A Bunch of Set Theory and a Bit of Logic

CPSC 509: Programming Language Principles

Ronald Garcia*

14 January 2014
(**Time Stamp**: 11:12, Monday 9th November, 2020)

1 Representing the world using sets

Set theory is the "machine language" of mathematics. If you think about it, every program that you have ever run on a computer has ultimately produced instructions that get sent to a CPU and the CPU just churns through them. Those CPU instructions are really really low-level: add this 32 bit number to this 32 bit number, grab these bits from memory and move them somewhere else, check this number to see if it's zero, and grab these other set of bits from memory and interpret them like a machine instruction. Somehow, these little primitive instructions make it possible to write programs that download pictures of kittens from the Internet and thereby make your life more fulfilling. At the end of the day, though, it's all bits: 1's and 0's. Your kittens are *represented* by a bunch of bits that your graphics hardware knows how to translate into a bundle of furry joy. Your tragic poetry is written in ASCII, which is really a bunch of characters, each of which is represented by a number, which in turn is represented by bits. So your computer consistently operates on *representations* of the things that you really care about.

Set theory is like that. It's *painfully* low-level, and it doesn't understand high-level concepts like programming languages, or numbers for that matter, but it's very very powerful! Essentially, mathematicians take set theory as *the* tool for building representations of things. If you can't represent it using sets, then it's not mathematical.

2 Abstractions make this tractable

On computers and in mathematics, we keep this low-level stuff from exploding our brains by building up layers of *abstraction*. Much of the time we just ignore what's going on at the low-levels, but instead think in terms of higher-level concepts that we really care about. As long as your abstractions are well-designed and clearly explained, you can ignore the details. If your abstractions are "leaky", then you might write down things that make no sense to you but say something meaningful at the lower levels. For example, I'm willing to believe that the letter `'a'` comes before `'b'`, so in a sense: `'a' < 'b'`, and the C programming language tells me this, but why would I think that `')' < '+'`??? The C language tells me that this is true too, and if I didn't know anything about the *ASCII encoding of characters as numbers*, then I wouldn't understand what's going on. In short, C's abstraction of characters is leaky. If the language didn't let you compare characters in terms of their underlying machine representations, this weird stuff wouldn't happen.[1] Later, we'll see some examples of leaky abstractions in set theory.

Now don't get me wrong, sometimes it's useful to know what's going on under the hood, if only so you can debug problems or implement abstractions of your own. This is true on both computers and in

[1] In fact, some errors in C code arise from assuming that ASCII is always the relevant character encoding! See for example `https://stackoverflow.com/questions/16400009/why-the-char-comparison-ifc-a-c-z-is-not-portable`

mathematics. In essence, the ideal is to be a *full stack mathematician* who can work efficiently at a high level, but understand the underpinnings that ensure that this high-level work makes sense.

Finally, bear in mind that we sometimes use low-level concepts in our high-level reasoning. In programming, sometimes all you want is a set of bits. So bits appear in our upper-level abstraction. But be warned: just because you are talking about bits in your high-level, it doesn't mean that they are being *represented* exactly as bits at the low level! Perhaps each of your high-level bits is being represented by *an entire* 32-bit machine word, for convenience of implementation. If you don't care about the space usage, then maybe that's not a big deal. Similarly, we still use sets in our higher-level reasoning too: sometimes you just need a set of kittens to make your day. But your high-level notion of "set of kittens" is going to be interpreted somehow in set theory, and that interpretation may not be readily recognizable after you compile it down. It will surely be a set, because sets are all there is at the bottom, but a set of what? That is to say, we can really think about everything we're doing in mathematics as operations on sets.[2]

Now, to keep the navel gazing to a reasonable minimum, we are not going to build *all* of the mathematical concepts we need from the ground up. In practice, we are going to chose a few primitive notions in addition to sets and work with them. In this document, however, we will do a bit of wallowing in the set-theoretic muck, partly so you have some experience looking under the hood at the low-level parts, partly so you can appreciate the high-level abstractions we use, and partly so that you can do your own trouble-shooting and discern whether your high-level reasoning makes sense, we start at the bottom for a while. It's good to set some ground-rules for creating and manipulating sets. You may have seen some of these concepts before, but maybe not explained in this painful of detail. Enjoy!

3 Sets and Logic

Set theory, as we will use it in class, is really an embedding of notions of sets into a logical language called *first-order logic*, which gives us the tools that we need to actually say stuff about sets and prove that the things we say make sense. The two combine to create what we technically call *first-order set theory*. So they are necessarily intertwined. Ultimately we want to be relatively informal about our discussions of sets, just like mathematicians tend to be, but we want to make sure that if we really *had* to, we could get super-pedantic and formal about it, literally using first-order logic all the way down. For this reason, I will intersperse the casual informal set notation with the super-pedantic statements within logic, so you can tell precisely what we mean when we write some informal stuff down.

4 Sets are about Stuff

At its heart, set theory is a way of talking about these containers of stuff. So *everything* in set theory boils down to the *"element of"* relation $\in$. If one set A is an element of another set B, then we write $A \in B$. That's really all there is to set theory, but we will build up a lot of stuff on top of that single concept (see, we don't even have 1's and 0's just $\in$). First, we need to know how this $\in$ relationship works...what are the rules that constrain it? These are captured by axioms of set theory. I'm not going to give *all* of the axioms of set theory, just the ones that we will use a lot. In fact, there are lots of set theories out there, but many have a pretty common core. We'll be using a set of axioms for what is called *Zermelo-Frankel Set Theory* or ZFC for short [3]. Now let's talk about some of the axioms, which tell us how sets work as well as what kinds of sets are out there, i.e. what sets *exist*.[4]

5 When are two sets really the same set?

We start with a basic property of sets:

[2] One can argue about whether this is a good idea, just as one can argue whether the X86 instruction set is a good idea. Philosophers and logicians have proposed other foundations for mathematics, and with good reason! But from the early 20th century until now, set theory rules the day, much like the X86 ISA.

[3] don't worry about the "C" in ZFC, we won't be concerned with it.

[4] Note that there are many different equivalent sets of axioms for ZFC. I'm choosing the ones here for simplicity and utility.

A set is fully determined by its elements.

Think of a set as an invisible bag of stuff...all we identify are the stuff, not the material that the bag is made of. This is called the *Axiom of Extensionality*.[5]:

$$\forall S_1.\forall S_2.S_1 = S_2 \iff \forall S.S \in S_1 \iff S \in S_2.$$

In words, this reads: "for any sets S_1 and S_2, they are the same set if and only if they contain the same elements." Remember: the only elements that arise in set theory are sets, so we're just worried about what sets are contained inside a set. For our purposes, this axiom *defines* equality of sets.

Note the use of the symbol $\forall$ to mean "forall", and $\iff$ which means "if and only if". $\iff$ is really an abbreviation:

$$A \iff B \quad \equiv \quad (A \implies B) \wedge (B \implies A).$$

Here the wedge $\wedge$ is the symbol for "and", and the arrow $\implies$ is just "if-then" or "implies"; The $\equiv$ symbol can be read "is shorthand/longhand for." Think of it as a *macro*, like what you would write in the C Preprocessor. For now don't worry about how to read the above logical formula, we'll get into the details of how to understand these logical statements later. As a side note, we could have, in principle, treated equality $=$ it as a *macro* rather than a definition, but most first-order logics (like first-order set theory) assume that you have an explicitly defined concept of equality in addition to the other operations (in our case just $\in$), so set theory follows that lead by axiomatizing it.

But unpacking the axiom above a little more, we now get: "for any sets S_1 and S_2, we know that: 1) if they are equal, then any element of S_1 is an element of S_2 and vice versa; and 2) if any element of S_1 is an element of S_2 and vice versa, then S_1 and S_2 are equal, i.e. the same set."

Now we know how to judge that two sets are the same set. Since equality is a pretty key idea in logical systems, it's important that we lay this out good and early. Without a notion of equality, we wouldn't be able to talk about things like "*the* set that has no elements" as we will do in the next section.

The name of this axiom (extensionality) leads us the idea of an *extensional definition* of a set. When you want to describe a set with a finite number of elements, you can just use the list of elements as a way to refer to the set. For example, we might write the expression:

$$\{\,\text{cheese}, \text{crackers}, \text{vegemite}\,\}$$

to describe the *unique* set that contains exactly cheese, crackers, and vegemite. An expression of this form, which serves as a definition of a set,[6] and given such a definition, we get some logical reasoning principles "for free":

$$\forall S.S \in \{\,\text{cheese}, \text{crackers}, \text{vegemite}\,\} \iff S = \text{cheese} \vee S = \text{crackers} \vee S = \text{vegemite}.$$

Here, the $\vee$ symbol represents *inclusive* "or". The formula $A \vee B$ is true if and only if A is true, B is true, *or both A and B are true*. This use of inclusive or can be confusing since in informal speech we often say "or" to mean "either A or B but not both". The above extensional principle seems to say something obvious: that the only elements in an extensionally-described set are the ones listed. It's nice to have a formal account of this idea though, because we (or our computer programs) can manipulate this idea via symbol pushing (i.e. formal reasoning). In any case here is our first incident of a definition affecting our reasoning principles: an extensional definition of a set licenses you to reason about its elements by cases on its (finite) members.

A subtlety to keep in mind: the notation notation above is just a *description* of a set, sometimes called a *name* of a set, not the set itself. For instance, the expressions $\{\,\text{cheese}\,\}$ and $\{\,\text{cheese}, \text{cheese}\,\}$ both describe the exact same set: the one containing cheese and nothing else. However, each description leads to a subtly different reasoning principle. Can you write down the reasoning principles and observe how they are (somewhat boringly) different? Later we will find that having multiple different definitions of the same set can give us *useful* reasoning power.

[5] Roughly speaking, a set's *extent* is "the stuff that it is made up of."

[6] For our purposes, the words "definition" and "description" mean exactly the same thing. I actually prefer the word "description", but mathematics seems to prefer "definition."

6 The Empty Set

So we now know how to tell if two sets are really the same, *but we don't know if there are any sets*. Don't let the "forall" fool you: we can state that "All flying pigs prefer to drive Teslas." which is true, but not because flying pigs don't like bicycles, but rather because there are no flying pigs in the first place![7] So we need to fix this issue at least and ensure that there's a set.

One set that's very important to us is the set with nothing in it: *the empty set*. So we declare the *Axiom of the Empty Set*.:

> There is an empty set, i.e. there exists a set with nothing in it.

In logical form, this looks like the following:

$$\exists S_1.\forall S_2.S_2 \in S_1 \implies \bot.$$

Here, the logical symbol $\bot$ represents *falsehood*. In our logic, $A \implies \bot$ is our way of writing "not A." You can read it as "Well if A then pigs fly!" as in, there is no way that A holds. If you have experience with logic, then this may look a bit odd compared to other presentations of negation, but there is some actual sense to this. In fact, we sometimes abbreviate this kind of negation. So for instance $\neg A$ is a syntactic shorthand for $A \implies \bot$. Also, we abbreviate the idea that some relation doesn't hold by crossing it out. For example, $A \notin B$ is shorthand for $A \in B \implies \bot$ and $X \neq Y$ is shorthand for $X = Y \implies \bot$. In short, negation is defined in terms of implication and falsehood. Keeping this in mind will help you prove theorems later.

Now, we can combine the empty set axiom with the axiom of extensionality to prove that the empty set is unique, i.e., there is at least one and at most one set that is empty.

Proposition 1. *The empty set is unique, i.e.:*

$$(\exists S_1.\forall S_2.S_2 \in S_1 \implies \bot) \wedge \forall S_1, S_2.\,(\forall S.S \in S_1 \implies \bot) \wedge (\forall S.S \in S_2 \implies \bot) \implies S_1 = S_2.$$

Reading the above formula literally, it reads "there exists a set that has nothing in it, and any two sets with nothing in them are actually the same set." This is why we get to call it *the* empty set, rather than *an* empty set, by the way. Thus the concept of empty-setness is a *definite description* of a set: it suffices to point out a particular set. On the other hand an *indefinite description* describes one of many possible entities.

Before I prove this proposition, let's talk notation. We can simplify the logical formula using a couple of abbreviations. First, we use $\notin$ to mean "not an element": $S_1 \notin S_2 \equiv S_1 \in S_2 \implies \bot$. This gives:

$$\exists S_1.(\forall S_2.S_2 \notin S_1) \wedge \forall S_3.(\forall S_2.S_2 \notin S_3) \implies S_3 = S_1.$$

Then we can introduce an abbreviation for uniqueness. Let $\Phi(X)$ be some *predicate*, i.e. logical statement, about S, written in terms of X.[8] For example, we could have $\Phi(X) \equiv (\forall S_2.S_2 \notin X)$. Then when we write

$$\exists! X.\Phi(X)$$

To mean that there is a *unique* X such that $\Phi(X)$ holds. Using these two abbreviations, we can succinctly rewrite our proposition as:

$$\exists! S_1.\forall S_2.S_2 \notin S_1$$

In short: "There's a unique set that has no elements." This means *exactly the same thing* as the original proposition (because it "macro-expands" to the original). Now that we know that there is only one empty set, we give it a special name of its own: $\emptyset$. By giving it a name, we can ignore the $\exists! S.$ stuff, and just remember the fact that: $\forall S.S \notin \emptyset$. We can use that fact to prove things about the empty set. We will call this the *reasoning principle* that we associate with the set name (a.k.a. *description*) $\emptyset$.

I'll write a proof of the above proposition without much explanation. We'll get into how to prove logical formulas later. For now the thing to keep in mind is that I am thinking in terms of the original proposition above when I am writing the proof, not the small and abbreviated statement just above.

[7] Barring recent advances in biology of which I am unaware.

[8] Technically, X doesn't have to appear in it at all, but then whatever you choose to replace X with doesn't matter.

Proof. Consider the empty set $\emptyset$. By the axiom of the empty set, we know that $\forall S.S \notin \emptyset$. That handles the first part (existence).

Now suppose two sets, S_1, S_2 such that $\forall S.S \notin S_1$ and $\forall S.S \notin S_2$. Then to prove that $S_1 = S_2$ it suffices, but the axiom of extensionality, to prove that for any set S, if $S \in S_1$ then $S \in S_2$ and vice-versa.

Let's do the first: suppose S is some set and $S \in S_1$. Then applying our assumption $\forall S.S \notin S_1$ to it yields a contradiction $\perp$, from which we immediately deduce $S \in S_2$. We can perform the analogous deduction in the opposite direction to prove that if $S \in S_2$ then $S \in S_1$. Thus we've proven this for all sets S so $S_1 = S_2$. □

Now wait a second! Why didn't we just make the uniqueness of the empty set an axiom? Well, because we didn't need to: the axiom of extensionality already implied it, so why say something more complicated? That's how mathematicians tend to think in my experience: keep the number and strength of axioms super-small so that you can get your work done with as small of a *trusted base* as possible. It doesn't make your working set of properties particularly ergonomic, but the idea is that you can build up higher-level tools (i.e. propositions) from here with confidence and then use those in your day-to-day work.

7 Sets in Sets in Sets

Okay: so far we know what it means for two sets to be equal and we know that there's an empty set. Any other sets out there? Not so far! Let's fix that, first with a simple axiom, the *Axiom of Pairing*:

> For any sets A and B, there is a set C that contains both of them as elements.

i.e.,

$$\forall A.\forall B.\exists C.A \in C \wedge B \in C$$

This axiom is a bit obnoxious, because it tells us that there's a set with A and B, but it says *absolutely nothing* about whether there is anything else in the set C. There may be tons of stuff! This is reminiscent of how we didn't say that the empty set is unique: later we will be given some axioms that we can use to prove that there is indeed a set with *exactly* A and B in it. We would write such a set $\{ A, B \}$. In essence, $\{ A, B \}$ is a *name* for "the set that has A in it, and B in it, and nothing else. Note though, that if A and B are the same set, then we are talking about a set that actually has only one element in it. That is to say, $\{ A, A \} = \{ A \}$. It's important to keep this possibility in mind when developing abstract arguments.

Using what we know so far, we know that the set $\{ \emptyset \}$, that is, the set containing one element, namely the empty set, exists. Using the axiom of pairing again, we can show that $\{ \emptyset, \{ \emptyset \} \}$ also exists. Look there are sets! And if we keep repeating this kind of pairing, we can show that there are an infinite number of sets. Phew that's a lot! However, each one of these sets only has a finite number of elements, which will not be enough for our purposes.[9]

Another useful way to "find" sets that combine other sets is to take the *union* of a collection of sets, i.e., a set of sets. The *Axiom of Union* supports this:

> For every set S there is a set G that contains as elements the contents of every set that S contains.

i.e.,

$$\forall S.\exists G.\forall A.\forall a.(a \in A \wedge A \in S) \implies a \in G.$$

To make sense of the formal notation, let's consider an example. Let $S = \{ \{ 1, 2 \}, \{ 3, 4 \} \}$. Then from the axiom, we know that there is some set G such that each of the given numbers is an element of G, i.e., $1 \in G, 2 \in G$, and so on.

Again, we'll be able to show later that there is a (unique) set that consists of *exactly* those elements of elements of S. We call that the union, and use the notation $\bigcup S$ for that set. Then, the notation $A \cup B$ is just another way to write $\bigcup \{ A, B \}$. Why define the binary union in terms of the bit one? Well, because we can't form the union of an infinite number of sets by doing it two at a time...we'll get bored eventually. However, if we have a set with an infinite number of things (see below!), then we can form an infinite union whiz bang!

[9] hint: how many C programs are there? So how big is the set of all C programs?

In any case, this gives us a reasoning principle for sets formed using union.

$$\forall S. \forall A. A \in \bigcup S \iff \exists G. A \in G \wedge G \in S.$$

Thus we've turned the axiom of union into a reasoning principle about *the* union of a family of sets.

8 Extensional set definitions from the ground up

So now we have enough machinery to justify one of the common ways that we write sets: extensionally! We've already used such definitions as examples, but now we have enough machinery to build them up from the axioms of set theory. You can think of this as akin to bootstrapping a compiler (I'll leave it to you to work out the analogy in full). A set is a collection of stuff, and the most obvious way to write it is with set notation. I'm sure you've seen this before. For example, the set of even positive integers smaller than 10: $\{ 2, 4, 6, 8, 10 \}$. You can think of this notation as a very descriptive *name* for a particular set.

Do we know that a set like $\{ a, b, c \}$ exists? In pedantic form the statement is:

$$\exists A. \forall S. S \in A \iff S = a \vee S = b \vee S = c,$$

The answer is clearly yes. One way to do it is:

1. use the axiom of pairing to form $\{ a, b \}$;
2. use the axiom of pairing to form $\{ c \}$;
3. use the axiom of pairing to form $\{ \{ a, b \}, \{ c \} \}$;
4. use the axiom of union to form $\{ a, b, c \}$.

Well that was exhausting! I will never bother doing it again. But note, that we have a reasoning principle for this set based simply on its name:

$$\forall S. S \in \{ a, b, c \} \iff S = a \vee S = b \vee S = c,$$

Notice that this makes sense for a set name like $\{ a, a \}$, though it's mighty redundant: we like the name $\{ a \}$ better. But this observation matters particularly when you are making an abstract argument. For instance, if I assume a and b, and then mention "the set $\{ a, b \}$", you can't be sure that the set $\{ a, b \}$ has two elements: what if $a = 5$ and $b = 5$? On the other hand, if you know a priori that $a \neq b$, then the set $\{ a, b \}$ definitely has two elements. Notice, though, that in both cases it's true that $a \in \{ a, b \}$ and $b \in \{ a, b \}$.

9 Infinite stuff!

Given the tools we have so far, we can only build finite sets! That's because all of our logical arguments have to have a finite number of steps, which can really be a pain sometimes, but that's how logicians roll. This is also how computer programmers roll: ever seen an infinite-length program?

So how do we get infinite stuff? Well, like we did with the empty set, we *assume it*, via the *Axiom of Infinity*:

> There's a set that (1) contains the empty set and (2) contains the set $a \cup \{ a \}$ whenever it contains a.

i.e.,

$$\exists S. (\emptyset \in S) \wedge (\forall A. A \in S \implies A \cup \{ A \} \in S).$$

Notice how I used $A \cup \{ A \}$ in the above formula, rather than spelling the whole thing out in terms of $\in$, $\exists$, uniqueness, etc. That would be way too painful! Luckily we can already exploit our paucity of abstractions to make life more pleasant. You may as an exercise try to boil this all the way down to just statements about elementhood.

Set theorists like to call the set that has *only* the elements above ω, and they use it to represent the set of natural numbers:[10]

$$\begin{aligned} 0 &\equiv \emptyset \\ 1 &\equiv \{\, 0 \,\} \cup 0 = \{\, \emptyset \,\} \\ 2 &\equiv \{\, 1 \,\} \cup 1 = \{\, \{\, \emptyset \,\}, \emptyset \,\} \\ &\vdots \end{aligned}$$

First, notice the leaky abstraction: when was the last time you took the union of a set with a number??? Let's not do that any more! instead, we will just consider the nice abstract set of *natural numbers* as the set,

$$\mathbb{N} = \{\, 0, 1, 2, \ldots \,\},$$

and say that questions like $1 \in 2$ are off limits...that would be exploiting the *representation* of numbers as sets, rather than the properties of numbers *as an abstraction* that we care about. Notice that 0 is a natural number. All right-thinking computer scientists will use this definition.[11]

10 All the subsets

Now we add yet another axiom that lets us build bigger sets: the *Axiom of Power Set*:

> Given any set A, there exists a set that contains all subsets of A.

i.e.,

$$\forall A. \exists S. \forall B. (\forall C. C \in B \implies C \in A) \implies B \in S.$$

Good grapes, just looking at that makes my eyes bleed! In the above formalism, B is a subset of A because every element C of B is also an element of A. Because we use subsets so often, there's a convenient notation for them:

$$X \subseteq Y \equiv \forall C. C \in X \implies C \in Y.$$

With this we could have written:

$$\forall A. \exists S. \forall B. B \subseteq A \implies B \in S.$$

Once again, this axiom doesn't say that the set *only* contains subsets of A, but that set exists. We call it the *powerset* of A, and give it the notation $\mathcal{P}(A)$. Some books use the notation 2^A, which sort of explains the name "powerset" (2 to the power of A): the reason for that notation is that if some set A has n elements, then $\mathcal{P}(A)$ has 2^n elements.

The axiom above and the uniqueness of the powerset give us another reasoning principle:

$$X \in \mathcal{P}(A) \iff X \subseteq A.$$

Notice how in the above I didn't quantify the X or the A. This is a common practice of mathematicians. If you see something like this with missing quantifiers, assume that there are a bunch of forall quantifiers in the front, so this is shorthand for:

$$\forall X. \forall A. X \in \mathcal{P}(A) \iff X \subseteq A.$$

[10] This encoding is due to the great logician and computer scientist John von Neumann.

[11] I once went to a wedding of two computer scientists, and during dinner they sat at "Table 0" because, as the table manifest said: "computer scientists start counting from zero."

11 Picking Things Out

The axioms that we have given so far allow us to identify bigger sets in terms of smaller sets. As we'll discover, most of the time we instead identify *smaller* sets by filtering desirable elements out of some bigger set. The tool that we use for that is called the *Axiom Schema of Specification* (and also the *Axiom Schema of Separation*)

> Let A be some set and let $\Phi(X)$ be some logical predicate on sets X. Then there is a set A of all and only the elements $a \in A$ such that $\Phi(a)$.

i.e. for every predicate $\Phi(X)$ we have the axiom:

$$\forall A. \exists S. \forall B. B \in S \iff B \in A \wedge \Phi(B).$$

This one is called an axiom *schema* because it really stands for an infinite number of axioms: one for every Φ that you can come up with. That's a little crazy, but it's how it works. Not my fault!

Notice how this axiom schema is phrased as an if-and-only if, so we can pick out a set whose contents *exactly* match the result of filtering, unlike earlier axioms like the axiom of pairing. This axiom is the one that we can use to winnow things down to exact sets.

The set that we get via specification in question is unique, and we give it a name using what we call a *set comprehension*. given a set A and a property Φ, we use the name $\{ a \in A \mid \Phi(a) \}$ for the set that you get by filtering A with respect to Φ. Combining this name with our axiom gives us the following reasoning principle:

$$B \in \{ a \in A \mid \Phi(a) \} \iff B \in A \wedge \Phi(B).$$

This concludes the axioms of set theory that we will focus on. ZFC has more axioms, but they are mostly geared toward details we will barely touch on, if at all, so we can ignore them.

You may want to mess around with the axioms that we've covered so far to try and build various sets and see what a pain it is to hack in machine code.

Note that each mechanism that we have for defining/naming sets comes with its own reasoning principle. This is a recurring theme in mathematics and in this course in particular: *the structure of my definitions determines the structure of my reasoning.* Often we can describe reasoning principles directly in terms of some definition for picking out a set of interest.

Next, we'll consider some of the pragmatics of using set theory.

12 The Practice of working with Sets

In this section, I'll describe how we will actually informally work with sets. Some of this rehashes concepts that we covered above, but ideally this version is written in a more high-level, actionable way, corresponding to how we will use set theory in real practice.

12.1 Assumed Sets

From time to time we will simply *assume* the existence of a set.

It will look like:

> Suppose $a \in \textsc{Atom}$...

What we mean by this is that there must be some set with an infinite number of things in it that we can tell apart, and distinguish from the elements of other sets. So I can always pull out some a_i and then grab some other $a_j \neq a_i$. When we want to talk about specific members, we will usually use the same letter as the metavariable but typeset it in blue, as in $\mathtt{a}_1$. In the case of program variables, like $x \in \textsc{Var}$, we may be more creative with our choice of elements.

We can justify this because we already know that there is a set with an infinite number of things, and one of the nice/crazy things about such a set is that you can take an infinite number of things out of that set and still have *another* infinite amount of things left over. For example, if I take the odd numbers out of

the set of natural numbers, how many numbers are left over? An infinite amount: the even numbers! My head explodes when I think about this kind of thing.[12]

13 Set formations

Aside from naming a set by explicitly saying which elements are in it, we assume a number of ways of building sets. We're not trying to be super-primitive, or make sure that there's only one way to build a set. Rather, we just want to establish a set of primitive rules that we can always use for building new sets.

13.1 Union

We can always combine sets to form new sets, forming the union of two or any number of sets. If A is a set, and B is a set, then there is a set that we call $A \cup B$ with the properties that $c \in A \cup B$ if and only if $c \in A$ *or* $c \in B$.

Example 1. If $A = \{ \mathsf{meat}, \mathsf{cheese} \}$ and $B = \{ \mathsf{vegetables}, \mathsf{fruits} \}$ then $A \cup B = \{ \mathsf{meat}, \mathsf{cheese}, \mathsf{vegetables}, \mathsf{fruits} \}$.

We can extend this to collections of sets. If F is a set of sets, then we can form the set $\bigcup F$, which is the union of all the sets in F: $c \in \bigcup F$ if and only if $c \in C$ for *some* set $C \in F$.

Example 2. If $F = \{ \{ \mathsf{cow}, \mathsf{bull}, \mathsf{buffalo} \}, \{ \mathsf{sheep}, \mathsf{elpaca}, \mathsf{horse} \}, \{ \mathsf{frog}, \mathsf{toad}, \mathsf{turtle} \} \}$ then $\bigcup F = \{ \mathsf{cow}, \mathsf{bull}, \mathsf{buffalo}, \mathsf{sheep}, \mathsf{elpaca}, \mathsf{horse}, \mathsf{frog}, \mathsf{toad}, \mathsf{turtle} \}$.

Notice that $\bigcup \emptyset = \emptyset$, since we are taking the union of no sets, and $A \cup B = \bigcup \{ A, B \}$.

13.2 Intersection

In addition to forming a set by grabbing element that appear in *any* of a given set of sets, we can also consider the set of elements that appear in *every* set in a given set. That's the intersection. If A and B are sets, then there is a set we call $A \cap B$ with the property that $c \in A \cap B$ if and only if $c \in A$ and $c \in B$.

Example 3. If $A = \{ \mathsf{paper}, \mathsf{cut} \}$ and $B = \{ \mathsf{news}, \mathsf{paper} \}$ then $A \cap B = \{ \mathsf{paper} \}$.

Generalizing this idea, if F is a *non-empty* set of sets, then we can form the set $\bigcap F$, which is the intersection of all the sets in F: $c \in \bigcap F$ if and only if $C \in F$, and $c \in C$ for *every* set $C \in F$. Analogous to unions, $A \cap B = \bigcap \{ A, B \}$, but in stark contrast, there is no such set $\bigcap \emptyset$! This means that when you are reasoning about sets, you must establish that F is not empty before you take its intersection. In the case of both intersection and union, you must establish that F contains only sets before you take their union, otherwise it's an "abstraction error".[13]

13.3 Powerset

Given some set C, you can form it's *powerset*, which we call $\mathcal{P}(C)$, which is the set of all subsets of C: $A \in \mathcal{P}(C)$ if and only if $A \subseteq C$.

Example 4. $\mathcal{P}(\{ \mathsf{tic}, \mathsf{tac}, \mathsf{toe} \}) = \{ \emptyset, \{ \mathsf{tic} \}, \{ \mathsf{tac} \}, \{ \mathsf{toe} \}, \{ \mathsf{tic}, \mathsf{tac} \}, \{ \mathsf{tic}, \mathsf{toe} \}, \{ \mathsf{tac}, \mathsf{toe} \}, \{ \mathsf{tic}, \mathsf{tac}, \mathsf{toe} \} \}$.

Notice that the C has 3 elements, while $\mathcal{P}(C)$ has $2^3 = 8$ elements. Some texts will wrote 2^C as a different notation for $\mathcal{P}(C)$.

As a side note, we can characterize subset completely in terms of containment: If A and C are sets, then $A \subseteq C$ if and only if whenever $a \in A$, it follows that $a \in C$. We differentiate between subset $\subseteq$ and *strict subset* $\subset$, meaning that if $A \subset C$, then it cannot be the case that $A = C$.

[12] In the 19th century, the mathematician Richard Dedekind proposed roughly the above property as the defining characteristic of an infinite set.

[13] By "abstraction error" I mean that technically it's fine to take the union, because at the lowest level all we have are sets, but you would surely be violating an abstraction principle. For example, $\bigcup 1$ is nonsensical for numbers, but not for their von Neumann representation where $\bigcup 1 \equiv \bigcup \{ \emptyset \} = \emptyset \equiv 0$. Yuk!

13.4 Sequences and Products

In addition to sets, unordered collections of unique elements, we will often want to consider *sequences* of elements that have a specific order and are not necessarily unique. Some examples of this are *pairs* like $\langle 1, 2\rangle$ and $\langle 1, 1\rangle$, triples like $\langle 4, 5, 3\rangle$, and more generally *tuples*, meaning some ordered sequence of n elements. Note that for our purpose, sequences are *not* sets, so we don't talk about $1 \in \langle 1, 2\rangle$: that's an "abstraction error."[14]

Now that we have a notion of sequences/tuples, we can form sets of them. If A is a set, and B is a set, then we can refer to the *product* of A and B, which we name $A \times B$, and it has the property that $c \in A \times B$ if and only if $c = \langle a, b\rangle$ for some $a \in A$ and $b \in B$.

Example 5. If $A = \{ \mathsf{eat}, \mathsf{throw} \}$ and $B = \{ \mathsf{chicken}, \mathsf{darts} \}$ then
$A \times B = \{ \langle \mathsf{eat}, \mathsf{chicken}\rangle , \langle \mathsf{eat}, \mathsf{darts}\rangle , \langle \mathsf{throw}, \mathsf{chicken}\rangle , \langle \mathsf{throw}, \mathsf{darts}\rangle \}$.

There are several generalizations of products. You can write $A \times B \times C$ for the set of triples from these three sets: if you *really* want to write the product of $A \times B$ with C, you should parenthesize as $(A \times B) \times C$. This is just a convention since we are far more likely to want triples than pairs that include pairs in the first position.

Also, in analogy to arithmetic, the set $A^2 = A \times A$ and $A^3 = A \times A \times A$ and so on. Note that $A^0 = \{ \langle\rangle \}$, the set containing only the empty sequence (since it's the only "0-long" tuple).

Given a set A, we use A^* to refer to the set of all finite sequences of elements of A. We sometimes use the name ε to denote the empty sequence $\langle\rangle$.

$$A^* = \bigcup_{n\in\mathbb{N}} A^n = A^0 \cup A^1 \cup A^2 \cup \ldots .$$

Technically, I haven't told you how to construct this set (we need one more axiom of set theory (Replacement) to do it, but we're not going to use it often, so I'm eliding it)

13.5 Total Functions, Partial Functions, and Relations

From your days taking math classes you have a rough idea of what a function is: a mapping that takes one value to another. We can formalize that idea in set theory: if A is a set, and B is a set, then we have a set $A \to B$ of *total functions from A to B*. It is characterized as follows: $f \in A \to B$ if and only if

1. $f \subseteq A \times B$.
2. If $\langle a, b_1\rangle \in f$ and $\langle a, b_2\rangle \in f$ then $b_1 = b_2$. This says that f maps elements of A uniquely to elements of b. Notice that what this is saying is that $\langle a, b_1\rangle = \langle a, b_2\rangle$, so more informally, there is at most one pair in f with a in its first position.
3. If $a \in A$ then $\langle a, b\rangle \in f$ for some $b \in B$. This means informally that there is *at least* one pair in f with a in its first position.

We can relax the notion of total functions to get the notion of *partial functions*. Intuitively, a partial function is a mapping that doesn't necessarily map *every* element. From A and B, we can form the set of partial functions $A \rightharpoonup B$, where elements $f \in A \rightharpoonup B$ need only satisfy the first two properties of total functions: they don't have to map every element of A to some element in B, but any element that they do map, they map uniquely. It should be clear from this description that every total function is a partial function. Note as well, when we just say "function", we mean "total function."

A little bit of terminology about functions. Given a partial function $f : A \rightharpoonup B$, where this is just a common notation for $f \in A \rightharpoonup B$, we call A the function's *domain*; we call B the function's *codomain* (others call it the function's *range*). Sometimes when we need to talk about it, we may write $dom(f)$ and $cod(f)$ for the function's domain and codomain respectively.

[14] Bear in mind that since all we *really* have to work with are sets, we end up encoding pairs and other sequences using, well, sets. See `https://en.wikipedia.org/wiki/Ordered_pair#Defining_the_ordered_pair_using_set_theory` for a variety of pair encodings in set theory that have been developed by a variety of mathematicians and logicians.

Even more general than partial functions is the idea of *relations*. A relation captures many-to-many correspondences between elements of sets. The most direct analogue to a partial function is a *binary relation*, which is technically any set of pairs: a binary relation R on A and B is some subset $R \subseteq A \times B$. Relations need not be only binary: you can have a relation between three sets, i.e. $R \subseteq A \times B \times C$, and even a unary relation $R \subseteq A$, which is basically a way of representing some property of $A's$ by simply giving the subset of A that satisfies that property.

Notation We treat functions, partial functions, and relations with special notation sometimes.

In the case of a partial function f, we write $f(a) = b$ to mean that $\langle a, b\rangle \in f$. We also write $f(a)$ to refer to the element b such that $\langle a, b\rangle \in f$. In the case of total functions, the expression $f(a)$ always makes sense, because there must be some b that fits the bill, but for partial functions, that may be nonsense, because the partial function is undefined at a. If a partial function f is undefined at a, we write $(f(a)\uparrow$ which literally means $\forall b \in cod(f).\, \langle a, b\rangle \notin f$. The opposite, that f is defined at a is written $f(a)\downarrow$.

We often write binary relations using *infix* notation: we write $a\ R\ b$ to mean that $\langle a, b\rangle \in R$. For a familiar example, $1 \leq 2$ just means $\langle 1, 2\rangle \in \leq$ where $\leq \subseteq \mathbb{N} \times \mathbb{N}$. For relations among more than two sets, we will often write them in "mixed-fix" notation. For example, when we discuss type systems, we will write the *typing judgment* $\Gamma \vdash t : T$ to mean that $\langle \Gamma, t, T\rangle \in \cdot \vdash \cdot : \cdot$ where $\cdot \vdash \cdot : \cdot \subseteq \textsc{Tenv} \times \textsc{Term} \times \textsc{Type}$. As with this example, we often use dots $\cdot$ in the names of functions and relations to indicate the positions where arguments go. So we will refer to the typing judgment either by a usage, $\Gamma \vdash t : T$, or by the dotted name $\cdot \vdash \cdot : \cdot$. The nice thing about the usage version is that you can deduce the sets that make up the relation from its metavariables. The dot notation, though, is clearer about which elements are really parameters.

Mathematical Functions Don't Run! One thing that it might take time to get used to in Set Theory is that functions are not like functions in programming languages...they don't take an input, churn on it, and then spit out a result after a bunch of computation. They're NOT procedures! Rather, they are (possibly) infinite lookup tables. Often when we write down a function definition, using a bunch of *equations*, it looks almost exactly like a program in a functional programming language like Haskell,[15] but really that is just a way of *naming* a particular function, just like we treat $\{ a, b, c \}$ as the *name* of a set. Such names, or descriptions, give us certain reasoning principles, and it's that process of reasoning that corresponds to the kind of computation that a function definition in a computer does. So two mottos that arise are:

- *Functions are like database tables, not like procedures.*
- *Computation arises from deduction, not from definition.*

We'll return to this discussion later because it's a common sticking point for programmers when it comes to math. At least it was a sticking point for *this* programmer!

13.6 Set Comprehensions

Most of the set constructions that we have been describing so far make "bigger" sets from smaller sets. We are very careful about the ways that we can build sets so that we don't write down a description of a set that can't possibly exist.[16]

However, once you have a set, you can always build a smaller set by choosing some of the elements of the set. We call this *set comprehension*: Given some set A, and some *predicate* $P(a)$ that describes properties of elements a of A, we can form the set $\{ a \in A \mid P(a) \}$ which is the set of elements of A for which the predicate P holds.

One of the constructions we already developed, the intersection of two sets, can be described this way:

$$A \cap B = \{ a \in A \mid a \in B \}.$$

In this case, the predicate $P(a) \equiv a \in B$. Notice that you can't use set comprehension to describe the *union* of two sets without already having something that contains the union. That's why we have an axiom for union, but not one for comprehension.

[15] That's no coincidence: they purposely stole the notation!

[16] For more on this, look at the Wikipedia page on "Paradoxes of set theory."

Another example, which we are introducing for the first time here, is the notion of *set difference*:

$$A \setminus B = \{\, a \in A \mid a \notin B \,\}.$$

Notice that set difference is always well defined for any two sets A and B.

At first you may not be clear on what counts as a legal predicate. In general, a predicate must be a formula in first-order logic about sets and their elements. Here I'll give some more examples of set descriptions using comprehensions. I recommend that you work your way through the logical statements to make sure that you understand how they work.

Partial Functions

$$A \rightharpoonup B = \{\, f \in \mathcal{P}(A \times B) \mid \forall a \in A.\, \forall b_1, b_2 \in B.\ \langle a, b_1\rangle \in f \wedge \langle a, b_2\rangle \in f \implies b_1 = b_2 \,\}$$

Total Functions

$$A \to B = \{\, f \in A \rightharpoonup B \mid \forall a \in A.\, \exists b \in B.\ f(a) = b \,\}$$

14 Where is this coming from?

So....what hole did I pull these rules out of? Well, technically, "set theory" is not just one thing. There are lots of sets of rules that logicians and mathematicians over time have cobbled together, just like X86, Z80, and Power architectures have different instruction sets. However they all tend to have a pretty common core of things: you can build sets, take their union, intersection, yadayada.

The guiding principles behind these informal rules come from what is called Zermelo-Fraenkel set theory, after two logicians who came up with the rules[17].

15 The Syntax of First-Order Set Theory

So far we have been simply using logical notation without explanation, in the hope that seeing it in context would provide *some* understanding, much like an immersive natural language course. However logical notation is precise, and a precise understanding is necessary for mastery. Let me try for moment to be a bit more precise about at least the low-level parts of first-order set theory. We will not be formal about all of it, but let me introduce some of it.

A word of warning though. In the following I use formal notation like Backus-Naur Form (BNF) simply to suggest the structure of the language, assuming that you have seen BNF notation before and can intuitively interpret it. Later I will give a very formal interpretation to BNF for programming languages. One can apply the same analysis to the following development as well, but doing so would amount to "defining set theory using set theory", kind of like writing a Python interpreter in Python.[18]

Representing a language inside itself does not amount to a fully-formal definition, but thanks to our human intuition and prior learning, we can still learn a lot from it. That's the goal of this section.

The core of the syntax of logic can be described as follows:

$$\begin{array}{l} S \in \textsc{SetName}, \quad \Phi \in \textsc{Formula} \\ \Phi \quad ::= \quad S \in S \mid \bot \mid \top \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \Phi \implies \Phi \mid \forall S.\Phi \mid \exists S.\Phi \end{array}$$

Here the metavariable S denotes a *name* for a set. Some example names that we use are TERM, or $\{\,1, 2, 3\,\}$. The latter is not a set in and of itself, but is a *descriptive* name, but in the latter case we deduce properties of the set being named based on the structure of that name (whereas TERM is not immediately telling without knowing that the name TERM is equivalent to a more contentful set description).

[17] Technically another logician named Skolem came up with the same part that Fraenkel did, but he got left out. Poor guy, that happens a lot in science and other forms of knowledge production.

[18] Nothing wrong with that! The PyPy project (`http://pypy.org`) has amply demonstrated benefits to doing this, and the LISP family of programming languages has a strong tradition of defining *metacircular interpreters* [McCarthy, 1960].

Now for a few common abbreviations. First, since we are not set-theorists—exploring the properties of *arbitrary sets*—but rather programming language theorists—exploiting set theory to study sets of programs and related structures—we will mostly be considering elements of *particular sets*. This leads to a desire to qualify our quantifiers. We do that by using the following notations that expand as described:

$$\forall S_1 \in S_2.\, \Phi \equiv \forall S_1.\, S_1 \in S_2 \implies \Phi;$$
$$\exists S_1 \in S_2.\, \Phi \equiv \exists S_1.\, S_1 \in S_2 \wedge \Phi.$$

Take a moment to convince yourself that the expansions will have the intended meaning. It's indeed a bit curious that the expansion is not the same for universal and existential quantification, but this is necessary to induce the desired interpretation.

Also, since we often introduce many elements of the same set, we abbreviate this in terms of the above abbreviations:

$$\forall S_1, S_2, \ldots, S_n \in S.\, \Phi \equiv \forall S_1 \in S.\, \forall S_2 \in S.\, \ldots\, \forall S_n \in S.\, \Phi;$$
$$\exists S_1 \in S_2.\, \Phi. \equiv \exists S_1 \in S.\, \exists S_2 \in S.\, \ldots\, \exists S_n \in S.\, \Phi.$$

Furthermore, we can abbreviate using the same kind of quantifier for different kinds of elements:

$$\forall S_1, S_2, \ldots, S_n \in S, T_1, T_2, \ldots, T_n \in T, .\, \Phi \equiv \forall S_1, S_2, \ldots, S_n \in S. \forall T_1, T_2, \ldots, T_n \in T, .\, \Phi;$$
$$\exists S_1, S_2, \ldots, S_n \in S, T_1, T_2, \ldots, T_n \in T, .\, \Phi \equiv \exists S_1, S_2, \ldots, S_n \in S. \exists T_1, T_2, \ldots, T_n \in T, .\, \Phi;$$

Sometimes we wish to quantify elements of some n-ary relation. For example, Given some binary relation $\odot \subseteq S \times T$, we write write $S_1 \odot T_1 \equiv \langle S_1, T_2 \rangle \in (\cdot \odot \cdot)$ (occasionally using the $\cdot$ placeholders to indicate that we use *infix*, or more generally *mixed fix* notation) to indicate membership. Given such a relation, we use the following abbreviation to quantify elements of the relation in terms of elements of the subcomponents:

$$\forall (S_1 \odot T_2).\Phi \equiv \forall S_1 \in S, T_2 \in T.\, S_1 \odot T_2 \implies \Phi;$$
$$\exists (S_1 \odot T_2).\Phi \equiv \exists S_1 \in S, T_2 \in T.\, S_1 \odot T_2 \wedge \Phi.$$

Finally, we introduce some richer logical forms that are interpreted using the basic ones above:

$$\neg\Phi \equiv \Phi \implies \bot;$$
$$\Phi_1 \iff \Phi_2 \equiv (\Phi_1 \iff \Phi_2) \wedge (\Phi_2 \iff \Phi_1);$$
$$\exists! S.\, \Phi(S) \equiv (\exists S. \Phi(S)) \wedge (\forall S_1, S_2.\, \Phi(S_1) \wedge \Phi(S_2) \implies S_1 = S_2).$$

The first one represents "not Φ, the second represents "if and only if"; and the third represents "there exists a unique set S such that Φ." The notation $\Phi(S)$ Says that Φ is a formula that *may* refer to S freely in it. Once this notation is introduced, then $\Phi(T)$ Refers to the same formula but with T substituted for S.

References

J. Bagaria. Set theory. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. The Metaphysics Research Lab, winter 2014 edition, 2014. URL `http://plato.stanford.edu/archives/win2014/entries/set-theory/`.

J. Ferreirós. The early development of set theory. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, summer 2019 edition, 2019. URL `https://plato.stanford.edu/archives/sum2019/entries/settheory-early/`.

P. R. Halmos. *Naive Set Theory*. Springer-Verlag, first edition, Jan. 1960. ISBN 0387900926.
A classic introductory textbook on set theory.

P. Maddy. Believing the axioms. I. *The Journal of Symbolic Logic*, 53(02):481–511, 1988.
An interesting (though complicated) analysis of why set theorists believe in their axioms.

J. McCarthy. Recursive functions of symbolic expressions and their computation by machine, part I. *Commun. ACM*, 3(4):184–195, 1960.

S. Stenlund. Descriptions in intuitionistic logic. In S. Kanger, editor, *Proceedings of the Third Scandinavian Logic Symposium*, volume 82 of *Studies in Logic and the Foundations of Mathematics*, pages 197 – 212. Elsevier, 1975. URL `http://www.sciencedirect.com/science/article/pii/S0049237X08707328`.

M. Tiles. Book Review: Stephen Pollard. Philosophical Introduction to Set Theory. *Notre Dame Journal of Formal Logic*, 32(1):161–166, 1990.
A brief introduction to the philosophical issues underlying set theory as a foundation for mathematics.

Wikipedia. Zermelo-Fraenkel set theory, 2015. URL `https://en.wikipedia.org/wiki/ZFC`.