Handout 1: Logic and Set Theory

The purpose of this handout is mostly to review the basic concepts of set theory and logic that will be used in CS 360. I assume you have studied basic logic and set theory at some point in your education, so this material should not be entirely new. As you are reading the handout, it might help to keep in mind the following question:

What does it mean to *prove* something?

Given that we will be proving many facts about computation throughout the course, it is good to have some sense of the answer to this question.

To begin to form an answer, let's draw an analogy with a modern personal computer. When you first turn on a PC, it has very limited capabilities and has to *bootstrap*: it starts out by running some sort of firmware embedded directly on a chip, which (perhaps) gives it enough information to find a boot loader on a hard disk, which then gives it the ability to boot an operating system. Simple processes invoke more complicated ones that take over.

Mathematics works in a similar way. When you want to prove something about complex objects such as differential equations or error correcting codes, you need to reason based on simpler concepts. This reasoning cannot be circular, which means you need to start somewhere—you need a basic formal system for reasoning and some very simple initial assumptions. By developing the system, it becomes possible to establish the validity of more powerful and more efficient methods for reasoning. This can be thought of in the same way as bootstrapping.

The obvious question at this point is: *what is the basic system?* The answer is: *set theory*. To be more precise, axiomatic set theory is just one formal logical system on which much of mathematics can be based, but it is the most common one. This is why you will often see mathematics books and courses starting with basic logic and set theory—sets lie at the heart of mathematics, including the theory of computation.

1 Propositional Logic

Before we get to set theory, let's talk about *Propositional logic*. Propositional logic is not powerful enough to prove much of anything, but it nevertheless helps to illustrate how more powerful logical systems work.

In propositional logic, *formulas* are formed from finite collections of *atomic propositions* (which you can think of as Boolean variables if you like), *logical operations*, and *parentheses* to make the meaning of formulas clear. Usually atomic propositions are just denoted by letters such as A, B, C, etc., and commonly we take our logical operations to be *and* ($\wedge$), *or* ($\vee$), *not* ($\neg$), *implies* ($\Rightarrow$), and *logical equivalence* ($\Leftrightarrow$). For example, this is a formula involving the atomic propositions A and B:

$$\neg(A \vee B) \Leftrightarrow (\neg A \wedge \neg B). \tag{1}$$

The only thing you really need to remember about the order of precedence for logical operations is that *not* comes first. It is natural to say that $\wedge$ comes next, $\vee$ comes third, and $\Rightarrow$ and $\Leftrightarrow$ together have lowest precedence, but I will not test you on trivialities such as whether $\wedge$ or $\vee$ has higher precedence. It is best to just use parentheses appropriately.

I assume that you understand the actual meaning of the logical operations $\wedge$, $\vee$, $\neg$, $\Rightarrow$, and $\Leftrightarrow$. Just to be sure, let's go through them once. Only one of the operations takes one argument: *not* ($\neg$). It just flips the Boolean value of its argument:

A	$\neg A$
0	1
1	0

The remaining operations take two arguments, and their values are determined by the following table:

A	B	$A \vee B$	$A \wedge B$	$A \Rightarrow B$	$A \Leftrightarrow B$
0	0	0	0	1	1
0	1	1	0	1	0
1	0	1	0	0	0
1	1	1	1	1	1

By the way, from here after we will associate 0 with false and 1 with true in the usual way without ever bothering to mention it again.

Whether or not a formula is true or false for different settings of Boolean values 0 and 1 for the atomic propositions does not change the fact that it is a formula. For instance, these are also formulas:

$$A \wedge \neg A, \quad (\neg A \wedge B) \vee B, \quad \text{and} \quad (A \wedge A) \Rightarrow (B \vee C).$$

It so happens that the first one always evaluates to 0 (for any Boolean assignment to A), while the second and third can evaluate to either 0 or 1 depending on the assignments to the inputs. The example (1) above happens to be true for all Boolean values substituted for A and B, which means it is a *tautology*. You probably recognize it as one of *De Morgan's Laws*.

Now, suppose now you want to *prove* that a particular formula is a tautology. For instance, you want to prove that the above formula (1) really is true for all Boolean values substituted for A and B. Of course in that simple case you can just check all of the possibilities, but in general that could take a long time. (For example, imagine a formula with several hundred atomic propositions rather than just two.) More importantly, this is not an option for more interesting logical systems in which variables can take on an infinite number of values.

So, with this in mind, what we can do instead is to develop a *logical system* for reasoning about formulas. Specifically, we take some set of *axioms* that are assumed to be true, as well as some set of *rules of inference*. Here is an example of a set of axioms (which you should not worry too much about because it is just an example):

1. $P \Rightarrow P \vee Q$ and $P \Rightarrow Q \vee P$
2. $P \wedge Q \Rightarrow P$ and $P \wedge Q \Rightarrow Q$
3. $P \Rightarrow (Q \Rightarrow P)$
4. $P \Rightarrow (Q \Rightarrow (P \wedge Q))$
5. $P \vee \neg P$
6. $(P \Rightarrow (Q \Rightarrow R)) \Rightarrow ((P \Rightarrow Q) \Rightarrow (P \Rightarrow R))$
7. $(P \Rightarrow Q) \Rightarrow ((P \Rightarrow \neg Q) \Rightarrow \neg P)$
8. $P \Rightarrow (\neg P \Rightarrow Q)$
9. $(P \Leftrightarrow Q) \Rightarrow (P \Rightarrow Q)$ and $(P \Leftrightarrow Q) \Rightarrow (Q \Rightarrow P)$
10. $(P \Rightarrow Q) \Rightarrow ((Q \Rightarrow P) \Rightarrow (P \Leftrightarrow Q))$

Here, P, Q, and R are not atomic propositions like A and B above—the axioms are true when any *formulas* are substituted for P, Q and R. An example of a rule of inference is *modus ponens*:

From P and $P \Rightarrow Q$, we derive Q.

It can sometimes be difficult to prove that even very simple formulas are true using a particular set of axioms and inference rules. (If you are interested in trying to prove something with the above axioms and inference rule, try $A \Rightarrow A$.) Usually people start by proving general theorems, such as something called the Deduction Theorem, before tackling specific formulas.

Once we have formal system like the example above, we can ask which formulas can be proved (meaning derived from the axioms and inference rules) and which cannot. Here is what we *obviously* would want to be the case:

1. If we have a formula that *evaluates* to true for all Boolean assignments to the atomic propositions, then it should be possible to *derive* the truth of the formula from the axioms and rules of inference. This property is called *completeness*.

2. If a formula can be *deduced* from the axioms and rules of inference, then it should be the case that the formula *evaluates* to true for all Boolean assignments to the atomic propositions. This property is called *soundness*.

For some random collection of axioms and inference rules, either or both of these properties might fail. For the particular example above, both properties are satisfied. The completeness property happens to be the harder property to prove: for this it is necessary to establish that *every* tautology can be derived from the axioms and inference rules, which is not so simple.

The problem with propositional logic is that it doesn't lead anywhere. Consider a statement like this:

$$2^n \geq n^2 \text{ for all integers } n \geq 4.$$

This is a true statement, but you could never prove it with propositional logic. Propositional logic is not powerful enough to say anything at all about numbers, in fact.

2 Set Theory

Now let us move on to set theory, starting with a simple, but ultimately problematic, version of set theory called *naive set theory*. For CS 360, it is enough that you understand just this version of set theory.

2.1 Definition and examples of sets

What is a set? That turns out to be a question with a more complicated answer than you might think. One way to define sets is to treat the notion as basically being self-evident, along the lines of this definition:

A set is an unordered collection of distinct objects.

Sometimes people call set theory based on this sort of definition *naive set theory*, because it turns out to be naive to think that the notion of a set really is so simple and self-evident. However, although there are serious problems with this approach that strongly motivate basing the theory on a more firm mathematical foundation, you will not run into problems in this course by thinking about sets in the naive manner.

An example of a set is this one:

$$\{a, b, c\}$$

It has three *members* or *elements*: a, b, and c. This set is no different from $\{b, a, c\}$ because the ordering of the elements in a set does not matter. It is also no different from $\{a, a, b, c\}$ because we only care about *distinct* objects. Another way of thinking about this is that the only concept that really matters for a set is *membership*: if A is a set and x is an object, then either x is an element of A or it isn't. If x is an element of A, then we write

$$x \in A$$

and if not we write

$$x \notin A.$$

There should always be a definite answer to the question of whether or not some object is contained in a given set. If that is not the case, then the set is not *well-defined* and you won't be able to reason about the set using set theory. In other words, the set is really not a set at all.

Sets can have a finite number of elements or an infinite number. For example, here are some important infinite sets whose elements are numbers:

1. The *natural numbers*:
$$\mathbb{N} = \{0, 1, 2, 3, \ldots\}.$$
(Some people choose not to include 0 in this set. This is nothing more than a definition, so neither way is right or wrong.)

2. The *integers*:
$$\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}.$$

3. The *rational numbers*:
$$\mathbb{Q} = \left\{ x : x = \frac{n}{m} \text{ for integers } n, m \text{ with } m \neq 0 \right\}.$$

4. The *real numbers* $\mathbb{R}$. (I will not define these formally because it would be a complicated digression. I assume you have some intuition about real numbers, which is enough for now.)

In the first and second examples, we use some dots like this: "$\ldots$". They mean: continue the pattern in the obvious way. In the third example, we use this sort of notation:

$$\{\boxed{} : \boxed{}\}$$

where the boxes are replaced by something. Sometimes people use a vertical line instead of a colon ":". You can think of the second box as representing some general conditions, and when those conditions are satisfied the first box represents the actual elements of the set. For example, the set of even natural numbers could be written like this:

$$\{n \in \mathbb{N} \;:\; n \text{ is divisible by } 2\},$$

or like this:

$$\{2n \;:\; n \in \mathbb{N}\}.$$

The notations above are useful for defining both finite and infinite sets, especially where it is not possible or practical to write down all of the elements.

2.2 Basic concepts

At this point let us go through some very basic concepts about sets that I assume you know. The purpose of doing this is (i) to make certain you know or remember these things, and (ii) to make clear the specific notation we will use in the course.

The *empty set* is the set containing no elements at all. We write it like this: $\varnothing$. For every possible object x we have $x \notin \varnothing$.

The *union* of two sets A and B is the set that contains exactly those objects that are contained in either A or B (or both). This set is denoted

$$A \cup B.$$

The *intersection* of two sets A and B is the set that contains exactly those objects that are contained in both A and B. This set is denoted

$$A \cap B.$$

We say that a set A is a *subset* of another set B if every element of A is also contained in B. We write

$$A \subseteq B$$

in this case, and we write

$$A \nsubseteq B$$

when it is not the case.

By the way, if you want to prove $A \subseteq B$ in some specific situation, you would often start by writing this: "Let $x \in A$". Here x is some arbitrary element of A that you don't place any restrictions on. From there, based on the definitions of A and B, you argue that $x \in B$. If you can do this, and all along the way no restrictions have been placed on x beyond the assumption that it is an element of A, then you've proved $A \subseteq B$.

We can write

$$A \subsetneq B$$

when $A \subseteq B$ and $B \nsubseteq A$ are both true. Some people use the symbol $\subset$ to mean $\subsetneq$ while others use it to mean $\subseteq$. I will simply not use the symbol $\subset$ in order to avoid confusion.

Two sets are *equal* when they have exactly the same elements, and in this case we write $A = B$. When you want to prove this in some situation, it usually means you have to do two things: first prove $A \subseteq B$ and then prove $B \subseteq A$.

If A and B are sets, then the *set difference* $A \backslash B$ is the set of all elements in A that are *not* in B:

$$A \backslash B = \{x \in A \,:\, x \notin B\}.$$

Some people write $A - B$ to mean the same thing.

Sometimes we work in a situation in which all sets of interest are subsets of some fixed set U called the *universe*. For example, our universe might be $U = \mathbb{N}$ when we are talking about properties of natural numbers. In CS 360, it will be common to work in a universe consisting of all possible strings over some set of symbols. When some universe U is specified, and we have a set $A \subseteq U$, we define the *complement of* A to be the set

$$\overline{A} = U \backslash A.$$

When A and B are subsets of some universe U, we have

$$\overline{A \cup B} = \overline{A} \cap \overline{B} \qquad \text{and} \qquad \overline{A \cap B} = \overline{A} \cup \overline{B}.$$

These are called *De Morgan's Laws*.

For every set A, we define the *power set* of A to be the set $\mathcal{P}(A)$ containing all subsets of A as its elements:

$$\mathcal{P}(A) = \{B \,:\, B \subseteq A\}\,.$$

For example, if $A = \{1,2,3\}$ then

$$\mathcal{P}(A) = \{\varnothing, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}.$$

The *Cartesian product* of a set A with a set B is formed by pairing elements from the two sets. Specifically, we define

$$A \times B = \{(x,y) \,:\, x \in A \text{ and } y \in B\}\,.$$

The notation (x,y) denotes an *ordered pair*, meaning that the order matters. More generally we can take Cartesian products of any finite number of sets: if $A_1, \ldots, A_n$ are sets for some integer $n \geq 2$, then

$$A_1 \times A_2 \times \cdots \times A_n = \{(x_1, \ldots, x_n) \,:\, x_i \in A_i \text{ for all } i = 1, \ldots, n\}\,.$$

2.3 Functions

A *function* f from a set A to a set B is, intuitively speaking, a way of transforming elements of A into elements of B. For example, we can define a function f from $\mathbb{N}$ to $\mathbb{N}$ that transforms n to $5n^2$. We would normally express this function as follows:

$$f(n) = 5n^2.$$

If we just want to express that, say, g is a function from A to B for certain sets A and B, we write

$$g : A \to B.$$

When you see this, you cannot tell very much about g, other than that it transforms elements of A into elements of B.

Formally speaking, a function f from A to B is defined as a subset of $f \subseteq A \times B$ with the property that if $(a,b) \in f$ and $(a,c) \in f$, then $b = c$. If $(a,b) \in f$, then we use the more familiar notation $f(a)$ to mean b. This is sensible because there cannot be more than one such element b.

Notice that according to the formal definition of a function, it is possible that a function $f : A \to B$ could be such that some element $a \in A$ has no corresponding element $b \in B$ satisfying $(a,b) \in f$. This means that f is *not defined* at a, and in this case we write this:

$$f(a) \uparrow$$

A function like this is called a *partial function*. If $f(a)$ is defined for all $a \in A$, then f is called a *total function*. You are probably used to working mostly with total functions, but partial functions will come up in this course. By default, whenever we refer to a "function" in this course, we mean a total function. When we do want to talk about partial functions, we will explicitly use the term "partial function".

If $f : A \to B$ is a total or partial function, then the *domain* of f is defined as follows:

$$\text{dom}(f) = \{a \in A \,:\, (a,b) \in f \text{ for some } b \in B\}.$$

Equivalently

$$\text{dom}(f) = \{a \in A \,:\, f(a) \text{ is defined}\}.$$

This means that if $f : A \to B$ is a (total) function, then the domain of f is just A.

The *range* of a total or partial function $f : A \to B$ is defined as follows:

$$\text{range}(f) = \{f(a) \,:\, a \in A\}.$$

Equivalently,

$$\text{range}(f) = \{b \in B \,:\, (a, b) \in f \text{ for some } b \in B\}.$$

A function $f : A \to B$ is *onto* if $\text{range}(f) = B$, and is *1-to-1* if it is the case that no two distinct elements of A are transformed to the same element $b \in B$. Formally, f is 1-to-1 if, for all $a, c \in A$ and $b \in B$, we have

$$[(a, b) \in f \text{ and } (c, b) \in f] \;\Rightarrow\; [a = c].$$

2.4 Predicates and Quantifiers

A *predicate* is just a function whose output is a Boolean value. For examples, the *equals* predicate defined on natural numbers has the form

$$\text{Equal} : \mathbb{N} \times \mathbb{N} \to \{0, 1\},$$

and could be expressed as follows:

$$\text{Equal}(n, m) = \begin{cases} 1 & \text{if } n = m \\ 0 & \text{if } n \neq m. \end{cases}$$

Another example is the predicate

$$\text{Zero} : \mathbb{N} \to \{0, 1\}$$

defined as

$$\text{Zero}(n) = \begin{cases} 1 & \text{if } n = 0 \\ 0 & \text{if } n \neq 0. \end{cases}$$

There are two *quantifiers* that we need in this course: *universal* quantifiers and *existential* quantifiers. You might know them better as "for all" and "there exists". Suppose, for the sake of example, that

$$P : \mathbb{N} \to \{0, 1\}$$

is a predicate defined on the natural numbers. Then the formula

$$(\forall n) P(n)$$

evaluates to *true* if the predicate $P(n)$ is true *for all* choices of $n \in \mathbb{N}$, and evaluates to *false* otherwise. The formula

$$(\exists n) P(n)$$

evaluates to *true* if the predicate $P(n)$ is true for *at least one* choice of $n \in \mathbb{N}$, and evaluates to *false* otherwise. For example,

$$(\forall n)\text{Zero}(n)$$

evaluates to *false* because not every natural number equals 0, while

$$(\exists n)\text{Zero}(n)$$

evaluates to *true* because there is at least one choice of n for which $\text{Zero}(n) = 1$, namely $n = 0$.

You can have more than one quantifier in a formula, but every variable should only be quantified by one quantifier. It is important to understand that quantifiers are read *from left to right*. For example, the formula

$$(\forall n)(\exists m)\text{Equal}(n, m)$$

reads "for all n, there exists an m, such that $n = m$." This formula evaluates to 1. On the other hand, the formula

$$(\exists m)(\forall n)\text{Equal}(n, m)$$

evaluates to 0, because there does not exist any choice of a natural number m with the property that $n = m$ for all choices of n.

2.5 Sizes of sets

The *size* of a finite set is just the number of elements it contains. We denote the size of a set A by $|A|$. For example,

$$|\{2, 3, 5, 7, 11\}| = 5.$$

We could just say that the size of an infinite set is ∞, but we will need a more refined notion than this. The reason is that there are different notions of infinity, and there is a precise sense in which one infinite set can be considered to be larger than another.

First, suppose that A is set for which there exists an onto function of the form $f : \mathbb{N} \to A$. Then we say that A is *countable*. If in addition the function f is one-to-one, then we say that A is *countably infinite* or *denumerable*.

For example, finite sets are obviously countable. The set $\mathbb{N}$ is itself countably infinite, because we can take f to be the function $f(n) = n$ for all $n \in \mathbb{N}$, which is one-to-one and onto.

The set $\mathbb{Z}$ is also countably infinite. To see this, define $f : \mathbb{N} \to \mathbb{Z}$ as follows:

$$f(n) = \begin{cases} \frac{n+1}{2} & \text{if } n \text{ is odd} \\ -\frac{n}{2} & \text{if } n \text{ is even.} \end{cases}$$

Listing the first few values of f should convince you that f is one-to-one and onto:

$$f(0) = 0,\ f(1) = 1,\ f(2) = -1,\ f(3) = 2,\ f(4) = -2,\ \ldots.$$

Every integer is hit exactly once.

The set of rational numbers $\mathbb{Q}$ is also countably infinite. To show this, imagine an infinite sequence of lists of rational numbers like this:

$$\begin{gathered} -1,\ 0,\ 1 \\ -2,\ -1/2,\ 1/2,\ 2 \\ -3,\ -3/2,\ -2/3,\ -1/3,\ 1/3,\ 2/3,\ 3/2,\ 3 \end{gathered}$$

The first list is list 1, the second is list 2, and so on. List k consists of all the rational numbers that can be formed as n/m with $n, m \in \{-k, \ldots, k\}$ and $m \neq 0$, in order and skipping over any numbers that appear in previous lists. Now define f to be the function you get by joining all of the above lists together, one after the other:

$$f(0) = -1,\ f(1) = 0,\ f(2) = 1,\ f(3) = -2,\ f(4) = -1/2,\ f(5) = 1/2,\ \ldots$$

Similar to before, you will hit every rational number exactly once.

A set is said to be *uncountable* when it is not countable. Is it possible to have an uncountable set? The answer is yes.

Theorem 1 (Cantor)**.** *The set $\mathcal{P}(\mathbb{N})$ is uncountable.*

Proof. The proof is by contradiction: we will assume $\mathcal{P}(\mathbb{N})$ is countable and derive a contradiction. This will imply that $\mathcal{P}(\mathbb{N})$ could not possibly be countable after all, proving the theorem.

The assumption that $\mathcal{P}(\mathbb{N})$ is countable means that there exists some onto function

$$f : \mathbb{N} \rightarrow \mathcal{P}(\mathbb{N}).$$

Given such a function, define a set S as follows:

$$S = \{n \in \mathbb{N} \,:\, n \notin f(n)\}\,.$$

(Note that the definition of S makes perfect sense: $f(n)$ is an element of the power set of $\mathbb{N}$, which means it is a subset of natural numbers.)

Now, because S is a subset of the natural numbers, we have $S \in \mathcal{P}(\mathbb{N})$ by the definition of the power set. Given that f is onto, it must be the case that $S = f(n)$ for at least one choice of $n \in \mathbb{N}$. For the rest of the proof, fix n to be the smallest number for which $S = f(n)$.

If it so happens that $n \in S$, then $n \notin f(n)$ by the definition of S. But that means that $n \notin S$ because $S = f(n)$. Thus

$$n \in S \Rightarrow n \notin S.$$

On the other hand, if $n \notin S$, then $n \in f(n)$, again by the definition of S. But because $S = f(n)$, we therefore have that $n \in S$. So, we have

$$n \notin S \Rightarrow n \in S.$$

The two implications together give us $n \in S \Leftrightarrow n \notin S$, which gives us the contradiction we were looking for. □

Remark 2. In mathematics, people sometimes discuss the *beauty* of certain proofs, which may seem strange to people who don't study mathematics. But the above proof is a stunning example of how beautiful a proof can be—and if you don't see its beauty you should study it until you do. I don't like to ask students to memorize things, but this proof is an exception: I consider that it is fair game to ask you to prove that $\mathcal{P}(\mathbb{N})$ is uncountable on an exam, expecting you to give me this proof.

Another example of an uncountable set the real numbers $\mathbb{R}$, also proved by Cantor using a similar method to the proof above. (The method is called *diagonalization*, and we will see it again later in the course.) In the case of the real numbers the proof must deal with some complications not present in the above proof, based on the fact that a single real number can have more than one decimal expansion.

3 First-Order Logic and Axiomatic Set Theory

This section of the handout is not required reading for CS 360—it is only here for those that are interested. Its point is to illustrate that the "self-evident" concept of a set used in naive set theory has some problems, for which there is a solution: *axiomatic set theory*. This is the "real" system on which much of mathematics is based, but for this course it is enough that you understand the principle and the fact that it exists. At a technical level it falls outside of the scope of this course.

3.1 Russell's Paradox

Consider the following peculiar example:

Let A be the set of all sets that are not elements of themselves.

For example, $\varnothing$ is not an element of itself, so it is included in A. All of the examples we have seen so far are contained in A in fact. Come to think of it, is there any set that is not contained in A? Maybe we could have a set like

$$B = \{\{\{\cdots\}\}\}.$$

That would not be an element of A, because it is an element of itself. The set of all sets is another example.

So what about A itself? Is $A \in A$? If so, then $A \notin A$, because A only contains sets that are not members of themselves. However, if $A \notin A$, then A should be included in A, so $A \in A$. We therefore have a contradiction, known as *Russell's Paradox*.

The fact that the definition of A leads to a contradiction means that we cannot hope to reason about A, so we shouldn't actually consider it to be a set at all. (Remember, there should be a definite answer to the question of whether a given object is contained in a given set, and A fails to have this property.) The set B we defined above, as well as the "set of all sets", are problematic as well—they shouldn't be considered to be valid sets either.

Hopefully this example convinces you that this is a problem that requires a careful solution. One solution is *axiomatic set theory*, which will allow us to define sets in a more precise way that makes clear which naively-defined sets are really valid sets.

3.2 First-Order Logic

First-order logics are similar to propositional logic, except that they allow variables and quantifiers in logical formulas. Specifically, first-order logics allows for:

1. Logic operations: $\wedge$, $\vee$, $\neg$, $\Rightarrow$, and $\Leftrightarrow$.

2. Variables.

3. Functions and predicates on the variables, including a special *equals* predicate ($=$) as well as *constant* functions and predicates.

4. Quantifiers.

5. Parentheses to make the meaning of formulas clear.

Axiomatic set theory is one specific first-order logic that puts the notion of a set on a firm mathematical foundation. In this case, the variables correspond to sets, and there is only one predicate other than the *equals* predicate, corresponding to set membership.

By the way, it might look like we are about to define the basic concept of a set with more complicated notions like functions and predicates, but it is not really the case. When things are done carefully (in gory detail that we will not worry about) nothing is circular.

3.3 The axioms of set theory

Now let us take a look at a more formal mathematical version of set theory—specifically *Zermelo-Fraenkel Set Theory with the Axiom of Choice* (abbreviated ZFC). It is a first-order logic with the following axioms:

1. **Axiom of the Empty Set:** There exists a set that contains no elements.
$$(\exists A)(\forall x)[\neg(x \in A)].$$

2. **Axiom of Extension:** Two sets are the same when they contain the same elements.
$$(\forall A)(\forall B)[(\forall x)(x \in A \Leftrightarrow x \in B) \Rightarrow (A = B)]$$

3. **Axiom of Pairing:** For any two sets A and B, there exists a set $\mathcal{C}$ containing the sets A and B.
$$(\forall A)(\forall B)(\exists \mathcal{C})[(A \in \mathcal{C}) \wedge (B \in \mathcal{C})].$$

4. **Axiom of Union:** Let $\mathcal{C}$ be a set (whose elements are sets). Then there exists a set A such that, for all sets B, we have $B \in \mathcal{C}$ implies $B \subseteq A$.
$$(\forall \mathcal{C})(\exists A)(\forall B)(\forall x)[((B \in \mathcal{C}) \wedge (x \in B)) \Rightarrow (x \in A)]$$

5. **Axiom of Powers:** For every set A there exists a set B whose elements are precisely the subsets of A.
$$(\forall A)(\exists \mathcal{C})(\forall B)[(\forall x)(x \in B \Rightarrow x \in A) \Rightarrow B \in \mathcal{C}]$$

6. **Axiom of Infinity:** There exists a set $\mathcal{A}$ such that $\mathcal{A}$ contains $\varnothing$, and such that, for every set $B \in \mathcal{A}$, it holds that $B \cup \{B\} \in \mathcal{A}$.
$$(\exists \mathcal{A})[(\varnothing \in \mathcal{A}) \wedge (\forall B)(B \in \mathcal{A} \Rightarrow B \cup \{B\} \in \mathcal{A})]$$

7. **Axiom of Replacement:** Suppose $F(x,y)$ is a formula involving variables x and y, and suppose A is a set. If, for every $x \in A$, there is a unique y for which $F(x,y)$ is true, then there exists a set B such that
$$B = \{y \,:\, (\exists x)(x \in A \;\wedge\; F(x,y) \text{ is true})\}.$$
Note: this is really an infinite collection of axioms, or an *axiom schema*.

8. **Axiom of Regularity:** Every non-empty set $\mathcal{A}$ contains some member B such that $\mathcal{A}$ and B are disjoint sets.
$$(\forall \mathcal{A})[(\exists x)(x \in \mathcal{A}) \Rightarrow (\exists B)[(B \in \mathcal{A}) \wedge (\forall y)[(y \in \mathcal{A}) \Leftrightarrow \neg(y \in B)]]]$$

9. **Axiom of Choice:** Let $\mathcal{C}$ be a set whose elements are nonempty sets. Then there exists a function f such that $f(A) \in A$ for every $A \in \mathcal{C}$.

I am not giving the completely formal versions of these axioms, although I have included some formulas that should give you a sense for how the axioms can be phrased using quantifiers, logical operations, and so on.

The first five axioms should appear to be very simple. The sixth one is basically just a technical way of saying that infinite sets exist. The seventh and eighth axioms are sort of technical, but serve to let us do some simple things like selecting certain subsets of set, rule out crazy sets like $\{\{\{\cdots\}\}\}$ and the set of all sets, and so on. The last one, the Axiom of Choice, is the source of much discussion and research—it says that from any set of non-empty sets, it is possible to *choose* one element from each subset. That seems like it should obviously be true, but it cannot be derived from the others and it turns out to have some surprising consequences when we are talking about infinite sets.

These axioms form a first-order logic when we add *modus ponens* as an inference rule (along with something simple having to do with the interpretation of quantifiers that I don't want to get into).

3.4 The purpose of ZFC and the connection to CS 360

What ZFC does is to reduce the notion of a mathematical proof to the lowest possible level you can imagine. Even complicated statements about numbers, equations, and so on, can be translated into formulas about sets, which might potentially be derived from the above axioms.

Now, If you have a proof that some formula is true in ZFC, then absolutely no creativity or even intelligence is required to check its validity. Rather it becomes a completely mechanical task that only involves manipulating symbols. In other words, checking the validity of proofs is a *computational* task.

What about *finding* a proof of some formula, or deciding that one does not exist? As we will see (much later in the course), this is rather different. It turns out that it is not possible to program a computer to determine whether or not a given formula can be proved to be true in ZFC. Any computer program that tries to do this must either be wrong sometimes, or have the possibility of running forever on some inputs. This is closely related to something called *Gödel's Incompleteness Theorem*. We will not study this theorem in CS 360, but we will later discuss the fundamental relationship between formal proofs and computation.

Of course, when people prove mathematical theorems, they do not do this using ZFC—this takes an enormous amount of time, even for very simple theorems, and is not well-suited to other people who read the proof. (Computer programs that check the validity of proofs in ZFC and related logical systems are a different story, however, and there has been a lot of interesting research done in this area.) Instead, you can think of a typical proof of a mathematical theorem as really being a sketch of a formal proof, giving someone with little creativity but a lot of stamina enough information that they could translate the proof to ZFC if they were so inclined. We often do something similar when describing an algorithm: translating high-level pseudo-code for some algorithm into assembly language is a lot like translating an ordinary proof into ZFC.