbb{Z}^-, 0, \mathbb{Z}^+\}$ a valid partition of $\mathbb{Z}$? Explain. If it is not, change the partition to make it valid.
7. Is $\{\{1, 3, 5, 7, 9, \cdots\}, \{0, 2, 4, 6, 8, \cdots\}\}$ a valid partition of $\mathbb{N}$? Explain. If it is not, change the partition to make it valid.
8. Provide a set T which is a valid partition of $\mathbb{N}$ such that $|T| = 5$.
9. For each of the following, answer true or false:
   (a) $\mathbb{Z} \supset \mathbb{N}$
   (b) $\mathbb{N} \in \mathbb{Z}$
   (c) $0 \in \emptyset$
   (d) $\emptyset \in \emptyset$
   (e) $\emptyset \subseteq \emptyset$
   (f) $\emptyset \subset \emptyset$
   (g) $\mathcal{P}(\emptyset) = \emptyset$
10. Simplify the expression $A^{\mathsf{c}} \cup (B \cup A)^{\mathsf{c}}$.
11. Prove the two Absorption theorem statements are equivalent.

Chapter 3

Number Theory

> Time to get number *freaky*.
>
> Justin Goodman

Contents

3.1 Introduction

Definition 3.1.1 Number Theory
The mathematical study of integers and their properties

This chapter is intended to introduce you to the most important tool in number theory – proofs. We will further explain how to execute a proof, however we must begin with basic definitions. It is important to see what is involved in a proof, so we will show you some proofs while introducing the basic principles.

3.2 Basic Principles

3.2.1 Summations and Products

We first introduce a form of notation that helps us simplify and express long equations.

Definition 3.2.1 Summation
The sum over a sequence of numbers. Example:

$$\sum_{\substack{i=0\\ i \text{ even}}}^{3} \frac{i+2}{i+1} = \frac{0+2}{0+1} + \frac{2+2}{2+1} + \frac{4+2}{4+1} + \frac{6+2}{6+1}$$

Sometimes there is a condition (placed under the sum), which means you only evaluate the summation when the condition is *satisfied*. Example:

$$\sum_{\substack{i=0\\ i \text{ even}}}^{6} \frac{i+2}{i+1} = \frac{0+2}{0+1} + \frac{2+2}{2+1} + \frac{4+2}{4+1} + \frac{6+2}{6+1}$$

The empty sum sums to zero and is denoted, for $a > b$:

$$\sum_{i=a}^{b} a_i = 0$$

Definition 3.2.2 Product
The product over a sequence of numbers. Example:

Again, sometimes there is a condition (placed under the product), the same rules apply. Example:

$$\prod_{\substack{i=1\\ i \text{ odd}}}^{8} i^2 = 1^2 \times 3^2 \times 5^2 \times 7^2$$

The empty product multiplies to 1 and is denoted, for $a > b$:

$$\prod_{i=a}^{b} a_i = 1$$

Remark 3.2.3
We can think of the indices in a sum or product as being elements of a set. Since the $+$ operator is commutative, then the order that we evaluate the sum with respect to the indices *does not matter*. For example, the sum $\sum_{i=2}^{6} i$ iterates over the indices $i \in \{2, 3, 4, 5, 6\}$, and equals $2 + 3 + 4 + 5 + 6$ which can be re-ordered however we like (eg $4 + 5 + 2 + 3 + 6$). The condition placed on a sum or product simply becomes a restriction on the set. For example, the sum $\sum_{\substack{i=2 \\ i \equiv 0 \pmod 3}}^{6} i$ iterates over the indices $\{i \in \{2, 3, 4, 5, 6\} \mid i \equiv 0 \pmod 3\}$.

Abstractly, we can write the sum (and similarly, the product) as a sum over elements of a set. We write this as follows:

$$\sum_{\substack{i=a \\ P(i)}}^{b} a_i = \sum_{i \in I} a_i$$

where

$$I = \{i \in \mathbb{Z} \mid (a \leq i \leq b) \wedge P(i)\}$$

and $P(i)$ is the condition placed on the sum (or product).

Interestingly, the condition placed on the summation (or product) can depend on the actual *values* in the sum (or product). We could represent this as $P(i) \equiv Q(a_i)$.

Another interesting point, if we ever have $a > b$ then the condition $a \leq i \leq b$ is never satisfied, which means $I = \emptyset$. We also have that if $P(i)$ is false for all values of i such that $a \leq i \leq b$, then again $I = \emptyset$. If this is the case, then we sum (or product) over no indices. This is where the term **empty sum (or product)** comes from.

The empty sum is equal to the additive identity 0, and the empty product is equal to the multiplicative identity 1 (as we have seen before).

Example: Re-write the following sum as a summation, then evaluate it for $n = 2$:

$$\frac{1}{1} + \frac{2}{2} + \frac{3}{4} + \frac{4}{8} + \frac{5}{16} + \cdots + \frac{n+1}{2^n}$$

Solution: The sum equals

$$\sum_{i=0}^{n} \frac{i+1}{2^i}$$

and for $n = 2$ evaluates to

$$\frac{1}{1} + \frac{2}{2} + \frac{3}{4} = 2.75$$

3.2.2 Sequences and Series

We discuss sequences and series in a later chapter, however for now we introduce the basic ideas.

Definition 3.2.4 Sequence
An ordered list of numbers, each one associated with a specific *position* (index)

Example: The following is a sequence:

$$1, 2, 3, 4, 5, 6, \cdots$$

We can also write this surrounded by curly braces:

$$\{1, 2, 3, 4, 5, 6, \cdots\}$$

We can also give this sequence as a function, which takes indices as the input:

$$f(i) = i$$

We can also denote the sequence with a name:

$$a_i = i$$

Definition 3.2.5 Series
The sum of all terms in a sequence. Typically a series is infinite, however it can be finite.

Example: Let the sequence $a_i = 2i$. Then we denote the infinite series:

$$\sum_{i=1}^{\infty} a_i$$

Methods from calculus can help us evaluate and understand infinite series, however that is out of scope of this course.
If we give an upper bound, like $n = 4$, then we can denote and evaluate finite series (which just becomes a summation):

$$\sum_{i=1}^{4} a_i = 2(1) + 2(2) + 2(3) + 2(4) = 20$$

3.2.3 Floors and Ceilings

3.2.4 Closure

Closure has two parts: a **set**, and an **operation**. Closure is a *property* of a set paired with the given operation. The basic idea is that if you take **any** two elements (or however are required for the operation – typically two) and apply the given operation to those elements, then the result will be an element also in the set. Some examples will be helpful to illustrate the point here.

Definition 3.2.6 Closure
For a given set S and operation $\circ$, S is *closed under* $\circ$ if and only if

$$(\forall a, b \in S)[a \circ b \in S]$$

Note, the given operator could be unary, binary, trinary, etc, and the amount of variables pulled from the set correspond accordingly.

Theorem 3.2.1
The natural numbers $\mathbb{N}$ *are closed under addition. Formally,*

$$(\forall n, m \in \mathbb{N})[n + m \in \mathbb{N}]$$

Example: Try and think of some natural numbers where if you add them you *do not* get a natural number. If you came up with an answer, it would be a *counterexample* to the theorem.
You should not be able to come up with a counterexample.

Remark 3.2.7
For this book, we will accept the above theorem as a fact, since the proof relies on proposition 2.5.1 (the inductive definition of $\mathbb{N}$) which we are not requiring you to know. The proof is relatively straightforward though, so we include it in a few sections.

Okay, onto more examples of closure.

Example: The natural numbers are closed under multiplication. Formally,

$$(\forall n, m \in \mathbb{N})[n \cdot m \in \mathbb{N}]$$

Proof. Informal sketch: since m is a discrete counting number, we can think of $n \cdot m = \underbrace{n + \cdots + n}_{m \text{ times}}$. Since $\mathbb{N}$ is closed under addition, and multiplication here is just repeated addition, we are done. □

Example: The natural numbers are **not** closed under subtraction. Formally,

$$(\exists n, m \in \mathbb{N})[n - m \notin \mathbb{N}]$$

Proof. Take any natural numbers n, m such that $n < m$. Then $n < m \Rightarrow n - m < 0$ and we get a negative number, which is by definition not a natural number. □

Remark 3.2.8
This is somewhat rigorous. All we need here is a counterexample to $(\forall n, m \in \mathbb{N})[n - m \in \mathbb{N}]$. So, just take any numbers that disprove the statement: $5 - 6 = -1 \notin \mathbb{N}$, done!

Theorem 3.2.2

$$(\forall a, b \in \mathbb{Z})[a + b \in \mathbb{Z}]$$

Remark 3.2.9
For the scope of this book, we can take it by definition that integers are closed under addition (the previous theorem).

Example: Prove or disprove

$$(\forall a, b \in \mathbb{Z})[a \div b \in \mathbb{Z}]$$

Proof. Consider $1, 4 \in \mathbb{Z}$ but $1 \div 4 = 0.25 \notin \mathbb{Z}$, thus we have dis-proven the statement. □

Example: Let $S = \{1, 2, 3, 4\}$. Show that S is not closed under addition.

Proof. Well, $3, 4 \in S$ but $3 + 4 = 7 \notin S$. Another counterexample: $4 + 4 = 8 \notin S$ (this is just to illustrate that you can pull **the same element** from the set since closure applies to **all** elements). □

Closure can be of **any** set and **any** operation. We can get fancy with this, as with the previous example and the next example.

Example: Let $S = \{a, b, c, d\}$. Define the following table as the outcome of performing a $\blacklozenge$ operation:

	a	b	c	d
a	d	c	b	a
b	b	d	a	c
c	c	a	d	b
d	a	b	c	d

For example, $a \blacklozenge d = a$ (the first operand comes from the row, and the second operand comes from the column).
Is S closed under $\blacklozenge$? Justify.

Proof. S is closed under $\blacklozenge$ since every possible outcome (as listed in the table) of the $\blacklozenge$ operation results in an element in S. □

Finally, we present an example of a unary operator.

Example: Show that the integers are closed under taking the negative (multiplying by -1),

$$(\forall x \in \mathbb{Z})[-x \in \mathbb{Z}]$$

Proof. Consider that $-1 \in \mathbb{Z}$ and integers are closed under multiplication.
□

3.2.5 Parity

We introduce the idea with the definition.

Definition 3.2.10 Parity
An inherit property of the integers which says that *every* integer is *exclusively* either **even** or **odd** (note the *exclusive or* – an integer must be even or odd, however it cannot be both and it cannot be neither)

This is an important property and will be useful for later proofs. Imagine you want to show some statement is true for *all* integers. Sometimes it is easier to break the statement into two cases – odd integers and even integers – and prove both cases separately. If both cases turn out to be true, then you can use parity to conclude the original statement is true.

We will show this idea in later proofs, however we should first explain what is even and odd.

Definition 3.2.11 Even
An integer n is even if and only if $(\exists k \in \mathbb{Z})[n = 2k]$

Definition 3.2.12 Odd
An integer m is odd if and only if $(\exists l \in \mathbb{Z})[m = 2l + 1]$

By *parity*, we can reformat the definitions of even and odd to depend on each other. The following statement is true:

An integer is even if and only if it is **not** odd

Equivalently:

An integer is odd if and only if it is **not** even

These statements both require that you know one of the two definitions. If this was not true, then the statements would be circular and would be useless.

We will come back to these statements soon.

3.2.6 Divisibility

Previously we showed that the integers are **not** closed under division. However, some integers when divided output an integer, like 3/1 and 4/2. It thus might make sense for us to study division.

Definition 3.2.13 Divisibility
The property that a number can be evenly divided (with no remainder). a is divisible by b if and only if $a \div b \in \mathbb{Z}$. Note that $a \div b = \frac{a}{b}$.

We say that b **divides** a if and only if a **is divisible by** b, and we note this as $b \mid a$. We will always use this notation.

$$b \mid a \Leftrightarrow \frac{a}{b} \in \mathbb{Z} \Leftrightarrow (\exists x \in \mathbb{Z})[a = bx]$$

Remark 3.2.14
We like to think of $b \mid a \Leftrightarrow \frac{a}{b}$ as a rotation counterclockwise:

$$\overset{\curvearrowleft}{b \mid a} \;\mapsto\; \frac{a}{b}$$

Definition 3.2.15 Remainder
An alternate way to represent (integer) division. When $b \nmid a$ then $\exists r \in \mathbb{Z}$ where $0 < r < a$ is the *remainder* and $b \mid (a - r)$. Sometimes we write the division as qRr – this is just the integer part of the division $q = (a - r)/b$ appended with R appended with the remainder

Theorem 3.2.3
For any $a, b \in \mathbb{Z}^+$, if $b \mid a$ then $b \leq a$

Proof. $b \mid a$ so by definition $(\exists k \in \mathbb{Z})[\frac{a}{b} = k]$. Since $a, b > 0$ we conclude that $k > 0$. Since $\frac{a}{b} = k$ then $a = bk$. Since $b, k > 0$ we conclude that $bk \geq b$ and thus $a = bk \geq b \Rightarrow b \leq a$ □

These definitions are important, however soon we will see that they fit into a wider scope of number theory. First, some examples.

Example: Give all natural numbers that divide 6.

Solution: We solve this by checking numbers. We do not want to check **all** numbers, though. Fortunately, we only have to check natural numbers, and the previous theorem tells us we only have to check numbers ≤ 6. Thus, we only need to check $0, 1, 2, 3, 4, 5, 6$.

- We cannot divide by 0
- 1 divides everything
- 6 is even so $2 \mid 6$
- $\frac{6}{3} = 2 \in \mathbb{N}$
- $\frac{6}{4} = 1.5 \notin \mathbb{N}$
- $\frac{6}{5} = 1.2 \notin \mathbb{N}$
- Any integer divides itself, so $6 \mid 6$

Thus $1, 2, 3, 6$ divide 6.

Example: Find the remainder of 10 divided by 4.

Solution: We can repeatedly take away 4 from 11 until we're left with r such that $0 < r < 4$.
$11 - 4 = 7$
$7 - 4 = 3$
We did this process 2 times, so $\frac{11}{4} = 2R3$. Also notice that $\frac{11-3}{4} = 8/4 = 2$

3.2.7 Modular Arithmetic

Modular arithmetic gives us a unified way to think about division between integers. Recall our definition of a remainder – this is essentially what a modulus is, but with an important difference. We show some notation first, then the definitions.

$$a \equiv b \pmod{m}$$

a, b, m are all integers here, with $m > 0$. Notice that we use the $\equiv$ symbol, which we have seen is used for *logical equivalence*. We will come back to this.

You may have seen modular arithmetic previously in a programming lens. In this case, the previous notation may be translated to the following.

$$a \,\%\, m = b$$

Still a, b, m are all integers but here we must have $b < m$. This is exactly taking the remainder.

Here is the subtle difference – in modular arithmetic, b **is not necessarily** less than m.

Definition 3.2.16 Congruence
Two integers a and b are congruent with respect to an integer m if and only if

$$\frac{b-a}{m}$$

is an integer. This may be re-written as

$$\big(\exists k \in \mathbb{Z}\big)\left[\frac{b-a}{m} = k\right]$$

Definition 3.2.17 Modulus
The divisor m in the congruence of two integers

Example: Find an integer congruent to 5 with respect to 3.

Solution: We can take the remainder of 5/3, which is 2. We will verify that $2 \equiv 5 \pmod{3}$. Notice that

$$\frac{5-2}{3} = \frac{3}{3} = 1$$

which is an integer. Also notice that the order of a and b does not matter:

$$\frac{2-5}{3} = \frac{-3}{3} = -1$$

which is still an integer

The problem asked for *an* integer, so maybe other integers satisfy the congruence. We need to find an integer b such that $\frac{b-5}{3} \in \mathbb{Z}$. We choose some random integer, maybe 8, and set it equal to the previous fraction, and solve for b.

$$\frac{b-5}{3} = 8 \Leftrightarrow b - 5 = 24 \Leftrightarrow b = 29$$

so 29 is congruent to 5, which is also congruent to 2, mod 3

You may have noticed how two integers are *congruent*, not equal. We save *equality* for things that are actually equal. Clearly $29 \neq 2 \neq 5$ yet all of these integers are *congruent* mod 3. This motivates a further expansion on congruence.

Definition 3.2.18 Congruence Class
The set of all integers that are congruent to each other with respect to a given modulus m. For a given integer a, the congruence class of a modulo m is the set of all integer solutions to $\frac{b-a}{m} \in \mathbb{Z}$

Example: Write the set of all integers congruent to 3 modulo 5.

Solution: We need all integer solutions to $\frac{b-3}{5}$, or equivalently $b = 5k+3$ for any integer k. This is the set

$$\{b \in \mathbb{Z} \mid (\exists k \in \mathbb{Z})[b = 5k + 3]\}$$

which, if we plug in values for k, equals

$$\{\cdots, -12, -7, -2, 3, 8, 13, \cdots\}$$

Remark 3.2.19
The congruence class of a modulo m is the set of all integers b such that $a \equiv b \pmod{m}$.

Now that you understand what *is* modular arithmetic, we now discuss the difference between *equivalence* and *congruence.* In logic, as you have seen before, an *equivalence* between two statements means that the two statements output the same truth values for the same inputs (the statements are *logically equivalent*). In modular arithmetic, however, a *congruence* between two integers means they fall in the same *congruence class* modulo some number. We overload the $\equiv$ symbol with these two meanings. Think of the $\equiv$ operator as a function that takes two arguments. In the *equivalence* definition the arguments are logical statements, however in the *congruence* definition the arguments are integers. Logical statements and integers will never overlap, so you should never be confused by our usage of the operator.

Recall that we can use integer parity to relate our definitions of even and odd. We will see now *why* it works.

Theorem 3.2.4
If two numbers are in different modulo classes, then they cannot be the same number. Formally,

$$(\forall a, b \in \mathbb{Z})(\forall m \in \mathbb{Z}^+)[a \not\equiv b \pmod{m} \Rightarrow a \neq b]$$

Proof. This is directly the contrapositive of an intuitive fact: $a = b \Rightarrow a \equiv b \pmod{m}$. Consider that $a \equiv a \pmod{m}$ and since $a = b$ then $a \equiv b \pmod{m}$ □

Example: Show that an even number cannot be an odd number.

Proof. Take an even number $2k$ and an odd number $2l+1$ for some $k, l \in \mathbb{Z}$. Now mod-2 those numbers: $2k \equiv 0 \pmod{2}$ and $2l + 1 \equiv 1 \pmod{2}$. Clearly $0 \neq 1$ and hence by the previous theorem $2k \neq 2l + 1$. Thus an even number cannot be an odd number □

Finally, three important theorems.

Theorem 3.2.5
For all integers a, b, c, d, m with $m > 0$, if

$$a \equiv b \pmod{m}$$

and

$$c \equiv d \pmod{m}$$

then

$$a + c \equiv b + d \pmod{m}$$

Proof. $a \equiv b \pmod{m} \Leftrightarrow (\exists k \in \mathbb{Z})[a = b + mk]$, $c \equiv d \pmod{m} \Leftrightarrow (\exists l \in \mathbb{Z})[c = d + ml]$. Then, we can add these two equations to get $a + c = b + mk + d + ml = b + d + m(k + l)$. By closure of integers under addition, $k + l \in \mathbb{Z}$, therefore $a + c \equiv b + d \pmod{m}$ □

Theorem 3.2.6
For all integers a, b, c, d, m with $m > 0$, if

$$a \equiv b \pmod{m}$$

and

$$c \equiv d \pmod{m}$$

then

$$ac \equiv bd \pmod{m}$$

Proof. $a \equiv b \pmod{m} \Leftrightarrow (\exists k \in \mathbb{Z})[a = b + mk]$, $c \equiv d \pmod{m} \Leftrightarrow (\exists l \in \mathbb{Z})[c = d + ml]$. Then, we can multiply these two equations to get $ac = (b+mk)(d+ml) = bd+bml+dmk+m^2lk = bd+m(bl+dk+mlk)$. By closure of integers under addition and multiplication, $bl + dk + mlk \in \mathbb{Z}$, therefore $ac \equiv bd \pmod{m}$ □

Theorem 3.2.7
For all integers a, b, m, n with $m > 0$ and $n \geq 0$, if

$$a \equiv b \pmod{m}$$

then

$$a^n \equiv b^n \pmod{m}$$

Proof. For $n = 0$ we have $a^0 = 1$ and $b^0 = 1$ and clearly $1 \equiv 1 \pmod{m}$. For $n = 1$ we have as given that $a \equiv b \pmod{m}$. Finally $n > 1$ follows by repeatedly taking the previous theorem letting $c = a$ and $d = b$. □

Remark 3.2.20
The above theorem is a one-way implication. We cannot go the reverse direction. For example, $2^4 = 16 \equiv 1 = 1^4 \pmod{3}$ yet $2 \not\equiv 1 \pmod{3}$

Remark 3.2.21
In all of the above theorems, the modulo m is the same. That is, we cannot do the following: $8 \equiv 2 \pmod{6}$ and $7 \equiv 1 \pmod{2}$ therefore $8+7 \equiv 2+1$ (mod ???)

3.2.8 Number Bases

Our familiar decimal system is known as *base-10*, since we have 10 possible digits spanning 0 to 9. We can represent base-10 numbers as an addition of numbers multiplied by the associated decimal position (the ones place, tens, hundreds, thousands, etc...).

Example: $123 = 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0$

There are three other bases important to computer science: base-2, base-8, and base-16.

Definition 3.2.22 Binary (base-2)
Representing numbers with two digits, 0 and 1

Definition 3.2.23 Octal (base-8)
Representing numbers with digits 0 to 7

Definition 3.2.24 Hexadecimal (base-16)
Representing numbers with 16 digits. After the 9th digit, we move to letters A through F

We implicitly denote any number as base-10. If we want to denote a number x as base-b, then we write x_b. For example, the base-10 number 24_{10} equals the base-2 number 11000_2.

Techniques for Translation

We generally work in base-10, so we will start with going from base-10 to any arbitrary base. For an arbitrary number x in base-10, translating it into a number in base b entails taking the least positive integer of x modulo b, integer dividing with respect to the base (by taking $\lfloor x/b \rfloor$, and repeating until the number becomes zero. This will yield a sequence of integers, which make up the digits of the number in base-b from least to most significant (right to left).

Example: Find the base-5 representation of the base-10 number 12345.

Solution: Follow the given algorithm. Step-wise, we get:

1. $12345 \equiv 0 \pmod 5$, and $\lfloor 12345/5 \rfloor = 2469$
2. $2469 \equiv 4 \pmod 5$, and $\lfloor 2469/5 \rfloor = 493$
3. $493 \equiv 3 \pmod 5$, and $\lfloor 493/5 \rfloor = 98$
4. $98 \equiv 3 \pmod 5$, and $\lfloor 98/5 \rfloor = 19$
5. $19 \equiv 4 \pmod 5$, and $\lfloor 19/5 \rfloor = 3$
6. $3 \equiv 3 \pmod 5$, and $\lfloor 3/5 \rfloor = 0$

So $12345_{10} = 343340_5$

We care more about the special cases of b here – 2, 8, and 16. We start with base-2.

A binary number's digits represent powers of 2 in base-10. For example, $18_{10} = 10010_2 = 16_{10} + 2_{10} = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$

$\cdots$	16	8	4	2	1
$\cdots$	1	0	0	1	0

Translating base-10 to base-2: Associate each binary digit with a power of 2 – $\{1, 2, 4, 8, 16, 32, \cdots\}$ – then repeatedly, until the input number becomes zero, subtract the highest power of 2 lower than the input number. The powers of 2 subtracted get a 1 for their digit, and the others get a 0.

Example: Translate 43 to binary.

Solution:

1. The highest power of 2 lower than 43 is 32, so the 32 digit gets a 1. $43 - 32 = 11$
2. The highest power of 2 lower than 11 is 8, so the 8 digit gets a 1. $11 - 8 = 3$
3. The highest power of 2 lower than 3 is 2, so the 2 digit gets a 1. $3 - 2 = 1$
4. The highest power of 2 lower than 1 is 1, so the 1 digit gets a 1. $1 - 1 = 0$

So $43_{10} = 101011_2$

Translating base-10 to base-16: First, translate the number to binary. Append zeros to the left-side of the binary number until the number of digits is a multiple of 4. Next, partition the number into groups of four (4) digits (bits). Finally, translate each 4-digit binary number group into a hexadecimal digit 0-15 (0-F).

Example: Translate 43 to hexadecimal.

Solution: From before, $43_{10} = 101011_2$. Left-fill with zeros: $= 00101011_2$. Partition into groups of 4: $= 0010 \mid 1011$. Translate each group into a hex number: $0010_2 = 2_{10} = 2_{16}$ and $1011_2 = 11_{10} = B_{16}$. So $43_{10} = 2B_{16}$

Translating base-10 to base-8: (*similar to the previous method*) First, translate the number to binary. Append zeros to the left-side of the binary number until the number of digits is a multiple of 3. Next, partition the number into groups of three (3) digits (bits). Finally, translate each 3-digit binary number group into an octal digit 0-7.

Example: Translate 43 to octal.

Solution: From before, $43_{10} = 101011_2$. Left-fill with zeros (none required): $= 101011_2$. Partition into groups of 3: $= 101 \mid 011$. Translate each group into an octal number: $101_2 = 5_{10} = 5_8$ and $011_2 = 3_{10} = 3_8$. So $43_{10} = 53_8$

Remark 3.2.25
We need four (4) binary digits to represent numbers 0-15, and three (3) binary digits to represent numbers 0-7.

Now, how do we go back from base-b to base-10? Well, associate each digit into an index position i, then multiply that digit by the base b raised to the power i.

Example: $2B_{16} = 2 \cdot 16^1 + B \cdot 16^0 = 2 \cdot 16 + 11 = 32 + 11 = 43$

Finally, how do we arbitrarily translate between bases? Well, usually the fastest way is to translate to base-10 or base-2, then translate to the other base you need.

Fast Mods

Taking mods with certain modulo can be made very easy, so long as you understand the previous modular arithmetic theorems and base-10 expansion.

Mod 1: Just return 1, since everything is divisible by 1.

Mod 2: Return 0 if the integer is even, and 1 if the integer is odd.

Mod 3: Notice that $10^k \equiv 1 \pmod 3$ for all $k \in \mathbb{N}$. So we expand an input number as the base-10 representation $a_n \cdot 10^n + \cdots + a_2 \cdot 10^2 + a_1 \cdot$

$10^1 + a_0 \cdot 10^0$ where the a_i's make up the associated digits of the input number. Then we take this summation $\pmod 3$ and we get

$$a_n \cdot 10^n + \cdots + a_2 \cdot 10^2 + a_1 \cdot 10^1 + a_0 \cdot 10^0 \equiv a_n + \cdots + a_2 + a_1 + a_0 \pmod 3$$

Thus, any integer is congruent to the sum of its digits $\pmod 3$.

Mod 5: Notice that $10^k \equiv 0 \pmod 5$ for all $k \in \mathbb{N}$ with $k > 0$. So, any digit past the one's place is equivalent to $0 \pmod 5$,

$$a_n \cdot 10^n + \cdots + a_2 \cdot 10^2 + a_1 \cdot 10^1 + a_0 \cdot 10^0 \equiv 0a_n + \cdots + 0a_2 + 0a_1 + 1a_0 \equiv a_0 \pmod 5$$

Thus, any integer is congruent to its one's place digit $\pmod 5$.

Mod 9: Works exactly the same way as $\pmod 3$, since $10^k \equiv 1 \pmod 9$ for all $k \in \mathbb{N}$.

Mod 10: Return the digit in the one's place (the last digit) of the input integer.

Applications to Circuits

One application which we promised to discuss is binary addition and its relation to circuits.

We can build a circuit that adds two single-digit binary numbers? There are only 4 possibilities for adding single-digit numbers, so let's examine what a truth table for this process might look like (all numbers are in base-2):

input a	input b	output $a+b$
0	0	0
0	1	1
1	0	1
1	1	10

In the output, we can always append zeros to the beginning of the binary number without changing the actual number:

input a	input b	output $a+b$
0	0	00
0	1	01
1	0	01
1	1	10

Now let's separate our output column into two columns – the sum-bit (right) and the carry-bit (left):

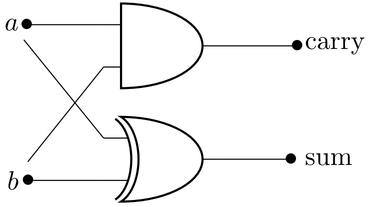
input a	input b	carry	sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

We know how to generate Boolean expressions for the output columns:

$$\begin{aligned} \text{sum} &\equiv a \oplus b \\ \text{carry} &\equiv a \wedge b \end{aligned}$$

Which gives us the **half adder**:

Adding two bits $a + b = cs$ is equivalent to the circuit



Now we can add two 1-bit numbers together. What if we want to add multiple-bit numbers together? Consider adding two 2-bit numbers:

$$\begin{array}{ccc} & c & \\ & x & y \\ + & z & w \\ \hline o_2 & o_1 & s \end{array}$$

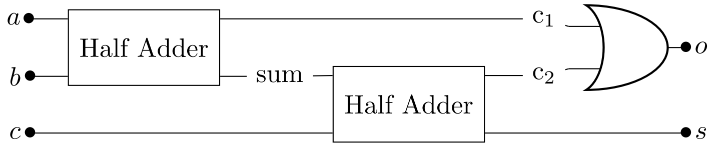
We see here that after adding $y+w$ we are left with a carry bit c for our addition $x + z$. How do we add three bits $c + x + z$? Well, we can separate it into two 1-bit additions: $c + x$ which, via a half-adder, yields a sum bit s_1 and carry bit c_1, then $s_1 + z$ which, via another half-adder, yields a sum bit s_2 and carry bit c_2. Then, we can let $o_1 = s_2$. Unfortunately this leaves us two carry bits that somehow need to be combined into a final carry bit.

Consider now the truth-table for adding three bits:

c	x	z	$c+x+z$	carry$(c+x)$	sum$(c+x)=s_1$	carry(s_1+z)
0	0	0	00	0	0	0
0	0	1	01	0	0	0
0	1	0	01	0	1	0
0	1	1	10	0	1	1
1	0	0	01	0	1	0
1	0	1	10	0	1	1
1	1	0	10	1	0	0
1	1	1	11	1	0	0

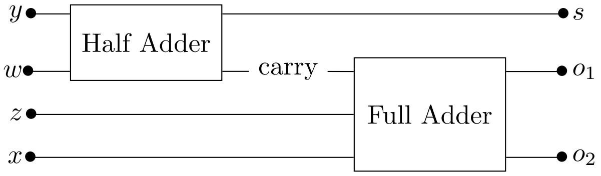
Now notice the carry bit in the output column $c+x+z$ is the same as OR-ing the two carry bits from $c+x$ and s_1+z. This completes our circuit for adding three bits. We call this the **full adder**:

Adding three bits $a+b+c=os$ is equivalent to the circuit



Putting these two structures, the half adder and full adder, together, we can construct a 2-bit adder, which solves our previous problem of doing $xy+zw=o_2o_1s$:

Adding three bits $a+b+c=os$ is equivalent to the circuit



This solves our 2-bit addition problem. What if we want to add 3-bit numbers? 4-bit numbers? n-bit numbers? Well, after the first bit column (adding two bits) we must add three bits. After this second column, we get a sum bit and a carry bit. If we tack on another column, which would make a 3-bit adder, then we add in the previous carry to the two new bits. This same process repeats for all further columns. So, tacking on another bit solely entails tacking on another full adder!

Example: How many half adders are required for an n-bit adder?

Solution: Note, here we implicitly assumed $n > 0$.
We need 1 half adder to start, and we need $n - 1$ full adders which each contain 2 half adders. This totals to $2(n-1)+1 = 2n-1$ half adders.

3.3 Proofs

Recall that number theory studies integers *and their properties*. We want to think about certain properties and be able to explain why they hold. We want our explanations, *proofs*, to be valid. We want to take our knowledge base of basic principles and logically derive true/false statements. The whole idea behind a proof is to creatively apply the set of rules you know to show whether a statement hold or not.

Definition 3.3.1 Proof
A formal and logical explanation for why a statement is true or false.

Let's examine this definition. We want to explain why a statement is *true* or *false*. Recall that in first-order logic (predicate logic) we have *existential* statements and *universal* statements. This gives us four proof "bins":

	Proving True	Proving False
Existential Statement	Find an example that makes the statement True	Show the statement is False for every element
Universal Statement	Show the statement is True for every element	Find an example that makes the statement False

Proving an existential statement true, as well as proving a universal statement false, is relatively easy – all you must do is search the statement domain for an example. Comparatively, proving an existential statement false, as well as proving a universal statement true, is much harder since it takes some form of creativity – that is, if you believe $P \neq NP$[1].

[1] You will learn about this in an Algorithms course. Here is the gist: problems in the Non-deterministic Polynomial (NP) class are ones whose solution is *easy* to verify in Polynomial time. Polynomial class problems have Polynomial-time (relatively fast) algorithms. If $P = NP$ then problems whose solution is easy to check *also* have fast algorithms. Typically you must use creativity to come up with an algorithm, however if it is just as easy to check a solution as it is to come up with a solution generator (algorithm), then creativity becomes meaningless. Suggested reading.

3.3.1 Format

Before we continue, we must show you how to format your proofs. The structure in this book may be different than what your instructor prefers, so make sure you pay attention in class and follow what they do. Our structure, however, is quite flexible with minimal rules, so it should be easy to adapt. You have likely already seen our style of proof in this book. We always follow a simple structural format that puts "*Proof.*" at the start, followed by what technique we are using (typically), followed by our explanation, finished by a simple square signifying the end of the proof.

> *Proof.* ← this signifies the start of a proof
>
> . . . put logically-sound proof here . . .
>
> . . . . and this signifies the end of a proof → □

The "*Proof.*" header clearly signifies the following text will be a proof and should be read as such.

The ending square is shorthand for the latin phrase *quod erot demonstrandum* which, used in this context, roughly means *it is shown/demonstrated.* You may put any of the following at the end of your proofs: QED, ■, □.

This general structure is easy to read, easy to adapt, and is easily formal. It is *absolutely fine* to deviate from our structure, **so long as** your proof is still easy to read and makes clear it **is** a proof.

3.3.2 Techniques

We have a handful of proof techniques that will help you structure your proofs.

Direct Proof

Direct proofs are classical applications of valid reasoning (see chapter 1.2.5 about deductions). We start with the easier techniques: examples and counterexamples.

Definition 3.3.2 Proof by Example
Showing a statement is true by presenting an example. This is used for proving existential statements true

Example: Prove that there is an integer that divides 6.

Proof. $\frac{6}{2} = 3 \in \mathbb{Z}$ so by definition $2 \mid 6$, so 2 is our example. □

Example: Show that

$$(\exists z \in \mathbb{Z})(\forall n \in \mathbb{N})[z \leq n]$$

Proof. The naturals are all positive, so any negative integer works. Let $z = -1 \leq n$ for all $n \in \mathbb{N}$ be our example. □

Definition 3.3.3 Proof by Counterexample
Showing a statement is false by presenting a counterexample. This is used for proving universal statements false

Example: Prove false that that all integers are divisible by 3.

Proof. $3 \nmid 2$ so 2 is our counterexample that proves the statement false. □

Remark 3.3.4
Proof by example and counterexample are the same. To see this, consider a general existential statement we want to prove true:

$$(\exists x)[P(x)] \equiv t$$

Now consider a general universal statement we want to prove false, and negate the entire thing:

$$\neg((\forall x)[Q(x)] \equiv f)$$

$$(\exists x)[\neg Q(x)] \equiv t$$

By letting $Q(x) \equiv \neg P(x)$ then we recover a proof by example.

Definition 3.3.5 Proof by Exhaustion
Proving a statement by reasoning about every possibility. Typically used for statements about small sets

Example: Prove that $(\forall e \in \{3,6,9\})[3 \mid e]$

Proof. We proceed via proof by exhaustion.
There are three possible cases in this statement: $3 \mid 3$, $3 \mid 6$, $3 \mid 9$
Case 1: $e = 3$, well $3 \mid 3$ since $\frac{3}{3} = 1 \in \mathbb{Z}$
Case 2: $e = 6$, well $3 \mid 6$ since $\frac{6}{3} = 2 \in \mathbb{Z}$
Case 3: $e = 9$, well $3 \mid 9$ since $\frac{9}{3} = 3 \in \mathbb{Z}$
All cases cover the domain, and all are true, therefore the statement is true. □

Definition 3.3.6 Proof by Construction

Definition 3.3.7 Proof by Non-Construction

Definition 3.3.8 General Direct Proof
Any proof that directly uses implications and/or bi-conditionals to directly show a statement is true. These proofs are done as a straightforward application of known-to-be-true facts, rules, propositions, and theorems

Proposition 3.3.1

$$(\forall a,b,c \in \mathbb{Z})[\big((a \mid b) \land (a \mid c)\big) \Rightarrow (\forall s,t \in \mathbb{Z})[a \mid (sb+tc)]]$$

Before we prove it, let's understand what it means. The statement says that if a number divides two other numbers, then it must also divide any linear combination of those two other numbers. We will use this proposition in a later example.

Proof. By definition of divisibility, $a \mid b \Leftrightarrow b = ak$ for some $k \in \mathbb{Z}$ and $a \mid c \Leftrightarrow c = al$ for some $l \in \mathbb{Z}$. Take any (all) $s,t \in \mathbb{Z}$. By substitution, and our previous definitions, $sb+tc = sak+tal = a(sk+tl)$. By closure of integers under addition and multiplication, $sk+tl = m \in \mathbb{Z}$. Thus $sb+tc = am \Leftrightarrow a \mid (sb+tc)$ □

Proposition 3.3.2

Indirect Proof

Advanced applications of valid reasoning. These techniques are "direct proofs in disguise" because you do, in-fact, prove *a statement* directly, however the original statement you *want to prove* is proven *indirectly* as a result.

We have two main techniques for indirect proof – contraposition and contradiction.

Definition 3.3.9 Proof by Contraposition
A proof technique that allows you to somewhat *reverse* an implication statement:

$$(p \Rightarrow q) \equiv ((\neg q) \Rightarrow (\neg p))$$

In a proof by contraposition, with the goal of proving $p \Rightarrow q$, one would prove the equivalent statement $((\neg q) \Rightarrow (\neg p))$

The proof by contraposition is useful in proving implication statements when it might be easier to go the opposite direction.

Proposition 3.3.3
For all integers a, if a^2 is even then a is even.

Proof. By way of contraposition, we prove the equivalent statement: if a is odd then a^2 is odd.
a is odd so by definition $a = 2k + 1$ for some integer k. Thus,

$$\begin{aligned} a^2 &= (2k+1)^2 \\ &= 4k^2 + 2k + 1 \\ &= 2(2k^2 + k) + 1 \end{aligned}$$

By closure of integers under addition and multiplication, $2k^2+k = m \in \mathbb{Z}$.
Then, $a^2 = 2m + 1$ is odd by definition.
The contrapositive of the statement is true, hence the original statement is true. □

We will use this proposition later.

Definition 3.3.10 Proof by Contradiction
A proof technique based on the following fact:

$$\begin{aligned} (\neg p) &\Rightarrow c \\ &\therefore p \end{aligned}$$

where c is some *logical contradiction* (ie something that is always false – for example, $p \wedge \neg p$).

A proof by contradiction exploits this fact – if one can show that the negation of a statement *must* lead to a logical contradiction, then that negation of a statement must be false.

Consider the contrapositive of the proof by contradiction:

$$(\neg p) \Rightarrow c \equiv t \Rightarrow p$$

where t is a tautology (ie something that is always true – for example, $p \vee \neg p$). Consider the truth table:

p	$\neg p$	t	$t \Rightarrow p$
F	T	T	F
T	F	**T**	**T**

The contrapositive statement is logically equivalent to p. Thus, a proof by contradiction is equivalent to directly proving the original statement.

The proof by contradiction is useful for proving statements that may "seem obvious" yet have no seemingly obvious direct proof.

Example: Prove there is no greatest integer.

Proof. By way of contradiction, assume there is a greatest integer. Call it $N \in \mathbb{Z}$. By closure of integers under addition, $N+1 \in \mathbb{Z}$. $N+1 > N$, so we have found an integer greater than N. But this contradicts our assumption that N is the greatest integer (and clearly, $N+1 \neq N$). Our assumption has lead to a contradiction, therefore our assumption must be false. Hence there is no greatest integer. □

The following argument, called the Euclidean Argument, is an important example of a proof by contradiction.

Example: Prove that $\sqrt{2} \notin \mathbb{Q}$

Proof. By way of contradiction, assume $\sqrt{2} \in \mathbb{Q}$. Then, by definition $\sqrt{2} = \frac{a}{b}$ for some $a, b \in \mathbb{Z}$ with $b \neq 0$. We necessarily assume that the ratio $\frac{a}{b}$ is in its most-reduced form (that a and b share no common divisors).
Then, $\sqrt{2} = \frac{a}{b} \Rightarrow 2 = \frac{a^2}{b^2} \Leftrightarrow 2b^2 = a^2$. b^2 is an integer by closure, thus a^2 is even. By proposition 3.3.3, it follows that a is even, so by definition $a = 2k$ for some $k \in \mathbb{Z}$.
Then, $2b^2 = a^2 = (2k)^2 = 4k^2$ so by dividing out 2, $b^2 = 2k^2$. k^2 is an integer by closure, thus b^2 is even. Again, by proposition 3.3.3, it follows that b is even, so by definition $b = 2l$ for some $l \in \mathbb{Z}$.
Then, $\sqrt{2} = \frac{a}{b} = \frac{2k}{2l} = \frac{k}{l}$ is reducible. This contradicts our assumption, hence $\sqrt{2} \notin \mathbb{Q}$. □

There is a point in this proof that may be confusing – why exactly do we need the assumption that $\sqrt{2} = \frac{a}{b}$ is in its most-reduced form? An excellent question. Let us do the proof **without** this assumption, and see what happens.

Proof. This proof is a sketch, and will skip over details noted in the previous proof.
Assume $\sqrt{2} = \frac{a}{b} \in \mathbb{Q}$. Then … *fill in the previous proof* … $\sqrt{2} = \frac{a}{b} = \frac{2k}{2l} = \frac{k}{l}$. Okay, then we proceed the same proof where $\sqrt{2} = \frac{k}{l}$.
If we solely examine the numerator, this gives us an infinite sequence of decreasing integers. In fact, since $\sqrt{2} > 0$ we know that all numbers in this sequence are non-negative. The sequence is as follows: $a, \frac{a}{2}, \frac{a}{4}, \frac{a}{8}, \cdots$
From the axioms that build all of mathematics, we can prove that any decreasing sequence of non-negative integers must stop. This proof is out of scope of this book.
Thus, our numerator, and by a similar argument our denominator, *must* eventually get to a form where they are irreducible. Then, we insert our original proof, and we are done. □

Euclid's Argument can in-fact be extended to show other roots of numbers are irrational.

Example: Prove that $\sqrt{7} \notin \mathbb{Q}$

Proof. This proof will rely on the following lemma, which is given as an exercise to the reader: $(\forall a \in \mathbb{Z})[7 \mid a^2 \Rightarrow 7 \mid a]$
By way of contradiction, assume $\sqrt{7} \in \mathbb{Q}$. Then, by definition $\sqrt{7} = \frac{a}{b}$ for some $a, b \in \mathbb{Z}$ with $b \neq 0$. We necessarily assume that the ratio $\frac{a}{b}$ is in its most-reduced form (that a and b share no common divisors).
Then, $\sqrt{7} = \frac{a}{b} \Rightarrow 7 = \frac{a^2}{b^2} \Leftrightarrow 7b^2 = a^2$. b^2 is an integer by closure, thus a^2 is divisible by 7. By the above lemma, it follows that a is divisible by 7, so by definition $a = 7k$ for some $k \in \mathbb{Z}$.
Then, $7b^2 = a^2 = (7k)^2 = 49k^2$ so by dividing out 7, $b^2 = 7k^2$. k^2 is an integer by closure, thus b^2 is divisible by 7. Again, by the above lemma, it follows that b is divisible by 7, so by definition $b = 7l$ for some $l \in \mathbb{Z}$.
Then, $\sqrt{7} = \frac{a}{b} = \frac{7k}{7l} = \frac{k}{l}$ is reducible. This contradicts our assumption, hence $\sqrt{7} \notin \mathbb{Q}$. □

At this point, you may be able to see a general "machinery" at work here, and you may be wondering whether it works for all square roots.

Example: Spot the logical fallacy in the following proof that $\sqrt{4} \notin \mathbb{Q}$

Proof. Assume the following lemma holds: $(\forall a \in \mathbb{Z})[4 \mid a^2 \Rightarrow 4 \mid a]$ By way of contradiction, assume $\sqrt{4} \in \mathbb{Q}$. Then, by definition $\sqrt{4} = \frac{a}{b}$ for some $a, b \in \mathbb{Z}$ with $b \neq 0$. We necessarily assume that the ratio $\frac{a}{b}$ is in its most-reduced form (that a and b share no common divisors).
Then, $\sqrt{4} = \frac{a}{b} \Rightarrow 4 = \frac{a^2}{b^2} \Leftrightarrow 4b^2 = a^2$. b^2 is an integer by closure, thus a^2 is divisible by 4. By the above lemma, it follows that a is divisible by 4, so by definition $a = 4k$ for some $k \in \mathbb{Z}$.
Then, $4b^2 = a^2 = (4k)^2 = 16k^2$ so by dividing out 4, $b^2 = 4k^2$. k^2 is an integer by closure, thus b^2 is divisible by 4. Again, by the above lemma, it follows that b is divisible by 4, so by definition $b = 4l$ for some $l \in \mathbb{Z}$.
Then, $\sqrt{4} = \frac{a}{b} = \frac{4k}{4l} = \frac{k}{l}$ is reducible. This contradicts our assumption, hence $\sqrt{4} \notin \mathbb{Q}$. □

Solution: This is clearly a false statement, since $\sqrt{4} = 2 = \frac{2}{1} \in \mathbb{Q}$. The fallacy occurs at the beginning – assuming a lemma that does not in-fact hold. Why? Well, $4 \mid 2^2$ but $4 \nmid 2$.

Theorem 3.3.4
The Unique Prime Factorization Theorem Every integer strictly bigger than 1 can be written uniquely as a product of primes. Formally,

$$(\forall n \in \mathbb{N}^{\geq 2})(\exists p_i \in \mathbf{P}, e_i \in \mathbb{N}^{\geq 1})[n = p_1^{e_1} \cdot p_2^{e_2} \cdot \ldots \cdot p_k^{e_k}]$$

where $1 \leq i \leq k$, $\mathbf{P}$ *is the set of prime numbers, and* $p_1 < p_2 < \cdots < p_k$

There are two parts to this proof – namely, that the factorization **exists** and the factorization is **unique**.

Proposition 3.3.5 Euclid's Lemma
For two integers a, b, if a prime number p divides ab, then p divides a or b (or both).

The proof of this is out of scope for this book. It requires fundamental theorems and definitions in number theory (namely, Bezout's Lemma, the Well-ordering Principle, Greatest Common Divisor definition, and Relatively Prime definition). If you are interested, to an internet search for "Euclid's Lemma".

We leave the *existential* part of the Unique Prime Factorization Theorem for later sections in this chapter. Below, we prove the *uniqueness*.

Proof. Suppose the factorization of some integer z is **not** unique, so it has two factorizations. To make things easier,

□

Now we can use the unique factorization theorem:

Example: Prove that there are an infinite number of prime numbers

Proof. By way of contradiction, assume there are a finite number of prime numbers. Then we can list all the prime numbers: $p_1, p_2, \cdots, p_k$. Let $P = p_1 \cdot p_2 \cdot \ldots \cdot p_k$ be the product of all primes in our list. We construct a new number Q by adding 1 to P: $Q = P + 1 = p_1 \cdot p_2 \cdot \ldots \cdot p_k + 1$. Now, either Q is prime or it is not prime. If it is prime, then we have found another prime number not in our original list of primes, which contradicts our assumption.
So Q is not prime. Then, by UPFT, there is some prime factor p that divides Q. p must be in our list of primes, as per our original assumption. Thus, $p \mid P$. But $p \mid (P + 1 = Q)$. By proposition 3.3.1, we must have that $p \mid (Q - P = (P + 1) - P)$, so $p \mid 1$. But by definition no prime number can divide 1. This is a contradiction, so our original assumption must be false. Hence, primes are infinite. □

UPFT gives us another way to prove that $\sqrt{2} \notin \mathbb{Q}$

Example: Prove that $\sqrt{2} \notin \mathbb{Q}$

Proof. Suppose not, then by definition $\sqrt{2} = \frac{a}{b}$ for some $a, b \in \mathbb{Z}$ with $b \neq 0$. In order to use UPFT, we need to show that $a, b > 1$. Since $\sqrt{2} > 0$ then it follows that $a, b > 0$ (if they were both negative, then simplify to make them positive). It suffices to explain why $a, b \neq 1$. Suppose $a = 1$, then if $b = 1$ we have $\sqrt{2} = 1$ which is not true, and if $b > 1$ then $\sqrt{2} < 1 \Rightarrow 2 < 1$ is also not true. Thus $a > 1$, and we need to show that $b \neq 1$. Suppose $b = 1$, then $\sqrt{2} = a \geq 2 \Rightarrow 2 \geq 4$ is not true. So we have both $a, b > 1$.
By UPFT, both a and b have unique prime factorizations:

$$a = p_1^{e_1} \cdots p_k^{e_k}$$

$$b = q_1^{f_1} \cdots q_m^{f_m}$$

with p_i, q_j prime and e_i, f_j positive integers.

Now, since 2 is prime, it appears inside both a and b's factorization, possibly with a zero exponent $2^0 = 1$. So, re-write the factorizations with all of the 2's pulled out:

$$a = 2^e \cdot A$$

$$b = 2^f \cdot B$$

where $e, f \geq 0$ and A, B constitute the rest of the factorization.
Now, we have

$$\begin{aligned}\sqrt{2} &= \frac{a}{b} \\ 2 &= \frac{a^2}{b^2} \\ 2b^2 &= a^2 \\ 2(2^f \cdot B)^2 &= (2^e \cdot A)^2 \\ 2^{2f+1}B^2 &= 2^{2e}A^2\end{aligned}$$

By UPFT, the only way for this equality to be satisfied is if $2f + 1 = 2e$. This means, by definition, that an odd number equals an even number, which is impossible. We have arrived at a contradiction, so our assumption was wrong, and hence $\sqrt{2} \notin \mathbb{Q}$ □

Induction

Historically induction is difficult for students to learn, *at first.* We have found, though, that as students progress through the course they exclusively either

- Fully understand the principle of induction, or
- Learn how to pattern-match and guess

We hope our explanations allow you to fall in the first category.

Definition 3.3.11 Weak Induction
A proof technique to show that some predicate, mathematical statement, $P(n)$ is true for all n in a discrete lower-bounded increasing set. Typically $n \in \mathbb{N}$.

The proof technique of weak induction is entirely based on the following theorem:

Theorem 3.3.6
The Principle of (weak) Mathematical Induction

$$\begin{array}{rl} & P(0) \\ & (\forall n \in \mathbb{N})[P(n) \Rightarrow P(n+1)] \\ \hline \therefore & (\forall n \in \mathbb{N})[P(n)] \end{array}$$

Included below is an equivalent definition (note the only difference is the second line):

Theorem 3.3.7
The Principle of (weak) Mathematical Induction

$$\begin{array}{rl} & P(0) \\ & (\forall n \in \mathbb{N}^{\geq 1})[P(n-1) \Rightarrow P(n)] \\ \hline \therefore & (\forall n \in \mathbb{N})[P(n)] \end{array}$$

We included both styles because sometimes one style is easier than the other when doing inductive proofs.

A classic inductive proof has 3 parts:

- The base case
- The hypothesis
- The step

Each part has a specific relation to theorems 3.3.6 and 3.3.7. We will explain these relations after the following example.

Example: Prove, $\forall n \in \mathbb{N}$,

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

Proof. We proceed using the principle of mathematical induction (3.3.6).
Base Case: We show the statement is true for $n = 0$. Indeed,

$$\sum_{i=0}^{0} i = 0 \text{ and } \frac{0(0+1)}{2} = 0$$

so the equality holds for $n = 0$
Hypothesis: Assume the statement is true for an arbitrary $n \in \mathbb{N}$. That is,

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

Step: Prove the implication. That is, using the hypothesis, show

$$\sum_{i=0}^{n+1} i = \frac{(n+1)((n+1)+1)}{2}$$

We know that

$$\sum_{i=0}^{n+1} i = (\sum_{i=0}^{n} i) + (n+1)$$

using properties of summation. From the hypothesis we know

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

Thus,

$$\begin{aligned} \sum_{i=0}^{n+1} i &= \big(\sum_{i=0}^{n} i\big) + (n+1) \\ &= \big(\frac{n(n+1)}{2}\big) + (n+1) && \text{hypothesis} \\ &= \big(\frac{n}{2}+1\big)(n+1) = (\frac{n+2}{2})(n+1) \\ &= \frac{(n+1)(n+2)}{2} \end{aligned}$$

Therefore, by the principle of mathematical induction we conclude the statement holds $\forall n \in \mathbb{N}$ □

Here is the same example, but we use 3.3.7 instead.

Example: Prove, $\forall n \in \mathbb{N}$,

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

Proof. We proceed using the principle of mathematical induction (3.3.7).
Base Case: We show the statement is true for $n = 0$. Indeed,

$$\sum_{i=0}^{0} i = 0 \text{ and } \frac{0(0+1)}{2} = 0$$

so the equality holds for $n = 0$

Hypothesis: Assume the statement is true for an arbitrary $n \in \mathbb{N}^{\geq 1}$. That is,

$$\sum_{i=0}^{n-1} i = \frac{(n-1)((n-1)+1)}{2}$$

Step: Prove the implication. That is, using the hypothesis, show

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

We know that

$$\sum_{i=0}^{n} i = (\sum_{i=0}^{n-1} i) + n$$

using properties of summation. From the hypothesis we know

$$\sum_{i=0}^{n-1} i = \frac{(n-1)((n-1)+1)}{2}$$

Thus,

$$\begin{aligned}
\sum_{i=0}^{n} i &= \Big(\sum_{i=0}^{n-1} i\Big) + n \\
&= \Big(\frac{(n-1)((n-1)+1)}{2}\Big) + n && \text{hypothesis} \\
&= \Big(\frac{n-1}{2} + 1\Big)(n) = (\frac{n+1}{2})(n) \\
&= \frac{n(n+1)}{2}
\end{aligned}$$

Therefore, by the principle of mathematical induction we conclude the statement holds $\forall n \in \mathbb{N}$ □

We now explain how each part of an inductive proof fits into the theorem. First notice that the theorem is a rule of inference – we are *deducing* that the theorem holds *for all* natural numbers. So, we need to *prove* the two premises to deduce that the theorem holds. The first premise, $P(0)$, is precisely the **base case**. We initially need to show that the statement we are proving is true for the lowest possible value of n (typically we use $\mathbb{N}$, but our base case could start at 4 for example). After the base case, we need to show the second premise, that $P(n) \Rightarrow P(n+1)$. If we recall the truth table for $p \Rightarrow q$, then we recover the hypothesis. Recall that if p is *false* then the implication $p \Rightarrow q$ is *true* regardless of what q is. Now, if p is *true* then the implication is dictated entirely by q.

The upshot of this is that to prove $P(n) \Rightarrow P(n+1)$ we first assume $P(n)$ is *true* (this is the inductive **hypothesis**), then we use this (since we are proving the $\Rightarrow$) to show that $P(n+1)$ is true (this is the inductive **step**).

So how does induction *work*? Well, suppose we have used induction to prove $P(n)$ for all $n \in \mathbb{N}$. So we have shown that $P(0)$ is true, and $P(n) \Rightarrow P(n+1)$. Using these two facts, we can plug in $n = 0$ into the implication to get $P(0) \Rightarrow P(1)$, and since we know $P(0)$ is true then we deduce that $P(1)$ is true. Again, we can plug in $n = 1$ into the implication to get $P(1) \Rightarrow P(2)$, and since, from before, we know $P(1)$ is true then we deduce that $P(2)$ is true. We can repeatedly do this to generate *any* $P(n)$, and hence we have that $P(n)$ holds for **all** n. Notice that this only worked because we had:

1. the base case
2. an implication, where the domain of input *includes* the base case

One final note about the inductive hypothesis. We want to assume the theorem holds *for some arbitrary* $n \in \mathbb{N}$. We want our inductive step to apply to *any* value of n, however we do **not** want to quantify n with $\forall, \exists$. We want it to be a *generic particular*. Consider the alternative – in the hypothesis you assume $P(n)$ holds for all n. In this case, you have just assumed the entire mathematical statement is true, so why do you need to prove it?

The starting point of induction is arbitrary – we could show some statement is true $\forall n \in \mathbb{N}^{\geq 4}$. The base case and hypothesis should adjust accordingly.

Example: Prove

$$(\forall n \in \mathbb{N}^{\geq 4})[2^n < n!]$$

Proof. We proceed using the principle of mathematical induction (3.3.7).
Base Case: We show the statement is true for $n = 0$. Indeed,

$$2^4 = 16 < 24 = 4!$$

so the inequality holds for $n = 4$
Hypothesis: Assume the statement is true for some arbitrary $n-1 \geq 4$. That is,

$$2^{n-1} < (n-1)!$$

Step: Prove the implication. That is, using the hypothesis, show

$$2^n < n!$$

We know that

$$2^n = 2 \cdot 2^{n-1}$$

and from the hypothesis

$$2^{n-1} < (n-1)!$$

and since $n-1 \geq 4 \Leftrightarrow n \geq 5 > 2$ then

$$\begin{aligned} 2^n &= 2 \cdot 2^{n-1} \\ &< 2(n-1)! && \text{hypothesis} \\ &< n(n-1)! \\ &= n! \end{aligned}$$

Therefore, by the principle of mathematical induction we conclude the statement holds $\forall n \in \mathbb{N}$ □

We can prove any sort of statement with n in a discrete lower-bounded increasing set. (note that sets are by definition un-ordered, however in this case we can place the natural ordering on $\mathbb{N}$ – sorted order)

Example: Prove

$$(\forall n \in \mathbb{N})[8 \mid (2n+1)^2 - 1]$$

Proof. We proceed using the principle of mathematical induction (3.3.6).
Base Case: We show the statement is true for $n = 0$. Indeed,

$$(2 \cdot 0 + 1)^2 - 1 = 1 - 1 = 0$$

and zero is divisible by anything, so the statement holds for $n = 0$
Hypothesis: Assume the statement is true for some arbitrary $n \geq 0$. That is,

$$8 \mid (2n+1)^2 - 1$$

Step: Prove the implication. That is, using the hypothesis, show

$$8 \mid (2(n+1)+1)^2 - 1$$

From the hypothesis we have that

$$8 \mid (2n+1)^2 - 1 \Leftrightarrow (\exists k \in \mathbb{Z})[(2n+1)^2 - 1 = 8k]$$

thus,

$$\begin{aligned} (2(n+1)+1)^2 - 1 &= (2n+3)^2 - 1 \\ &= 4n^2 + 12n + 9 - 1 \\ &= 4n^2 + 4n + 1 - 1 + 8n + 8 \\ &= (2n+1)^2 - 1 + 8n + 8 \\ &= 8k + 8n + 8 && \text{hypothesis} \\ &= 8(k + n + 1) \end{aligned}$$

By closure we have $k+n+1 \in \mathbb{Z}$, so by definition $8 \mid 8(k+n+1)$, which means $8 \mid (2(n+1)+1)^2 - 1$
Therefore, by the principle of mathematical induction we conclude the statement holds $\forall n \in \mathbb{N}$ □

***

Definition 3.3.12 Strong Induction

Example: Prove the existence part of the UPFT – ie prove that every integer above 1 can be factored into a product of primes.

Proof.
□

***

Definition 3.3.13 Structural Induction

***

Definition 3.3.14 Constructive Induction

Combination of Techniques

We can combine our previously-defined proof techniques to accomplish any goal we want.

Definition 3.3.15 Proof by Cases
A more general proof technique where you break a statement into multiple sub-

statements, and prove each sub-statement individually. The sub-statements **must** fully cover the original statement (there cannot be anything missing)

Example: Explain, for the statement $(\forall x \in \mathbb{Z})[2 \mid x^2 + x]$, why the following proof is invalid.

> *Proof.* When x is even then $\exists k \in \mathbb{Z}$ such that $x = 2k$, and thus $x^2 + x = (2k)^2 + 2k = 4k^2 + 2k = 2(2k^2 + k) = 2q$ where $q = 2k^2 + k \in \mathbb{Z}$ by closure of integers under addition and multiplication. So $x^2 + x$ is divisible by 2 by definition □

Solution: It is invalid because the proof does not fully cover all possibilities of the statement. The proof covers all even integers, however odd integers are nowhere found in the proof. Clearly $\mathbb{Z} \neq \mathbb{Z}^{\text{even}}$.
This proof should have been a proof by cases. One case is the proof already given, and the other case should be when x is odd.

Example: Finish the previous proof.

Proof. (continued)
Case 2: x is odd, then $\exists l \in \mathbb{Z}$ such that $x = 2l + 1$, then $x^2 + x = (2l+1)^2 + 2l + 1 = 4l^2 + 4l + 1 + 2l + 1 = 4l^2 + 6l + 2 = 2(2l^2 + 3l + 1) = 2r$ where $r = 2l^2 + 3l + 1 \in \mathbb{Z}$ by closure of integers under addition and multiplication. So $x^2 + x$ is divisible by 2 by definition
Both cases cover the entirety of the integers by parity, thus the statement holds for all integers □

Remark 3.3.16
Both cases in the previous proof were direct proofs, however in other examples there could be cases that are proven indirectly. The previous proof used two cases, however there could be examples using more (e.g. if you want to prove $(\forall a \in \mathbb{Z})[7 \mid a^2 \Rightarrow 7 \mid a]$).

3.4 Summary

- one
- two

- three

3.5 Practice

1. Re-write the following sum as a summation:

$$1 - 4 + 7 - 10 + 13 - 16 + 19 - 22 + \cdots \pm (3n - 2)$$

2. Re-write the following product as a product:

$$n^{1/2} \times (2n)^{1/4} \times (3n)^{1/6} \times (4n)^{1/8} \times \cdots$$

3.

4. Explain the difference between $\equiv$ (congruence) and %.

5. Fill in the following number-base translation table:

Base 2	Base 8	Base 16	Base 10
10110			
	703		
		0xDAB	
			1000

6. Come up with a rule for $\pmod 4$ and $\pmod{11}$.

7. Show that $(\forall m, n \in \mathbb{Z}^{\neq 0})[(n \mid m \wedge m \mid n) \Rightarrow m = \pm n]$

8. Prove there is no smallest integer.

9. Prove $7 \mid a^2 \Rightarrow 7 \mid a$ for any $a \in \mathbb{Z}$.

10. Prove $2 \mid a^3 \Rightarrow 2 \mid a$ for any $a \in \mathbb{Z}$. Then prove that $\sqrt[3]{2} \notin \mathbb{Q}$ by the Euclidean Argument and by UPFT.

11. Generalize the lemmas used in the root-irrationality proofs – namely, prove that for any prime p, integer a, and integer $n > 1$, $p \mid a^n \Rightarrow p \mid a$. Then, prove the general claim that $\sqrt[n]{p} \notin \mathbb{Q}$ by the Euclidean Argument and by UPFT.

12. Prove that $\sqrt[n]{n} < 2 - \frac{1}{n}$ for all $n \geq 2$.

13. Prove $\forall x \neq 0,\ \forall n \in \mathbb{N}^{>1}$ that $x^2 \mid ((x+1)^n - nx - 1)$

14.

Chapter 4

Combinatorics and Probability

What?!

Lil Jon

Contents

4.1 Introduction

This is *Discrete* Mathematics, so we will be focusing on *discrete* probability.

Let's put some more text here for now

4.2 Combinatorics

Imagine you are a software developer for a large company, *Macrosoft*. Your manager assigned you to a project that needs test cases written. Your test cases must cover all execution paths. At first this seems like a daunting task, however you took discrete mathematics and bleh bleh

(just say a story that shows that counting techniques are useful for creating test cases)

Definition 4.2.1 Combinatorics

> **Example:** A problem here

Definition 4.2.2 Permutations

> **Example:** A problem here

Definition 4.2.3 Combinations

> **Example:** A problem here

A mini-discussion on the differences between ordering and replacement.

Insert that one table here.

4.3 Pigeonhole Principle

Theorem 4.3.1 The Pigeonhole Principle

Theorem 4.3.2 Generalized Pigeonhole Principle

4.4 Discrete Probability

Intro text here

Definition 4.4.1 Probability
The likelihood (percent chance) that an event occurs. Probabilities of every possible event should add to 1

Example:

Something else here

Example: Harder example

Joint/Disjoint

Definition 4.4.2 Joint Probability

Definition 4.4.3 Independence

Some discussion on conditional probability

Definition 4.4.4 Conditional Probability

Definition 4.4.5 Conditional Probability

Probably talk about

4.5 Basic Statistics

Statistics are an important application of probability. Statistics give us a way to mathematically model probabilities of big events/experiments. We cover three fundamental statistics principles, along with a few related topics.

Definition 4.5.1 Expected Value
$\mathbb{E}[X] = \mu_X = \sum_x x \cdot \Pr(X = x)$. This is a generalization of the *arithmetic mean*, which is defined in scenarios where outcomes are equally likely. If we denote $x_1,\ x_2,\ x_3,\ \cdots,\ x_n$ to be the values outputted by X, and $p_1,\ p_2,\ p_3,\ \cdots,\ p_n$ to be their respective probabilities, then $\mathbb{E}[X] = \sum_{i=1}^{n} x_i p_i$

Sometimes students confuse *expected value* with *average*. First, the "average" is not well-defined – it could refer to the mean, median, or mode (typically it refers to the mean). Second, as stated in the definition, the expected value is a generalized *mean*.

Definition 4.5.2 Variance

Definition 4.5.3 Standard Deviation

4.6 Summary

- 1
- 2
- 3

4.7 Practice

1. q
2. 2 of 20 light-bulbs are defective. You select 2 light-bulbs at random. What is the probability that neither bulb selected is defective?
3. q

Chapter 5

Functions and Relations

They don't think it be like it is,
but it do.

Oscar Gamble

Contents

5.1 Introduction

Sometimes in life you will have two (or more) things that are somehow related. Sometimes people make these relations for you. Sometimes you will be given two things and you have to come up with the relation. Our goal in this section is to give you the tools to solve these problems life throws at you. And remember, when life gets you down, just chuck some lemons back at him.

5.2 Functions

We start simply with the definition.

Definition 5.2.1 Function
An explicit rule, or set of rules, that can be applied to a specified input to yield a specified output. There are two rules that all functions must follow:

1. Every input must be mapped to an output

2. No input can map to more than one output

Basically, functions allow us to define a *mapping* between sets. Typically we define functions on sets of numbers (we will restrict it to subsets of $\mathbb{R}$). We use the following notation to denote a function f:

$$f(x) = \text{some rules}$$

and we pair the input/output sets as follows:

$$f : D \mapsto R$$

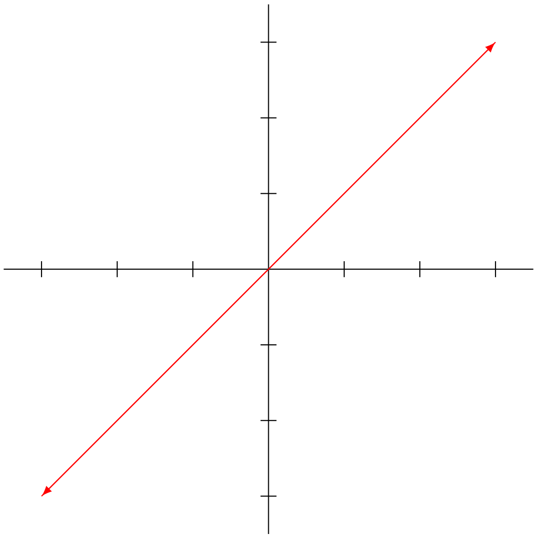
Definition 5.2.2 Domain
The set of inputs to a function

Definition 5.2.3 Range
The set containing all outputs to a function. This may also be called the *co-domain*

Definition 5.2.4 Image
The set of all outputs to a function. This may be a subset of the function's range

Let's look at some basic examples. First are some graphs of functions. Hopefully you learned how to plot 2-dimensional functions in previous math classes.

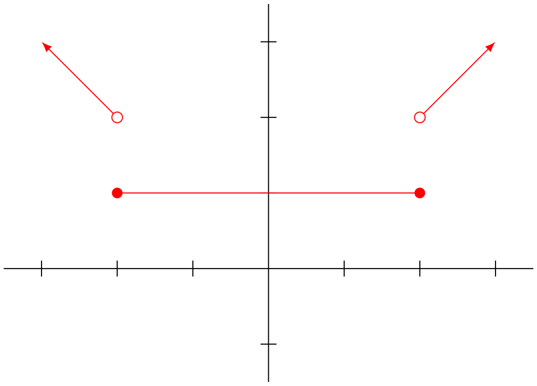
Example: Here is a graph of the function $f(x) = x$ where $f : \mathbb{R} \mapsto \mathbb{R}$



Example: Here is a graph of the function

$$f(x) = \begin{cases} x, & x > 2 \\ 1, & -2 \leq x \leq 2 \\ -x, & x < -2 \end{cases}$$

where $f : \mathbb{R} \mapsto \mathbb{R}$



Next we present examples of classifying whether a function is a function. To do this, recall the two requirements:

1. Every input must be mapped to an output

2. No input can map to more than one output

You should check both of these requirements. For the first, recall that to disprove a universal statement you need a single counter-example. Thus, search the domain and make sure each input maps to an output

Example: Which of the following are functions? For the non-functions, change them such that they become functions.

(a) $a(x) = x$ where $a : \mathbb{Z} \mapsto \mathbb{R}$
(b) $b(x) = \sqrt{x}$ where $b : \mathbb{Z} \mapsto \mathbb{R}$
(c) $c(y) = \log_2(y)$ where $c : \mathbb{N} \mapsto \mathbb{Z}$
(d) $d(y) = y^2$ where $d : \mathbb{Z} \mapsto \mathbb{N}$
(e) $e(z) = \begin{cases} z+1 & z \geq 0 \\ z-1 & z \leq 0 \end{cases}$ where $e : \mathbb{R} \mapsto \mathbb{Z}$

Solution:

(a) Function
(b) Not a function – b does not work for any negative input, however the domain is set to $\mathbb{Z}$. Simply change the domain to $\mathbb{N}$
(c) Not a function – $c(3) \notin \mathbb{Z}$ thus not every input is mapped to an output. Change the co-domain to positive real numbers
(d) Function
(e) Not a function – (1) the input $z = 0$ has two different outputs $e(0) = 1$ and $e(0) = -1$, (2) any real number that is not an integer does not have a mapping in the range (ex: $e(1.5) = 1.5 + 1 = 2.5 \notin \mathbb{Z}$). Change one of the cases to not include 0, then either change the range to $\mathbb{R}$ or change the domain to $\mathbb{Z}$

There exist 3 properties that any function can have.

Definition 5.2.5 Injection (One-to-One)
A function where each co-domain (range) value is mapped to by at most one value in the domain. In other words, each value in the range has either 0 or 1 pointers from the domain. Formally, $f : D \mapsto R$ is injective iff

$$(\forall x_1, x_2 \in D)[f(x_1) = f(x_2) \Rightarrow x_1 = x_2]$$

The horizontal line test can be used to see if a function is injective

Definition 5.2.6 Surjection (Onto)
A function where each co-domain (range) value is mapped to by at least one value in the domain. In other words, there are no un-mapped values in the range. Formally, $f : D \mapsto R$ is surjective iff

$$(\forall y \in R)(\exists x \in D)[y = f(x)]$$

Definition 5.2.7 Bijection (One-to-One Correspondence)
A function that is both an injection and a surjection. Every input is mapped to exactly one unique output

You should know how to identify these properties in functions. To test injectivity, perform the horizontal line test – run a horizontal line vertically along the graph of the function; the function is injective if the horizontal line never touches more than one point. To test surjectivity, you can BLANK. To be bijective, you must both be injective and surjective. A function can be neither injective nor surjective, but still be a function.

We can represent a function by drawing ovals representing the domain and range, drawing points in each oval representing the elements in each set, and drawing arrows between the points representing the function mapping. In this way, we present a visual guide to the above definitions.

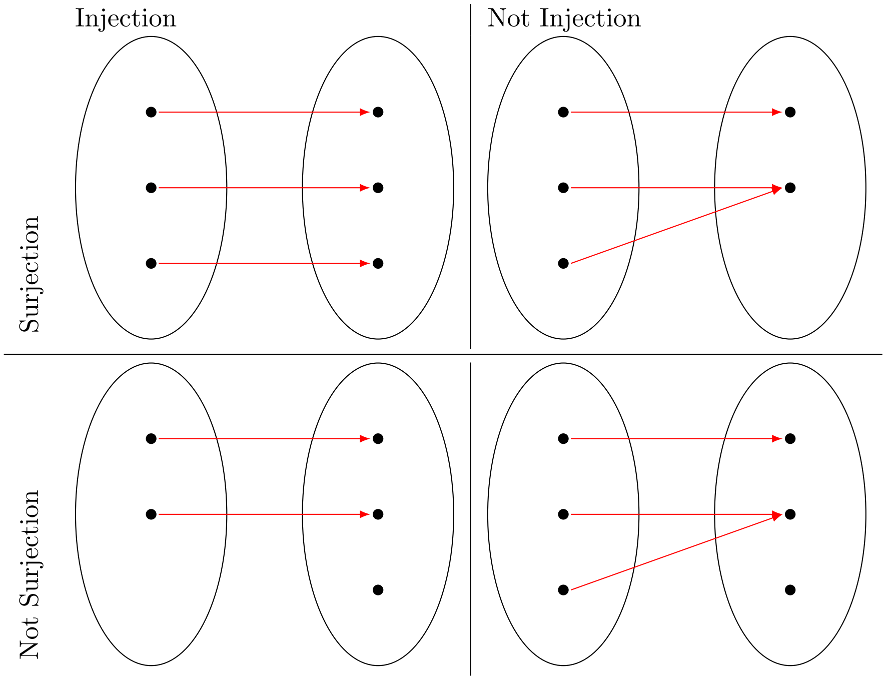


Figure 5.1: Injection and Surjection

Example: Classify the following:

		func.	inj.	surj.	bij.
$f(x) = x$	$f : \mathbb{N} \mapsto \mathbb{Z}$	$\Diamond$	$\Diamond$	$\Diamond$	$\Diamond$
$g(x) = x$	$g : \mathbb{Z} \mapsto \mathbb{Z}$	$\Diamond$	$\Diamond$	$\Diamond$	$\Diamond$
$h(x) = x^2$	$h : \mathbb{R} \mapsto \mathbb{N}$	$\Diamond$	$\Diamond$	$\Diamond$	$\Diamond$
$i(x) = \lfloor x^2 \rfloor$	$i : \mathbb{R} \mapsto \mathbb{N}$	$\Diamond$	$\Diamond$	$\Diamond$	$\Diamond$
$j(x) = x^3 - x$	$j : \mathbb{Z} \mapsto \mathbb{R}$	$\Diamond$	$\Diamond$	$\Diamond$	$\Diamond$

Solution: $\blacklozenge$ indicates YES, $\Diamond$ indicates NO

		func.	inj.	surj.	bij.
$f(x) = x$	$f : \mathbb{N} \mapsto \mathbb{Z}$	$\blacklozenge$	$\blacklozenge$	$\Diamond$	$\Diamond$
$g(x) = x$	$g : \mathbb{Z} \mapsto \mathbb{Z}$	$\blacklozenge$	$\blacklozenge$	$\blacklozenge$	$\blacklozenge$
$h(x) = x^2$	$h : \mathbb{R} \mapsto \mathbb{N}$	$\Diamond$	$\Diamond$	$\Diamond$	$\Diamond$
$i(x) = \lfloor x^2 \rfloor$	$i : \mathbb{R} \mapsto \mathbb{N}$	$\blacklozenge$	$\Diamond$	$\blacklozenge$	$\Diamond$
$j(x) = x^3 - x$	$j : \mathbb{Z} \mapsto \mathbb{R}$	$\blacklozenge$	$\Diamond$	$\Diamond$	$\Diamond$

Hash tables are an important application of functions in computer science. A hash table is a data structure; it is a fancy array. A hash table has an internal array, and a hash function. This hash function maps table elements (some set of objects) to indices ($\mathbb{N}$). The modulus of the hash function output, with respect to the table size, is taken as the input object's key. This is where the object is stored in the table. Ideally, you want your hash function to be an injection on the set of indices.

5.3 Sequences

Sequences make up an interesting sub-part of functions. In calculus, we build up continuous functions as limits of sequences. This is discrete mathematics, however, so sequences will be of more interest to us.

Definition 5.3.1 Sequence
An ordered list of numbers, each one associated with a specific *position* (index)

For the purposes of this class, you can think of a sequence as a mapping between the natural numbers and the associated index in the sequence. Sometimes you can express a sequence as a closed-form function, and sometimes a sequence can have multiple interpretations. Let's look at a few examples to get the notation down.

Example: The following three notations describe the same sequence:
(a) $1, 3, 7, 15, 31, \ldots$

(b) $a_n = \begin{cases} 1 & , \ n = 1 \\ 2a_{n-1} + 1 & , \ n > 1 \end{cases}, \ n \in \mathbb{Z}^+$

(c) $f(n) = 2^n - 1$ where $f : \mathbb{Z}^+ \mapsto \mathbb{Z}^+$

We have explicit names for the notation types in the previous example.

Definition 5.3.2 General Form
A written list of the elements of a sequence, in order, typically with ... at the end (and at the beginning, if your sequence elements start in the middle – typically this does not happen). Sometimes mathematicians put curly braces {} around their general-form sequences – try not to get this confused with a set! Other times mathematicians will give you a rule, written in plain English, for generating sequence terms

Definition 5.3.3 Recursive Form
An explicit formula that depends on previous elements in the sequence, and necessarily has one or more base cases. Typically this is expressed as

$$x_n = \begin{cases} \text{base case(s)} & , n \in \{\text{some set}\} \\ \text{some formula dependent on } n & , n \notin \{\text{the previous set}\} \end{cases}$$

Definition 5.3.4 Closed Form
An explicit formula to generate a specific term in a sequence. This formula is solely dependent on a given index. Typically this is expressed as

$$y_k = \text{some formula dependent on } k$$

Remark 5.3.5
Sequence numbers do not need to be in any particular order, nor do they necessarily have to fit some sort of function. Sequences elements can also be any sort of number (we will restrict to $\mathbb{R}$).

Example: Some sequences:

$$1, 0, -1, 0, 1, 0, -1, 0, \ldots$$

$$8, 6, 4, 2, 8, 6, 4, 2, 8, \ldots$$

$$0, 0.5, 0.333, 0.25, 0.2, 0.125, \ldots$$

Sometimes we will ask you to generate terms of a sequence. This is a relatively straightforward task so long as you avoid simple arithmetic errors.

Example: Generate the first 7 terms of the following sequences:

(a) The nth term is its index tripled (start your indices at 0)

(b) $f_k = \begin{cases} 1 & ,k = 1,2 \\ f_{k-1} + f_{k-2} & ,k \geq 3 \end{cases}$

(c) $l_1 = l_2 = l_3 = 1$ and $l_i = l_{i-1} + l_{i-2} + l_{i-3}$

Solution: We will use $\{\}$ notation here:

(a) A simple rule (one that we could express in closed form) – $\{0, 3, 6, 9, 12, 15, 18, \dots\}$

(b) This is the Fibonacci sequence – $\{1, 1, 2, 3, 5, 8, 13, \dots\}$

(c) In this case, the index is implied to be $\in \mathbb{Z}^+$ – $\{1, 1, 1, 3, 5, 9, 17, \dots\}$

Other times we will ask you to find a closed form for a sequence given in general or recursive form. Typically we ask this about general forms, however occasionally you will see a recursive form. This task is akin to pattern-matching and sometimes requires much more critical thinking. Sometimes sequences can have multiple answers – any correct answer suffices. Answers that are unreasonable – i.e. answers that only fit the provided sequence digits – will receive a good chuckle along with zero marks.

Example: Find closed-forms for the following sequences:

(a) $\{1, 4, 9, 16, 25, \dots\}$

(b) $\{3, 6, 9, 12, 15, \dots\}$

(c) $\{1, 4, 9, 6, 5, 6, 9, \dots\}$

(d) $\{1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 4, \dots\}$

(e) $\{2, 5, 8, 0, 3, 6, 9, \dots\}$

Solution: We attempt to explain the patterns that led us to the solutions:

(a) The pattern here is squares. Each sequence number is the square of its index

(b) $f(n) = 3(n + 1)$ where $f : \mathbb{N} \mapsto \mathbb{Z}$

(c) Hopefully you still had the first sequence in mind – squares modulo 10

(d) Notice a pattern with the indices – specifically look at the index where there is a switch. Index 1 starts the 1s, index 4 starts the 2s, index 9 starts the 3s, index 16 starts the 4s. What is the relationship between these numbers? Take the square root of the index. Then realize that square roots maintain inequalities – $1 = \sqrt{1} < \{\sqrt{2}, \sqrt{3}\} < \sqrt{4} = 2$, so look towards floors and ceilings to yield integers. Thus, $f(n) = \lfloor \sqrt{n} \rfloor$ where $f : \mathbb{Z}^+ \mapsto \mathbb{Z}^+$

(e) This one can have multiple answers. One solution: $f(n) = 3n - 1 \pmod{11}$ (taking least positive remainder) where $f : \mathbb{Z}^+ \mapsto \mathbb{Z}^+$. Typically if you see a somewhat circular sequence, especially if there is a zero, then consider applying a modulus. Start at the zero and look forward until the sequence reverts. In this case, we have $0, 3, 6, 9$ which is simply $f(n) = 3n$. Then play around with the start point and the modulus

Finding a closed-form function that matches a sequence is difficult. It requires a deep understanding of how all mathematical operators work, and requires a knack for finding patterns. There are multiple methods for estimating functions, including polynomial interpolation, piece-wise functions, floor-ceiling interpolation, and least-squares. These methods usually yield a result significantly more complicated than finding a simpler function. However, there are merits to using these methods as a starting-point.

5.3.1 Series

Definition 5.3.6 Series
The sum of all terms in a sequence

Typically we try to understand the limiting behavior of infinite series. There are a number of tests you can use to study this behavior, however this topic is typically taught in Calculus II classes. We will thus omit this topic here, however we do recommend at least understanding the idea behind limits. We will come back to infinite objects when we discuss countability in chapter 6.

5.4 Relations

Relations are a formal way for us to map elements together in a similar fashion to functions. Unlike functions, however, a relation simply gives the *relationship*. Sometimes you can construct a function that yields the relation. Other times, a function that would represent the relation would not be a function by definition.

Definition 5.4.1 Relation
For sets A and B, a relation R is any subset of the cross product of A and B:

$$R \subseteq \{(a, b) \in A \times B\}$$

To specify that two elements relate to each other under a relation R we say xRy which means $(x, y) \in R$

Here are a few examples:

Example: The *equality* relation $\{(a,b) \mid a=b\} \subseteq \mathbb{R} \times \mathbb{R}$

Remark 5.4.2
We will only consider relations of sets with themselves. We *can* define relations between different sets, however this will make the following definitions significantly more challenging to define.

- Relation - (Binary Relation) if A and B are sets, then a Relation from A to B is any subset of $A \times B$. aRb means $(a,b) \in R$ where $a \in A$ and $b \in B$ and $R \subseteq A \times B$. A Relation over a set A is simply a Relation from A to A.
- Reflexive - a Relation R on a set A is Reflexive if $(\forall a \in A)[(a,a) \in R]$.
- Symmetric - a Relation R on a set A is Symmetric if $(\forall a,b \in A)[(a,b) \in R \Rightarrow (b,a) \in R]$.
- Antisymmetric - a Relation R on a set A is Antisymmetric if $(\forall a,b \in A)[(a,b) \in R \wedge (b,a) \in R \Rightarrow a=b]$.
- Transitive - a Relation R on a set A is Transitive if $(\forall a,b,c \in A)[(a,b) \in R \wedge (b,c) \in R \Rightarrow (a,c) \in R]$.
- Composite - (just something cool to note) if R is a Relation from sets A to B and S is a Relation from B to C, then the Composite of R and S, denoted $S \circ R$, is $\{(a,c) \mid a \in A \wedge c \in C \wedge (b \in B \Rightarrow (a,b) \in R \wedge (b,c) \in S)\}$
- Powers - (just something cool to note) if $R \subseteq A \times A$, then for $n \in \mathbb{N}^{\geq 1}, R^n = R^{n-1} \circ R$ where $R^1 = R$.

  $R \subseteq A \times A$ is Transitive iff $R^n \subseteq R$ (this is a theorem from your textbook)
- Equivalence Relation - a Relation R on a set A is an Equivalence Relation if it is Reflexive, Symmetric, and Transitive.

Definition 5.4.3 Reflexive
A relation R on a set A is reflexive if

$$(\forall a \in A)[(a,a) \in R]$$

Definition 5.4.4 Symmetric

Definition 5.4.5 Transitive

Segway to an example

Example: exaples

Solution: solutions

Segway to equivalence relations

Definition 5.4.6 Equivalence Relation
A relation that is reflexive, symmetric, and transitive

I guess done?

Put something discussing n-ary relations

5.5 Summary

-
-
-

5.6 Practice

1. question

Chapter 6

Countability

$\aleph_0$ is my favorite character!

Justin Goodman

Contents

6.1 Introduction

Our motivation here is that we want to describe the sizes of incredibly large sets. What is the cardinality of $\mathbb{N}$? So far, we have not learned any means to count the natural numbers. That is what this chapter is for!

Additional reading: The Banach-Tarski Paradox – Vsauce (relevant content starts at 2:06)

6.2 Definitions

Definition 6.2.1 Enumeration of a Set
A specified ordering of a set

Example: We could enumerate the set of integers as follows:

$$0, 1, -1, 2, -2, 3, -3, 4, -4, \cdots$$

Definition 6.2.2 Finite
A set is finite if there exists a bound on the number of elements. If you start counting the elements in a finite set, there should be a precise stopping point

Example: $\{1, 2, 3\}$ and $\mathbb{N}^{<10,000,000,000}$ are finite sets

Definition 6.2.3 Countably Infinite
A set is countably infinte if you can find a one-to-one correspondence (bijection) between the set and $\mathbb{N}$. Alternatively, a set is countably infinite if you can enumerate the set such that every element appears at a finite position

Remark 6.2.4
We denote $|S| = \aleph_0$ for any (every) countably infinite set S

There exist other definitions for countably infinite, however they all share the same idea that *countably infinite* sets can be *counted in a finite amount of time.*

Definition 6.2.5 Countable
A set is countable if it is finite or countably infinite

Example: $\mathbb{N}$ is countably infinite

Proof. The function $f(x) = x$ where $f : \mathbb{N} \mapsto \mathbb{N}$ is a bijection □

Example: $\mathbb{Z}$ is countably infinite

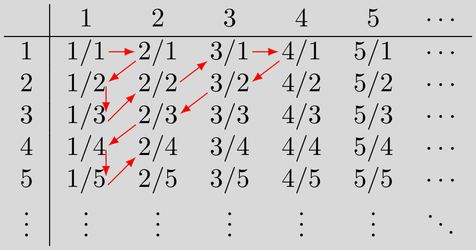
Proof. Enumerate $\mathbb{Z}$ as follows: $\{0, -1, 1, -2, 2, -3, 3, \cdots\}$
Then

$$f(x) = \begin{cases} \frac{n}{2} & x \equiv 0 \pmod{2} \\ -\frac{n+1}{2} & x \equiv 1 \pmod{2} \end{cases}$$

where $f : \mathbb{N} \mapsto \mathbb{Z}$ is a bijection □

Example: $\mathbb{Q}^+$ is countably infinite

Proof. We use what is called a *snaking argument*. The idea is to build a grid that describes all positive rational numbers, then provide a bijection between that grid and the natural numbers.
Grid:



	1	2	3	4	5	⋯
1	1/1	2/1	3/1	4/1	5/1	⋯
2	1/2	2/2	3/2	4/2	5/2	⋯
3	1/3	2/3	3/3	4/3	5/3	⋯
4	1/4	2/4	3/4	4/4	5/4	⋯
5	1/5	2/5	3/5	4/5	5/5	⋯
⋮	⋮	⋮	⋮	⋮	⋮	⋱

Let $f(n)$ yield the rational number at position n along the snake in the table (e.g. $f(0) = 1/1$, $f(1) = 2/1$, $f(2) = 1/2$). Then $f : \mathbb{N} \mapsto \mathbb{Q}^+$ is a bijection □

Note: this proof does not account for the fact that we can reduce fractions (like how $4/2 = 2/1$*). A more rigorous proof uses the Schröder-Bernstein Theorem, which is out of the scope of this course*

Definition 6.2.6 Uncountable
A set that is not countable

6.3 Cantor's Diagonal Argument

We begin our study of this famous proof by diving into the actual proof.

Example: Prove $(0, 1)$ is uncountable

Proof. Assume $(0, 1)$ is countable. Then, we can enumerate the entire set. One possible enumeration follows:

0.	0	1	2	6	5	9	8	7	⋯
0.	1	8	4	3	1	3	0	8	⋯
0.	1	9	9	5	9	9	6	6	⋯
0.	1	9	0	3	2	5	2	7	⋯
0.	1	9	0	4	2	0	4	1	⋯
0.	1	9	0	4	3	3	8	2	⋯
0.	1	9	0	4	3	4	8	2	⋯
⋮									

□

6.4 Useful Theorems

We give some useful theorems of countable and uncountable sets. These will help you in determining whether a set is countable or uncountable.

6.5 Summary

-
-
-

6.6 Practice

1.
2.

Chapter 7

Graph Theory

> The origins of graph theory are humble, even frivolous.
>
> Norman L. Biggs

Contents

7.1 Introduction

Graph theory is traditionally introduced using the Seven Bridges of Königsberg (see fig. 7.1). The scene is Königsberg, Prussia, 1700s. Leonhard Euler sees this area – two land masses separated by a river, with two islands in-between, all connected by 7 bridges – and thinks, *hmm, I wonder if I could walk through this city crossing each bridge only once.* Euler mathematically proved that no such walk exists due to the nature of this area. This was the birth of graph theory.

Euler did not brute-force the solution by examining all possible paths. Instead, Euler noticed that he could abstract out the large landmasses into single points, called vertices, and connect them by simple lines, called edges, which represent

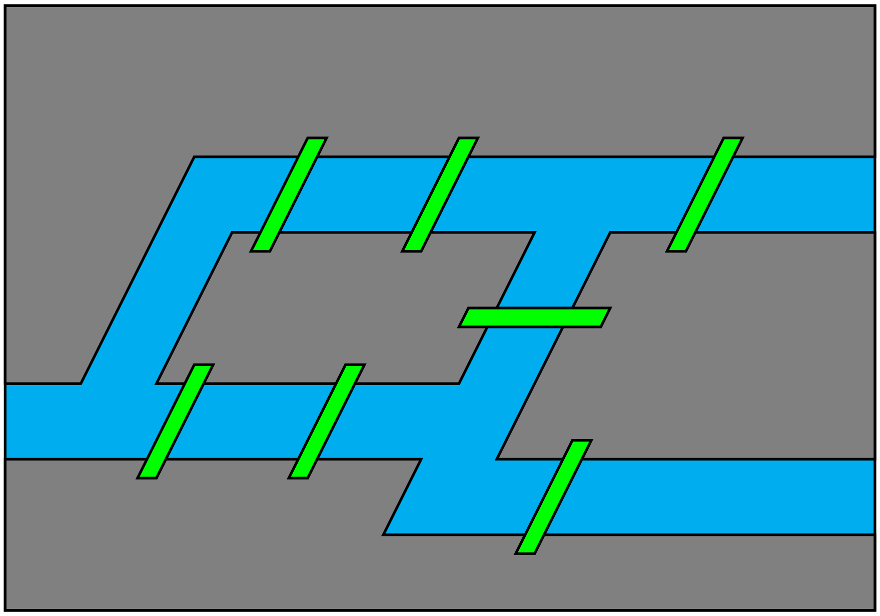


Figure 7.1: The Seven Bridges of Königsberg

the bridges. This mathematical structure is a *graph*. The graph version of the Seven Bridges is presented in figure 7.2.

The beautiful thing about graphs is that we can move the vertices around in space, and so long as the edges remain connected to their respective vertices, without changing the fundamental structure of the graph!

Graphs are applicable in numerous fields, including

- Network (computer, water, electric) flow
- Facebook friendships
- Google Maps path-finding
- E-commerce similarity predictions
- Neural networks
- ...

7.2 Key Terms

Definition 7.2.1 Graph
A set of abstract points, called vertices V, and connections between those points,

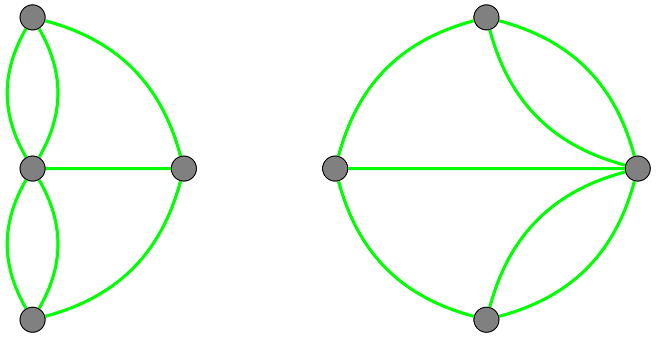


Figure 7.2: Two graphified versions of The Seven Bridges of Königsberg. The left graph is roughly a direct translation of the actual landmasses. The right graph is the same graph with the vertices moved around (take the middle-left vertex in the first graph, move it out to the left, then rotate the resulting graph 180 degrees)

called edges E. We denote a graph as $G = \{V, E\}$ or $G = (V, E)$. The order that you list V and E does not matter, however typically V is listed before E

Definition 7.2.2 Vertex

Definition 7.2.3 Edge
Directed vs Un-directed

Definition 7.2.4 Di-graph
A graph where each edge is directed.

Definition 7.2.5 Cycle
A thing. Only defined for directed edges

Definition 7.2.6 DAG
A Directed Acyclic Graph – a di-graph with no cycles

7.3 Graph Representations

There exist multiple ways to represent a graph. Each way has benefits and drawbacks. Typically algorithmic analysis involving graphs uses the representation that provides the best run-time.

Definition 7.3.1 Adjacency Matrix
An $m \times m$ (square) symmetric matrix M of 0s and 1s. Each row/column represents a node, and the position i, j in M represents the connection. For an edge

7.4 Problems

7.5 Summary

7.6 Practice

Chapter 8

Asymptotic Analysis

In algorithms, as in life,
persistence usually pays off.

Steven S. Skiena

Contents

8.1 Introduction

What is an algorithm?

Definition 8.1.1 Algorithm
A recipe/description of a process that accomplishes a goal

Why study algorithms? First, you will gain tools to *compare* algorithms. Second, studying algorithms will help you to creatively make *new* algorithms.

Why study algorithms in discrete math? Computers are discrete. Computers run programs, which are based on algorithms. Hence, algorithms must be discrete.

8.2 $\mathcal{O}$, Ω, and Θ Notations

Our motivation here is to describe the *growth-rate* of functions. The growth-rate class gives us key insights to what the actual function does on large inputs.

Definition 8.2.1 Big-Oh
An upper-bound on the growth-rate of a function. Formally:

$$f(n) \in \mathcal{O}(g(n)) \Leftrightarrow (\exists n_0 \in \mathbb{N}, \exists c \in \mathbb{R}^+, \forall n \in \mathbb{N}^{\geq n_0})[f(n) \leq c \cdot g(n)]$$

Definition 8.2.2 Big-Omega
A lower-bound on the growth-rate of a function. Formally:

$$f(n) \in \Omega(g(n)) \Leftrightarrow (\exists n_0 \in \mathbb{N}, \exists c \in \mathbb{R}^+, \forall n \in \mathbb{N}^{\geq n_0})[f(n) \geq c \cdot g(n)]$$

Definition 8.2.3 Big-Theta
An exact growth-rate of a function. Formally:

$$f(n) \in \Theta(g(n)) \Leftrightarrow f(n) \in \mathcal{O}(g(n)) \wedge f(n) \in \Omega(g(n))$$

8.3 Analyzing Algorithms

We can express algorithms in terms of functions. The function input is typically n, the number of elements inputted into the algorithm, and the function output is typically time. Usually *time* will be the number of certain operations performed during the algorithm.

Example: Come up with a function $T(n)$ that describes the run-time of the following algorithm. Define run-time as the number of conditional evaluations. In the following algorithm, L is a list of integers of length n.

```
function GetMax($L$, $n$)
    $M \leftarrow -\infty$
    $i \leftarrow 0$
    while $i < n$ do
        if $L[i] > M$ then
            $M \leftarrow L[i]$
        end if
        $i \leftarrow i + 1$
    end while
    return $M$
end function
```

Solution: We evaluate the if-statement in line 5 exactly n times, and we evaluate the while-loop condition exactly $n+1$ times (the last time is when we break out of the loop). Thus, $T(n) = n+n+1 = 2n+1$

Now come up with a function $U(n)$ that outputs the number of $\leftarrow$ operations. Compare $T(n)$ to $U(n)$.

Solution: We have two initial stores from lines 1 and 2. Each while-loop iteration has a store command in line 8. The while-loop runs exactly n times, so this gives us n stores. Line 6 has a store command which is run $\leq n$ times. We can model this using a natural number $c \leq n$ where c is the number of times line 5 evaluates to true. Our function is thus $U(n) = n + c + 2$
Since $c \leq n$ we have that $U(n) \leq T(n)$ for all $n > 1$. Note that when $n = 1$ we have $T(n) = 3$ and $U(n) = 4$ (since $c = 1$). In any case, we have that the growth-rates of both functions are solely dependent on n and are hence the same. Both functions are in $\Theta(n)$

Definition 8.3.1 The Loop-Heuristic
(algorithm analysis) For each nested loop in an algorithm, multiply n by the number of nested levels. This can be helpful in an initial analysis, however it will only take you so far.

Remark 8.3.2
We state the loop-heuristic as a definition, however it can be rephrased as a provable theorem.

It is important to think about what is going on during the algorithm instead of blindly following the loop-heuristic. We present a few related examples.

Example: Analyze the following algorithm:

```
function AnalyzeList($L$, $n$)
    $M \leftarrow 0$
    for $i \leftarrow 0$; $i < n$; $i \leftarrow i + 1$ do
        $k \leftarrow L[i]$
        for $j \leftarrow i$; $j < n$; $j \leftarrow j + 1$ do
            $k \leftarrow k * L[j]$
        end for
        $M \leftarrow M + k$
    end for
    return $M$
end function
```

Solution: The algorithm contains 2 nested loops, so by the loop-heuristic our run-time is n^2. We examine further. The outer loop goes through the list n times. For each iteration i, we have a loop that goes through the list $n-i$ times. The actual *work* is done in the inner loop. Thus, the amount of work equals $1+2+3+\cdots+n = \frac{n(n+1)}{2} = \frac{1}{2}n^2 + \frac{1}{2}n$ by the Gaussian sum. This has a growth rate of n^2, so we see that the loop-heuristic worked and the algorithm runs in $\mathcal{O}(n^2)$. In fact, the amount of work does not change with respect to different input ordering, so the algorithm runs in $\Theta(n^2)$

Example: Examine the following algorithm:

```
function AnalyzeGraph($G = (V, E)$)
    $c \leftarrow 0$
    for $v \in V$ do
        for $w \in$ Neighbors($v$) do
            $c \leftarrow c + 1$
        end for
    end for
    return $c$
end function
```

Solution: G represents a graph

Example:

8.4 Common Complexities

There are a handful of common run-time complexities that you should be familiar with. In this book, we give them to you in ascending order of growth rate in Big Theta notation.

Definition 8.4.1 Constant

$$\Theta(1)$$

Definition 8.4.2 Logarithmic

$$\Theta(\log n)$$

Definition 8.4.3 Linear

$$\Theta(n)$$

Definition 8.4.4 Polynomial

$$\Theta(n^c),\ c > 1$$

We call $\Theta(n^2)$ Quadratic

Definition 8.4.5 Exponential

$$\Theta(c^n),\ c > 1$$

8.5 Summary

-
-
-

8.6 Practice

1. Analyze the following algorithm:

   ```
   function SomeFunc($n$)
       something
   end function
   ```

2. Order the following complexities: $\Theta(n \log n)$, $\Theta(n \log(n^2))$, $\Theta(n)$, $\Theta(n^2)$. Explain why your ordering is correct.

3. Order the following complexities: $\Theta(\log(3n))$, $\Theta(n^n)$, $\Theta(n!)$, $\Theta(\log(\log n))$.

Chapter 9

Conclusion

So, that's it?

Justin Goodman

Intro outro content.

9.1 Closing Remarks

Some content here

9.2 Summary

- This is a long and difficult course
- Be proud of yourself

Index