2 An Introduction to the Theory of Sets

Intuitively, the word *"set"* is meant to describe a *"collection of things."* The way that we will *formally* describe what sets are and what properties they have is by introducing the *axioms of set theory.* Before we get to the point where we can write and understand the axioms, we need to establish a couple of things.

$\Leftrightarrow$

$\Rightarrow$

In formal contexts, we will primarily speak using the *first-order logic.* A *first-order theory*[1] is a formal language built upon the first-order logic by establishing a *universe of discourse,* introducing *predicate symbols,*[2][3] and then writing down *axioms*[4] that describe the basic truths.[5] Since we are no longer doing logic *per se,* and thanks to the many theorems we proved about the first-order logic, we will now switch away from using the $\equiv$ and $\vdash$ symbols, which are not typically used by mathematicians. When referring to *logical equivalences* and *biconditionals,* we will now consolidate the $\equiv$ and $\leftrightarrow$ symbols into the $\Leftrightarrow$ symbol. Similarly, when referring to *entailments* or *conditionals,* we replace the $\vdash$ and $\rightarrow$ symbols by the $\Rightarrow$ symbol.

$:=$

There are two special *meta-symbols* that we will use for introducing new *definitions.* When we want to define a new kind of object, we use the $:=$ symbol; this can be used in one of two ways. If, in the context of something like a proof or an article, we want to give the label x to the number 5, we would use the syntax $x := 5$. This is usually understood to be a temporary label that only exists for the context of "the discussion," whatever it may be; that way, we could reuse the symbol for a different definition like $x := 6$ in a *different* proof or article. However, within a context (*e.g.,* a proof), once a symbol has been defined to mean something, you *can not modify what it means.* If you define $x := 5$ at the top of your proof, you can't then change your mind within the same proof and *"update"* x to be $5 + 1$. This is one important way in which mathematics has different *semantics* to programming.[6] The other way this can be used is to define a *general operation.* For example, we will soon define what the *union of two sets is:* given two sets A and B, we define their union to be $A \cup B := \{x \mid x \in A \ \vee \ x \in B\}$.[7]

$:\Leftrightarrow$

When we want to introduce a new *predicate,* we use the $:\Leftrightarrow$ symbol. We will see examples of this below when we introduce the basic predicate symbols of set theory.

2.1 Equality

Our first fundamental predicate is called *equality*, denoted by the $=$ symbol. Given two terms x and y, the syntax for using this predicate looks like either $x = y$ or $x \neq y$.

$$\begin{aligned} x = y \quad &:\Leftrightarrow \quad x \text{ and } y \text{ refer to the same object} \\ x \neq y \quad &:\Leftrightarrow \quad \neg(x = y) \end{aligned}$$

This predicate has three important properties that, despite probably being *"obvious"* to those with prior experience using this symbol, we will formalize explicitly below.

1. $\forall x(x = x)$. *Reflexivity*
2. $\forall x \forall y(x = y \ \Rightarrow \ y = x)$. *Symmetry*
3. $\forall x \forall y \forall z\big((x = y \ \wedge \ y = z) \ \Rightarrow \ x = z\big)$. *Transitivity*

The significance of these properties is that, whenever we have two terms x and y such that $x = y$, we can *replace* any occurrence of x with y (and *vice versa*).

[1] … like *ZFC set theory*…

[2] … and their properties…

[3] In other contexts, it may also be important to introduce *functional symbols* and *constant symbols.*

[4] … which are first-order sentences written in terms of the predicate (and possibly other) symbols…

[5] In the same way that *predicate symbols* transform terms into truth values, *functional symbols* transform terms into other terms. *Constant symbols* are just constant terms that are considered important enough to specify upfront for the whole theory.

[6] … unless you're working with a pure (usually functional) programming language (*e.g.,* Agda, Haskell, Idris, Lean) in which state is *immutable*…

[7] We don't understand what this means yet; that's okay. This was just meant to showcase how we use $:=$ to define a new operation that takes objects as inputs and produces objects as outputs.

2.2 *Elementhood*

Since *"sets"* are meant to be *"collections of things,"* our second fundamental predicate helps us describe when one object is *"inside of"* another one. This predicate is usually called *elementhood*, denoted by the $\in$ symbol. Given two terms x and y, the syntax for using this predicate looks like either $x \in y$ or $x \notin y$.

$$x \in y \quad :\Leftrightarrow \quad x \text{ is an element of } y$$
$$x \notin y \quad :\Leftrightarrow \quad \neg(x \in y)$$

… in other words, x is *"contained physically inside of" y*…

This predicate doesn't have any stand-alone properties like equality does. Instead, we will understand how $\in$ works based on how it interacts with the $=$ symbol. We will encode this understanding formally by introducing some new *notation* and some *axioms for set theory.*

2.3 *Notation*

Given finitely many terms x_0, x_1, …, x_{n-1}, we introduce the notation $\{x_0, x_1, \ldots, x_n\}$ to denote the set containing exactly those elements x_0, x_1, …, x_{n-1}. This is sometimes called *set roster notation* or *set builder notation*.[8] We define below what this notation formally means by explaining what it means for an arbitrary term x to be an *element* of the set described by the $\{x_0, x_1, \ldots, x_{n-1}\}$ notation.

$$x \in \{x_0, x_1, \ldots, x_{n-1}\} \quad :\Leftrightarrow \quad (x = x_0) \lor (x = x_1) \lor \cdots \lor (x = x_{n-1})$$

For example, consider three objects a, b, and c. Then, for any x, we say $x \in \{a, b, c\}$ is equivalent to saying $x = a$ or $x = b$ or $x = c$. The biggest restriction of this notation is that *there can only be finitely many elements,* and they must *all be written out explicitly* between the brackets. Trying to describe a set by writing an expression that looks like $\{0, 1, 2, \ldots, 127\}$ is an ***abuse of notation*** and is fundamentally *informal.*

The above notation is clearly cumbersome when dealing with a set containing a *"large"* amount of elements and can not be used at all when describing *"infinite"* sets.

2.4 *The Axiom of "Infinity" (Informal)*

The only existentially quantified axiom we will introduce, which we call the *Axiom of Infinity,* is unfortunately an axiom that we are not yet ready to fully understand or appreciate. We will instead introduce this axiom *informally* for now; when we are ready, after studying the rest of the relevant axioms of set theory, we will come back to this.

Axiom 0: Infinity (Informal).
The "set of all natural numbers" exists.

We use the symbol $\mathbb{N}$ to refer to this "set of all natural numbers," and *informally speaking* it looks like $\{0, 1, 2, \ldots\}$. We will revisit this in more detail later.

2.5 *The Axiom of Extensionality*

Our first significant axiom begins formalizing what the elementhood predicate $\in$ *means* in terms of equality $=$ by asserting that *sets are equal when they have the same elements.*

[8] … though in most other contexts, *"set builder notation"* is actually synonymous with *"set comprehension notation"* as we later describe it…

Axiom 1: Extensionality.
$\forall x \forall y \big(x = y \iff \forall z (z \in x \iff z \in y) \big)$.

In English, this says that, given two sets x and y, we determine $x = y$ if and only if every element of x is also an element of y and *vice versa.* For example, the sets $\{0, 0, 1, 2\}$ and $\{2, 1, 0\}$ are equal because they each contain *all* and *only* the elements 0, 1, and 2.

$\subseteq$

Given two sets x and y, we define x to be a *subset* of y when every element of x is also an element of y. Formally, we say $x \subseteq y :\iff \forall z (z \in x \Rightarrow z \in y)$. You might now be able to notice that this definition is closely related to the Axiom of Extensionality.

Theorem 2.1.
$\forall x \forall y \big(x = y \iff (x \subseteq y \land y \subseteq x) \big)$.

Proof. Let x and y be sets. Suppose $x = y$ and let z be an arbitrary set. Then, by the *axiom of extensionality,* we know $z \in x \iff z \in y$. This means $z \in x \Rightarrow z \in y$, and also $z \in y \Rightarrow z \in x$. Clearly, since z was arbitrary, we can then say $\forall w (w \in x \Rightarrow w \in y)$ and $\forall w (w \in y \Rightarrow w \in x)$. Therefore, $x \subseteq y$ and $y \subseteq x$ by definition.

We will demonstrate that $x = y \iff (x \subseteq y \land y \subseteq x)$ by showing the *forward* and *backward* directions separately.

Now, in the other direction, suppose $x \subseteq y$ and $y \subseteq x$ and let z be an arbitrary set. By definition, we know $\forall w (w \in x \Rightarrow w \in y)$, so that $z \in x \Rightarrow z \in y$. Similarly, we know $\forall w (w \in y \Rightarrow w \in x)$ by definition, so that $z \in y \Rightarrow z \in x$. Thus, $z \in x \iff z \in y$. Since z was arbitrary, we can then say $\forall w (w \in x \iff w \in y)$, from which we conclude $x = y$ by the *axiom of extensionality.*

Therefore, $x = y \iff (x \subseteq y \land y \subseteq x)$.

QED

2.6 *The Axiom of Pairing*

Our next axiom encodes something that should feel very intuitive based on the idea that sets are *"collections of things."* If we take two things that we know exist (because they're objects in our universe of discourse), we should be able to put those two things together in a collection by themselves, and that collection should exist. In other words, we should be allowed to construct *unordered pairs* of things.

Axiom 2: Pairing.
$\forall x \forall y \exists z \big(z = \{x, y\} \big)$.