Discrete Mathematics

Tamir Enkhjargal

September 2017

Contents

Chapter 1

Introduction

Computer Science has several foundational roots from Mathematics. Several applications of mathematics, not limited to just algebra, but statistics and probability, matrices and structures, and more are all important in programming and Computer Science.

Most examples and information is being relayed from Yale's 2013 *CS202: Mathematical Tools for Computer Science* course notes. Any use or distribution of this or their work is protected by "Fair Use" for Educational Purposes under Copyright Laws.

1.1 Mathematics and Its Roots in Computer Science

The requirement of knowing Mathematics in Computer Science comes down to the relative difficulty of computations. Computations are as simply put, the process and rules that a computer follows to fulfill any problem. As necessary, all computations must follow a consistent, strict, and uncomplicated rules.

1.2 What You Already Know

The level of difficulty of mathematics will come down to your own mathematical reasoning abilities. However, I will break the mathematical reasoning down to a simple logical reasoning level at best.

For example, there is very little *reasoning* difference between:

1. If x is in S, then $x + 1$ is in S.
2. If x is of royal blood, then $x's$ child is of royal blood.

The second statement is easier to make sense of and understand. We can easily understand that anyone blood related to the King is also royalty.

The first statement is structurally and logically very similar, but possibly doesn't make sense in the same way statement two does. It's all about your understanding of mathematical reasoning. If you want, you can think of x being any person, and fitting into a description S, defined by *royal blood, race, etc.*, then any child of x will also be of the same *royal blood, race, etc.*

The benefits of these notes will come as benefits to your mathematical reasoning and logic, as well as the vice versa effect on logical reasoning, and how to convert logic into math.

1.3 Applications of Discrete Mathematics

Applications of Discrete Mathematics are far and wide, but these will mostly appear in Computer Science, Engineering, and Cryptography:

1.3.1 Foundations and Logic

This is where everything begins. The "starter code" of all mathematics.

- Propositional Logic*
- Predicate Logic*
- Axioms, Theories, and Models*
- Proofs*
- Induction and Recursion

1.3.2 Basic Mathematics and Functions

A necessity in all mathematics.

- Standard functions and their special properties. Addition, Multiplication, Exponents, Logarithms, etc.*
- Special Functions, such as Maximum, Minimum, Floor, Ceiling.
- Inequalities
- Subsets of Real Numbers: Rationals, Integers, and Natural Numbers.
- Induction and Recursion

1.3.3 Objects in Mathematics

How to organize large data structures and objects with Mathematics.

- Set Theory*
- Functions on Sets*

- Relations*
- Universes of Mathematics*

1.3.4 Modular Arithmetic and Polynomials

This is the mathematical basis of modern cryptography

- Modular Arithmetic
- Primes and Factorization
- Euclid's Algorithm
- Chinese Remainder Theorem, Fermat's Little Theorem, and Euler's Theorem
- RSA Encryption

1.3.5 Linear Algebra

Matrices and Vectors will appear in many places, especially in C programming and engineering mathematics.

- Vectors and Matrices
- Matrix Operations and Functions
- Elimination Methods and Inverses

1.3.6 Probability and Counting

Counting is useful for combinatorics and computations. Useful especially for resource and memory management.
Probability is used for advanced algorithms and analyses.

- Combinatorial Counting: Sums, Products, Exponents, etc
- Combinatorial Functions
- Probability Events and Independence
- Expectation and Variance

Chapter 2

Mathematical Logic

This is the set of followed rules in mathematics, which mathematicians have created as the universal language of mathematics.

2.1 Modeling Mathematics Into the Real World

An example of how the real world can be put into a mathematical model and theory are as shown:

Real		Model		Theory
tally marks city population number of rocks	$\Longrightarrow$	$\mathbb{N} = \{0, 1, 2, \ldots\}$	$\Longrightarrow$	$\forall x \colon \exists y \colon y = x + 1$

2.2 Basic Terminology

In the above example, we model the real world examples with the **Natural Numbers** model. However, what can we say about the natural numbers? How do we know any number exists in the natural numbers without a set of rules to define the model?

This is approached by the use of **axioms**, or given true statements about the model, and a list of **inference rules** that we can use to derive more information about the axioms. These axioms and inference rules combine to turn into a **theory**, that consists of all of the statements that can be created from the axioms and inference rules. Any some statement that we claim is true from a theory is a **theorem**, something we assume or derive as a part of the theory.

Using the course notes' example of axioms and inference rules:

All fish are green	Statement (Axiom)
George Washington is a fish	Statement (Axiom)
"All X are Y" and "Z is an X" implies "Z is a Y"	Inference Rule
Therefore, George Washington is green	Culmination of Steps 1-3 (Theorem)

Theories try to describe **Models**. A model is a set or collection of objects that are somewhat related to each other.

2.3 Consistency

Consistency is very important when it comes to theories. **Consistency** means that a theory cannot prove both P and not-P for any P. To put it simply, there are no contradictions to be found with a theory.

An example of what can go wrong, using the same example:

All fish are green.
All sharks are **not green**.
All sharks are fish.
George Washington is a fish.

As you can see, many problems can arise if you add too many axioms that we claim to be true. Stating sharks are not green, but all fish are green contradict with the truth that sharks are also fish. We also don't even know where the George Washington statement lies. Therefore, we will stick to axioms that have been proof-read and debugged of all inconsistencies.

Onto the basis of mathematical logic, the simplest form of logic is **Propositional Logic**, which was invented by Aristotle. Propositional logic is the model that uses a collection of **statements** that can be **true** or **false**. This is a starting point of logical thinking. We can only make statements, and see what we can infer from those statements.

The next level of logic is the **Predicate Logic**. Predicate logic adds **constants** and **predicates**. Constants are variables you can use instead of "George Washington", you typically use x or y. Predicates are the equivalent of properties of constants, such as "is a fish".

Additionally, with predicate logic, you get functions and equality. You are able to make functions such as $f(x)$ = the number of times Bob sends a message to

Alice. You are also even able to make statements such as "Bob" = 5, and apply inference rules to imply "Bob + 5 = 10". Predicate logic is sufficient enough to solve and use all of modern mathematics.

2.4 General Axioms and Models

Mathematicians have debugged and solved their own axioms and models, which we know as modern mathematics. They are well known to be consistent and universal.

We will now cover the most popular models that are known to all mathematicians:

- $\mathbb{N}$: The **Naturals**. These are defined with the *Peano Axioms*, which allows us to count, add, and multiply. What's missing is that we can't subtract, since negatives don't exist in the natural numbers.
- $\mathbb{Z}$: The **Integers**. These are the same as the naturals, with the addition that we can subtract. We can't divide with integers.
- $\mathbb{Q}$: The **Rationals**. We can divide integers now, but irrationals exist outside of this model.
- $\mathbb{R}$: The **Reals**. We are able to solve things such as $\sqrt{2}$ and operations with π. What about the imaginary numbers, i?
- $\mathbb{C}$: The **Complexes**. We are just about able to model any number that exists. However, what if we want to define more than one complex number at the same time?
- The **Universe of Sets**. These are defined using the axioms of set theory, which can hold any collection of naturals, rationals, complexes, etc. as much as we want. These sets can even go into infinity, but we'll stick to finite sets.
- There are other models that can also answer the question for collections of objects. These alternatives include lambda calculus, category theory, or second-order arithmetic. However, most of these are ignored or skipped, since set theory is able to do anything these models can do. You might encounter these if you are looking into the basis of programming language theory.

Chapter 3

Propositional Logic

If you remember, propositional logic is the model that uses a collection of true or false statements called **propositions**. Propositions may also never contain a variable, such as x.

Examples of propositions:

- $2 + 2 = 4$ (always true)
- $2 + 2 = 2$ (always false)

Examples of non-propositions:

- $x + 2 = 4$ (may or may not be true, depends on the value of x)
- $x \cdot 0 = 0$ (always true, but not a proposition because it contains x)
- $x \cdot 0 = 1$ (always false, but still not a proposition)

3.1 Operations

You can do operations on propositions, usually through logical operators. Disclaimer: propositions are so simple by themselves, logicians and mathematicians by themselves use placeholders p, q, or r to indicate a proposition. Remember that these also take on the boolean value, true or false. The logical operators include:

Negation: The simplest of the operators. The **negation** of p is $\neg p$, or sometimes references as p, $-p$, or $\overline{p}$. The programmers equivalent would be $!p$. These are true when p is false, and false when p is true.

Or: The **or** between the propositions p and q is written as $p \vee q$ and is true as long as *one of the propositions is true.* It is false when *both of the propositions* are false. This is also logically the **inclusive or**.

Exclusive Or: The **exclusive or**, or **XOR** between the propositions p and q is written as $p \veebar q$. It returns true, if and only if **one** of the propositions is true.

And: The **and** between p and q is written as $p \wedge q$, and is only true when both of p **and** q are true.

Implication: This is the most important operator when dealing with proofs. An **implication** is closest to the English "if... then" phrase. If p **implies** q, then $p \implies q$ or $p \to q$. Implications are can be very easily true, and difficult to be false. A false proposition implying a false proposition is *true*, a true proposition implying a proposition makes the second one *true*. For example:

"If $2 + 2 = 5$, then my name is Bob" is true.

If $2 + 2 = 4$, then $3 + 3 = 6$ is also true.

Biconditional: Biconditionals exist when $p \to q$ *and* $q \to p$, and therefore becomes $p \leftrightarrow q$. Note that $p \leftrightarrow q$ is equivalent to $q \leftrightarrow p$. This is only true when **both propositions are true** or **both propositions are false**. This is the English equivalent of "If and only if..."

Logical Operator	General Symbol	Alternatives
NOT p	$\neg p$	$\sim p$, $\overline{p}$, $!p$
p AND q	$p \wedge q$	$p \,\&\, q$
p XOR q	$p \veebar q$	$p \oplus q$
p OR q	$p \vee q$	$p \parallel q$
p implies q	$p \to q$	$p \implies q$
p $\mathbb{IF}$ q	$p \leftrightarrow q$	$p \iff q$

This table shows all of the **compound propositions**, and is ordered in terms of strength of precedence. $\neg$ is the strongest while $\leftrightarrow$ is the weakest.

3.1.1 Precendence Rules

The ordering of precedence, as seen in the table is closest to the logical ordering in C-like programming languages. Here is an example of how precedence works in action:

$$\neg p \wedge q \vee r \to s \leftrightarrow t$$

is the exact same as

$$(((\neg p) \wedge (q \vee r)) \to s) \leftrightarrow t$$

Some other rules about these logical operators is that AND and OR are both **associative**, meaning that $(p \wedge q \wedge r)$ is the same as $p \wedge (q \wedge r)$ and $(p \wedge q) \wedge r$. The exact same thing applies to $(p \vee q \vee r)$. Implication is not completely associative, but it is **right associative** so $p \to q \to r$ is the same as $p \to (q \to r)$. If there is any other issue with how precedences work, make sure to put as many parentheses as much as you need to.

3.2 Truth Tables

So far, in propositional logic, we have defined what kind of operations we can use with propositions, with precedence, or the order of operations with propositions defined. Now we can create a **truth table**, which is the output of compound propositions given each kind of input.

For example the truth table for the negation of p is:

p	$\neg p$
0	1
1	0

And this is the truth table for each logical operator:

p	q	$p \wedge q$	$p \veebar q$	$p \vee q$	$p \rightarrow q$	$p \leftrightarrow q$
0	0	0	0	0	1	1
0	1	0	1	1	1	0
1	0	0	1	1	0	0
1	1	1	0	1	1	1

Although these truth truth tables are extremely slow to input and output by hand, a computer would handle all of these in mere milliseconds. On another fact, these truth tables are the stand-in **model** for propositional logic. Remember a model is the all-together collection of objects, and by creating every single possibility through a truth table, we are **model checking**. Truth tables are also useful in solving long and complex compound propositions.

In predicate logic, model checking becomes much more complicated, because we need to also check each combination of systems of axioms, resulting in infinitely many models, all of which could be infinitely large. Predicate model checking is thus done mostly through applying inference rules.

3.3 Propositional Tautologies and Equivalences

Additionally, when compound propositions are found to **always be true**, regardless of p or q, then it is called a **tautology**. When compound propositions are found to be **always false**, it is called a **contradiction**. The opposite of a tautology is a contradiction and vice versa. Tautologies are useful when comparing compound propositions to be **logically equivalent**. This means that $X \leftrightarrow Y$, given X and Y are both compound propositions, resulting in the ability to substitute these in for another. This also means we can write $X \equiv Y$. The main use of these **logical equivalences** is that we can construct a truth table to show tautologies using previously-known logical equivalences.

Examples of such equivalences are:

- $p \wedge \neg p \equiv 0$ using the truth table:

p	$\neg p$	$p \wedge \neg p$	0
0	1	0	0
1	0	0	0

  As you can see, the last two columns are always equal, meaning that $p \wedge \neg p$ and 0 are logically equivalent

- $p \vee p \equiv p$ using the truth table:

p	$p \vee p$
0	0
1	1

- $p \to q \equiv \neg p \vee q$ again using the table:

p	q	$p \to q$	$\neg p \vee q$
0	0	1	1
0	1	1	1
1	0	0	0
1	1	1	1

This leads us to be able to solve things such as:

$$\begin{aligned}(p \to q \leftrightarrow \neg p \vee q) \wedge (p \vee p \leftrightarrow p) &\equiv ?\\ (1) \wedge (1) &\equiv ?\\ 1 &\equiv 1\end{aligned}$$

We were able to solve that because we knew each of the biconditional comparisons were comparing logically equivalent propositions, so we can easily lead them to be always true and always true.

3.4 Contrapositives, Inverses, and Converses

The **contrapositive** of $p \to q$ is $\neg q \to \neg p$, and is logically equivalent to each other. Using an example, "If I am human, then I am a mammal" is logically equivalent to "If I am **not** a mammal then I am **not** human". Contrapositives are very useful in proofs, specifically through the process of **proof by contraposition**, which shows that p implies q by assuming $\neg q$ and then proving $\neg p$. It is also similar to an **indirect proof**.

The **inverse** of $p \to q$ is $\neg p \to \neg q$. For example, the inverse of "If you are human, you must be a mammal" is "If you are not human, you are not a mammal".

As you can see, the validity of the propositions does not often match with the validity of the inverse.

The **converse** of $p \to q$ is $q \to p$. The converse of "If I am human then I am a mammal" is "If I am a mammal then I am human". As you see, the implication is not logically equivalent to the converse. However, the converse is **always logically equivalent** to the inverse. "If I am a mammal then I am human" is logically equivalent to "If you are not human , you are not a mammal".

$$q \to p \equiv \neg p \to \neg q$$

There are also laws of tautologies and contradictions. The **law of the excluded middle** and the **law of non-contradiction** are used for **case analysis** and proofs.

The **law of the excluded middle** states that:

$$P \vee \neg P \equiv 1$$

and its dual, the **law of non-contradiction**:

$$P \wedge \neg P \equiv 0$$

We won't go into case analysis and heavy proofs, since those require more precedence rules and properties of propositions. However, just note that these laws are used for simplifying large logical expressions and proofs.

Chapter 4

Predicate Logic

We learned how to use propositional logic, for the most part, and we can use that to express a famous argument:

- Socrates is a man.
- If Socrates is a man, then Socrates is mortal.
- Therefore, Socrates is mortal.

Using propositions, we applied an application of the inference rule called the **modus ponens**. The rule states that by proposing p and $p \to q$, you can deduce the validity of q.

We are also showing that the first two statements are *axioms*, meaning that we assume they are always true, and the last statement is a result from an inference rule.

4.1 Variables and Predicates

However, on the basis of predicate logic, we can now use variables, instead of always using Socrates as an example.

We can let x, y, and z to stand in for **any** element of our **universe of discourse**, or **domain**—meaning that we can state any value of x as something at any time.

We also can now state things like:

- "x is human."
- "x is a child of y."
- "$x + 2 = x^2$."

Remember, these are not propositions anymore, because of the addition of variables, but rather they are now **predicates**. Predicates are statements whose validity and truth depends on what object replaces the variable.

Predicates also are abbreviated and appear as "function notation", e.g.:

- $H(x)$ ="x is human."
- $J(x)$ ="x is a child of y."
- $K(x)$ ="$x + 2 = x^2$."

The variable(s), x and more, that appears between paragraphs right after a capital letter are called **arguments**. The variable(s) that appear in the arguments also appear in the predicate. Now that we have a completed function notation, we can input any specific value, such as $H(\text{Socrates})$ = "Socrates is human". Once we bring specific values in, we have a proposition again, and we can verify if that proposition is true or false, from what we've learned.

It's also important to note that we are using **first-order logic**, where variables refer to objects or specific values, and never predicates, while predicates are effectively a constant (i.e. H, J, K).

4.2 Quantifiers

Now that we are using predicates, the variables can be any value or object, and we can "select" or **bind** the variables through **quantifiers**. We can claim a predicate works for all values of a variable with the **universal quantifier**, or some values of a variable with the **existential quantifier**.

4.2.1 Universial Quantifier

The universal quantifier $\forall$ ("for all...") says that a predicate or statement must be true for all values of a variable within a set or collection of values. For example:

- "All humans are mortal", is the same as
- $\forall x$: Human(x) $\rightarrow$ Mortal(x)

We can also explicitly call what kind of universe we are using with set notation $\in$.

- "If x is positive, then $x + 1$ must also be positive" is the same as
- $\forall x \in \mathbb{Z} : x > 0 \rightarrow x + 1 > 0$

The use of "in", or $\in$ limits the **range** of x to an integer, natural number, etc.

4.2.2 Existential Quantifier

The existential quantifier $\exists$ ("there exists...") claims that a predicate or statement must be true for at least one value of the variable.

- "Some human is mortal", is the same as
- $\exists x$: Human$(x) \wedge$ Mortal(x)

Note that we use $\wedge$, AND, instead of an implication, because it becomes a weaker claim. "There exists a human that is mortal" is better than "There exists a human, such that if it is human, then it is mortal".

As the same as $\forall$, $\exists$ can also be limited to an explicit universe with set notation.

4.2.3 Negation of Quantifiers

The following are logically equivalent:

$$\neg\forall x : H(x) \equiv \exists x : \neg H(x)$$
$$\neg\exists x : H(x) \equiv \forall x : \neg H(x)$$

These are essentially the quantifier version of De Morgan's law. The first statement states that "not all humans are mortal" is the same as "some human are not mortal". The second statement states that "no human is mortal" is the same as "all humans are not mortal".

4.2.4 Nested Quantifiers

When using more than one variable, we can nest quantifiers as such:

$$\forall x \exists y : y > x \wedge \text{Prime}(y)$$

We can read that as "for any x there is a y that is larger than x and is prime".

The precedence of ordering of quantifiers here relies on order. Since $\forall$ is first, we can choose any x we want, and then there will have to exist a y that is bigger than the x we chose and is also prime.

Also note that the universal quantifier as a picky variable, while the existential quantifier will try to make do with what the universal quantifier chooses. Coming back to the ordering, an example shows that the statements

$\forall x \exists y : \text{likes}(x, y) =$ Every x must like a y that exists.
$\exists y \forall x : \text{likes}(x, y) =$ There is some y that exists that every x likes y.

As we see, these two statements are very different, just from the ordering of quantifiers. The thing about logically equivalent nested quantifiers, is that there are many, many more ways you can write a single statement in different manners.

4.2.4.1 Examples

We will get some practice with reading nested quantifiers with some English sentences.

All crows are black	$\forall x :$	$\text{Crow}(x)$	$\rightarrow$	$\text{Black}(x)$
No ID, no service	$\forall x :$	$\neg\text{ID}(x)$	$\rightarrow$	$\neg\text{Served}(x)$
All that glitters is not gold	$\neg\forall x :$	$\text{Glitters}(x)$	$\rightarrow$	$\text{Gold}(x)$

4.3 Functions and Equality

A **function symbol** by itself looks like a predicate, but instead of returning a boolean value, it returns an object or variable. The function symbol S could be a function that returns $x + 1$, so $S(x) = x + 1$. See that the function Sx or $S(x)$ returns an object, $x + 1$.

With predicates, there is a special predicate $=$, written as $x = y$. This can give us a lot of information about x or y. From that predicate, we know that x and y are the same *element* and from the same *domain*.

This predicate also satisfies the **reflexivity axiom**, $\forall x : x = x$ and the **substitution axiom schema**, $\forall x \forall y : (x = y \rightarrow (Px \leftrightarrow Py)$, where P is any predicate. Most importantly, we are given the ability to **substitute** between predicates now. For example, we can prove the **symmetry property**, $\forall x \forall y : (x = y \rightarrow y = x)$

$\forall x \forall y : (x = y \rightarrow (Py \leftrightarrow Px)$	Begin with the substitutuion axiom schema
$\forall x \forall y : (x = y \rightarrow (y = x \leftrightarrow x = x)$	Substitute in $Px \equiv y = x$ and the reflexivity axiom
$\forall x \forall y : (x = y \rightarrow y = x)$	We can get rid of the repetitive $x = x$

4.3.1 Uniqueness

Another special property with quantifiers is that there is an abbrevation, $\exists! x P(x)$, which means "there exists a *unique* x such that $P(x)$...". This can be coupled with examples such as:

$$\exists! x P(x) : x + 2 = 10,$$

where there is an *unique* value, or solution that would make this true (i.e. $x = 8$). Of course we could write $\exists x : x + 2 = 10$ and we would already assume the value is 8, but uniqueness calls for a more explicit amount of solutions.

4.3.2 Models

Model checking and modeling with propositional logic was solved by creating truth tables for every possible logical operator. In predicate logic, the closest equivalent to the truth table is called a **structure**. A structure is made up from a set of objects or elements (built using set theory), alongside a description of what kind and which elements fill in the variables. This description or information is called the **signature** of the structure. As truth tables are the models of propositional logic, structures are the **model** of a particular **theory** (set of statements).

As mentioned many times before, model checking with predicate logic is extremely difficult and time-consuming, but any preexisting or general model gives us two important kinds of information. If we know a model of a particular theory, then the model shows that the theory is consistent. We can also prove the opposite, that if

a statement S isn't true in that particular theory, then we can state that we can't prove S from that theory.

4.3.2.1 Examples of Models

We will be using more jargoned terms as we move on now.

- Consider the **axiom** $\neg\exists x$. This axiom has exactly **one model** (it's empty, no values exist).
- Then consider the axiom $\exists!x$. This axiom also contains only one model (one element).
- Add in a predicate P, turning the axiom into $\exists x P(x)$. Now we have many models. You can make the predicate solve for one value, such as $\exists x P(x) : x + 2 = 10$, or two values $\exists x P(x) : x^2 = 16$. You can change the model by changing the predicate(s) and axiom(s).

These models, or structures, of predicate logic can get easily messy, but set theory ties it up in a more efficient form than predicate models.

4.4 Proofs and Inferences

We will not be going into proofs and proving techniques, as these will most likely never be used, and on a practical level, will only confuse you and myself simultaneously. We will, however, just talk about vocabulary the mathematical terms we should know.

A **proof** is a way to derive new statements from other statements. They start with **axioms**, statements we assume are always true. Then, we have **theorems** and **lemmas**, statements that have already been proven for us. Lemmas are usually the mid-steps in proofs, while theorems are defined as the final conclusion or final step of a proof.

Next, we have **premises** and **inference rules**. Premises (typically depicted as P), are what we are trying to prove, since we don't need to prove axioms, theorems, or lemmas. Inference rules are what we make from these axioms, theorems, and lemmas.

We also define anything that isn't proved in the proof itself as a **hypothesis**, and the resultant (generally Q) is the **conclusion**.

When a proof does exist of Q from the premises P_1, P_2, ..., we write that as

$$P_1, P_2, \ldots, \vdash Q,$$

and that Q is **provable** or **deducible** from P_1, P_2, ...,. If we can just prove Q from out inference rules without making premises or assumptions, we can just write

$$\vdash Q$$

We don't need to really know every kind of inference rule out there, but the most important is the **modus ponens**, which states that $(p \wedge (p \rightarrow q)) \rightarrow q$. If you remember our example at the beginning:

- Socrates is a man. (Premise)
- If Socrates is a man, then Socrates is mortal. (Axiom)
- Therefore, Socrates is mortal. (Modus Ponens)

There are large topics in proofs, such as the **Deduction Theorem** and **Natural Deduction** which are extensions of what inference rules are and what they can do.

Chapter 5

Set Theory

Set theory, created from everything that has been building up to this point, is one of the biggest foundations of mathematics. This is because anything in mathematics can be written in terms of sets; numbers, functions, structures, etc. can all be collected into sets. "If predicate logic is the machine code of mathematics, set theory would be the assembly language".

5.1 Naive Set Theory

Naive set theory is more or less an informal version of set theory so we can give easy examples about collections of objects (generally called **elements**) with no duplicate elements. A set is written with the curly braces { }:

- {} = the **empty set** $\emptyset$, which contains no elements.
- {Moe, Curly, Larry} the **The Three Stooges**.
- {0, 1, 2, ...} = $\mathbb{N}$, the **natural numbers**. Also note we are assuming the read can continue the sequence here.
- {{},{0},{1},{0,1},{0,1,2},7} = a set of sets of natural numbers, alongside a stray natural number that is an element of the "outer" set.

To say an element is in a set, membership is written with the $\in$ symbol, ("is in", "is a member of", or "is an element of"). So we can write Moe $\in$ The Three Stooges, or 4 $\in \mathbb{N}$. We can also say an element is **not** within a set using $\notin$. For example, Moe $\notin \mathbb{N}$.

A fundamental axiom in set theory (the **Axiom of Extensionality**) is the only distinguishing property of a set that states that if two sets contain the same members, they are the same set. However, it's important to distinguish the difference between nested sets.

The set {{1}} has a single member, the **set** {1}, not the natural number 1.

5.1.1 Set Notation

Instead of explicitly defining a set with elements directly, we can use rules, "set comprehension" to generate all of its elements. With rules, we can explicitly define infinite sets without having to guess. Counting infinite sets is a different matter.

Set-builder notation is written as such:

- $\{2x \mid x \in \mathbb{N}\}$ = the even numbers.
- $\{x \mid x \in \mathbb{N} \land x < 8\} = \{0, 1, 2, 3, 4, 5, 6, 7\}$.
- $\{x \mid x \in \mathbb{N} \land x > 1 \land (\forall y \in \mathbb{N} : \forall z \in \mathbb{N} : yz = x \to y = 1 \lor z)\} = 1$ = the prime numbers

With this set comprehension, we can see that every set in naive set theory is equivalent to some predicate P. Given a set S, the corresponding predicate is $x \in S$. Given a predicate P, the corresponding set is $\{x \mid P(x)\}$. An interesting situation with sets is **Russell's Paradox**, what happens when you have $\{S \mid S \notin S\}$?

A description from Wikipedia shows "If R is not a member of itself, then its definition dictates that it must contain itself, and if it contains itself, then it contradicts its own definition as the set of all sets that are not members of themselves. This contradiction is Russell's paradox"

5.2 Operations on Sets

There are logical operations that you can use with sets.

Union of A and B	$A \cup B =$	$\{x \mid x \in A \lor x \in B\}$
Intersection of A and B	$A \cap B =$	$\{x \mid x \in A \land x \in B\}$
Set Difference of A and B	$A \setminus B =$	$\{x \mid x \in A \land x \notin B\}$
Symmetric Difference of A and B	$A \triangle B =$	$\{x \mid x \in A \oplus x \in B\}$
Subset A of B	$A \subseteq B =$	$\forall x : x \in A \to x \in B$

The **union** of A and B is a new set that contains all of the elements that appear at least once in one of the two sets.

The **intersection** of A and B is a new set that contains elements that appear in both sets.

The **set difference** of A and B is a new set that contains elements from A, but not B.

The **symmetric difference** of A and B is a new set that contains elements which appear in either of the sets **excluding** elements that appear in the **intersection**.

The **subset** A of B states that A is **contained in** B is all of the elements of A are also found in B. It's important to distinguish that $A \subseteq B$ is **not** the same as $A \in B$. If A is a set with the element $\{12\}$, and B is a set with the elements {Moe,

Larry, Curly, {12}}, then A is **not** a subset ($\not\subseteq$) of B, but rather $A \in B$.

There is also the sets-equivalent of negation, called the complement. The **complement** of A is written as $\bar{A} = \{x \mid x \notin A\}$. Also, if we are given ability to use complements, we also must assume we are working with a **finite or fixed universe** (U. We will have some issues if we state $\bar{\emptyset} = U$, then what is U? What can U hold? How big is U?

5.2.1 Proofs With Sets

Onto another section about sets, we will go very shortly over how to prove a predicate given a set, and proving a set given a predicate, etc. There are basically about three things we can try to prove things about sets:

1. Given x and S, show $x \in S$. See the description/model of S, and see if x matches or not.
2. Given S and T, show $S \subseteq T$. Show for any x that exists in S, it can also exist in T.
3. Given S and T, show $S = T$. Prove $S \subseteq T$ and $T \subseteq S$ separately. If both prove to be true, then $S = T$. Use instructions from Step 2.

There are also the corresponding negative statements:

1. Show $x \notin S$. Show x doesn't match the description of S.
2. Show $S \not\subseteq T$. Show that there exists an x that is in S but not T.
3. Show $S \neq T$. Prove either $S \not\subseteq T$ or $T \not\subseteq S$.

5.3 Axiomatic Set Theory

Throughout our descriptions of **naive set theory**, we warned about how finite or infinite an universe could get to be, if we don't make a stricter or more restrictive description of sets. **Axiomatic Set Theory** fulfills this problem by becoming more restrictive. The general or most commonly used axioms are known as the **Zermelo-Fraenkel set theory with choice**, or **ZFC**, from Zermelo and Fraenkel's first revised proposition of axiomatic set theory in the early 1920s.

5.3.1 Properties and Axioms of ZFC

Extensionality: Any two sets with the same elements are equal sets.

Existence: The empty set $\emptyset$ is a set.

Pairing: Given distinct sets x and y, $\{x, y\}$ is a set.

Union: Given any set of sets $S = \{x, y, z, ...\}$, the set $\bigcup S = x \cup y \cup z \cup ...$ exists.

Power Set: Given any set S, the power set $\mathcal{P} = \{A \mid A \subseteq S\}$ exists.

Specification: Given any set S and predicate P, the set $\{x \in S \mid P(x)\}$ exists.

Infinity: There is a set that has $\emptyset$ as an element and also has $x \cup \{x\}$ whenever it has x. In essence, this gives us a description of $\mathbb{N}$. The $\emptyset$ represents 0 and $x \cup \{x\}$ represents a recursive $x + 1$, which allows us to have infinite sets.

These next three axioms are more technicalities and don't appear much in computer science, but are still important to mention.

Foundation: Every nonempty set A contains a set B where $A \cap B = \emptyset$. Through this odd definition, we are allowed to create recursive sets, such as $A_0 \ni A_1 \ni A_2 \ni \ldots$; These allow us to make **induction arguments**.

Replacement: If S is a set, and $R(x, y)$ is a predicate with the particular property that $\forall x : \exists! y : R(x, y,)$, then $\{y \mid \exists x \in S R(x, y)\}$ is a set. These allow us to create extremely large infinite sets.

Choice: For any nonempty set S, there is a function f that assigns each x in S to some $f(x) \in x$. This is one of the least important axioms in ZFC, since it tells you that f exists, but doesn't give you any information about f. However, as it is our foundation of axiomatic set theory, it's still too important to leave out.

We just defined several axioms that produce a model to define our theory, the Set Theory. It's not too obvious, but with the axioms, we are able to do anything we know how to do in mathematics.

5.3.2 Cartesian Product

Naturally, sets are unordered. The set $\{a, b\}$ is the same as $\{b, a\}$. However, there are situations where the order is important. We can very explicitly state the order of (a, b) as the set $\{a\}, \{a, b\}\}$, which was proposed by Kuratowski. We can "multiply" two sets together, defined as their **Cartesian product**. If A is of size n and B of m, then the Cartesian product $A \times B$ is of size nm. For example, the Cartesian product:

$$\{1,2\}\times\{3,4\} = \{(1,3), (1,4), (2,3), (2,4)\}$$

The ordering matters here, as $A \times B \neq B \times A$. This principle is the same with matrices:

$$\mathrm{A} = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} \text{ and } \mathrm{B} = \begin{bmatrix} 6 & 3 \\ 2 & 7 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} 6 & 3 \\ 2 & 7 \end{bmatrix} = AB = \begin{bmatrix} 12 & 24 \\ 20 & 34 \end{bmatrix}$$

$$\begin{bmatrix} 6 & 3 \\ 2 & 7 \end{bmatrix} \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} = BA = \begin{bmatrix} 12 & 30 \\ 16 & 34 \end{bmatrix}$$

Chapter 6

Sets of Numbers

With sets, and their properties and functions (e.g. power set, Cartesian product, etc.) we can create the universe of mathematics.

6.1 Integers

The integers are denoted as the set $\mathbb{Z} = \{\dots, \text{-2, -1, 0, 1, 2, } \dots\}$. Every integer z is defined as an ordered pair (x, y). Positives are simply represented as $(z,\ 0)$ while negatives are represented as $(0,\ -z)$. These allow for addition, subtraction, multiplication, etc.

6.2 Rationals

The rational numbers $\mathbb{Q}$ are all fractions in a p/q form. p is an integer (negatives and positives), while q needs to be a natural number that is not equal to 0. Another requirement is that p and q are not common factors (meaning the fraction doesn't become an integer).

6.3 Reals

The real numbers $\mathbb{R}$ can be expressed in many different ways. The simplest way to describe the real numbers is a pair of sets:

$$\{y \in \mathbb{Q} \mid y < x\} \text{ and } \{y \in \mathbb{Q} \mid y \geq x\}$$

or formally called the **Dedekind cut**. This is called a cut because it can split off specifically at an irrational number. One set is **closed upward** and the other **closed downward**.

6.4 Complex

6.5 The Universe of Sets

Chapter 7

Sizes of Sets