Advanced Logic

Auburn University, 2024

Open Logic Project

Contents

Part I

Sets, Relations, Functions

Chapter 1

Sets

1.1 Extensionality

A *set* is a collection of objects, considered as a single object. The objects making up the set are called *elements* or *members* of the set. If x is an element of a set a, we write $x \in a$; if not, we write $x \notin a$. The set which has no elements is called the *empty* set and denoted "$\emptyset$".

It does not matter how we *specify* the set, or how we *order* its elements, or indeed how *many times* we count its elements. All that matters are what its elements are. We codify this in the following principle.

Definition 1.1 (Extensionality). If A and B are sets, then $A = B$ iff every element of A is also an element of B, and vice versa.

Extensionality licenses some notation. In general, when we have some objects $a_1, \dots, a_n$, then $\{a_1, \dots, a_n\}$ is *the* set whose elements are $a_1, \dots, a_n$. We emphasise the word "*the*", since extensionality tells us that there can be only *one* such set. Indeed, extensionality also licenses the following:

$$\{a, a, b\} = \{a, b\} = \{b, a\}.$$

This delivers on the point that, when we consider sets, we don't care about the order of their elements, or how many times they are specified.

Example 1.2. Whenever you have a bunch of objects, you can collect them together in a set. The set of Richard's siblings, for instance, is a set that contains one person, and we could write it as $S = \{\text{Ruth}\}$. The set of positive integers less than 4 is $\{1, 2, 3\}$, but it can also be written as $\{3, 2, 1\}$ or even as $\{1, 2, 1, 2, 3\}$. These are all the same set, by extensionality. For every element of $\{1, 2, 3\}$ is also an element of $\{3, 2, 1\}$ (and of $\{1, 2, 1, 2, 3\}$), and vice versa.

Frequently we'll specify a set by some property that its elements share. We'll use the following shorthand notation for that: $\{x \mid \phi(x)\}$, where the

$\phi(x)$ stands for the property that x has to have in order to be counted among the elements of the set.

Example 1.3. In our example, we could have specified S also as

$$S = \{x \mid x \text{ is a sibling of Richard}\}.$$

Example 1.4. A number is called *perfect* iff it is equal to the sum of its proper divisors (i.e., numbers that evenly divide it but aren't identical to the number). For instance, 6 is perfect because its proper divisors are 1, 2, and 3, and $6 = 1 + 2 + 3$. In fact, 6 is the only positive integer less than 10 that is perfect. So, using extensionality, we can say:

$$\{6\} = \{x \mid x \text{ is perfect and } 0 \leq x \leq 10\}$$

We read the notation on the right as "the set of x's such that x is perfect and $0 \leq x \leq 10$". The identity here confirms that, when we consider sets, we don't care about how they are specified. And, more generally, extensionality guarantees that there is always only one set of x's such that $\phi(x)$. So, extensionality justifies calling $\{x \mid \phi(x)\}$ *the* set of x's such that $\phi(x)$.

Extensionality gives us a way for showing that sets are identical: to show that $A = B$, show that whenever $x \in A$ then also $x \in B$, and whenever $y \in B$ then also $y \in A$.

1.2 Subsets and Power Sets

We will often want to compare sets. And one obvious kind of comparison one might make is as follows: *everything in one set is in the other too*. This situation is sufficiently important for us to introduce some new notation.

Definition 1.5 (Subset). If every element of a set A is also an element of B, then we say that A is a *subset* of B, and write $A \subseteq B$. If A is not a subset of B we write $A \not\subseteq B$. If $A \subseteq B$ but $A \neq B$, we write $A \subsetneq B$ and say that A is a *proper subset* of B.

Example 1.6. Every set is a subset of itself, and $\emptyset$ is a subset of every set. The set of even numbers is a subset of the set of natural numbers. Also, $\{a, b\} \subseteq \{a, b, c\}$. But $\{a, b, e\}$ is not a subset of $\{a, b, c\}$.

Example 1.7. The number 2 is an element of the set of integers, whereas the set of even numbers is a subset of the set of integers. However, a set may happen to *both* be an element and a subset of some other set, e.g., $\{0\} \in \{0, \{0\}\}$ and also $\{0\} \subseteq \{0, \{0\}\}$.

Extensionality gives a criterion of identity for sets: $A = B$ iff every element of A is also an element of B and vice versa. The definition of "subset" defines $A \subseteq B$ precisely as the first half of this criterion: every element of A is also an element of B. Of course the definition also applies if we switch A and B: that is, $B \subseteq A$ iff every element of B is also an element of A. And that, in turn, is exactly the "vice versa" part of extensionality. In other words, extensionality entails that sets are equal iff they are subsets of one another.

Proposition 1.8. *$A = B$ iff both $A \subseteq B$ and $B \subseteq A$.*

Now is also a good opportunity to introduce some further bits of helpful notation. In defining when A is a subset of B we said that "every element of A is ...," and filled the "..." with "an element of B". But this is such a common *shape* of expression that it will be helpful to introduce some formal notation for it.

Definition 1.9. $(\forall x \in A)\phi$ abbreviates $\forall x(x \in A \supset \phi)$. Similarly, $(\exists x \in A)\phi$ abbreviates $\exists x(x \in A \,\&\, \phi)$.

Using this notation, we can say that $A \subseteq B$ iff $(\forall x \in A)x \in B$.

Now we move on to considering a certain kind of set: the set of all subsets of a given set.

Definition 1.10 (Power Set). The set consisting of all subsets of a set A is called the *power set of A*, written $\wp(A)$.

$$\wp(A) = \{B \mid B \subseteq A\}$$

Example 1.11. What are all the possible subsets of $\{a, b, c\}$? They are: $\emptyset$, $\{a\}$, $\{b\}$, $\{c\}$, $\{a, b\}$, $\{a, c\}$, $\{b, c\}$, $\{a, b, c\}$. The set of all these subsets is $\wp(\{a, b, c\})$:

$$\wp(\{a, b, c\}) = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{b, c\}, \{a, c\}, \{a, b, c\}\}$$

1.3 Some Important Sets

Example 1.12. We will mostly be dealing with sets whose elements are mathematical objects. Four such sets are important enough to have specific names:

$$\mathbb{N} = \{0, 1, 2, 3, \ldots\} \qquad \text{the set of natural numbers}$$

$$\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\} \qquad \text{the set of integers}$$

$$\mathbb{Q} = \{{}^{m}/_{n} \mid m, n \in \mathbb{Z} \text{ and } n \neq 0\} \qquad \text{the set of rationals}$$

$$\mathbb{R} = (-\infty, \infty) \qquad \text{the set of real numbers (the continuum)}$$

These are all *infinite* sets, that is, they each have infinitely many elements.

As we move through these sets, we are adding *more* numbers to our stock. Indeed, it should be clear that $\mathbb{N} \subseteq \mathbb{Z} \subseteq \mathbb{Q} \subseteq \mathbb{R}$: after all, every natural number is an integer; every integer is a rational; and every rational is a real. Equally, it should be clear that $\mathbb{N} \subsetneq \mathbb{Z} \subsetneq \mathbb{Q}$, since -1 is an integer but not a natural number, and ${}^{1}/_{2}$ is rational but not integer. It is less obvious that $\mathbb{R} \subsetneq \mathbb{Q}$, i.e., that there are some real numbers which are not rational.

We'll sometimes also use the set of positive integers $\mathbb{Z}^{+} = \{1, 2, 3, \ldots\}$ and the set containing just the first two natural numbers $\mathbb{B} = \{0, 1\}$.

Example 1.13 (Strings). Another interesting example is the set A^* of *finite strings* over an alphabet A: any finite sequence of elements of A is a string over A. We include the *empty string* Λ among the strings over A, for every alphabet A. For instance,

$$\mathbb{B}^* = \{\Lambda, 0, 1, 00, 01, 10, 11, \\ 000, 001, 010, 011, 100, 101, 110, 111, 0000, \ldots\}.$$

If $x = x_1 \ldots x_n \in A^*$is a string consisting of n "letters" from A, then we say *length* of the string is n and write $\mathrm{len}(x) = n$.

Example 1.14 (Infinite sequences). For any set A we may also consider the set A^{ω} of infinite sequences of elements of A. An infinite sequence $a_1 a_2 a_3 a_4 \ldots$ consists of a one-way infinite list of objects, each one of which is an element of A.

1.4 Unions and Intersections

In section 1.1, we introduced definitions of sets by abstraction, i.e., definitions of the form $\{x \mid \phi(x)\}$. Here, we invoke some property ϕ, and this property

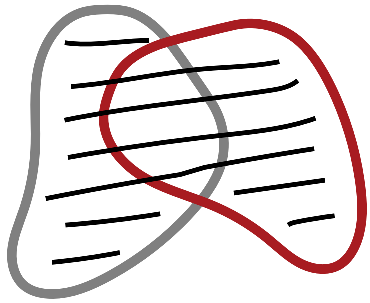


Figure 1.1: The union $A \cup B$ of two sets is set of elements of A together with those of B.

can mention sets we've already defined. So for instance, if A and B are sets, the set $\{x \mid x \in A \lor x \in B\}$ consists of all those objects which are elements of either A or B, i.e., it's the set that combines the elements of A and B. We can visualize this as in Figure 1.1, where the highlighted area indicates the elements of the two sets A and B together.

This operation on sets—combining them—is very useful and common, and so we give it a formal name and a symbol.

Definition 1.15 (Union). The *union* of two sets A and B, written $A \cup B$, is the set of all things which are elements of A, B, or both.

$$A \cup B = \{x \mid x \in A \lor x \in B\}$$

Example 1.16. Since the multiplicity of elements doesn't matter, the union of two sets which have an element in common contains that element only once, e.g., $\{a, b, c\} \cup \{a, 0, 1\} = \{a, b, c, 0, 1\}$.

The union of a set and one of its subsets is just the bigger set: $\{a, b, c\} \cup \{a\} = \{a, b, c\}$.

The union of a set with the empty set is identical to the set: $\{a, b, c\} \cup \emptyset = \{a, b, c\}$.

We can also consider a "dual" operation to union. This is the operation that forms the set of all elements that are elements of A and are also elements of B. This operation is called *intersection*, and can be depicted as in Figure 1.2.

Definition 1.17 (Intersection). The *intersection* of two sets A and B, written $A \cap B$, is the set of all things which are elements of both A and B.

$$A \cap B = \{x \mid x \in A \mathbin{\&} x \in B\}$$

Two sets are called *disjoint* if their intersection is empty. This means they have no elements in common.

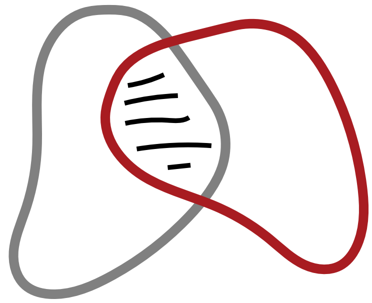


Figure 1.2: The intersection $A \cap B$ of two sets is the set of elements they have in common.

Example 1.18. If two sets have no elements in common, their intersection is empty: $\{a, b, c\} \cap \{0, 1\} = \emptyset$.

If two sets do have elements in common, their intersection is the set of all those: $\{a, b, c\} \cap \{a, b, d\} = \{a, b\}$.

The intersection of a set with one of its subsets is just the smaller set: $\{a, b, c\} \cap \{a, b\} = \{a, b\}$.

The intersection of any set with the empty set is empty: $\{a, b, c\} \cap \emptyset = \emptyset$.

We can also form the union or intersection of more than two sets. An elegant way of dealing with this in general is the following: suppose you collect all the sets you want to form the union (or intersection) of into a single set. Then we can define the union of all our original sets as the set of all objects which belong to at least one element of the set, and the intersection as the set of all objects which belong to every element of the set.

Definition 1.19. If A is a set of sets, then $\bigcup A$ is the set of elements of elements of A:

$$\begin{aligned}\bigcup A &= \{x \mid x \text{ belongs to an element of } A\}, \text{ i.e.,}\\ &= \{x \mid \text{there is a } B \in A \text{ so that } x \in B\}\end{aligned}$$

Definition 1.20. If A is a set of sets, then $\bigcap A$ is the set of objects which all elements of A have in common:

$$\begin{aligned}\bigcap A &= \{x \mid x \text{ belongs to every element of } A\}, \text{ i.e.,}\\ &= \{x \mid \text{for all } B \in A, x \in B\}\end{aligned}$$

Example 1.21. Suppose $A = \{\{a, b\}, \{a, d, e\}, \{a, d\}\}$. Then $\bigcup A = \{a, b, d, e\}$ and $\bigcap A = \{a\}$.

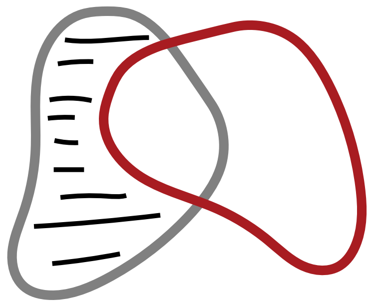


Figure 1.3: The difference $A \setminus B$ of two sets is the set of those elements of A which are not also elements of B.

We could also do the same for a sequence of sets $A_1, A_2, \ldots$

$$\bigcup_i A_i = \{x \mid x \text{ belongs to one of the } A_i\}$$
$$\bigcap_i A_i = \{x \mid x \text{ belongs to every } A_i\}.$$

When we have an *index* of sets, i.e., some set I such that we are considering A_i for each $i \in I$, we may also use these abbreviations:

$$\bigcup_{i \in I} A_i = \bigcup\{A_i \mid i \in I\}$$
$$\bigcap_{i \in I} A_i = \bigcap\{A_i \mid i \in I\}$$

Finally, we may want to think about the set of all elements in A which are not in B. We can depict this as in Figure 1.3.

Definition 1.22 (Difference). The *set difference* $A \setminus B$ is the set of all elements of A which are not also elements of B, i.e.,

$$A \setminus B = \{x \mid x \in A \text{ and } x \notin B\}.$$

1.5 Pairs, Tuples, Cartesian Products

It follows from extensionality that sets have no order to their elements. So if we want to represent order, we use *ordered pairs* $\langle x, y\rangle$. In an unordered pair $\{x, y\}$, the order does not matter: $\{x, y\} = \{y, x\}$. In an ordered pair, it does: if $x \neq y$, then $\langle x, y\rangle \neq \langle y, x\rangle$.

How should we think about ordered pairs in set theory? Crucially, we want to preserve the idea that ordered pairs are identical iff they share the same first element and share the same second element, i.e.:

$$\langle a, b\rangle = \langle c, d\rangle \text{ iff both } a = c \text{ and } b = d.$$

We can define ordered pairs in set theory using the Wiener–Kuratowski definition.

Definition 1.23 (Ordered pair). $\langle a, b\rangle = \{\{a\}, \{a, b\}\}$.

Having fixed a definition of an ordered pair, we can use it to define further sets. For example, sometimes we also want ordered sequences of more than two objects, e.g., *triples* $\langle x, y, z\rangle$, *quadruples* $\langle x, y, z, u\rangle$, and so on. We can think of triples as special ordered pairs, where the first element is itself an ordered pair: $\langle x, y, z\rangle$ is $\langle\langle x, y\rangle, z\rangle$. The same is true for quadruples: $\langle x, y, z, u\rangle$ is $\langle\langle\langle x, y\rangle, z\rangle, u\rangle$, and so on. In general, we talk of *ordered n-tuples* $\langle x_1, \dots, x_n\rangle$.

Certain sets of ordered pairs, or other ordered n-tuples, will be useful.

Definition 1.24 (Cartesian product). Given sets A and B, their *Cartesian product* $A \times B$ is defined by

$$A \times B = \{\langle x, y\rangle \mid x \in A \text{ and } y \in B\}.$$

Example 1.25. If $A = \{0, 1\}$, and $B = \{1, a, b\}$, then their product is

$$A \times B = \{\langle 0, 1\rangle, \langle 0, a\rangle, \langle 0, b\rangle, \langle 1, 1\rangle, \langle 1, a\rangle, \langle 1, b\rangle\}.$$

Example 1.26. If A is a set, the product of A with itself, $A \times A$, is also written A^2. It is the set of *all* pairs $\langle x, y\rangle$ with $x, y \in A$. The set of all triples $\langle x, y, z\rangle$ is A^3, and so on. We can give a recursive definition:

$$\begin{aligned} A^1 &= A \\ A^{k+1} &= A^k \times A \end{aligned}$$

Proposition 1.27. *If A has n elements and B has m elements, then $A \times B$ has $n \cdot m$ elements.*

Proof. For every element x in A, there are m elements of the form $\langle x, y\rangle \in A \times B$. Let $B_x = \{\langle x, y\rangle \mid y \in B\}$. Since whenever $x_1 \neq x_2$, $\langle x_1, y\rangle \neq \langle x_2, y\rangle$, $B_{x_1} \cap B_{x_2} = \emptyset$. But if $A = \{x_1, \dots, x_n\}$, then $A \times B = B_{x_1} \cup \dots \cup B_{x_n}$, and so has $n \cdot m$ elements.

To visualize this, arrange the elements of $A \times B$ in a grid:

$$\begin{array}{rllll} B_{x_1} = & \{\langle x_1, y_1\rangle & \langle x_1, y_2\rangle & \dots & \langle x_1, y_m\rangle\} \\ B_{x_2} = & \{\langle x_2, y_1\rangle & \langle x_2, y_2\rangle & \dots & \langle x_2, y_m\rangle\} \\ \vdots & & \vdots & & \\ B_{x_n} = & \{\langle x_n, y_1\rangle & \langle x_n, y_2\rangle & \dots & \langle x_n, y_m\rangle\} \end{array}$$

Since the x_i are all different, and the y_j are all different, no two of the pairs in this grid are the same, and there are $n \cdot m$ of them. □

Example 1.28. If A is a set, a *word* over A is any sequence of elements of A. A sequence can be thought of as an n-tuple of elements of A. For instance, if $A = \{a, b, c\}$, then the sequence "bac" can be thought of as the triple $\langle b, a, c\rangle$. Words, i.e., sequences of symbols, are of crucial importance in computer science. By convention, we count elements of A as sequences of length 1, and $\emptyset$ as the sequence of length 0. The set of *all* words over A then is

$$A^* = \{\emptyset\} \cup A \cup A^2 \cup A^3 \cup \dots$$

1.6 Russell's Paradox

Extensionality licenses the notation $\{x \mid \phi(x)\}$, for *the* set of x's such that $\phi(x)$. However, all that extensionality *really* licenses is the following thought. *If* there is a set whose members are all and only the ϕ's, *then* there is only one such set. Otherwise put: having fixed some ϕ, the set $\{x \mid \phi(x)\}$ is unique, *if it exists*.

But this conditional is important! Crucially, not every property lends itself to *comprehension*. That is, some properties do *not* define sets. If they all did, then we would run into outright contradictions. The most famous example of this is Russell's Paradox.

Sets may be elements of other sets—for instance, the power set of a set A is made up of sets. And so it makes sense to ask or investigate whether a set is an element of another set. Can a set be a member of itself? Nothing about the idea of a set seems to rule this out. For instance, if *all* sets form a collection of objects, one might think that they can be collected into a single set—the set of all sets. And it, being a set, would be an element of the set of all sets.

Russell's Paradox arises when we consider the property of not having itself as an element, of being *non-self-membered*. What if we suppose that there is a set of all sets that do not have themselves as an element? Does

$$R = \{x \mid x \notin x\}$$

exist? It turns out that we can prove that it does not.

Theorem 1.29 (Russell's Paradox). *There is no set $R = \{x \mid x \notin x\}$.*

Proof. If $R = \{x \mid x \notin x\}$ exists, then $R \in R$ iff $R \notin R$, which is a contradiction. $\square$

Let's run through this proof more slowly. If R exists, it makes sense to ask whether $R \in R$ or not. Suppose that indeed $R \in R$. Now, R was defined as the set of all sets that are not elements of themselves. So, if $R \in R$, then R does not itself have R's defining property. But only sets that have this property are in R, hence, R cannot be an element of R, i.e., $R \notin R$. But R can't both be and not be an element of R, so we have a contradiction.

Since the assumption that $R \in R$ leads to a contradiction, we have $R \notin R$. But this also leads to a contradiction! For if $R \notin R$, then R itself does have R's defining property, and so R would be an element of R just like all the other non-self-membered sets. And again, it can't both not be and be an element of R.

How do we set up a set theory which avoids falling into Russell's Paradox, i.e., which avoids making the *inconsistent* claim that $R = \{x \mid x \notin x\}$ exists? Well, we would need to lay down axioms which give us very precise conditions for stating when sets exist (and when they don't).

The set theory sketched in this chapter doesn't do this. It's *genuinely naïve*. It tells you only that sets obey extensionality and that, if you have some sets, you can form their union, intersection, etc. It is possible to develop set theory more rigorously than this.

Problems

Problem 1.1. Prove that there is at most one empty set, i.e., show that if A and B are sets without elements, then $A = B$.

Problem 1.2. List all subsets of $\{a, b, c, d\}$.

Problem 1.3. Show that if A has n elements, then $\wp(A)$ has 2^n elements.

Problem 1.4. Prove that if $A \subseteq B$, then $A \cup B = B$.

Problem 1.5. Prove rigorously that if $A \subseteq B$, then $A \cap B = A$.

Problem 1.6. Show that if A is a set and $A \in B$, then $A \subseteq \bigcup B$.

Problem 1.7. Prove that if $A \subsetneq B$, then $B \setminus A \neq \emptyset$.

Problem 1.8. Using Definition 1.23, prove that $\langle a, b \rangle = \langle c, d \rangle$ iff both $a = c$ and $b = d$.

Problem 1.9. List all elements of $\{1, 2, 3\}^3$.

Problem 1.10. Show, by induction on k, that for all $k \geq 1$, if A has n elements, then A^k has n^k elements.

Chapter 2

Relations

2.1 Relations as Sets

In section 1.3, we mentioned some important sets: $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$. You will no doubt remember some interesting relations between the elements of some of these sets. For instance, each of these sets has a completely standard *order relation* on it. There is also the relation *is identical with* that every object bears to itself and to no other thing. There are many more interesting relations that we'll encounter, and even more possible relations. Before we review them, though, we will start by pointing out that we can look at relations as a special sort of set.

For this, recall two things from section 1.5. First, recall the notion of a *ordered pair*: given a and b, we can form $\langle a, b\rangle$. Importantly, the order of elements *does* matter here. So if $a \neq b$ then $\langle a, b\rangle \neq \langle b, a\rangle$. (Contrast this with unordered pairs, i.e., 2-element sets, where $\{a, b\} = \{b, a\}$.) Second, recall the notion of a *Cartesian product*: if A and B are sets, then we can form $A \times B$, the set of all pairs $\langle x, y\rangle$ with $x \in A$ and $y \in B$. In particular, $A^2 = A \times A$ is the set of all ordered pairs from A.

Now we will consider a particular relation on a set: the $<$-relation on the set $\mathbb{N}$ of natural numbers. Consider the set of all pairs of numbers $\langle n, m\rangle$ where $n < m$, i.e.,

$$R = \{\langle n, m\rangle \mid n, m \in \mathbb{N} \text{ and } n < m\}.$$

There is a close connection between n being less than m, and the pair $\langle n, m\rangle$ being a member of R, namely:

$$n < m \text{ iff } \langle n, m\rangle \in R.$$

Indeed, without any loss of information, we can consider the set R to *be* the $<$-relation on $\mathbb{N}$.

In the same way we can construct a subset of $\mathbb{N}^2$ for any relation between numbers. Conversely, given any set of pairs of numbers $S \subseteq \mathbb{N}^2$, there is a corresponding relation between numbers, namely, the relationship n bears to m if and only if $\langle n, m \rangle \in S$. This justifies the following definition:

Definition 2.1 (Binary relation). A *binary relation* on a set A is a subset of A^2. If $R \subseteq A^2$ is a binary relation on A and $x, y \in A$, we sometimes write Rxy (or xRy) for $\langle x, y \rangle \in R$.

Example 2.2. The set $\mathbb{N}^2$ of pairs of natural numbers can be listed in a 2-dimensional matrix like this:

$$\begin{array}{ccccc} \langle \mathbf{0}, \mathbf{0} \rangle & \langle 0, 1 \rangle & \langle 0, 2 \rangle & \langle 0, 3 \rangle & \dots \\ \langle 1, 0 \rangle & \langle \mathbf{1}, \mathbf{1} \rangle & \langle 1, 2 \rangle & \langle 1, 3 \rangle & \dots \\ \langle 2, 0 \rangle & \langle 2, 1 \rangle & \langle \mathbf{2}, \mathbf{2} \rangle & \langle 2, 3 \rangle & \dots \\ \langle 3, 0 \rangle & \langle 3, 1 \rangle & \langle 3, 2 \rangle & \langle \mathbf{3}, \mathbf{3} \rangle & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{array}$$

We have put the diagonal, here, in bold, since the subset of $\mathbb{N}^2$ consisting of the pairs lying on the diagonal, i.e.,

$$\{\langle 0, 0 \rangle, \langle 1, 1 \rangle, \langle 2, 2 \rangle, \dots\},$$

is the *identity relation on* $\mathbb{N}$. (Since the identity relation is popular, let's define $\mathrm{Id}_A = \{\langle x, x \rangle \mid x \in A\}$ for any set A.) The subset of all pairs lying above the diagonal, i.e.,

$$L = \{\langle 0, 1 \rangle, \langle 0, 2 \rangle, \dots, \langle 1, 2 \rangle, \langle 1, 3 \rangle, \dots, \langle 2, 3 \rangle, \langle 2, 4 \rangle, \dots\},$$

is the *less than* relation, i.e., Lnm iff $n < m$. The subset of pairs below the diagonal, i.e.,

$$G = \{\langle 1, 0 \rangle, \langle 2, 0 \rangle, \langle 2, 1 \rangle, \langle 3, 0 \rangle, \langle 3, 1 \rangle, \langle 3, 2 \rangle, \dots\},$$

is the *greater than* relation, i.e., Gnm iff $n > m$. The union of L with I, which we might call $K = L \cup I$, is the *less than or equal to* relation: Knm iff $n \leq m$. Similarly, $H = G \cup I$ is the *greater than or equal to relation*. These relations L, G, K, and H are special kinds of relations called *orders*. L and G have the property that no number bears L or G to itself (i.e., for all n, neither Lnn nor Gnn). Relations with this property are called *irreflexive*, and, if they also happen to be orders, they are called *strict orders.*

Although orders and identity are important and natural relations, it should be emphasized that according to our definition *any* subset of A^2 is a relation on A, regardless of how unnatural or contrived it seems. In particular, $\emptyset$ is a relation on any set (the *empty relation*, which no pair of elements bears), and A^2 itself is a relation on A as well (one which every pair bears), called the *universal relation*. But also something like $E = \{\langle n, m \rangle \mid n > 5 \text{ or } m \times n \geq 34\}$ counts as a relation.

2.2 Special Properties of Relations

Some kinds of relations turn out to be so common that they have been given special names. For instance, $\leq$ and $\subseteq$ both relate their respective domains (say, $\mathbb{N}$ in the case of $\leq$ and $\wp(A)$ in the case of $\subseteq$) in similar ways. To get at exactly how these relations are similar, and how they differ, we categorize them according to some special properties that relations can have. It turns out that (combinations of) some of these special properties are especially important: orders and equivalence relations.

Definition 2.3 (Reflexivity). A relation $R \subseteq A^2$ is *reflexive* iff, for every $x \in A$, Rxx.

Definition 2.4 (Transitivity). A relation $R \subseteq A^2$ is *transitive* iff, whenever Rxy and Ryz, then also Rxz.

Definition 2.5 (Symmetry). A relation $R \subseteq A^2$ is *symmetric* iff, whenever Rxy, then also Ryx.

Definition 2.6 (Anti-symmetry). A relation $R \subseteq A^2$ is *anti-symmetric* iff, whenever both Rxy and Ryx, then $x = y$ (or, in other words: if $x \neq y$ then either $\sim Rxy$ or $\sim Ryx$).

In a symmetric relation, Rxy and Ryx always hold together, or neither holds. In an anti-symmetric relation, the only way for Rxy and Ryx to hold together is if $x = y$. Note that this does not *require* that Rxy and Ryx holds when $x = y$, only that it isn't ruled out. So an anti-symmetric relation can be reflexive, but it is not the case that every anti-symmetric relation is reflexive. Also note that being anti-symmetric and merely not being symmetric are different conditions. In fact, a relation can be both symmetric and anti-symmetric at the same time (e.g., the identity relation is).

Definition 2.7 (Connectivity). A relation $R \subseteq A^2$ is *connected* if for all $x, y \in A$, if $x \neq y$, then either Rxy or Ryx.

Definition 2.8 (Irreflexivity). A relation $R \subseteq A^2$ is called *irreflexive* if, for all $x \in A$, not Rxx.

Definition 2.9 (Asymmetry). A relation $R \subseteq A^2$ is called *asymmetric* if for no pair $x, y \in A$ we have both Rxy and Ryx.

Note that if $A \neq \emptyset$, then no irreflexive relation on A is reflexive and every asymmetric relation on A is also anti-symmetric. However, there are $R \subseteq A^2$ that are not reflexive and also not irreflexive, and there are anti-symmetric relations that are not asymmetric.

2.3 Equivalence Relations

The identity relation on a set is reflexive, symmetric, and transitive. Relations R that have all three of these properties are very common.

Definition 2.10 (Equivalence relation). A relation $R \subseteq A^2$ that is reflexive, symmetric, and transitive is called an *equivalence relation*. Elements x and y of A are said to be *R-equivalent* if Rxy.

Equivalence relations give rise to the notion of an *equivalence class*. An equivalence relation "chunks up" the domain into different partitions. Within each partition, all the objects are related to one another; and no objects from different partitions relate to one another. Sometimes, it's helpful just to talk about these partitions *directly*. To that end, we introduce a definition:

Definition 2.11. Let $R \subseteq A^2$ be an equivalence relation. For each $x \in A$, the *equivalence class* of x in A is the set $[x]_R = \{y \in A \mid Rxy\}$. The *quotient* of A under R is $A/_R = \{[x]_R \mid x \in A\}$, i.e., the set of these equivalence classes.

The next result vindicates the definition of an equivalence class, in proving that the equivalence classes are indeed the partitions of A:

Proposition 2.12. *If $R \subseteq A^2$ is an equivalence relation, then Rxy iff $[x]_R = [y]_R$.*

Proof. For the left-to-right direction, suppose Rxy, and let $z \in [x]_R$. By definition, then, Rxz. Since R is an equivalence relation, Ryz. (Spelling this out: as Rxy and R is symmetric we have Ryx, and as Rxz and R is transitive we have Ryz.) So $z \in [y]_R$. Generalising, $[x]_R \subseteq [y]_R$. But exactly similarly, $[y]_R \subseteq [x]_R$. So $[x]_R = [y]_R$, by extensionality.

For the right-to-left direction, suppose $[x]_R = [y]_R$. Since R is reflexive, Ryy, so $y \in [y]_R$. Thus also $y \in [x]_R$ by the assumption that $[x]_R = [y]_R$. So Rxy. □

Example 2.13. A nice example of equivalence relations comes from modular arithmetic. For any a, b, and $n \in \mathbb{N}$, say that $a \equiv_n b$ iff dividing a by n gives the same remainder as dividing b by n. (Somewhat more symbolically: $a \equiv_n b$ iff, for some $k \in \mathbb{Z}$, $a - b = kn$.) Now, $\equiv_n$ is an equivalence relation, for any n. And there are exactly n distinct equivalence classes generated by $\equiv_n$; that is, $\mathbb{N}/_{\equiv_n}$ has n elements. These are: the set of numbers divisible by n without remainder, i.e., $[0]_{\equiv_n}$; the set of numbers divisible by n with remainder 1, i.e., $[1]_{\equiv_n}$; . . . ; and the set of numbers divisible by n with remainder $n - 1$, i.e., $[n - 1]_{\equiv_n}$.

2.4 Orders

Many of our comparisons involve describing some objects as being "less than", "equal to", or "greater than" other objects, in a certain respect. These involve *order* relations. But there are different kinds of order relations. For instance, some require that any two objects be comparable, others don't. Some include identity (like $\leq$) and some exclude it (like $<$). It will help us to have a taxonomy here.

Definition 2.14 (Preorder). A relation which is both reflexive and transitive is called a *preorder.*

Definition 2.15 (Partial order). A preorder which is also anti-symmetric is called a *partial order*.

Definition 2.16 (Linear order). A partial order which is also connected is called a *total order* or *linear order.*

Example 2.17. Every linear order is also a partial order, and every partial order is also a preorder, but the converses don't hold. The universal relation on A is a preorder, since it is reflexive and transitive. But, if A has more than one element, the universal relation is not anti-symmetric, and so not a partial order.

Example 2.18. Consider the *no longer than* relation $\preccurlyeq$ on $\mathbb{B}^*$: $x \preccurlyeq y$ iff $\text{len}(x) \leq \text{len}(y)$. This is a preorder (reflexive and transitive), and even connected, but not a partial order, since it is not anti-symmetric. For instance, $01 \preccurlyeq 10$ and $10 \preccurlyeq 01$, but $01 \neq 10$.

Example 2.19. An important partial order is the relation $\subseteq$ on a set of sets. This is not in general a linear order, since if $a \neq b$ and we consider $\wp(\{a, b\}) = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$, we see that $\{a\} \nsubseteq \{b\}$ and $\{a\} \neq \{b\}$ and $\{b\} \nsubseteq \{a\}$.

Example 2.20. The relation of *divisibility without remainder* gives us a partial order which isn't a linear order. For integers n, m, we write $n \mid m$ to mean n (evenly) divides m, i.e., iff there is some integer k so that $m = kn$. On $\mathbb{N}$, this is a partial order, but not a linear order: for instance, $2 \nmid 3$ and also $3 \nmid 2$. Considered as a relation on $\mathbb{Z}$, divisibility is only a preorder since it is not anti-symmetric: $1 \mid -1$ and $-1 \mid 1$ but $1 \neq -1$.

Definition 2.21 (Strict order). A *strict order* is a relation which is irreflexive, asymmetric, and transitive.

Definition 2.22 (Strict linear order). A strict order which is also connected is called a *strict total order* or *strict linear order.*

Example 2.23. $\leq$ is the linear order corresponding to the strict linear order $<$. $\subseteq$ is the partial order corresponding to the strict order $\subsetneq$.

Any strict order R on A can be turned into a partial order by adding the diagonal Id_A, i.e., adding all the pairs $\langle x, x\rangle$. (This is called the *reflexive closure* of R.) Conversely, starting from a partial order, one can get a strict order by removing Id_A. These next two results make this precise.

Proposition 2.24. *If R is a strict order on A, then $R^+ = R \cup \mathrm{Id}_A$ is a partial order. Moreover, if R is a strict linear order, then R^+ is a linear order.*

Proof. Suppose R is a strict order, i.e., $R \subseteq A^2$ and R is irreflexive, asymmetric, and transitive. Let $R^+ = R \cup \mathrm{Id}_A$. We have to show that R^+ is reflexive, anti-symmetric, and transitive.

R^+ is clearly reflexive, since $\langle x, x\rangle \in \mathrm{Id}_A \subseteq R^+$ for all $x \in A$.

To show R^+ is anti-symmetric, suppose for reductio that R^+xy and R^+yx but $x \neq y$. Since $\langle x, y\rangle \in R \cup \mathrm{Id}_A$, but $\langle x, y\rangle \notin \mathrm{Id}_A$, we must have $\langle x, y\rangle \in R$, i.e., Rxy. Similarly, Ryx. But this contradicts the assumption that R is asymmetric.

To establish transitivity, suppose that R^+xy and R^+yz. If both $\langle x, y\rangle \in R$ and $\langle y, z\rangle \in R$, then $\langle x, z\rangle \in R$ since R is transitive. Otherwise, either $\langle x, y\rangle \in \mathrm{Id}_A$, i.e., $x = y$, or $\langle y, z\rangle \in \mathrm{Id}_A$, i.e., $y = z$. In the first case, we have that R^+yz by assumption, $x = y$, hence R^+xz. Similarly in the second case. In either case, R^+xz, thus, R^+ is also transitive.

Concerning the "moreover" clause, suppose that R is also connected. So for all $x \neq y$, either Rxy or Ryx, i.e., either $\langle x, y\rangle \in R$ or $\langle y, x\rangle \in R$. Since $R \subseteq R^+$, this remains true of R^+, so R^+ is connected as well. □

Proposition 2.25. *If R is a partial order on A, then $R^- = R \setminus \mathrm{Id}_A$ is a strict order. Moreover, if R is a linear order, then R^- is a strict linear order.*

Proof. This is left as an exercise. □

The following simple result establishes that strict linear orders satisfy an extensionality-like property:

Proposition 2.26. *If $<$ is a strict linear order on A, then:*

$$(\forall a, b \in A)((\forall x \in A)(x < a \equiv x < b) \supset a = b).$$

Proof. Suppose $(\forall x \in A)(x < a \equiv x < b)$. If $a < b$, then $a < a$, contradicting the fact that $<$ is irreflexive; so $a \not< b$. Exactly similarly, $b \not< a$. So $a = b$, as $<$ is connected. □

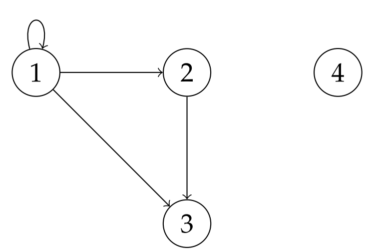
2.5 Graphs

A *graph* is a diagram in which points—called "nodes" or "vertices" (plural of "vertex")—are connected by edges. Graphs are a ubiquitous tool in discrete mathematics and in computer science. They are incredibly useful for representing, and visualizing, relationships and structures, from concrete things like networks of various kinds to abstract structures such as the possible outcomes of decisions. There are many different kinds of graphs in the literature which differ, e.g., according to whether the edges are directed or not, have labels or not, whether there can be edges from a node to the same node, multiple edges between the same nodes, etc. *Directed graphs* have a special connection to relations.

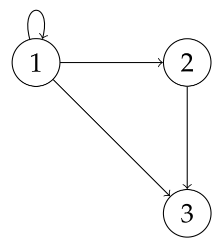
Definition 2.27 (Directed graph). A *directed graph* $G = \langle V, E \rangle$ is a set of *vertices* V and a set of *edges* $E \subseteq V^2$.

According to our definition, a graph just is a set together with a relation on that set. Of course, when talking about graphs, it's only natural to expect that they are graphically represented: we can draw a graph by connecting two vertices v_1 and v_2 by an arrow iff $\langle v_1, v_2 \rangle \in E$. The only difference between a relation by itself and a graph is that a graph specifies the set of vertices, i.e., a graph may have isolated vertices. The important point, however, is that every relation R on a set X can be seen as a directed graph $\langle X, R \rangle$, and conversely, a directed graph $\langle V, E \rangle$ can be seen as a relation $E \subseteq V^2$ with the set V explicitly specified.

Example 2.28. The graph $\langle V, E \rangle$ with $V = \{1, 2, 3, 4\}$ and $E = \{\langle 1, 1 \rangle, \langle 1, 2 \rangle, \langle 1, 3 \rangle, \langle 2, 3 \rangle\}$ looks like this:



This is a different graph than $\langle V', E\rangle$ with $V' = \{1, 2, 3\}$, which looks like this:



2.6 Operations on Relations

It is often useful to modify or combine relations. In Proposition 2.24, we considered the *union* of relations, which is just the union of two relations considered as sets of pairs. Similarly, in Proposition 2.25, we considered the relative difference of relations. Here are some other operations we can perform on relations.

Definition 2.29. Let R, S be relations, and A be any set.

The *inverse* of R is $R^{-1} = \{\langle y, x\rangle \mid \langle x, y\rangle \in R\}$.

The *relative product* of R and S is $(R \mid S) = \{\langle x, z\rangle : \exists y(Rxy \mathbin{\&} Syz)\}$.

The *restriction* of R to A is $R{\restriction}_A = R \cap A^2$.

The *application* of R to A is $R[A] = \{y : (\exists x \in A)Rxy\}$

Example 2.30. Let $S \subseteq \mathbb{Z}^2$ be the successor relation on $\mathbb{Z}$, i.e., $S = \{\langle x, y\rangle \in \mathbb{Z}^2 \mid x + 1 = y\}$, so that Sxy iff $x + 1 = y$.

S^{-1} is the predecessor relation on $\mathbb{Z}$, i.e., $\{\langle x, y\rangle \in \mathbb{Z}^2 \mid x - 1 = y\}$.

$S \mid S$ is $\{\langle x, y\rangle \in \mathbb{Z}^2 \mid x + 2 = y\}$

$S{\restriction}_{\mathbb{N}}$ is the successor relation on $\mathbb{N}$.

$S[\{1, 2, 3\}]$ is $\{2, 3, 4\}$.

Definition 2.31 (Transitive closure). Let $R \subseteq A^2$ be a binary relation.

The *transitive closure* of R is $R^+ = \bigcup_{0<n\in\mathbb{N}} R^n$, where we recursively define $R^1 = R$ and $R^{n+1} = R^n \mid R$.

The *reflexive transitive closure* of R is $R^* = R^+ \cup \mathrm{Id}_A$.

Example 2.32. Take the successor relation $S \subseteq \mathbb{Z}^2$. S^2xy iff $x + 2 = y$, S^3xy iff $x + 3 = y$, etc. So S^+xy iff $x + n = y$ for some $n \geq 1$. In other words, S^+xy iff $x < y$, and S^*xy iff $x \leq y$.

Problems

Problem 2.1. List the elements of the relation $\subseteq$ on the set $\wp(\{a, b, c\})$.

Problem 2.2. Give examples of relations that are (a) reflexive and symmetric but not transitive, (b) reflexive and anti-symmetric, (c) anti-symmetric, transitive, but not reflexive, and (d) reflexive, symmetric, and transitive. Do not use relations on numbers or sets.

Problem 2.3. Show that $\equiv_n$ is an equivalence relation, for any $n \in \mathbb{N}$, and that $\mathbb{N}/_{\equiv_n}$ has exactly n members.

Problem 2.4. Give a proof of Proposition 2.25.

Problem 2.5. Consider the less-than-or-equal-to relation $\leq$ on the set $\{1, 2, 3, 4\}$ as a graph and draw the corresponding diagram.

Problem 2.6. Show that the transitive closure of R is in fact transitive.

Chapter 3

Functions

3.1 Basics

A *function* is a map which sends each element of a given set to a specific element in some (other) given set. For instance, the operation of adding 1 defines a function: each number n is mapped to a unique number $n + 1$.

More generally, functions may take pairs, triples, etc., as inputs and return some kind of output. Many functions are familiar to us from basic arithmetic. For instance, addition and multiplication are functions. They take in two numbers and return a third.

In this mathematical, abstract sense, a function is a *black box*: what matters is only what output is paired with what input, not the method for calculating the output.

Definition 3.1 (Function). A *function* $f \colon A \to B$ is a mapping of each element of A to an element of B.

We call A the *domain* of f and B the *codomain* of f. The elements of A are called inputs or *arguments* of f, and the element of B that is paired with an argument x by f is called the *value of* f for argument x, written $f(x)$.

The *range* $\operatorname{ran}(f)$ of f is the subset of the codomain consisting of the values of f for some argument; $\operatorname{ran}(f) = \{f(x) \mid x \in A\}$.

The diagram in Figure 3.1 may help to think about functions. The ellipse on the left represents the function's *domain*; the ellipse on the right represents the function's *codomain*; and an arrow points from an *argument* in the domain to the corresponding *value* in the codomain.

Example 3.2. Multiplication takes pairs of natural numbers as inputs and maps them to natural numbers as outputs, so goes from $\mathbb{N} \times \mathbb{N}$ (the domain) to $\mathbb{N}$ (the codomain). As it turns out, the range is also $\mathbb{N}$, since every $n \in \mathbb{N}$ is $n \times 1$.

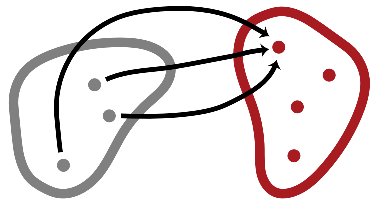


Figure 3.1: A function is a mapping of each element of one set to an element of another. An arrow points from an argument in the domain to the corresponding value in the codomain.

Example 3.3. Multiplication is a function because it pairs each input—each pair of natural numbers—with a single output: $\times\colon \mathbb{N}^2 \to \mathbb{N}$. By contrast, the square root operation applied to the domain $\mathbb{N}$ is not functional, since each positive integer n has two square roots: $\sqrt{n}$ and $-\sqrt{n}$. We can make it functional by only returning the positive square root: $\sqrt{\ }\colon \mathbb{N} \to \mathbb{R}$.

Example 3.4. The relation that pairs each student in a class with their final grade is a function—no student can get two different final grades in the same class. The relation that pairs each student in a class with their parents is not a function: students can have zero, or two, or more parents.

We can define functions by specifying in some precise way what the value of the function is for every possible argument. Different ways of doing this are by giving a formula, describing a method for computing the value, or listing the values for each argument. However functions are defined, we must make sure that for each argument we specify one, and only one, value.

Example 3.5. Let $f\colon \mathbb{N} \to \mathbb{N}$ be defined such that $f(x) = x + 1$. This is a definition that specifies f as a function which takes in natural numbers and outputs natural numbers. It tells us that, given a natural number x, f will output its successor $x + 1$. In this case, the codomain $\mathbb{N}$ is not the range of f, since the natural number 0 is not the successor of any natural number. The range of f is the set of all positive integers, $\mathbb{Z}^+$.

Example 3.6. Let $g\colon \mathbb{N} \to \mathbb{N}$ be defined such that $g(x) = x + 2 - 1$. This tells us that g is a function which takes in natural numbers and outputs natural numbers. Given a natural number n, g will output the predecessor of the successor of the successor of x, i.e., $x + 1$.

We just considered two functions, f and g, with different *definitions*. However, these are the *same function*. After all, for any natural number n, we have that $f(n) = n + 1 = n + 2 - 1 = g(n)$. Otherwise put: our definitions for f

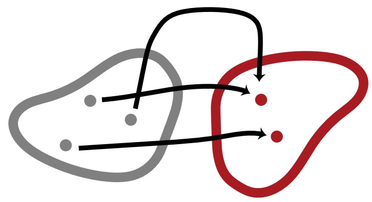


Figure 3.2: A surjective function has every element of the codomain as a value.

and g specify the same mapping by means of different equations. Implicitly, then, we are relying upon a principle of extensionality for functions,

$$\text{if } \forall x\, f(x) = g(x) \text{, then } f = g$$

provided that f and g share the same domain and codomain.

Example 3.7. We can also define functions by cases. For instance, we could define $h\colon \mathbb{N} \to \mathbb{N}$ by

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

Since every natural number is either even or odd, the output of this function will always be a natural number. Just remember that if you define a function by cases, every possible input must fall into exactly one case. In some cases, this will require a proof that the cases are exhaustive and exclusive.

3.2 Kinds of Functions

It will be useful to introduce a kind of taxonomy for some of the kinds of functions which we encounter most frequently.

To start, we might want to consider functions which have the property that every member of the codomain is a value of the function. Such functions are called surjective, and can be pictured as in Figure 3.2.

Definition 3.8 (Surjective function). A function $f\colon A \to B$ is *surjective* iff B is also the range of f, i.e., for every $y \in B$ there is at least one $x \in A$ such that $f(x) = y$, or in symbols:

$$(\forall y \in B)(\exists x \in A) f(x) = y.$$

We call such a function a surjection from A to B.

If you want to show that f is a surjection, then you need to show that every object in f's codomain is the value of $f(x)$ for some input x.

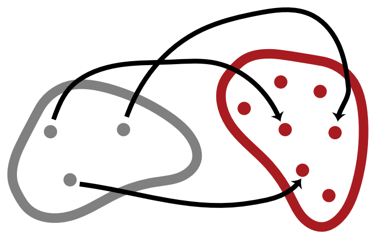


Figure 3.3: An injective function never maps two different arguments to the same value.

Note that any function *induces* a surjection. After all, given a function $f\colon A \to B$, let $f'\colon A \to \operatorname{ran}(f)$ be defined by $f'(x) = f(x)$. Since $\operatorname{ran}(f)$ is *defined* as $\{f(x) \in B \mid x \in A\}$, this function f' is guaranteed to be a surjection

Now, any function maps each possible input to a unique output. But there are also functions which never map different inputs to the same outputs. Such functions are called injective, and can be pictured as in Figure 3.3.

Definition 3.9 (Injective function). A function $f\colon A \to B$ is *injective* iff for each $y \in B$ there is at most one $x \in A$ such that $f(x) = y$. We call such a function an injection from A to B.

If you want to show that f is an injection, you need to show that for any elements x and y of f's domain, if $f(x) = f(y)$, then $x = y$.

Example 3.10. The constant function $f\colon \mathbb{N} \to \mathbb{N}$ given by $f(x) = 1$ is neither injective, nor surjective.

The identity function $f\colon \mathbb{N} \to \mathbb{N}$ given by $f(x) = x$ is both injective and surjective.

The successor function $f\colon \mathbb{N} \to \mathbb{N}$ given by $f(x) = x + 1$ is injective but not surjective.

The function $f\colon \mathbb{N} \to \mathbb{N}$ defined by:

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

is surjective, but not injective.

Often enough, we want to consider functions which are both injective and surjective. We call such functions bijective. They look like the function pictured in Figure 3.4. Bijections are also sometimes called *one-to-one correspondences*, since they uniquely pair elements of the codomain with elements of the domain.

Definition 3.11 (Bijection). A function $f\colon A \to B$ is *bijective* iff it is both surjective and injective. We call such a function a bijection from A to B (or between A and B).

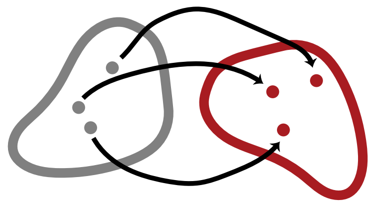


Figure 3.4: A bijective function uniquely pairs the elements of the codomain with those of the domain.

3.3 Functions as Relations

A function which maps elements of A to elements of B obviously defines a relation between A and B, namely the relation which holds between x and y iff $f(x) = y$. In fact, we might even—if we are interested in reducing the building blocks of mathematics for instance—*identify* the function f with this relation, i.e., with a set of pairs. This then raises the question: which relations define functions in this way?

Definition 3.12 (Graph of a function). Let $f\colon A \to B$ be a function. The *graph* of f is the relation $R_f \subseteq A \times B$ defined by

$$R_f = \{\langle x, y\rangle \mid f(x) = y\}.$$

The graph of a function is uniquely determined, by extensionality. Moreover, extensionality (on sets) will immediately vindicate the implicit principle of extensionality for functions, whereby if f and g share a domain and codomain then they are identical if they agree on all values.

Similarly, if a relation is "functional", then it is the graph of a function.

Proposition 3.13. *Let $R \subseteq A \times B$ be such that:*

1. *If Rxy and Rxz then $y = z$; and*
2. *for every $x \in A$ there is some $y \in B$ such that $\langle x, y\rangle \in R$.*

Then R is the graph of the function $f\colon A \to B$ defined by $f(x) = y$ iff Rxy.

Proof. Suppose there is a y such that Rxy. If there were another $z \neq y$ such that Rxz, the condition on R would be violated. Hence, if there is a y such that Rxy, this y is unique, and so f is well-defined. Obviously, $R_f = R$. □

Every function $f\colon A \to B$ has a graph, i.e., a relation on $A \times B$ defined by $f(x) = y$. On the other hand, every relation $R \subseteq A \times B$ with the properties given in Proposition 3.13 is the graph of a function $f\colon A \to B$. Because of this close connection between functions and their graphs, we can think of a function simply as its graph. In other words, functions can be identified with certain relations, i.e., with certain sets of tuples. We can now consider performing similar operations on functions as we performed on relations (see section 2.6). In particular:

Definition 3.14. Let $f\colon A \to B$ be a function with $C \subseteq A$.

The *restriction* of f to C is the function $f{\restriction_C}\colon C \to B$ defined by $(f{\restriction_C})(x) = f(x)$ for all $x \in C$. In other words, $f{\restriction_C} = \{\langle x, y\rangle \in R_f \mid x \in C\}$.

The *application* of f to C is $f[C] = \{f(x) \mid x \in C\}$. We also call this the *image* of C under f.

It follows from these definitions that $\operatorname{ran}(f) = f[\operatorname{dom}(f)]$, for any function f. These notions are exactly as one would expect, given the definitions in section 2.6 and our identification of functions with relations. But two other operations—inverses and relative products—require a little more detail. We will provide that in section 3.4 and section 3.5.

3.4 Inverses of Functions

We think of functions as maps. An obvious question to ask about functions, then, is whether the mapping can be "reversed." For instance, the successor function $f(x) = x + 1$ can be reversed, in the sense that the function $g(y) = y - 1$ "undoes" what f does.

But we must be careful. Although the definition of g defines a function $\mathbb{Z} \to \mathbb{Z}$, it does not define a *function* $\mathbb{N} \to \mathbb{N}$, since $g(0) \notin \mathbb{N}$. So even in simple cases, it is not quite obvious whether a function can be reversed; it may depend on the domain and codomain.

This is made more precise by the notion of an inverse of a function.

Definition 3.15. A function $g\colon B \to A$ is an *inverse* of a function $f\colon A \to B$ if $f(g(y)) = y$ and $g(f(x)) = x$ for all $x \in A$ and $y \in B$.

If f has an inverse g, we often write f^{-1} instead of g.

Now we will determine when functions have inverses. A good candidate for an inverse of $f\colon A \to B$ is $g\colon B \to A$ "defined by"

$$g(y) = \text{"the" } x \text{ such that } f(x) = y.$$

But the scare quotes around "defined by" (and "the") suggest that this is not a definition. At least, it will not always work, with complete generality. For, in order for this definition to specify a function, there has to be one and only one x such that $f(x) = y$—the output of g has to be uniquely specified. Moreover, it has to be specified for every $y \in B$. If there are x_1 and $x_2 \in A$ with $x_1 \neq x_2$ but $f(x_1) = f(x_2)$, then $g(y)$ would not be uniquely specified for $y = f(x_1) = f(x_2)$. And if there is no x at all such that $f(x) = y$, then $g(y)$ is not specified at all. In other words, for g to be defined, f must be both injective and surjective.

Let's go slowly. We'll divide the question into two: Given a function $f\colon A \to B$, when is there a function $g\colon B \to A$ so that $g(f(x)) = x$? Such a g "undoes" what f does, and is called a *left inverse* of f. Secondly, when is there a function $h\colon B \to A$ so that $f(h(y)) = y$? Such an h is called a *right inverse* of f—f "undoes" what h does.

Proposition 3.16. *If $f\colon A \to B$ is injective, then there is a* left inverse $g\colon B \to A$ *of f so that $g(f(x)) = x$ for all $x \in A$.*

Proof. Suppose that $f\colon A \to B$ is injective. Consider a $y \in B$. If $y \in \operatorname{ran}(f)$, there is an $x \in A$ so that $f(x) = y$. Because f is injective, there is only one such $x \in A$. Then we can define: $g(y) = x$, i.e., $g(y)$ is "the" $x \in A$ such that $f(x) = y$. If $y \notin \operatorname{ran}(f)$, we can map it to any $a \in A$. So, we can pick an $a \in A$ and define $g\colon B \to A$ by:

$$g(y) = \begin{cases} x & \text{if } f(x) = y \\ a & \text{if } y \notin \operatorname{ran}(f). \end{cases}$$

It is defined for all $y \in B$, since for each such $y \in \operatorname{ran}(f)$ there is exactly one $x \in A$ such that $f(x) = y$. By definition, if $y = f(x)$, then $g(y) = x$, i.e., $g(f(x)) = x$. □

Proposition 3.17. *If $f\colon A \to B$ is surjective, then there is a* right inverse $h\colon B \to A$ *of f so that $f(h(y)) = y$ for all $y \in B$.*

Proof. Suppose that $f\colon A \to B$ is surjective. Consider a $y \in B$. Since f is surjective, there is an $x_y \in A$ with $f(x_y) = y$. Then we can define: $h(y) = x_y$, i.e., for each $y \in B$ we choose some $x \in A$ so that $f(x) = y$; since f is surjective there is always at least one to choose from.[1] By definition, if $x = h(y)$, then $f(x) = y$, i.e., for any $y \in B$, $f(h(y)) = y$. □

By combining the ideas in the previous proof, we now get that every bijection has an inverse, i.e., there is a single function which is both a left and right inverse of f.

Proposition 3.18. *If $f\colon A \to B$ is bijective, there is a function $f^{-1}\colon B \to A$ so that for all $x \in A$, $f^{-1}(f(x)) = x$ and for all $y \in B$, $f(f^{-1}(y)) = y$.*

[1] Since f is surjective, for every $y \in B$ the set $\{x \mid f(x) = y\}$ is nonempty. Our definition of h requires that we choose a single x from each of these sets. That this is always possible is actually not obvious—the possibility of making these choices is simply assumed as an axiom. In other words, this proposition assumes the so-called Axiom of Choice, an issue we will gloss over. However, in many specific cases, e.g., when $A = \mathbb{N}$ or is finite, or when f is bijective, the Axiom of Choice is not required. (In the particular case when f is bijective, for each $y \in B$ the set $\{x \mid f(x) = y\}$ has exactly one element, so that there is no choice to make.)

Proof. Exercise. □

There is a slightly more general way to extract inverses. We saw in section 3.2 that every function f induces a surjection $f' \colon A \to \operatorname{ran}(f)$ by letting $f'(x) = f(x)$ for all $x \in A$. Clearly, if f is injective, then f' is bijective, so that it has a unique inverse by Proposition 3.18. By a very minor abuse of notation, we sometimes call the inverse of f' simply "the inverse of f."

Proposition 3.19. *Show that if $f \colon A \to B$ has a left inverse g and a right inverse h, then $h = g$.*

Proof. Exercise. □

Proposition 3.20. *Every function f has at most one inverse.*

Proof. Suppose g and h are both inverses of f. Then in particular g is a left inverse of f and h is a right inverse. By Proposition 3.19, $g = h$. □

3.5 Composition of Functions

We saw in section 3.4 that the inverse f^{-1} of a bijection f is itself a function. Another operation on functions is composition: we can define a new function by composing two functions, f and g, i.e., by first applying f and then g. Of course, this is only possible if the ranges and domains match, i.e., the range of f must be a subset of the domain of g. This operation on functions is the analogue of the operation of relative product on relations from section 2.6.

A diagram might help to explain the idea of composition. In Figure 3.5, we depict two functions $f \colon A \to B$ and $g \colon B \to C$ and their composition $(g \circ f)$. The function $(g \circ f) \colon A \to C$ pairs each element of A with an element of C. We specify which element of C an element of A is paired with as follows: given an input $x \in A$, first apply the function f to x, which will output some $f(x) = y \in B$, then apply the function g to y, which will output some $g(f(x)) = g(y) = z \in C$.

Definition 3.21 (Composition). Let $f \colon A \to B$ and $g \colon B \to C$ be functions. The *composition* of f with g is $g \circ f \colon A \to C$, where $(g \circ f)(x) = g(f(x))$.

Example 3.22. Consider the functions $f(x) = x + 1$, and $g(x) = 2x$. Since $(g \circ f)(x) = g(f(x))$, for each input x you must first take its successor, then multiply the result by two. So their composition is given by $(g \circ f)(x) = 2(x + 1)$.

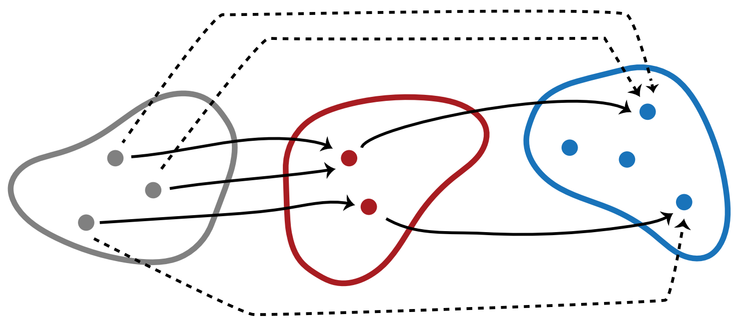


Figure 3.5: The composition $g \circ f$ of two functions f and g.

3.6 Partial Functions

It is sometimes useful to relax the definition of function so that it is not required that the output of the function is defined for all possible inputs. Such mappings are called *partial functions*.

Definition 3.23. A *partial function* $f : A \nrightarrow B$ is a mapping which assigns to every element of A at most one element of B. If f assigns an element of B to $x \in A$, we say $f(x)$ is *defined*, and otherwise *undefined*. If $f(x)$ is defined, we write $f(x) \downarrow$, otherwise $f(x) \uparrow$. The *domain* of a partial function f is the subset of A where it is defined, i.e., $\mathrm{dom}(f) = \{x \in A \mid f(x) \downarrow\}$.

Example 3.24. Every function $f : A \to B$ is also a partial function. Partial functions that are defined everywhere on A—i.e., what we so far have simply called a function—are also called *total* functions.

Example 3.25. The partial function $f : \mathbb{R} \nrightarrow \mathbb{R}$ given by $f(x) = 1/x$ is undefined for $x = 0$, and defined everywhere else.

Definition 3.26 (Graph of a partial function). Let $f : A \nrightarrow B$ be a partial function. The *graph* of f is the relation $R_f \subseteq A \times B$ defined by

$$R_f = \{\langle x, y \rangle \mid f(x) = y\}.$$

Proposition 3.27. *Suppose $R \subseteq A \times B$ has the property that whenever Rxy and Rxy' then $y = y'$. Then R is the graph of the partial function $f : X \nrightarrow Y$ defined by: if there is a y such that Rxy, then $f(x) = y$, otherwise $f(x) \uparrow$. If R is also* serial, *i.e., for each $x \in X$ there is a $y \in Y$ such that Rxy, then f is total.*

Proof. Suppose there is a y such that Rxy. If there were another $y' \neq y$ such that Rxy', the condition on R would be violated. Hence, if there is a y such that Rxy, that y is unique, and so f is well-defined. Obviously, $R_f = R$ and f is total if R is serial. □

Problems

Problem 3.1. Show that if $f\colon A \to B$ has a left inverse g, then f is injective.

Problem 3.2. Show that if $f\colon A \to B$ has a right inverse h, then f is surjective.

Problem 3.3. Prove Proposition 3.18. You have to define f^{-1}, show that it is a function, and show that it is an inverse of f, i.e., $f^{-1}(f(x)) = x$ and $f(f^{-1}(y)) = y$ for all $x \in A$ and $y \in B$.

Problem 3.4. Prove Proposition 3.19.

Problem 3.5. Show that if $f\colon A \to B$ and $g\colon B \to C$ are both injective, then $g \circ f\colon A \to C$ is injective.

Problem 3.6. Show that if $f\colon A \to B$ and $g\colon B \to C$ are both surjective, then $g \circ f\colon A \to C$ is surjective.

Problem 3.7. Suppose $f\colon A \to B$ and $g\colon B \to C$. Show that the graph of $g \circ f$ is $R_f \mid R_g$.

Problem 3.8. Given $f\colon A \nrightarrow B$, define the partial function $g\colon B \nrightarrow A$ by: for any $y \in B$, if there is a unique $x \in A$ such that $f(x) = y$, then $g(y) = x$; otherwise $g(y) \uparrow$. Show that if f is injective, then $g(f(x)) = x$ for all $x \in \operatorname{dom}(f)$, and $f(g(y)) = y$ for all $y \in \operatorname{ran}(f)$.

Chapter 4

The Size of Sets

4.1 Introduction

When Georg Cantor developed set theory in the 1870s, one of his aims was to make palatable the idea of an infinite collection—an actual infinity, as the medievals would say. A key part of this was his treatment of the *size* of different sets. If a, b and c are all distinct, then the set $\{a, b, c\}$ is intuitively *larger* than $\{a, b\}$. But what about infinite sets? Are they all as large as each other? It turns out that they are not.

The first important idea here is that of an enumeration. We can list every finite set by listing all its elements. For some infinite sets, we can also list all their elements if we allow the list itself to be infinite. Such sets are called countable. Cantor's surprising result, which we will fully understand by the end of this chapter, was that some infinite sets are not countable.

4.2 Enumerations and Countable Sets

We've already given examples of sets by listing their elements. Let's discuss in more general terms how and when we can list the elements of a set, even if that set is infinite.

Definition 4.1 (Enumeration, informally). Informally, an *enumeration* of a set A is a list (possibly infinite) of elements of A such that every element of A appears on the list at some finite position. If A has an enumeration, then A is said to be *countable*.

A couple of points about enumerations:

1. We count as enumerations only lists which have a beginning and in which every element other than the first has a single element immediately preceding it. In other words, there are only finitely many elements between the first element of the list and any other element. In particular,

this means that every element of an enumeration has a finite position: the first element has position 1, the second position 2, etc.

2. We can have different enumerations of the same set A which differ by the order in which the elements appear: 4, 1, 25, 16, 9 enumerates the (set of the) first five square numbers just as well as 1, 4, 9, 16, 25 does.

3. Redundant enumerations are still enumerations: 1, 1, 2, 2, 3, 3, ... enumerates the same set as 1, 2, 3, ... does.

4. Order and redundancy *do* matter when we specify an enumeration: we can enumerate the positive integers beginning with 1, 2, 3, 1, ..., but the pattern is easier to see when enumerated in the standard way as 1, 2, 3, 4, ...

5. Enumerations must have a beginning: ..., 3, 2, 1 is not an enumeration of the positive integers because it has no first element. To see how this follows from the informal definition, ask yourself, "at what position in the list does the number 76 appear?"

6. The following is not an enumeration of the positive integers: 1, 3, 5, ..., 2, 4, 6, ... The problem is that the even numbers occur at places $\infty + 1$, $\infty + 2$, $\infty + 3$, rather than at finite positions.

7. The empty set is enumerable: it is enumerated by the empty list!

Proposition 4.2. *If A has an enumeration, it has an enumeration without repetitions.*

Proof. Suppose A has an enumeration $x_1, x_2, \ldots$ in which each x_i is an element of A. We can remove repetitions from an enumeration by removing repeated elements. For instance, we can turn the enumeration into a new one in which we list x_i if it is an element of A that is not among $x_1, \ldots, x_{i-1}$ or remove x_i from the list if it already appears among $x_1, \ldots, x_{i-1}$. □

The last argument shows that in order to get a good handle on enumerations and countable sets and to prove things about them, we need a more precise definition. The following provides it.

Definition 4.3 (Enumeration, formally). An *enumeration* of a set $A \neq \emptyset$ is any surjective function $f\colon \mathbb{Z}^+ \to A$.

Let's convince ourselves that the formal definition and the informal definition using a possibly infinite list are equivalent. First, any surjective function from $\mathbb{Z}^+$ to a set A enumerates A. Such a function determines an enumeration as defined informally above: the list $f(1), f(2), f(3), \ldots$. Since f is surjective, every element of A is guaranteed to be the value of $f(n)$ for some $n \in \mathbb{Z}^+$.

Hence, every element of A appears at some finite position in the list. Since the function may not be injective, the list may be redundant, but that is acceptable (as noted above).

On the other hand, given a list that enumerates all elements of A, we can define a surjective function $f\colon \mathbb{Z}^+ \to A$ by letting $f(n)$ be the nth element of the list, or the final element of the list if there is no nth element. The only case where this does not produce a surjective function is when A is empty, and hence the list is empty. So, every non-empty list determines a surjective function $f\colon \mathbb{Z}^+ \to A$.

Definition 4.4. A set A is countable iff it is empty or has an enumeration.

Example 4.5. A function enumerating the positive integers ($\mathbb{Z}^+$) is simply the identity function given by $f(n) = n$. A function enumerating the natural numbers $\mathbb{N}$ is the function $g(n) = n - 1$.

Example 4.6. The functions $f\colon \mathbb{Z}^+ \to \mathbb{Z}^+$ and $g\colon \mathbb{Z}^+ \to \mathbb{Z}^+$ given by

$$\begin{aligned} f(n) &= 2n \text{ and} \\ g(n) &= 2n - 1 \end{aligned}$$

enumerate the even positive integers and the odd positive integers, respectively. However, neither function is an enumeration of $\mathbb{Z}^+$, since neither is surjective.

Example 4.7. The function $f(n) = (-1)^n \lceil \frac{(n-1)}{2} \rceil$ (where $\lceil x \rceil$ denotes the *ceiling* function, which rounds x up to the nearest integer) enumerates the set of integers $\mathbb{Z}$. Notice how f generates the values of $\mathbb{Z}$ by "hopping" back and forth between positive and negative integers:

$$\begin{array}{cccccccc} f(1) & f(2) & f(3) & f(4) & f(5) & f(6) & f(7) & \dots \\ \\ -\lceil \frac{0}{2} \rceil & \lceil \frac{1}{2} \rceil & -\lceil \frac{2}{2} \rceil & \lceil \frac{3}{2} \rceil & -\lceil \frac{4}{2} \rceil & \lceil \frac{5}{2} \rceil & -\lceil \frac{6}{2} \rceil & \dots \\ \\ 0 & 1 & -1 & 2 & -2 & 3 & \dots & \end{array}$$

You can also think of f as defined by cases as follows:

$$f(n) = \begin{cases} 0 & \text{if } n = 1 \\ n/2 & \text{if } n \text{ is even} \\ -(n-1)/2 & \text{if } n \text{ is odd and } > 1 \end{cases}$$

Although it is perhaps more natural when listing the elements of a set to start counting from the 1st element, mathematicians like to use the natural numbers $\mathbb{N}$ for counting things. They talk about the 0th, 1st, 2nd, and so on, elements of a list. Correspondingly, we can define an enumeration as a surjective function from $\mathbb{N}$ to A. Of course, the two definitions are equivalent.

Proposition 4.8. *There is a surjection* $f: \mathbb{Z}^+ \to A$ *iff there is a surjection* $g: \mathbb{N} \to A$.

Proof. Given a surjection $f: \mathbb{Z}^+ \to A$, we can define $g(n) = f(n+1)$ for all $n \in \mathbb{N}$. It is easy to see that $g: \mathbb{N} \to A$ is surjective. Conversely, given a surjection $g: \mathbb{N} \to A$, define $f(n) = g(n-1)$. □

This gives us the following result:

Corollary 4.9. *A set* A *is countable iff it is empty or there is a surjective function* $f: \mathbb{N} \to A$.

We discussed above than an list of elements of a set A can be turned into a list without repetitions. This is also true for enumerations, but a bit harder to formulate and prove rigorously. Any function $f: \mathbb{Z}^+ \to A$ must be defined for all $n \in \mathbb{Z}^+$. If there are only finitely many elements in A then we clearly cannot have a function defined on the infinitely many elements of $\mathbb{Z}^+$ that takes as values all the elements of A but never takes the same value twice. In that case, i.e., in the case where the list without repetitions is finite, we must choose a different domain for f, one with only finitely many elements. Not having repetitions means that f must be injective. Since it is also surjective, we are looking for a bijection between some finite set $\{1, \dots, n\}$ or $\mathbb{Z}^+$ and A.

Proposition 4.10. *If* $f: \mathbb{Z}^+ \to A$ *is surjective (i.e., an enumeration of* A*), there is a bijection* $g: Z \to A$ *where* Z *is either* $\mathbb{Z}^+$ *or* $\{1, \dots, n\}$ *for some* $n \in \mathbb{Z}^+$.

Proof. We define the function g recursively: Let $g(1) = f(1)$. If $g(i)$ has already been defined, let $g(i+1)$ be the first value of $f(1), f(2), \dots$ not already among $g(1), \dots, g(i)$, if there is one. If A has just n elements, then $g(1), \dots, g(n)$ are all defined, and so we have defined a function $g: \{1, \dots, n\} \to A$. If A has infinitely many elements, then for any i there must be an element of A in the enumeration $f(1), f(2), \dots$, which is not already among $g(1), \dots, g(i)$. In this case we have defined a function $g: \mathbb{Z}^+ \to A$.

The function g is surjective, since any element of A is among $f(1), f(2), \dots$ (since f is surjective) and so will eventually be a value of $g(i)$ for some i. It is also injective, since if there were $j < i$ such that $g(j) = g(i)$, then $g(i)$ would already be among $g(1), \dots, g(i-1)$, contrary to how we defined g. □

Corollary 4.11. *A set* A *is countable iff it is empty or there is a bijection* $f: N \to A$ *where either* $N = \mathbb{N}$ *or* $N = \{0, \dots, n\}$ *for some* $n \in \mathbb{N}$.

Proof. A is countable iff A is empty or there is a surjective $f: \mathbb{Z}^+ \to A$. By Proposition 4.10, the latter holds iff there is a bijective function $f: Z \to A$ where $Z = \mathbb{Z}^+$ or $Z = \{1, \dots, n\}$ for some $n \in \mathbb{Z}^+$. By the same argument as in the proof of Proposition 4.8, that in turn is the case iff there is a bijection $g: N \to A$ where either $N = \mathbb{N}$ or $N = \{0, \dots, n-1\}$. □

4.3 Cantor's Zig-Zag Method

We've already considered some "easy" enumerations. Now we will consider something a bit harder. Consider the set of pairs of natural numbers, which we defined in section 1.5 thus:

$$\mathbb{N} \times \mathbb{N} = \{\langle n, m \rangle \mid n, m \in \mathbb{N}\}$$

We can organize these ordered pairs into an *array*, like so:

	0	**1**	**2**	**3**	...
0	$\langle 0,0 \rangle$	$\langle 0,1 \rangle$	$\langle 0,2 \rangle$	$\langle 0,3 \rangle$	...
1	$\langle 1,0 \rangle$	$\langle 1,1 \rangle$	$\langle 1,2 \rangle$	$\langle 1,3 \rangle$	...
2	$\langle 2,0 \rangle$	$\langle 2,1 \rangle$	$\langle 2,2 \rangle$	$\langle 2,3 \rangle$	...
3	$\langle 3,0 \rangle$	$\langle 3,1 \rangle$	$\langle 3,2 \rangle$	$\langle 3,3 \rangle$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Clearly, every ordered pair in $\mathbb{N} \times \mathbb{N}$ will appear exactly once in the array. In particular, $\langle n, m \rangle$ will appear in the nth row and mth column. But how do we organize the elements of such an array into a "one-dimensional" list? The pattern in the array below demonstrates one way to do this (although of course there are many other options):

	0	**1**	**2**	**3**	**4**	...
0	0	1	3	6	10	...
1	2	4	7	11	...	...
2	5	8	12	...	...	...
3	9	13	...	...	...	...
4	14	...	...	...	...	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...	$\ddots$

This pattern is called *Cantor's zig-zag method*. It enumerates $\mathbb{N} \times \mathbb{N}$ as follows:

$$\langle 0,0 \rangle, \langle 0,1 \rangle, \langle 1,0 \rangle, \langle 0,2 \rangle, \langle 1,1 \rangle, \langle 2,0 \rangle, \langle 0,3 \rangle, \langle 1,2 \rangle, \langle 2,1 \rangle, \langle 3,0 \rangle, \ldots$$

And this establishes the following:

Proposition 4.12. $\mathbb{N} \times \mathbb{N}$ *is countable.*

Proof. Let $f\colon \mathbb{N} \to \mathbb{N} \times \mathbb{N}$ take each $k \in \mathbb{N}$ to the tuple $\langle n, m \rangle \in \mathbb{N} \times \mathbb{N}$ such that k is the value of the nth row and mth column in Cantor's zig-zag array. □

This technique also generalises rather nicely. For example, we can use it to enumerate the set of ordered triples of natural numbers, i.e.:

$$\mathbb{N} \times \mathbb{N} \times \mathbb{N} = \{\langle n, m, k \rangle \mid n, m, k \in \mathbb{N}\}$$

We think of $\mathbb{N} \times \mathbb{N} \times \mathbb{N}$ as the Cartesian product of $\mathbb{N} \times \mathbb{N}$ with $\mathbb{N}$, that is,

$$\mathbb{N}^3 = (\mathbb{N} \times \mathbb{N}) \times \mathbb{N} = \{\langle\langle n, m\rangle, k\rangle \mid n, m, k \in \mathbb{N}\}$$

and thus we can enumerate $\mathbb{N}^3$ with an array by labelling one axis with the enumeration of $\mathbb{N}$, and the other axis with the enumeration of $\mathbb{N}^2$:

	0	**1**	**2**	**3**	$\ldots$
$\langle \mathbf{0}, \mathbf{0}\rangle$	$\langle 0,0,0\rangle$	$\langle 0,0,1\rangle$	$\langle 0,0,2\rangle$	$\langle 0,0,3\rangle$	$\ldots$
$\langle \mathbf{0}, \mathbf{1}\rangle$	$\langle 0,1,0\rangle$	$\langle 0,1,1\rangle$	$\langle 0,1,2\rangle$	$\langle 0,1,3\rangle$	$\ldots$
$\langle \mathbf{1}, \mathbf{0}\rangle$	$\langle 1,0,0\rangle$	$\langle 1,0,1\rangle$	$\langle 1,0,2\rangle$	$\langle 1,0,3\rangle$	$\ldots$
$\langle \mathbf{0}, \mathbf{2}\rangle$	$\langle 0,2,0\rangle$	$\langle 0,2,1\rangle$	$\langle 0,2,2\rangle$	$\langle 0,2,3\rangle$	$\ldots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Thus, by using a method like Cantor's zig-zag method, we may similarly obtain an enumeration of $\mathbb{N}^3$. And we can keep going, obtaining enumerations of $\mathbb{N}^n$ for any natural number n. So, we have:

Proposition 4.13. *$\mathbb{N}^n$ is countable, for every $n \in \mathbb{N}$.*

4.4 Pairing Functions and Codes

Cantor's zig-zag method makes the enumerability of $\mathbb{N}^n$ visually evident. But let us focus on our array depicting $\mathbb{N}^2$. Following the zig-zag line in the array and counting the places, we can check that $\langle 1, 2\rangle$ is associated with the number 7. However, it would be nice if we could compute this more directly. That is, it would be nice to have to hand the *inverse* of the zig-zag enumeration, $g \colon \mathbb{N}^2 \to \mathbb{N}$, such that

$$g(\langle 0,0\rangle) = 0,\ g(\langle 0,1\rangle) = 1,\ g(\langle 1,0\rangle) = 2,\ \ldots, g(\langle 1,2\rangle) = 7,\ \ldots$$

This would enable us to calculate exactly where $\langle n, m\rangle$ will occur in our enumeration.

In fact, we can define g directly by making two observations. First: if the nth row and mth column contains value v, then the $(n+1)$st row and $(m-1)$st column contains value $v+1$. Second: the first row of our enumeration consists of the triangular numbers, starting with 0, 1, 3, 6, etc. The kth triangular number is the sum of the natural numbers $< k$, which can be computed as $k(k+1)/2$. Putting these two observations together, consider this function:

$$g(n, m) = \frac{(n+m+1)(n+m)}{2} + n$$

We often just write $g(n, m)$ rather that $g(\langle n, m\rangle)$, since it is easier on the eyes. This tells you first to determine the $(n+m)^{\text{th}}$ triangle number, and then add

n to it. And it populates the array in exactly the way we would like. So in particular, the pair $\langle 1, 2 \rangle$ is sent to $\frac{4 \times 3}{2} + 1 = 7$.

This function g is the *inverse* of an enumeration of a set of pairs. Such functions are called *pairing functions*.

Definition 4.14 (Pairing function). A function $f \colon A \times B \to \mathbb{N}$ is an arithmetical *pairing function* if f is injective. We also say that f *encodes* $A \times B$, and that $f(x, y)$ is the *code* for $\langle x, y \rangle$.

We can use pairing functions to encode, e.g., pairs of natural numbers; or, in other words, we can represent each *pair* of elements using a *single* number. Using the inverse of the pairing function, we can *decode* the number, i.e., find out which pair it represents.

4.5 An Alternative Pairing Function

There are other enumerations of $\mathbb{N}^2$ that make it easier to figure out what their inverses are. Here is one. Instead of visualizing the enumeration in an array, start with the list of positive integers associated with (initially) empty spaces. Imagine filling these spaces successively with pairs $\langle n, m \rangle$ as follows. Starting with the pairs that have 0 in the first place (i.e., pairs $\langle 0, m \rangle$), put the first (i.e., $\langle 0, 0 \rangle$) in the first empty place, then skip an empty space, put the second (i.e., $\langle 0, 2 \rangle$) in the next empty place, skip one again, and so forth. The (incomplete) beginning of our enumeration now looks like this

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	…
$\langle 0, 1 \rangle$		$\langle 0, 2 \rangle$		$\langle 0, 3 \rangle$		$\langle 0, 4 \rangle$		$\langle 0, 5 \rangle$		…

Repeat this with pairs $\langle 1, m \rangle$ for the place that still remain empty, again skipping every other empty place:

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	…
$\langle 0, 0 \rangle$	$\langle 1, 0 \rangle$	$\langle 0, 1 \rangle$		$\langle 0, 2 \rangle$	$\langle 1, 1 \rangle$	$\langle 0, 3 \rangle$		$\langle 0, 4 \rangle$	$\langle 1, 2 \rangle$	…

Enter pairs $\langle 2, m \rangle$, $\langle 2, m \rangle$, etc., in the same way. Our completed enumeration thus starts like this:

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	…
$\langle 0, 0 \rangle$	$\langle 1, 0 \rangle$	$\langle 0, 1 \rangle$	$\langle 2, 0 \rangle$	$\langle 0, 2 \rangle$	$\langle 1, 1 \rangle$	$\langle 0, 3 \rangle$	$\langle 3, 0 \rangle$	$\langle 0, 4 \rangle$	$\langle 1, 2 \rangle$	…

If we number the cells in the array above according to this enumeration, we will not find a neat zig-zag line, but this arrangement:

	0	**1**	**2**	**3**	**4**	**5**	...
0	1	3	5	7	9	11	...
1	2	6	10	14	18	...	...
2	4	12	20	28	...	...	...
3	8	24	40	...	...	...	...
4	16	48	...	...	...	...	...
5	32	...	...	...	...	...	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱

We can see that the pairs in row 0 are in the odd numbered places of our enumeration, i.e., pair $\langle 0, m\rangle$ is in place $2m+1$; pairs in the second row, $\langle 1, m\rangle$, are in places whose number is the double of an odd number, specifically, $2 \cdot (2m+1)$; pairs in the third row, $\langle 2, m\rangle$, are in places whose number is four times an odd number, $4 \cdot (2m+1)$; and so on. The factors of $(2m+1)$ for each row, 1, 2, 4, 8, $\ldots$, are exactly the powers of 2: $1 = 2^0$, $2 = 2^1$, $4 = 2^2$, $8 = 2^3$, $\ldots$ In fact, the relevant exponent is always the first member of the pair in question. Thus, for pair $\langle n, m\rangle$ the factor is 2^n. This gives us the general formula: $2^n \cdot (2m+1)$. However, this is a mapping of pairs to *positive* integers, i.e., $\langle 0, 0\rangle$ has position 1. If we want to begin at position 0 we must subtract 1 from the result. This gives us:

Example 4.15. The function $h\colon \mathbb{N}^2 \to \mathbb{N}$ given by

$$h(n, m) = 2^n(2m+1) - 1$$

is a pairing function for the set of pairs of natural numbers $\mathbb{N}^2$.

Accordingly, in our second enumeration of $\mathbb{N}^2$, the pair $\langle 0, 0\rangle$ has code $h(0, 0) = 2^0(2 \cdot 0 + 1) - 1 = 0$; $\langle 1, 2\rangle$ has code $2^1 \cdot (2 \cdot 2 + 1) - 1 = 2 \cdot 5 - 1 = 9$; $\langle 2, 6\rangle$ has code $2^2 \cdot (2 \cdot 6 + 1) - 1 = 51$.

Sometimes it is enough to encode pairs of natural numbers $\mathbb{N}^2$ without requiring that the encoding is surjective. Such encodings have inverses that are only partial functions.

Example 4.16. The function $j\colon \mathbb{N}^2 \to \mathbb{N}^+$ given by

$$j(n, m) = 2^n 3^m$$

is an injective function $\mathbb{N}^2 \to \mathbb{N}$.

4.6 Uncountable Sets

Some sets, such as the set $\mathbb{Z}^+$ of positive integers, are infinite. So far we've seen examples of infinite sets which were all countable. However, there are also infinite sets which do not have this property. Such sets are called *uncountable*.

First of all, it is perhaps already surprising that there are uncountable sets. For any countable set A there is a surjective function $f\colon \mathbb{Z}^+ \to A$. If a set is uncountable there is no such function. That is, no function mapping the infinitely many elements of $\mathbb{Z}^+$ to A can exhaust all of A. So there are "more" elements of A than the infinitely many positive integers.

How would one prove that a set is uncountable? You have to show that no such surjective function can exist. Equivalently, you have to show that the elements of A cannot be enumerated in a one way infinite list. The best way to do this is to show that every list of elements of A must leave at least one element out; or that no function $f\colon \mathbb{Z}^+ \to A$ can be surjective. We can do this using Cantor's *diagonal method*. Given a list of elements of A, say, x_1, x_2, $\dots$, we construct another element of A which, by its construction, cannot possibly be on that list.

Our first example is the set $\mathbb{B}^\omega$ of all infinite, non-gappy sequences of 0's and 1's.

Theorem 4.17. $\mathbb{B}^\omega$ *is uncountable.*

Proof. Suppose, by way of contradiction, that $\mathbb{B}^\omega$ is countable, i.e., suppose that there is a list s_1, s_2, s_3, s_4, $\dots$ of all elements of $\mathbb{B}^\omega$. Each of these s_i is itself an infinite sequence of 0's and 1's. Let's call the j-th element of the i-th sequence in this list $s_i(j)$. Then the i-th sequence s_i is

$$s_i(1), s_i(2), s_i(3), \dots$$

We may arrange this list, and the elements of each sequence s_i in it, in an array:

	1	2	3	4	$\dots$
1	$\mathbf{s_1(1)}$	$s_1(2)$	$s_1(3)$	$s_1(4)$	$\dots$
2	$s_2(1)$	$\mathbf{s_2(2)}$	$s_2(3)$	$s_2(4)$	$\dots$
3	$s_3(1)$	$s_3(2)$	$\mathbf{s_3(3)}$	$s_3(4)$	$\dots$
4	$s_4(1)$	$s_4(2)$	$s_4(3)$	$\mathbf{s_4(4)}$	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

The labels down the side give the number of the sequence in the list $s_1, s_2, \dots$; the numbers across the top label the elements of the individual sequences. For instance, $s_1(1)$ is a name for whatever number, a 0 or a 1, is the first element in the sequence s_1, and so on.

Now we construct an infinite sequence, $\bar{s}$, of 0's and 1's which cannot possibly be on this list. The definition of $\bar{s}$ will depend on the list $s_1, s_2, \dots$. Any infinite list of infinite sequences of 0's and 1's gives rise to an infinite sequence $\bar{s}$ which is guaranteed to not appear on the list.

To define $\bar{s}$, we specify what all its elements are, i.e., we specify $\bar{s}(n)$ for all $n \in \mathbb{Z}^+$. We do this by reading down the diagonal of the array above (hence the name "diagonal method") and then changing every 1 to a 0 and every 0 to a 1. More abstractly, we define $\bar{s}(n)$ to be 0 or 1 according to whether the n-th element of the diagonal, $s_n(n)$, is 1 or 0.

$$\bar{s}(n) = \begin{cases} 1 & \text{if } s_n(n) = 0 \\ 0 & \text{if } s_n(n) = 1. \end{cases}$$

If you like formulas better than definitions by cases, you could also define $\bar{s}(n) = 1 - s_n(n)$.

Clearly $\bar{s}$ is an infinite sequence of 0's and 1's, since it is just the mirror sequence to the sequence of 0's and 1's that appear on the diagonal of our array. So $\bar{s}$ is an element of $\mathbb{B}^\omega$. But it cannot be on the list $s_1, s_2, \dots$ Why not?

It can't be the first sequence in the list, s_1, because it differs from s_1 in the first element. Whatever $s_1(1)$ is, we defined $\bar{s}(1)$ to be the opposite. It can't be the second sequence in the list, because $\bar{s}$ differs from s_2 in the second element: if $s_2(2)$ is 0, $\bar{s}(2)$ is 1, and vice versa. And so on.

More precisely: if $\bar{s}$ were on the list, there would be some k so that $\bar{s} = s_k$. Two sequences are identical iff they agree at every place, i.e., for any n, $\bar{s}(n) = s_k(n)$. So in particular, taking $n = k$ as a special case, $\bar{s}(k) = s_k(k)$ would have to hold. $s_k(k)$ is either 0 or 1. If it is 0 then $\bar{s}(k)$ must be 1—that's how we defined $\bar{s}$. But if $s_k(k) = 1$ then, again because of the way we defined $\bar{s}$, $\bar{s}(k) = 0$. In either case $\bar{s}(k) \neq s_k(k)$.

We started by assuming that there is a list of elements of $\mathbb{B}^\omega$, $s_1, s_2, \dots$ From this list we constructed a sequence $\bar{s}$ which we proved cannot be on the list. But it definitely is a sequence of 0's and 1's if all the s_i are sequences of 0's and 1's, i.e., $\bar{s} \in \mathbb{B}^\omega$. This shows in particular that there can be no list of *all* elements of $\mathbb{B}^\omega$, since for any such list we could also construct a sequence $\bar{s}$ guaranteed to not be on the list, so the assumption that there is a list of all sequences in $\mathbb{B}^\omega$ leads to a contradiction. □

This proof method is called "diagonalization" because it uses the diagonal of the array to define $\bar{s}$. Diagonalization need not involve the presence of an array: we can show that sets are not countable by using a similar idea even when no array and no actual diagonal is involved.

Theorem 4.18. *$\wp(\mathbb{Z}^+)$ is not countable.*

Proof. We proceed in the same way, by showing that for every list of subsets of $\mathbb{Z}^+$ there is a subset of $\mathbb{Z}^+$ which cannot be on the list. Suppose the following is a given list of subsets of $\mathbb{Z}^+$:

$$Z_1, Z_2, Z_3, \dots$$

We now define a set $\overline{Z}$ such that for any $n \in \mathbb{Z}^+$, $n \in \overline{Z}$ iff $n \notin Z_n$:

$$\overline{Z} = \{n \in \mathbb{Z}^+ \mid n \notin Z_n\} \qquad \square$$

$\overline{Z}$ is clearly a set of positive integers, since by assumption each Z_n is, and thus $\overline{Z} \in \wp(\mathbb{Z}^+)$. But $\overline{Z}$ cannot be on the list. To show this, we'll establish that for each $k \in \mathbb{Z}^+$, $\overline{Z} \neq Z_k$.

So let $k \in \mathbb{Z}^+$ be arbitrary. We've defined $\overline{Z}$ so that for any $n \in \mathbb{Z}^+$, $n \in \overline{Z}$ iff $n \notin Z_n$. In particular, taking $n = k$, $k \in \overline{Z}$ iff $k \notin Z_k$. But this shows that $\overline{Z} \neq Z_k$, since k is an element of one but not the other, and so $\overline{Z}$ and Z_k have different elements. Since k was arbitrary, $\overline{Z}$ is not on the list Z_1, Z_2, …

The preceding proof did not mention a diagonal, but you can think of it as involving a diagonal if you picture it this way: Imagine the sets $Z_1, Z_2, \dots,$ written in an array, where each element $j \in Z_i$ is listed in the j-th column. Say the first four sets on that list are $\{1, 2, 3, \dots\}$, $\{2, 4, 6, \dots\}$, $\{1, 2, 5\}$, and $\{3, 4, 5, \dots\}$. Then the array would begin with

$$\begin{array}{rlllllll}
Z_1 = \{ & \mathbf{1}, & 2, & 3, & 4, & 5, & 6, & \dots\} \\
Z_2 = \{ & & \mathbf{2}, & & 4, & & 6, & \dots\} \\
Z_3 = \{ & 1, & 2, & & & 5 & & \} \\
Z_4 = \{ & & & 3, & \mathbf{4}, & 5, & 6, & \dots\} \\
\vdots & & & & & \ddots & &
\end{array}$$

Then $\overline{Z}$ is the set obtained by going down the diagonal, leaving out any numbers that appear along the diagonal and include those j where the array has a gap in the j-th row/column. In the above case, we would leave out 1 and 2, include 3, leave out 4, etc.

4.7 Reduction

We showed $\wp(\mathbb{Z}^+)$ to be uncountable by a diagonalization argument. We already had a proof that $\mathbb{B}^\omega$, the set of all infinite sequences of 0s and 1s, is uncountable. Here's another way we can prove that $\wp(\mathbb{Z}^+)$ is uncountable: Show that *if $\wp(\mathbb{Z}^+)$ is countable then $\mathbb{B}^\omega$ is also countable*. Since we know $\mathbb{B}^\omega$ is not countable, $\wp(\mathbb{Z}^+)$ can't be either. This is called *reducing* one problem to another—in this case, we reduce the problem of enumerating $\mathbb{B}^\omega$ to the problem of enumerating $\wp(\mathbb{Z}^+)$. A solution to the latter—an enumeration of $\wp(\mathbb{Z}^+)$—would yield a solution to the former—an enumeration of $\mathbb{B}^\omega$.

How do we reduce the problem of enumerating a set B to that of enumerating a set A? We provide a way of turning an enumeration of A into an enumeration of B. The easiest way to do that is to define a surjective function $f\colon A \to B$. If x_1, x_2, … enumerates A, then $f(x_1)$, $f(x_2)$, … would enumerate B. In our case, we are looking for a surjective function $f\colon \wp(\mathbb{Z}^+) \to \mathbb{B}^\omega$.

Proof of Theorem 4.18 by reduction. Suppose that $\wp(\mathbb{Z}^+)$ were countable, and thus that there is an enumeration of it, Z_1, Z_2, Z_3, …

Define the function $f\colon \wp(\mathbb{Z}^+) \to \mathbb{B}^\omega$ by letting $f(Z)$ be the sequence s_k such that $s_k(n) = 1$ iff $n \in Z$, and $s_k(n) = 0$ otherwise. This clearly defines a function, since whenever $Z \subseteq \mathbb{Z}^+$, any $n \in \mathbb{Z}^+$ either is an element of Z or isn't. For instance, the set $2\mathbb{Z}^+ = \{2, 4, 6, \dots\}$ of positive even numbers gets mapped to the sequence $010101\dots$, the empty set gets mapped to $0000\dots$ and the set $\mathbb{Z}^+$ itself to $1111\dots$.

It also is surjective: Every sequence of 0s and 1s corresponds to some set of positive integers, namely the one which has as its members those integers corresponding to the places where the sequence has 1s. More precisely, suppose $s \in \mathbb{B}^\omega$. Define $Z \subseteq \mathbb{Z}^+$ by:

$$Z = \{n \in \mathbb{Z}^+ \mid s(n) = 1\}$$

Then $f(Z) = s$, as can be verified by consulting the definition of f.

Now consider the list

$$f(Z_1), f(Z_2), f(Z_3), \dots$$

Since f is surjective, every member of $\mathbb{B}^\omega$ must appear as a value of f for some argument, and so must appear on the list. This list must therefore enumerate all of $\mathbb{B}^\omega$.

So if $\wp(\mathbb{Z}^+)$ were countable, $\mathbb{B}^\omega$ would be countable. But $\mathbb{B}^\omega$ is uncountable (Theorem 4.17). Hence $\wp(\mathbb{Z}^+)$ is uncountable. □

It is easy to be confused about the direction the reduction goes in. For instance, a surjective function $g\colon \mathbb{B}^\omega \to B$ does *not* establish that B is uncountable. (Consider $g\colon \mathbb{B}^\omega \to \mathbb{B}$ defined by $g(s) = s(1)$, the function that maps a sequence of 0's and 1's to its first element. It is surjective, because some sequences start with 0 and some start with 1. But $\mathbb{B}$ is finite.) Note also that the function f must be surjective, or otherwise the argument does not go through: $f(x_1), f(x_2), \dots$ would then not be guaranteed to include all the elements of B. For instance,

$$h(n) = \underbrace{000\dots0}_{n\ 0\text{'s}}$$

defines a function $h\colon \mathbb{Z}^+ \to \mathbb{B}^\omega$, but $\mathbb{Z}^+$ is countable.

4.8 Equinumerosity

We have an intuitive notion of "size" of sets, which works fine for finite sets. But what about infinite sets? If we want to come up with a formal way of comparing the sizes of two sets of *any* size, it is a good idea to start by defining when sets are the same size. Here is Frege:

> If a waiter wants to be sure that he has laid exactly as many knives as plates on the table, he does not need to count either of them, if he simply lays a knife to the right of each plate, so that every knife on the table lies to the right of some plate. The plates and knives are thus uniquely correlated to each other, and indeed through that same spatial relationship. (Frege, 1884, §70)

The insight of this passage can be brought out through a formal definition:

Definition 4.19. A is *equinumerous* with B, written $A \approx B$, iff there is a bijection $f\colon A \to B$.

Proposition 4.20. *Equinumerosity is an equivalence relation.*

Proof. We must show that equinumerosity is reflexive, symmetric, and transitive. Let A, B, and C be sets.

Reflexivity. The identity map $\mathrm{Id}_A\colon A \to A$, where $\mathrm{Id}_A(x) = x$ for all $x \in A$, is a bijection. So $A \approx A$.

Symmetry. Suppose $A \approx B$, i.e., there is a bijection $f\colon A \to B$. Since f is bijective, its inverse f^{-1} exists and is also bijective. Hence, $f^{-1}\colon B \to A$ is a bijection, so $B \approx A$.

Transitivity. Suppose that $A \approx B$ and $B \approx C$, i.e., there are bijections $f\colon A \to B$ and $g\colon B \to C$. Then the composition $g \circ f\colon A \to C$ is bijective, so that $A \approx C$. □

Proposition 4.21. *If $A \approx B$, then A is countable if and only if B is.*

Proof. Suppose $A \approx B$, so there is some bijection $f\colon A \to B$, and suppose that A is countable. Then either $A = \emptyset$ or there is a surjective function $g\colon \mathbb{Z}^+ \to A$. If $A = \emptyset$, then $B = \emptyset$ also (otherwise there would be an element $y \in B$ but no $x \in A$ with $g(x) = y$). If, on the other hand, $g\colon \mathbb{Z}^+ \to A$ is surjective, then $f \circ g\colon \mathbb{Z}^+ \to B$ is surjective. To see this, let $y \in B$. Since f is surjective, there is an $x \in A$ such that $f(x) = y$. Since g is surjective, there is an $n \in \mathbb{Z}^+$ such that $g(n) = x$. Hence,

$$(f \circ g)(n) = f(g(n)) = f(x) = y$$

and thus $f \circ g$ is surjective. We have that $f \circ g$ is an enumeration of B, and so B is countable.

If B is countable, we obtain that A is countable by repeating the argument with the bijection $f^{-1}\colon B \to A$ instead of f. □

4.9 Sets of Different Sizes, and Cantor's Theorem

We have offered a precise statement of the idea that two sets have the same size. We can also offer a precise statement of the idea that one set is smaller than another. Our definition of "is smaller than (or equinumerous)" will require, instead of a bijection between the sets, an injection from the first set to the second. If such a function exists, the size of the first set is less than or equal to the size of the second. Intuitively, an injection from one set to another guarantees that the range of the function has at least as many elements as the domain, since no two elements of the domain map to the same element of the range.

Definition 4.22. A is *no larger than* B, written $A \preceq B$, iff there is an injection $f\colon A \to B$.

It is clear that this is a reflexive and transitive relation, but that it is not symmetric (this is left as an exercise). We can also introduce a notion, which states that one set is (strictly) smaller than another.

Definition 4.23. A is *smaller than* B, written $A \prec B$, iff there is an injection $f\colon A \to B$ but no bijection $g\colon A \to B$, i.e., $A \preceq B$ and $A \not\approx B$.

It is clear that this relation is irreflexive and transitive. (This is left as an exercise.) Using this notation, we can say that a set A is countable iff $A \preceq \mathbb{N}$, and that A is uncountable iff $\mathbb{N} \prec A$. This allows us to restate Theorem 4.18 as the observation that $\mathbb{Z}^+ \prec \wp(\mathbb{Z}^+)$. In fact, Cantor (1892) proved that this last point is *perfectly general*:

Theorem 4.24 (Cantor). *$A \prec \wp(A)$, for any set A.*

Proof. The map $f(x) = \{x\}$ is an injection $f\colon A \to \wp(A)$, since if $x \neq y$, then also $\{x\} \neq \{y\}$ by extensionality, and so $f(x) \neq f(y)$. So we have that $A \preceq \wp(A)$.

We will now show that there cannot be a surjective function $g\colon A \to \wp(A)$, let alone a bijective one, and hence that $A \not\approx \wp(A)$. For suppose that $g\colon A \to \wp(A)$. Since g is total, every $x \in A$ is mapped to a subset $g(x) \subseteq A$. We can show that g cannot be surjective. To do this, we define a subset $\overline{A} \subseteq A$ which by definition cannot be in the range of g. Let

$$\overline{A} = \{x \in A \mid x \notin g(x)\}.$$

Since $g(x)$ is defined for all $x \in A$, $\overline{A}$ is clearly a well-defined subset of A. But, it cannot be in the range of g. Let $x \in A$ be arbitrary, we will show that $\overline{A} \neq g(x)$. If $x \in g(x)$, then it does not satisfy $x \notin g(x)$, and so by the definition of $\overline{A}$, we have $x \notin \overline{A}$. If $x \in \overline{A}$, it must satisfy the defining property of $\overline{A}$, i.e., $x \in A$ and $x \notin g(x)$. Since x was arbitrary, this shows that for each

$x \in \overline{A}$, $x \in g(x)$ iff $x \notin \overline{A}$, and so $g(x) \neq \overline{A}$. In other words, $\overline{A}$ cannot be in the range of g, contradicting the assumption that g is surjective. $\square$

It's instructive to compare the proof of Theorem 4.24 to that of Theorem 4.18. There we showed that for any list Z_1, Z_2, ..., of subsets of $\mathbb{Z}^+$ one can construct a set $\overline{Z}$ of numbers guaranteed not to be on the list. It was guaranteed not to be on the list because, for every $n \in \mathbb{Z}^+$, $n \in Z_n$ iff $n \notin \overline{Z}$. This way, there is always some number that is an element of one of Z_n or $\overline{Z}$ but not the other. We follow the same idea here, except the indices n are now elements of A instead of $\mathbb{Z}^+$. The set $\overline{B}$ is defined so that it is different from $g(x)$ for each $x \in A$, because $x \in g(x)$ iff $x \notin \overline{B}$. Again, there is always an element of A which is an element of one of $g(x)$ and $\overline{B}$ but not the other. And just as $\overline{Z}$ therefore cannot be on the list Z_1, Z_2, ..., $\overline{B}$ cannot be in the range of g.

The proof is also worth comparing with the proof of Russell's Paradox, Theorem 1.29. Indeed, Cantor's Theorem was the inspiration for Russell's own paradox.

4.10 The Notion of Size, and Schröder-Bernstein

Here is an intuitive thought: if A is no larger than B and B is no larger than A, then A and B are equinumerous. To be honest, if this thought were *wrong*, then we could scarcely justify the thought that our defined notion of equinumerosity has anything to do with comparisons of "sizes" between sets! Fortunately, though, the intuitive thought is correct. This is justified by the Schröder-Bernstein Theorem.

Theorem 4.25 (Schröder-Bernstein). *If $A \preceq B$ and $B \preceq A$, then $A \approx B$.*

In other words, if there is an injection from A to B, and an injection from B to A, then there is a bijection from A to B.

This result, however, is really rather *difficult* to prove. Indeed, although Cantor stated the result, others proved it.[1] For now, you can (and must) take it on trust.

Fortunately, Schröder-Bernstein is *correct*, and it vindicates our thinking of the relations we defined, i.e., $A \approx B$ and $A \preceq B$, as having something to do with "size". Moreover, Schröder-Bernstein is very *useful*. It can be difficult to think of a bijection between two equinumerous sets. The Schröder-Bernstein Theorem allows us to break the comparison down into cases so we only have to think of an injection from the first to the second, and vice-versa.

Problems

Problem 4.1. Define an enumeration of the positive squares 1, 4, 9, 16, ...

[1] For more on the history, see e.g., Potter (2004, pp. 165–6).

Problem 4.2. Show that if A and B are countable, so is $A \cup B$. To do this, suppose there are surjective functions $f\colon \mathbb{Z}^+ \to A$ and $g\colon \mathbb{Z}^+ \to B$, and define a surjective function $h\colon \mathbb{Z}^+ \to A \cup B$ and prove that it is surjective. Also consider the cases where A or $B = \emptyset$.

Problem 4.3. Show that if $B \subseteq A$ and A is countable, so is B. To do this, suppose there is a surjective function $f\colon \mathbb{Z}^+ \to A$. Define a surjective function $g\colon \mathbb{Z}^+ \to B$ and prove that it is surjective. What happens if $B = \emptyset$?

Problem 4.4. Show by induction on n that if $A_1, A_2, \dots, A_n$ are all countable, so is $A_1 \cup \dots \cup A_n$. You may assume the fact that if two sets A and B are countable, so is $A \cup B$.

Problem 4.5. According to Definition 4.4, a set A is enumerable iff $A = \emptyset$ or there is a surjective $f\colon \mathbb{Z}^+ \to A$. It is also possible to define "countable set" precisely by: a set is enumerable iff there is an injective function $g\colon A \to \mathbb{Z}^+$. Show that the definitions are equivalent, i.e., show that there is an injective function $g\colon A \to \mathbb{Z}^+$ iff either $A = \emptyset$ or there is a surjective $f\colon \mathbb{Z}^+ \to A$.

Problem 4.6. Show that $(\mathbb{Z}^+)^n$ is countable, for every $n \in \mathbb{N}$.

Problem 4.7. Show that $(\mathbb{Z}^+)^*$ is countable. You may assume problem 4.6.

Problem 4.8. Give an enumeration of the set of all non-negative rational numbers.

Problem 4.9. Show that $\mathbb{Q}$ is countable. Recall that any rational number can be written as a fraction z/m with $z \in \mathbb{Z}$, $m \in \mathbb{N}^+$.

Problem 4.10. Define an enumeration of $\mathbb{B}^*$.

Problem 4.11. Recall from your introductory logic course that each possible truth table expresses a truth function. In other words, the truth functions are all functions from $\mathbb{B}^k \to \mathbb{B}$ for some k. Prove that the set of all truth functions is enumerable.

Problem 4.12. Show that the set of all finite subsets of an arbitrary infinite countable set is countable.

Problem 4.13. A subset of $\mathbb{N}$ is said to be *cofinite* iff it is the complement of a finite set $\mathbb{N}$; that is, $A \subseteq \mathbb{N}$ is cofinite iff $\mathbb{N} \setminus A$ is finite. Let I be the set whose elements are exactly the finite and cofinite subsets of $\mathbb{N}$. Show that I is countable.

Problem 4.14. Show that the countable union of countable sets is countable. That is, whenever $A_1, A_2, \dots$ are sets, and each A_i is countable, then the union $\bigcup_{i=1}^{\infty} A_i$ of all of them is also countable. [NB: this is hard!]

Problem 4.15. Let $f\colon A \times B \to \mathbb{N}$ be an arbitrary pairing function. Show that the inverse of f is an enumeration of $A \times B$.

Problem 4.16. Specify a function that encodes $\mathbb{N}^3$.

Problem 4.17. Show that $\wp(\mathbb{N})$ is uncountable by a diagonal argument.

Problem 4.18. Show that the set of functions $f\colon \mathbb{Z}^+ \to \mathbb{Z}^+$ is uncountable by an explicit diagonal argument. That is, show that if f_1, f_2, …, is a list of functions and each $f_i\colon \mathbb{Z}^+ \to \mathbb{Z}^+$, then there is some $\overline{f}\colon \mathbb{Z}^+ \to \mathbb{Z}^+$ not on this list.

Problem 4.19. Show that if there is an injective function $g\colon B \to A$, and B is uncountable, then so is A. Do this by showing how you can use g to turn an enumeration of A into one of B.

Problem 4.20. Show that the set of all *sets of* pairs of positive integers is uncountable by a reduction argument.

Problem 4.21. Show that the set X of all functions $f\colon \mathbb{N} \to \mathbb{N}$ is uncountable by a reduction argument (Hint: give a surjective function from X to $\mathbb{B}^\omega$.)

Problem 4.22. Show that $\mathbb{N}^\omega$, the set of infinite sequences of natural numbers, is uncountable by a reduction argument.

Problem 4.23. Let P be the set of functions from the set of positive integers to the set $\{0\}$, and let Q be the set of *partial* functions from the set of positive integers to the set $\{0\}$. Show that P is countable and Q is not. (Hint: reduce the problem of enumerating $\mathbb{B}^\omega$ to enumerating Q).

Problem 4.24. Let S be the set of all surjective functions from the set of positive integers to the set $\{0,1\}$, i.e., S consists of all surjective $f\colon \mathbb{Z}^+ \to \mathbb{B}$. Show that S is uncountable.

Problem 4.25. Show that the set $\mathbb{R}$ of all real numbers is uncountable.

Problem 4.26. Show that if $A \approx C$ and $B \approx D$, and $A \cap B = C \cap D = \emptyset$, then $A \cup B \approx C \cup D$.

Problem 4.27. Show that if A is infinite and countable, then $A \approx \mathbb{N}$.

Problem 4.28. Show that there cannot be an injection $g\colon \wp(A) \to A$, for any set A. Hint: Suppose $g\colon \wp(A) \to A$ is injective. Consider $D = \{g(B) \mid B \subseteq A \text{ and } g(B) \notin B\}$. Let $x = g(D)$. Use the fact that g is injective to derive a contradiction.

Part II

First-order Logic

Chapter 5

Introduction to First-Order Logic

5.1 First-Order Logic

You are probably familiar with first-order logic from your first introduction to formal logic.[1] You may know it as "quantificational logic" or "predicate logic." First-order logic, first of all, is a formal language. That means, it has a certain vocabulary, and its expressions are strings from this vocabulary. But not every string is permitted. There are different kinds of permitted expressions: terms, formulae, and sentences. We are mainly interested in sentences of first-order logic: they provide us with a formal analogue of sentences of English, and about them we can ask the questions a logician typically is interested in. For instance:

- Does ψ follow from φ logically?
- Is φ logically true, logically false, or contingent?
- Are φ and ψ equivalent?

These questions are primarily questions about the "meaning" of sentences of first-order logic. For instance, a philosopher would analyze the question of whether ψ follows logically from φ as asking: is there a case where φ is true but ψ is false (ψ doesn't follow from φ), or does every case that makes φ true also make ψ true (ψ does follow from φ)? But we haven't been told yet what a "case" is—that is the job of *semantics*. The semantics of first-order logic provides a mathematically precise model of the philosopher's intuitive idea of "case," and also—and this is important—of what it is for a sentence φ to be *true in* a case. We call the mathematically precise model that we will develop a structure. The relation which makes "true in" precise, is called the relation of *satisfaction*. So what we will define is "φ is satisfied in $\mathfrak{M}$" (in symbols:

[1] In fact, we more or less assume you are! If you're not, you could review a more elementary textbook, such as *forall x* (Magnus et al., 2021).

$\mathfrak{M} \vDash \varphi$) for sentences φ and structures $\mathfrak{M}$. Once this is done, we can also give precise definitions of the other semantical terms such as "follows from" or "is logically true." These definitions will make it possible to settle, again with mathematical precision, whether, e.g., $\forall x\,(\varphi(x) \supset \psi(x)), \exists x\,\varphi(x) \vDash \exists x\,\psi(x)$. The answer will, of course, be "yes." If you've already been trained to symbolize sentences of English in first-order logic, you will recognize this as, e.g., the symbolizations of, say, "All ants are insects, there are ants, therefore there are insects." That is obviously a valid argument, and so our mathematical model of "follows from" for our formal language should give the same answer.

Another topic you probably remember from your first introduction to formal logic is that there are *derivations*. If you have taken a first formal logic course, your instructor will have made you practice finding such derivations, perhaps even a derivation that shows that the above entailment holds. There are many different ways to give derivations: you may have done something called "natural deduction" or "truth trees," but there are many others. The purpose of derivation systems is to provide tools using which the logicians' questions above can be answered: e.g., a natural deduction derivation in which $\forall x\,(\varphi(x) \supset \psi(x))$ and $\exists x\,\varphi(x)$ are premises and $\exists x\,\psi(x)$ is the conclusion (last line) *verifies* that $\exists x\,\psi(x)$ logically follows from $\forall x\,(\varphi(x) \supset \psi(x))$ and $\exists x\,\varphi(x)$.

But why is that? On the face of it, derivation systems have nothing to do with semantics: giving a formal derivation merely involves arranging symbols in certain rule-governed ways; they don't mention "cases" or "true in" at all. The connection between derivation systems and semantics has to be established by a meta-logical investigation. What's needed is a mathematical proof, e.g., that a formal derivation of $\exists x\,\psi(x)$ from premises $\forall x\,(\varphi(x) \supset \psi(x))$ and $\exists x\,\varphi(x)$ is possible, if, and only if, $\forall x\,(\varphi(x) \supset \psi(x))$ and $\exists x\,\varphi(x)$ together entail $\exists x\,\psi(x)$. Before this can be done, however, a lot of painstaking work has to be carried out to get the definitions of syntax and semantics correct.

5.2 Syntax

We first must make precise what strings of symbols count as sentences of first-order logic. We'll do this later; for now we'll just proceed by example. The basic building blocks—the vocabulary—of first-order logic divides into two parts. The first part is the symbols we use to say specific things or to pick out specific things. We pick out things using constant symbols, and we say stuff about the things we pick out using predicate symbols. E.g, we might use a as a constant symbol to pick out a single thing, and then say something about it using the sentence $P(a)$. If you have meanings for "a" and "P" in mind, you can read $P(a)$ as a sentence of English (and you probably have done so when you first learned formal logic). Once you have such simple sentences of first-order logic, you can build more complex ones using the second part of the vocabulary: the logical symbols (connectives and quantifiers). So, for instance, we can form expressions like $(P(a) \mathbin{\&} Q(b))$ or $\exists x\, P(x)$.

In order to provide the precise definitions of semantics and the rules of our derivation systems required for rigorous meta-logical study, we first of all have to give a precise definition of what counts as a sentence of first-order logic. The basic idea is easy enough to understand: there are some simple sentences we can form from just predicate symbols and constant symbols, such as $P(a)$. And then from these we form more complex ones using the connectives and quantifiers. But what exactly are the rules by which we are allowed to form more complex sentences? These must be specified, otherwise we have not defined "sentence of first-order logic" precisely enough. There are a few issues. The first one is to get the right strings to count as sentences. The second one is to do this in such a way that we can give mathematical proofs about *all* sentences. Finally, we'll have to also give precise definitions of some rudimentary operations with sentences, such as "replace every x in φ by b." The trouble is that the quantifiers and variables we have in first-order logic make it not entirely obvious how this should be done. E.g., should $\exists x\, P(a)$ count as a sentence? What about $\exists x\, \exists x\, P(x)$? What should the result of "replace x by b in $(P(x) \mathbin{\&} \exists x\, P(x))$" be?

5.3 Formulae

Here is the approach we will use to rigorously specify sentences of first-order logic and to deal with the issues arising from the use of variables. We first define a *different* set of expressions: formulae. Once we've done that, we can consider the role variables play in them—and on the basis of some other ideas, namely those of "free" and "bound" variables, we can define what a sentence is (namely, a formula without free variables). We do this not just because it makes the definition of "sentence" more manageable, but also because it will be crucial to the way we define the semantic notion of satisfaction.

Let's define "formula" for a simple first-order language, one containing only a single predicate symbol P and a single constant symbol a, and only the logical symbols $\sim$, &, and $\exists$. Our full definitions will be much more general: we'll allow infinitely many predicate symbols and constant symbols. In fact, we will also consider function symbols which can be combined with constant symbols and variables to form "terms." For now, a and the variables will be our only terms. We do need infinitely many variables. We'll officially use the symbols $v_0, v_1, \dots,$ as variables.

Definition 5.1. The set of *formulae* Frm is defined as follows:

1. $P(a)$ and $P(v_i)$ are formulae ($i \in \mathbb{N}$).
2. If φ is a formula, then $\sim\varphi$ is formula.
3. If φ and ψ are formulae, then $(\varphi \mathbin{\&} \psi)$ is a formula.

4. If φ is a formula and x is a variable, then $\exists x\, \varphi$ is a formula.

5. Nothing else is a formula.

(1) tells us that $P(a)$ and $P(v_i)$ are formulae, for any $i \in \mathbb{N}$. These are the so-called *atomic* formulae. They give us something to start from. The other clauses give us ways of forming new formulae from ones we have already formed. So for instance, by (2), we get that $\sim P(v_2)$ is a formula, since $P(v_2)$ is already a formula by (1). Then, by (4), we get that $\exists v_2 \sim P(v_2)$ is another formula, and so on. (5) tells us that *only* strings we can form in this way count as formulae. In particular, $\exists v_0\, P(a)$ and $\exists v_0\, \exists v_0\, P(a)$ *do* count as formulae, and $(\sim P(a))$ does not, because of the extraneous outer parentheses.

This way of defining formulae is called an *inductive definition*, and it allows us to prove things about formulae using a version of proof by induction called *structural induction*. These are discussed in a general way in appendix B.4 and appendix B.5, which you should review before delving into the proofs later on. Basically, the idea is that if you want to give a proof that something is true for all formulae, you show first that it is true for the atomic formulae, and then that *if* it's true for any formula φ (and ψ), it's *also* true for $\sim\varphi$, $(\varphi \,\&\, \psi)$, and $\exists x\, \varphi$. For instance, this proves that it's true for $\exists v_2 \sim P(v_2)$: from the first part you know that it's true for the atomic formula $P(v_2)$. Then you get that it's true for $\sim P(v_2)$ by the second part, and then again that it's true for $\exists v_2 \sim P(v_2)$ itself. Since all formulae are inductively generated from atomic formulae, this works for any of them.

5.4 Satisfaction

We can already skip ahead to the semantics of first-order logic once we know what formulae are: here, the basic definition is that of a structure. For our simple language, a structure $\mathfrak{M}$ has just three components: a non-empty set $|\mathfrak{M}|$ called the *domain*, what a picks out in $\mathfrak{M}$, and what P is true of in $\mathfrak{M}$. The object picked out by a is denoted $a^{\mathfrak{M}}$ and the set of things P is true of by $P^{\mathfrak{M}}$. A structure $\mathfrak{M}$ consists of just these three things: $|\mathfrak{M}|$, $a^{\mathfrak{M}} \in |\mathfrak{M}|$ and $P^{\mathfrak{M}} \subseteq |\mathfrak{M}|$. The general case will be more complicated, since there will be many predicate symbols and constant symbols, the constant symbols can have more than one place, and there will also be function symbols.

This is enough to give a definition of satisfaction for formulae that don't contain variables. The idea is to give an inductive definition that mirrors the way we have defined formulae. We specify when an atomic formula is satisfied in $\mathfrak{M}$, and then when, e.g., $\sim\varphi$ is satisfied in $\mathfrak{M}$ on the basis of whether or not φ is satisfied in $\mathfrak{M}$. E.g., we could define:

1. $P(a)$ is satisfied in $\mathfrak{M}$ iff $a^{\mathfrak{M}} \in P^{\mathfrak{M}}$.

2. $\sim\varphi$ is satisfied in $\mathfrak{M}$ iff φ is not satisfied in $\mathfrak{M}$.

3. $(\varphi \,\&\, \psi)$ is satisfied in $\mathfrak{M}$ iff φ is satisfied in $\mathfrak{M}$, and ψ is satisfied in $\mathfrak{M}$ as well.

Let's say that $|\mathfrak{M}| = \{0, 1, 2\}$, $a^{\mathfrak{M}} = 1$, and $P^{\mathfrak{M}} = \{1, 2\}$. This definition would tell us that $P(a)$ is satisfied in $\mathfrak{M}$ (since $a^{\mathfrak{M}} = 1 \in \{1, 2\} = P^{\mathfrak{M}}$). It tells us further that $\sim P(a)$ is not satisfied in $\mathfrak{M}$, and that in turn $\sim\sim P(a)$ is and $(\sim P(a) \,\&\, P(a))$ is not satisfied, and so on.

The trouble comes when we want to give a definition for the quantifiers: we'd like to say something like, "$\exists v_0\, P(v_0)$ is satisfied iff $P(v_0)$ is satisfied." But the structure $\mathfrak{M}$ doesn't tell us what to do about variables. What we actually want to say is that $P(v_0)$ is satisfied *for some value of* v_0. To make this precise we need a way to assign elements of $|\mathfrak{M}|$ not just to a but also to v_0. To this end, we introduce variable *assignments*. A variable assignment is simply a function s that maps variables to elements of $|\mathfrak{M}|$ (in our example, to one of 1, 2, or 3). Since we don't know beforehand which variables might appear in a formula we can't limit which variables s assigns values to. The simple solution is to require that s assigns values to *all* variables v_0, v_1, ... We'll just use only the ones we need.

Instead of defining satisfaction of formulae just relative to a structure, we'll define it relative to a structure $\mathfrak{M}$ *and* a variable assignment s, and write $\mathfrak{M}, s \vDash \varphi$ for short. Our definition will now include an additional clause to deal with atomic formulae containing variables:

1. $\mathfrak{M}, s \vDash P(a)$ iff $a^{\mathfrak{M}} \in P^{\mathfrak{M}}$.

2. $\mathfrak{M}, s \vDash P(v_i)$ iff $s(v_i) \in P^{\mathfrak{M}}$.

3. $\mathfrak{M}, s \vDash \sim\varphi$ iff not $\mathfrak{M}, s \vDash \varphi$.

4. $\mathfrak{M}, s \vDash (\varphi \,\&\, \psi)$ iff $\mathfrak{M}, s \vDash \varphi$ and $\mathfrak{M}, s \vDash \psi$.

Ok, this solves one problem: we can now say when $\mathfrak{M}$ satisfies $P(v_0)$ for the value $s(v_0)$. To get the definition right for $\exists v_0\, P(v_0)$ we have to do one more thing: We want to have that $\mathfrak{M}, s \vDash \exists v_0\, P(v_0)$ iff $\mathfrak{M}, s' \vDash P(v_0)$ for *some* way s' of assigning a value to v_0. But the value assigned to v_0 does not necessarily have to be the value that $s(v_0)$ picks out. We'll introduce a notation for that: if $m \in |\mathfrak{M}|$, then we let $s[m/v_0]$ be the assignment that is just like s (for all variables other than v_0), except to v_0 it assigns m. Now our definition can be:

5. $\mathfrak{M}, s \vDash \exists v_i\, \varphi$ iff $\mathfrak{M}, s[m/v_i] \vDash \varphi$ for some $m \in |\mathfrak{M}|$.

Does it work out? Let's say we let $s(v_i) = 0$ for all $i \in \mathbb{N}$. $\mathfrak{M}, s \vDash \exists v_0\, P(v_0)$ iff there is an $m \in |\mathfrak{M}|$ so that $\mathfrak{M}, s[m/v_0] \vDash P(v_0)$. And there is: we can choose $m = 1$ or $m = 2$. Note that this is true even if the value $s(v_0)$ assigned to v_0 by s itself—in this case, 0—doesn't do the job. We have $\mathfrak{M}, s[1/v_0] \vDash P(v_0)$ but not $\mathfrak{M}, s \vDash P(v_0)$.

If this looks confusing and cumbersome: it is. But the added complexity is required to give a precise, inductive definition of satisfaction for all formulae, and we need something like it to precisely define the semantic notions. There are other ways of doing it, but they are all equally (in)elegant.

5.5 Sentences

Ok, now we have a (sketch of a) definition of satisfaction ("true in") for structures and formulae. But it needs this additional bit—a variable assignment—and what we wanted is a definition of sentences. How do we get rid of assignments, and what are sentences?

You probably remember a discussion in your first introduction to formal logic about the relation between variables and quantifiers. A quantifier is always followed by a variable, and then in the part of the sentence to which that quantifier applies (its "scope"), we understand that the variable is "bound" by that quantifier. In formulae it was not required that every variable has a matching quantifier, and variables without matching quantifiers are "free" or "unbound." We will take sentences to be all those formulae that have no free variables.

Again, the intuitive idea of when an occurrence of a variable in a formula φ is bound, which quantifier binds it, and when it is free, is not difficult to get. You may have learned a method for testing this, perhaps involving counting parentheses. We have to insist on a precise definition—and because we have defined formulae by induction, we can give a definition of the free and bound occurrences of a variable x in a formula φ also by induction. E.g., it might look like this for our simplified language:

1. If φ is atomic, all occurrences of x in it are free (that is, the occurrence of x in $P(x)$ is free).

2. If φ is of the form $\sim\psi$, then an occurrence of x in $\sim\psi$ is free iff the corresponding occurrence of x is free in ψ (that is, the free occurrences of variables in ψ are exactly the corresponding occurrences in $\sim\psi$).

3. If φ is of the form $(\psi \,\&\, \chi)$, then an occurrence of x in $(\psi \,\&\, \chi)$ is free iff the corresponding occurrence of x is free in ψ or in χ.

4. If φ is of the form $\exists x\, \psi$, then no occurrence of x in φ is free; if it is of the form $\exists y\, \psi$ where y is a different variable than x, then an occurrence of x in $\exists y\, \psi$ is free iff the corresponding occurrence of x is free in ψ.

Once we have a precise definition of free and bound occurrences of variables, we can simply say: a sentence is any formula without free occurrences of variables.

5.6 Semantic Notions

We mentioned above that when we consider whether $\mathfrak{M}, s \vDash \varphi$ holds, we (for convenience) let s assign values to all variables, but only the values it assigns to variables in φ are used. In fact, it's only the values of *free* variables in φ that matter. Of course, because we're careful, we are going to prove this fact. Since sentences have no free variables, s doesn't matter at all when it comes to whether or not they are satisfied in a structure. So, when φ is a sentence we can define $\mathfrak{M} \vDash \varphi$ to mean "$\mathfrak{M}, s \vDash \varphi$ for all s," which as it happens is true iff $\mathfrak{M}, s \vDash \varphi$ for at least one s. We need to introduce variable assignments to get a working definition of satisfaction for formulae, but for sentences, satisfaction is independent of the variable assignments.

Once we have a definition of "$\mathfrak{M} \vDash \varphi$," we know what "case" and "true in" mean as far as sentences of first-order logic are concerned. On the basis of the definition of $\mathfrak{M} \vDash \varphi$ for sentences we can then define the basic semantic notions of validity, entailment, and satisfiability. A sentence is valid, $\vDash \varphi$, if every structure satisfies it. It is entailed by a set of sentences, $\Gamma \vDash \varphi$, if every structure that satisfies all the sentences in Γ also satisfies φ. And a set of sentences is satisfiable if some structure satisfies all sentences in it at the same time.

Because formulae are inductively defined, and satisfaction is in turn defined by induction on the structure of formulae, we can use induction to prove properties of our semantics and to relate the semantic notions defined. We'll collect and prove some of these properties, partly because they are individually interesting, but mainly because many of them will come in handy when we go on to investigate the relation between semantics and derivation systems. In order to do so, we'll also have to define (precisely, i.e., by induction) some syntactic notions and operations we haven't mentioned yet.

5.7 Substitution

We'll discuss an example to illustrate how things hang together, and how the development of syntax and semantics lays the foundation for our more advanced investigations later. Our derivation systems should let us derive $P(a)$ from $\forall v_0\, P(v_0)$. Maybe we even want to state this as a rule of inference. However, to do so, we must be able to state it in the most general terms: not just for P, a, and v_0, but for any formula φ, and term t, and variable x. (Recall that constant symbols are terms, but we'll consider also more complicated terms built from constant symbols and function symbols.) So we want to be able to say something like, "whenever you have derived $\forall x\, \varphi(x)$ you are justified in inferring $\varphi(t)$—the result of removing $\forall x$ and replacing x by t." But what exactly does "replacing x by t" mean? What is the relation between $\varphi(x)$ and $\varphi(t)$? Does this always work?

To make this precise, we define the operation of *substitution*. Substitution is actually tricky, because we can't just replace all x's in φ by t, and not every t can be substituted for any x. We'll deal with this, again, using inductive definitions. But once this is done, specifying an inference rule as "infer $\varphi(t)$ from $\forall x\, \varphi(x)$" becomes a precise definition. Moreover, we'll be able to show that this is a good inference rule in the sense that $\forall x\, \varphi(x)$ entails $\varphi(t)$. But to prove this, we have to again prove something that may at first glance prompt you to ask "why are we doing this?" That $\forall x\, \varphi(x)$ entails $\varphi(t)$ relies on the fact that whether or not $\mathfrak{M} \vDash \varphi(t)$ holds depends only on the value of the term t, i.e., if we let m be whatever element of $|\mathfrak{M}|$ is picked out by t, then $\mathfrak{M}, s \vDash \varphi(t)$ iff $\mathfrak{M}, s[m/x] \vDash \varphi(x)$. This holds even when t contains variables, but we'll have to be careful with how exactly we state the result.

5.8 Models and Theories

Once we've defined the syntax and semantics of first-order logic, we can get to work investigating the properties of structures and the semantic notions. We can also define derivation systems, and investigate those. For a set of sentences, we can ask: what structures make all the sentences in that set true? Given a set of sentences Γ, a structure $\mathfrak{M}$ that satisfies them is called a *model of* Γ. We might start from Γ and try to find its models—what do they look like? How big or small do they have to be? But we might also start with a single structure or collection of structures and ask: what sentences are true in them? Are there sentences that *characterize* these structures in the sense that they, and only they, are true in them? These kinds of questions are the domain of *model theory*. They also underlie the *axiomatic method*: describing a collection of structures by a set of sentences, the axioms of a theory. This is made possible by the observation that exactly those sentences entailed in first-order logic by the axioms are true in all models of the axioms.

As a very simple example, consider preorders. A preorder is a relation R on some set A which is both reflexive and transitive. A set A with a two-place relation $R \subseteq A \times A$ on it is exactly what we would need to give a structure for a first-order language with a single two-place relation symbol P: we would set $|\mathfrak{M}| = A$ and $\mathsf{P}^{\mathfrak{M}} = R$. Since R is a preorder, it is reflexive and transitive, and we can find a set Γ of sentences of first-order logic that say this:

$$\forall v_0\, \mathsf{P}(v_0, v_0)$$
$$\forall v_0\, \forall v_1\, \forall v_2\, ((\mathsf{P}(v_0, v_1) \mathbin{\&} \mathsf{P}(v_1, v_2)) \supset \mathsf{P}(v_0, v_2))$$

These sentences are just the symbolizations of "for any x, Rxx" (R is reflexive) and "whenever Rxy and Ryz then also Rxz" (R is transitive). We see that a structure $\mathfrak{M}$ is a model of these two sentences Γ iff R (i.e., $\mathsf{P}^{\mathfrak{M}}$), is a preorder on A (i.e., $|\mathfrak{M}|$). In other words, the models of Γ are exactly the preorders. Any property of all preorders that can be expressed in the first-order language with just P as predicate symbol (like reflexivity and transitivity above), is entailed by the two sentences in Γ and vice versa. So anything we can prove about models of Γ we have proved about all preorders.

For any particular theory and class of models (such as Γ and all preorders), there will be interesting questions about what can be expressed in the corresponding first-order language, and what cannot be expressed. There are some properties of structures that are interesting for all languages and classes of models, namely those concerning the size of the domain. One can always express, for instance, that the domain contains exactly n elements, for any $n \in \mathbb{Z}^+$. One can also express, using a set of infinitely many sentences, that the domain is infinite. But one cannot express that the domain is finite, or that the domain is uncountable. These results about the limitations of first-order languages are consequences of the compactness and Löwenheim–Skolem theorems.

5.9 Soundness and Completeness

We'll also introduce derivation systems for first-order logic. There are many derivation systems that logicians have developed, but they all define the same derivability relation between sentences. We say that Γ *derives* φ, $\Gamma \vdash \varphi$, if there is a derivation of a certain precisely defined sort. Derivations are always finite arrangements of symbols—perhaps a list of sentences, or some more complicated structure. The purpose of derivation systems is to provide a tool to determine if a sentence is entailed by some set Γ. In order to serve that purpose, it must be true that $\Gamma \vDash \varphi$ if, and only if, $\Gamma \vdash \varphi$.

If $\Gamma \vdash \varphi$ but not $\Gamma \vDash \varphi$, our derivation system would be too strong, prove too much. The property that if $\Gamma \vdash \varphi$ then $\Gamma \vDash \varphi$ is called *soundness*, and it is a minimal requirement on any good derivation system. On the other hand, if $\Gamma \vDash \varphi$ but not $\Gamma \vdash \varphi$, then our derivation system is too weak, it doesn't prove enough. The property that if $\Gamma \vDash \varphi$ then $\Gamma \vdash \varphi$ is called *completeness*. Soundness is usually relatively easy to prove (by induction on the structure of derivations, which are inductively defined). Completeness is harder to prove.

Soundness and completeness have a number of important consequences. If a set of sentences Γ derives a contradiction (such as $\varphi \mathbin{\&} {\sim}\varphi$) it is called *inconsistent*. Inconsistent Γs cannot have any models, they are unsatisfiable. From completeness the converse follows: any Γ that is not inconsistent—or, as we will say, *consistent*—has a model. In fact, this is equivalent to completeness, and is the form of completeness we will actually prove. It is a deep and perhaps surprising result: just because you cannot prove $\varphi \mathbin{\&} {\sim}\varphi$ from Γ guarantees that there is a structure that is as Γ describes it. So completeness gives an answer to the question: which sets of sentences have models? Answer: all and only consistent sets do.

The soundness and completeness theorems have two important consequences: the compactness and the Löwenheim–Skolem theorem. These are important results in the theory of models, and can be used to establish many interesting results. We've already mentioned two: first-order logic cannot express that the domain of a structure is finite or that it is uncountable.

Historically, all of this—how to define syntax and semantics of first-order logic, how to define good derivation systems, how to prove that they are sound and complete, getting clear about what can and cannot be expressed in first-order languages—took a long time to figure out and get right. We now know how to do it, but going through all the details can still be confusing and tedious. But it's also important, because the methods developed here for the formal language of first-order logic are applied all over the place in logic, computer science, and linguistics. So working through the details pays off in the long run.

Chapter 6

Syntax of First-Order Logic

6.1 Introduction

In order to develop the theory and metatheory of first-order logic, we must first define the syntax and semantics of its expressions. The expressions of first-order logic are terms and formulae. Terms are formed from variables, constant symbols, and function symbols. Formulae, in turn, are formed from predicate symbols together with terms (these form the smallest, "atomic" formulae), and then from atomic formulae we can form more complex ones using logical connectives and quantifiers. There are many different ways to set down the formation rules; we give just one possible one. Other systems will chose different symbols, will select different sets of connectives as primitive, will use parentheses differently (or even not at all, as in the case of so-called Polish notation). What all approaches have in common, though, is that the formation rules define the set of terms and formulae *inductively*. If done properly, every expression can result essentially in only one way according to the formation rules. The inductive definition resulting in expressions that are *uniquely readable* means we can give meanings to these expressions using the same method—inductive definition.

6.2 First-Order Languages

Expressions of first-order logic are built up from a basic vocabulary containing *variables*, *constant symbols*, *predicate symbols* and sometimes *function symbols*. From them, together with logical connectives, quantifiers, and punctuation symbols such as parentheses and commas, *terms* and *formulae* are formed.

Informally, predicate symbols are names for properties and relations, constant symbols are names for individual objects, and function symbols are names for mappings. These, except for the identity predicate $=$, are the *non-logical symbols* and together make up a language. Any first-order language $\mathcal{L}$ is determined by its non-logical symbols. In the most general case, $\mathcal{L}$ contains infinitely many symbols of each kind.

In the general case, we make use of the following symbols in first-order logic:

1. Logical symbols

   a) Logical connectives: $\sim$ (negation), & (conjunction), $\lor$ (disjunction), $\supset$ (conditional), $\forall$ (universal quantifier), $\exists$ (existential quantifier).

   b) The propositional constant for falsity $\bot$.

   c) The two-place identity predicate $=$.

   d) A countably infinite set of variables: $v_0, v_1, v_2, \ldots$

2. Non-logical symbols, making up the *standard language* of first-order logic

   a) A countably infinite set of n-place predicate symbols for each $n > 0$: $A_0^n, A_1^n, A_2^n, \ldots$

   b) A countably infinite set of constant symbols: $c_0, c_1, c_2, \ldots$.

   c) A countably infinite set of n-place function symbols for each $n > 0$: $f_0^n, f_1^n, f_2^n, \ldots$

3. Punctuation marks: (,), and the comma.

Most of our definitions and results will be formulated for the full standard language of first-order logic. However, depending on the application, we may also restrict the language to only a few predicate symbols, constant symbols, and function symbols.

Example 6.1. The language $\mathcal{L}_A$ of arithmetic contains a single two-place predicate symbol $<$, a single constant symbol 0, one one-place function symbol $'$, and two two-place function symbols $+$ and $\times$.

Example 6.2. The language of set theory $\mathcal{L}_Z$ contains only the single two-place predicate symbol $\in$.

Example 6.3. The language of orders $\mathcal{L}_{\leq}$ contains only the two-place predicate symbol $\leq$.

Again, these are conventions: officially, these are just aliases, e.g., $<$, $\in$, and $\leq$ are aliases for A_0^2, 0 for c_0, $'$ for f_0^1, $+$ for f_0^2, $\times$ for f_1^2.

In addition to the primitive connectives and quantifiers introduced above, we also use the following *defined* symbols: $\equiv$ (biconditional), truth $\top$

A defined symbol is not officially part of the language, but is introduced as an informal abbreviation: it allows us to abbreviate formulas which would, if we only used primitive symbols, get quite long. This is obviously an advantage. The bigger advantage, however, is that proofs become shorter. If a symbol is primitive, it has to be treated separately in proofs. The more primitive symbols, therefore, the longer our proofs.

You may be familiar with different terminology and symbols than the ones we use above. Logic texts (and teachers) commonly use $\sim$, $\neg$, or ! for "negation", $\wedge$, $\cdot$, or & for "conjunction". Commonly used symbols for the "conditional" or "implication" are $\rightarrow$, $\Rightarrow$, and $\supset$. Symbols for "biconditional," "bi-implication," or "(material) equivalence" are $\leftrightarrow$, $\Leftrightarrow$, and $\equiv$. The $\bot$ symbol is variously called "falsity," "falsum,", "absurdity," or "bottom." The $\top$ symbol is variously called "truth," "verum," or "top."

It is conventional to use lower case letters (e.g., a, b, c) from the beginning of the Latin alphabet for constant symbols (sometimes called names), and lower case letters from the end (e.g., x, y, z) for variables. Quantifiers combine with variables, e.g., x; notational variations include $\forall x$, $(\forall x)$, (x), Πx, $\bigwedge_x$ for the universal quantifier and $\exists x$, $(\exists x)$, (Ex), Σx, $\bigvee_x$ for the existential quantifier.

We might treat all the propositional operators and both quantifiers as primitive symbols of the language. We might instead choose a smaller stock of primitive symbols and treat the other logical operators as defined. "Truth functionally complete" sets of Boolean operators include $\{\sim, \vee\}$, $\{\sim, \&\}$, and $\{\sim, \supset\}$—these can be combined with either quantifier for an expressively complete first-order language.

You may be familiar with two other logical operators: the Sheffer stroke $|$ (named after Henry Sheffer), and Peirce's arrow $\downarrow$, also known as Quine's dagger. When given their usual readings of "nand" and "nor" (respectively), these operators are truth functionally complete by themselves.

6.3 Terms and Formulae

Once a first-order language $\mathcal{L}$ is given, we can define expressions built up from the basic vocabulary of $\mathcal{L}$. These include in particular *terms* and *formulae*.

Definition 6.4 (Terms). The set of *terms* $\mathrm{Trm}(\mathcal{L})$ of $\mathcal{L}$ is defined inductively by:

1. Every variable is a term.
2. Every constant symbol of $\mathcal{L}$ is a term.
3. If f is an n-place function symbol and $t_1, \dots, t_n$ are terms, then $f(t_1, \dots, t_n)$ is a term.
4. Nothing else is a term.

A term containing no variables is a *closed term*.

The constant symbols appear in our specification of the language and the terms as a separate category of symbols, but they could instead have been included as zero-place function symbols. We could then do without the second clause in the definition of terms. We just have to understand $f(t_1, \dots, t_n)$ as just f by itself if $n = 0$.

Definition 6.5 (Formulas). The set of *formulae* $\mathrm{Frm}(\mathcal{L})$ of the language $\mathcal{L}$ is defined inductively as follows:

1. $\bot$ is an atomic formula.
2. If R is an n-place predicate symbol of $\mathcal{L}$ and $t_1, \dots, t_n$ are terms of $\mathcal{L}$, then $R(t_1, \dots, t_n)$ is an atomic formula.
3. If t_1 and t_2 are terms of $\mathcal{L}$, then $=(t_1, t_2)$ is an atomic formula.
4. If φ is a formula, then $\sim\varphi$ is formula.
5. If φ and ψ are formulae, then $(\varphi \mathbin{\&} \psi)$ is a formula.
6. If φ and ψ are formulae, then $(\varphi \lor \psi)$ is a formula.
7. If φ and ψ are formulae, then $(\varphi \supset \psi)$ is a formula.
8. If φ is a formula and x is a variable, then $\forall x\, \varphi$ is a formula.
9. If φ is a formula and x is a variable, then $\exists x\, \varphi$ is a formula.
10. Nothing else is a formula.

The definitions of the set of terms and that of formulae are *inductive definitions*. Essentially, we construct the set of formulae in infinitely many stages. In the initial stage, we pronounce all atomic formulas to be formulas; this corresponds to the first few cases of the definition, i.e., the cases for $\bot$, $R(t_1, \dots, t_n)$ and $=(t_1, t_2)$. "Atomic formula" thus means any formula of this form.

The other cases of the definition give rules for constructing new formulae out of formulae already constructed. At the second stage, we can use them to construct formulae out of atomic formulae. At the third stage, we construct new formulas from the atomic formulas and those obtained in the second stage, and so on. A formula is anything that is eventually constructed at such a stage, and nothing else.

By convention, we write $=$ between its arguments and leave out the parentheses: $t_1 = t_2$ is an abbreviation for $=(t_1, t_2)$. Moreover, $\sim=(t_1, t_2)$ is abbreviated as $t_1 \neq t_2$. When writing a formula $(\psi * \chi)$ constructed from ψ, χ using a two-place connective $*$, we will often leave out the outermost pair of parentheses and write simply $\psi * \chi$.

Some logic texts require that the variable x must occur in φ in order for $\exists x\, \varphi$ and $\forall x\, \varphi$ to count as formulae. Nothing bad happens if you don't require this, and it makes things easier.

Definition 6.6. Formulas constructed using the defined operators are to be understood as follows:

1. $\top$ abbreviates $\sim\perp$.

2. $\varphi \equiv \psi$ abbreviates $(\varphi \supset \psi) \,\&\, (\psi \supset \varphi)$.

If we work in a language for a specific application, we will often write two-place predicate symbols and function symbols between the respective terms, e.g., $t_1 < t_2$ and $(t_1 + t_2)$ in the language of arithmetic and $t_1 \in t_2$ in the language of set theory. The successor function in the language of arithmetic is even written conventionally *after* its argument: t'. Officially, however, these are just conventional abbreviations for $A^2_0(t_1, t_2)$, $f^2_0(t_1, t_2)$, $A^2_0(t_1, t_2)$ and $f^1_0(t)$, respectively.

Definition 6.7 (Syntactic identity). The symbol $\equiv$ expresses syntactic identity between strings of symbols, i.e., $\varphi \equiv \psi$ iff φ and ψ are strings of symbols of the same length and which contain the same symbol in each place.

The $\equiv$ symbol may be flanked by strings obtained by concatenation, e.g., $\varphi \equiv (\psi \lor \chi)$ means: the string of symbols φ is the same string as the one obtained by concatenating an opening parenthesis, the string ψ, the $\lor$ symbol, the string χ, and a closing parenthesis, in this order. If this is the case, then we know that the first symbol of φ is an opening parenthesis, φ contains ψ as a substring (starting at the second symbol), that substring is followed by $\lor$, etc.

As terms and formulae are built up from basic elements via inductive definitions, we can use the following induction principles to prove things about them.

Lemma 6.8 (*Principle of induction on terms*). *Let $\mathcal{L}$ be a first-order language. If some property P is such that*

1. *it holds for every variable v,*

2. *it holds for every constant symbol a of $\mathcal{L}$, and*

3. *it holds for $f(t_1, \ldots, t_n)$ whenever it holds for $t_1, \ldots, t_n$ and f is an n-place function symbol of $\mathcal{L}$*

(assuming $t_1, \ldots, t_n$ are terms of $\mathcal{L}$), then P holds for every term in $\mathrm{Trm}(\mathcal{L})$.

Lemma 6.9 (*Principle of induction on formulae*). *Let $\mathcal{L}$ be a first-order language. If some property P holds for all the atomic formulae and is such that*

1. *it holds for $\sim\varphi$ whenever it holds for φ;*

2. *it holds for $(\varphi \,\&\, \psi)$ whenever it holds for φ and ψ;*

3. *it holds for* $(\varphi \lor \psi)$ *whenever it holds for* φ *and* ψ*;*

4. *it holds for* $(\varphi \supset \psi)$ *whenever it holds for* φ *and* ψ*;*

5. *it holds for* $\exists x\, \varphi$ *whenever it holds for* φ*;*

6. *it holds for* $\forall x\, \varphi$ *whenever it holds for* φ*;*

(assuming φ *and* ψ *are formulae of* $\mathcal{L}$*), then* P *holds for all formulas in* $\mathrm{Frm}(\mathcal{L})$*.*

6.4 Unique Readability

The way we defined formulae guarantees that every formula has a *unique reading*, i.e., there is essentially only one way of constructing it according to our formation rules for formulae and only one way of "interpreting" it. If this were not so, we would have ambiguous formulae, i.e., formulae that have more than one reading or intepretation—and that is clearly something we want to avoid. But more importantly, without this property, most of the definitions and proofs we are going to give will not go through.

Perhaps the best way to make this clear is to see what would happen if we had given bad rules for forming formulae that would not guarantee unique readability. For instance, we could have forgotten the parentheses in the formation rules for connectives, e.g., we might have allowed this:

> If φ and ψ are formulae, then so is $\varphi \supset \psi$.

Starting from an atomic formula θ, this would allow us to form $\theta \supset \theta$. From this, together with θ, we would get $\theta \supset \theta \supset \theta$. But there are two ways to do this:

1. We take θ to be φ and $\theta \supset \theta$ to be ψ.

2. We take φ to be $\theta \supset \theta$ and ψ is θ.

Correspondingly, there are two ways to "read" the formula $\theta \supset \theta \supset \theta$. It is of the form $\psi \supset \chi$ where ψ is θ and χ is $\theta \supset \theta$, but *it is also* of the form $\psi \supset \chi$ with ψ being $\theta \supset \theta$ and χ being θ.

If this happens, our definitions will not always work. For instance, when we define the main operator of a formula, we say: in a formula of the form $\psi \supset \chi$, the main operator is the indicated occurrence of $\supset$. But if we can match the formula $\theta \supset \theta \supset \theta$ with $\psi \supset \chi$ in the two different ways mentioned above, then in one case we get the first occurrence of $\supset$ as the main operator, and in the second case the second occurrence. But we intend the main operator to be a *function* of the formula, i.e., every formula must have exactly one main operator occurrence.

Lemma 6.10. *The number of left and right parentheses in a formula* φ *are equal.*

Proof. We prove this by induction on the way φ is constructed. This requires two things: (a) We have to prove first that all atomic formulas have the property in question (the induction basis). (b) Then we have to prove that when we construct new formulas out of given formulas, the new formulas have the property provided the old ones do.

Let $l(\varphi)$ be the number of left parentheses, and $r(\varphi)$ the number of right parentheses in φ, and $l(t)$ and $r(t)$ similarly the number of left and right parentheses in a term t.

1. $\varphi \equiv \bot$: φ has 0 left and 0 right parentheses.
2. $\varphi \equiv R(t_1, \dots, t_n)$: $l(\varphi) = 1 + l(t_1) + \dots + l(t_n) = 1 + r(t_1) + \dots + r(t_n) = r(\varphi)$. Here we make use of the fact, left as an exercise, that $l(t) = r(t)$ for any term t.
3. $\varphi \equiv t_1 = t_2$: $l(\varphi) = l(t_1) + l(t_2) = r(t_1) + r(t_2) = r(\varphi)$.
4. $\varphi \equiv {\sim}\psi$: By induction hypothesis, $l(\psi) = r(\psi)$. Thus $l(\varphi) = l(\psi) = r(\psi) = r(\varphi)$.
5. $\varphi \equiv (\psi * \chi)$: By induction hypothesis, $l(\psi) = r(\psi)$ and $l(\chi) = r(\chi)$. Thus $l(\varphi) = 1 + l(\psi) + l(\chi) = 1 + r(\psi) + r(\chi) = r(\varphi)$.
6. $\varphi \equiv \forall x\, \psi$: By induction hypothesis, $l(\psi) = r(\psi)$. Thus, $l(\varphi) = l(\psi) = r(\psi) = r(\varphi)$.
7. $\varphi \equiv \exists x\, \psi$: Similarly. □

Definition 6.11 (Proper prefix). A string of symbols ψ is a *proper prefix* of a string of symbols φ if concatenating ψ and a non-empty string of symbols yields φ.

Lemma 6.12. *If φ is a formula, and ψ is a proper prefix of φ, then ψ is not a formula.*

Proof. Exercise. □

Proposition 6.13. *If φ is an atomic formula, then it satisfies one, and only one of the following conditions.*

1. $\varphi \equiv \bot$.
2. $\varphi \equiv R(t_1, \dots, t_n)$ *where R is an n-place predicate symbol, $t_1, \dots, t_n$ are terms, and each of R, $t_1, \dots, t_n$ is uniquely determined.*
3. $\varphi \equiv t_1 = t_2$ *where t_1 and t_2 are uniquely determined terms.*

Proof. Exercise. □

Proposition 6.14 (Unique Readability). *Every formula satisfies one, and only one of the following conditions.*

1. *φ is atomic.*
2. *φ is of the form $\sim\psi$.*
3. *φ is of the form $(\psi \mathbin{\&} \chi)$.*
4. *φ is of the form $(\psi \lor \chi)$.*
5. *φ is of the form $(\psi \supset \chi)$.*
6. *φ is of the form $\forall x\, \psi$.*
7. *φ is of the form $\exists x\, \psi$.*

Moreover, in each case ψ, or ψ and χ, are uniquely determined. This means that, e.g., there are no different pairs ψ, χ and ψ', χ' so that φ is both of the form $(\psi \supset \chi)$ and $(\psi' \supset \chi')$.

Proof. The formation rules require that if a formula is not atomic, it must start with an opening parenthesis (, $\sim$, or a quantifier. On the other hand, every formula that starts with one of the following symbols must be atomic: a predicate symbol, a function symbol, a constant symbol, $\bot$.

So we really only have to show that if φ is of the form $(\psi * \chi)$ and also of the form $(\psi' *' \chi')$, then $\psi \equiv \psi'$, $\chi \equiv \chi'$, and $* = *'$.

So suppose both $\varphi \equiv (\psi * \chi)$ and $\varphi \equiv (\psi' *' \chi')$. Then either $\psi \equiv \psi'$ or not. If it is, clearly $* = *'$ and $\chi \equiv \chi'$, since they then are substrings of φ that begin in the same place and are of the same length. The other case is $\psi \not\equiv \psi'$. Since ψ and ψ' are both substrings of φ that begin at the same place, one must be a proper prefix of the other. But this is impossible by Lemma 6.12. $\square$

6.5 Main operator of a Formula

It is often useful to talk about the last operator used in constructing a formula φ. This operator is called the *main operator* of φ. Intuitively, it is the "outermost" operator of φ. For example, the main operator of $\sim\varphi$ is $\sim$, the main operator of $(\varphi \lor \psi)$ is $\lor$, etc.

Definition 6.15 (Main operator). The *main operator* of a formula φ is defined as follows:

1. φ is atomic: φ has no main operator.
2. $\varphi \equiv \sim\psi$: the main operator of φ is $\sim$.
3. $\varphi \equiv (\psi \mathbin{\&} \chi)$: the main operator of φ is &.

4. $\varphi \equiv (\psi \lor \chi)$: the main operator of φ is $\lor$.

5. $\varphi \equiv (\psi \supset \chi)$: the main operator of φ is $\supset$.

6. $\varphi \equiv \forall x\, \psi$: the main operator of φ is $\forall$.

7. $\varphi \equiv \exists x\, \psi$: the main operator of φ is $\exists$.

In each case, we intend the specific indicated *occurrence* of the main operator in the formula. For instance, since the formula $((\theta \supset \alpha) \supset (\alpha \supset \theta))$ is of the form $(\psi \supset \chi)$ where ψ is $(\theta \supset \alpha)$ and χ is $(\alpha \supset \theta)$, the second occurrence of $\supset$ is the main operator.

This is a *recursive* definition of a function which maps all non-atomic formulae to their main operator occurrence. Because of the way formulae are defined inductively, every formula φ satisfies one of the cases in Definition 6.15. This guarantees that for each non-atomic formula φ a main operator exists. Because each formula satisfies only one of these conditions, and because the smaller formulae from which φ is constructed are uniquely determined in each case, the main operator occurrence of φ is unique, and so we have defined a function.

We call formulae by the names in Table 6.1 depending on which symbol their main operator is.Recall, however, that defined operators do not officially appear in formulae. They are just abbreviations, so officially they cannot be the main operator of a formula. In proofs about all formulae they therefore do not have to be treated separately.

Main operator	Type of formula	Example
none	atomic (formula)	$\bot$, $R(t_1, \ldots, t_n)$, $t_1 = t_2$
$\sim$	negation	$\sim\varphi$
$\&$	conjunction	$(\varphi \mathbin{\&} \psi)$
$\lor$	disjunction	$(\varphi \lor \psi)$
$\supset$	conditional	$(\varphi \supset \psi)$
$\equiv$	biconditional	$(\varphi \equiv \psi)$
$\forall$	universal (formula)	$\forall x\, \varphi$
$\exists$	existential (formula)	$\exists x\, \varphi$

Table 6.1: Main operator and names of formulae

6.6 Subformulae

It is often useful to talk about the formulae that "make up" a given formula. We call these its *subformulae*. Any formula counts as a subformula of itself; a subformula of φ other than φ itself is a *proper subformula*.

Definition 6.16 (Immediate Subformula). If φ is a formula, the *immediate subformulae* of φ are defined inductively as follows:

1. Atomic formulae have no immediate subformulae.
2. $\varphi \equiv {\sim}\psi$: The only immediate subformula of φ is ψ.
3. $\varphi \equiv (\psi * \chi)$: The immediate subformulae of φ are ψ and χ ($*$ is any one of the two-place connectives).
4. $\varphi \equiv \forall x\, \psi$: The only immediate subformula of φ is ψ.
5. $\varphi \equiv \exists x\, \psi$: The only immediate subformula of φ is ψ.

Definition 6.17 (Proper Subformula). If φ is a formula, the *proper subformulae* of φ are defined recursively as follows:

1. Atomic formulae have no proper subformulae.
2. $\varphi \equiv {\sim}\psi$: The proper subformulae of φ are ψ together with all proper subformulae of ψ.
3. $\varphi \equiv (\psi * \chi)$: The proper subformulae of φ are ψ, χ, together with all proper subformulae of ψ and those of χ.
4. $\varphi \equiv \forall x\, \psi$: The proper subformulae of φ are ψ together with all proper subformulae of ψ.
5. $\varphi \equiv \exists x\, \psi$: The proper subformulae of φ are ψ together with all proper subformulae of ψ.

Definition 6.18 (Subformula). The subformulae of φ are φ itself together with all its proper subformulae.

Note the subtle difference in how we have defined immediate subformulae and proper subformulae. In the first case, we have directly defined the immediate subformulae of a formula φ for each possible form of φ. It is an explicit definition by cases, and the cases mirror the inductive definition of the set of formulae. In the second case, we have also mirrored the way the set of all formulae is defined, but in each case we have also included the proper subformulae of the smaller formulae ψ, χ in addition to these formulae themselves. This makes the definition *recursive*. In general, a definition of a function on an inductively defined set (in our case, formulae) is recursive if the cases in the definition of the function make use of the function itself. To be well defined, we must make sure, however, that we only ever use the values of the function for arguments that come "before" the one we are defining—in our case, when defining "proper subformula" for $(\psi * \chi)$ we only use the proper subformulae of the "earlier" formulae ψ and χ.

Proposition 6.19. *Suppose ψ is a subformula of φ and χ is a subformula of ψ. Then χ is a subformula of φ. In other words, the subformula relation is transitive.*

Proposition 6.20. *Suppose φ is a formula with n connectives and quantifiers. Then φ has at most $2n + 1$ subformulas.*

6.7 Formation Sequences

Defining formulae via an inductive definition, and the complementary technique of proving properties of formulae via induction, is an elegant and efficient approach. However, it can also be useful to consider a more bottom-up, step-by-step approach to the construction of formulae, which we do here using the notion of a *formation sequence*. To show how terms and formulae can be introduced in this way without needing to refer to their inductive definitions, we first introduce the notion of an arbitrary string of symbols drawn from some language $\mathcal{L}$.

Definition 6.21 (Strings). Suppose $\mathcal{L}$ is a first-order language. An $\mathcal{L}$-*string* is a finite sequence of symbols of $\mathcal{L}$. Where the language $\mathcal{L}$ is clearly fixed by the context, we will often refer to a $\mathcal{L}$-string simply as a *string*.

Example 6.22. For any first-order language $\mathcal{L}$, all $\mathcal{L}$-formulae are $\mathcal{L}$-strings, but not conversely. For example,

$$)(v_0 \supset \exists$$

is an $\mathcal{L}$-string but not an $\mathcal{L}$-formula.

Definition 6.23 (Formation sequences for terms). A finite sequence of $\mathcal{L}$-strings $\langle t_0, \ldots, t_n \rangle$ is a *formation sequence* for a term t if $t \equiv t_n$ and for all $i \leq n$, either t_i is a variable or a constant symbol, or $\mathcal{L}$ contains a k-ary function symbol f and there exist $m_0, \ldots, m_k < i$ such that $t_i \equiv f(t_{m_0}, \ldots, t_{m_k})$.

Example 6.24. The sequence

$$\langle c_0, v_0, f_0^2(c_0, v_0), f_0^1(f_0^2(c_0, v_0)) \rangle$$

is a formation sequence for the term $f_0^1(f_0^2(c_0, v_0))$, as is

$$\langle v_0, c_0, f_0^2(c_0, v_0), f_0^1(f_0^2(c_0, v_0)) \rangle.$$

Definition 6.25 (Formation sequences for formulas). A finite sequence of $\mathcal{L}$-strings $\langle \varphi_0, \ldots, \varphi_n \rangle$ is a *formation sequence* for φ if $\varphi \equiv \varphi_n$ and for all $i \leq n$, either φ_i is an atomic formula or there exist $j, k < i$ and a variable x such that one of the following holds:

1. $\varphi_i \equiv {\sim}\varphi_j$.
2. $\varphi_i \equiv (\varphi_j \mathbin{\&} \varphi_k)$.
3. $\varphi_i \equiv (\varphi_j \vee \varphi_k)$.
4. $\varphi_i \equiv (\varphi_j \supset \varphi_k)$.

5. $\varphi_i \equiv \forall x\, \varphi_j$.
6. $\varphi_i \equiv \exists x\, \varphi_j$.

Example 6.26.

$$\langle A^1_0(v_0), A^1_1(c_1), (A^1_1(c_1) \mathbin{\&} A^1_0(v_0)), \exists v_0\, (A^1_1(c_1) \mathbin{\&} A^1_0(v_0))\rangle$$

is a formation sequence of $\exists v_0\, (A^1_1(c_1) \mathbin{\&} A^1_0(v_0))$, as is

$$\langle A^1_0(v_0), A^1_1(c_1), (A^1_1(c_1) \mathbin{\&} A^1_0(v_0)), A^1_1(c_1), \\ \forall v_1\, A^1_0(v_0), \exists v_0\, (A^1_1(c_1) \mathbin{\&} A^1_0(v_0))\rangle.$$

As can be seen from the second example, formation sequences may contain "junk": formulae which are redundant or do not contribute to the construction.

Proposition 6.27. *Every formula φ in $\mathrm{Frm}(\mathcal{L})$ has a formation sequence.*

Proof. Suppose φ is atomic. Then the sequence $\langle\varphi\rangle$ is a formation sequence for φ. Now suppose that ψ and χ have formation sequences $\langle\psi_0, \dots, \psi_n\rangle$ and $\langle\chi_0, \dots, \chi_m\rangle$ respectively.

1. If $\varphi \equiv {\sim}\psi$, then $\langle\psi_0, \dots, \psi_n, {\sim}\psi_n\rangle$ is a formation sequence for φ.
2. If $\varphi \equiv (\psi \mathbin{\&} \chi)$, then $\langle\psi_0, \dots, \psi_n, \chi_0, \dots, \chi_m, (\psi_n \mathbin{\&} \chi_m)\rangle$ is a formation sequence for φ.
3. If $\varphi \equiv (\psi \lor \chi)$, then $\langle\psi_0, \dots, \psi_n, \chi_0, \dots, \chi_m, (\psi_n \lor \chi_m)\rangle$ is a formation sequence for φ.
4. If $\varphi \equiv (\psi \supset \chi)$, then $\langle\psi_0, \dots, \psi_n, \chi_0, \dots, \chi_m, (\psi_n \supset \chi_m)\rangle$ is a formation sequence for φ.
5. If $\varphi \equiv \forall x\, \psi$, then $\langle\psi_0, \dots, \psi_n, \forall x\, \psi_n\rangle$ is a formation sequence for φ.
6. If $\varphi \equiv \exists x\, \psi$, then $\langle\psi_0, \dots, \psi_n, \exists x\, \psi_n\rangle$ is a formation sequence for φ.

By the principle of induction on formulae, every formula has a formation sequence. □

We can also prove the converse. This is important because it shows that our two ways of defining formulas are equivalent: they give the same results. It also means that we can prove theorems about formulas by using ordinary induction on the length of formation sequences.

Lemma 6.28. *Suppose that $\langle\varphi_0, \dots, \varphi_n\rangle$ is a formation sequence for φ_n, and that $k \le n$. Then $\langle\varphi_0, \dots, \varphi_k\rangle$ is a formation sequence for φ_k.*

Proof. Exercise. □

Theorem 6.29. $\mathrm{Frm}(\mathcal{L})$ *is the set of all expressions (strings of symbols) in the language* $\mathcal{L}$ *with a formation sequence.*

Proof. Let F be the set of all strings of symbols in the language $\mathcal{L}$ that have a formation sequence. We have seen in Proposition 6.27 that $\mathrm{Frm}(\mathcal{L}) \subseteq F$, so now we prove the converse.

Suppose φ has a formation sequence $\langle \varphi_0, \dots, \varphi_n \rangle$. We prove that $\varphi \in \mathrm{Frm}(\mathcal{L})$ by strong induction on n. Our induction hypothesis is that every string of symbols with a formation sequence of length $m < n$ is in $\mathrm{Frm}(\mathcal{L})$. By the definition of a formation sequence, either $\varphi \equiv \varphi_n$ is atomic or there must exist $j, k < n$ such that one of the following is the case:

1. $\varphi \equiv {\sim}\varphi_j$.
2. $\varphi \equiv (\varphi_j \mathbin{\&} \varphi_k)$.
3. $\varphi \equiv (\varphi_j \lor \varphi_k)$.
4. $\varphi \equiv (\varphi_j \supset \varphi_k)$.
5. $\varphi \equiv \forall x\, \varphi_j$.
6. $\varphi \equiv \exists x\, \varphi_j$.

Now we reason by cases. If φ is atomic then $\varphi_n \in \mathrm{Frm}(\mathcal{L}_0)$. Suppose instead that $\varphi \equiv (\varphi_j \mathbin{\&} \varphi_k)$. By Lemma 6.28, $\langle \varphi_0, \dots, \varphi_j \rangle$ and $\langle \varphi_0, \dots, \varphi_k \rangle$ are formation sequences for φ_j and φ_k, respectively. Since these are proper initial subsequences of the formation sequence for φ, they both have length less than n. Therefore by the induction hypothesis, φ_j and φ_k are in $\mathrm{Frm}(\mathcal{L}_0)$, and by the definition of a formula, so is $(\varphi_j \mathbin{\&} \varphi_k)$. The other cases follow by parallel reasoning. □

Formation sequences for terms have similar properties to those for formulae.

Proposition 6.30. $\mathrm{Trm}(\mathcal{L})$ *is the set of all expressions* t *in the language* $\mathcal{L}$ *such that there exists a (term) formation sequence for* t*.*

Proof. Exercise. □

There are two types of "junk" that can appear in formation sequences: repeated elements, and elements that are irrelevant to the construction of the formation or term. We can eliminate both by looking at minimal formation sequences.

Definition 6.31 (Minimal formation sequences). A formation sequence $\langle \varphi_0, \dots, \varphi_n \rangle$ for φ is a *minimal formation sequence* for φ if for every other formation sequence s for φ, the length of s is greater than or equal to $n + 1$.

Proposition 6.32. *The following are equivalent:*

1. *ψ is a sub-formula of φ.*
2. *ψ occurs in every formation sequence of φ.*
3. *ψ occurs in a minimal formation sequence of φ.*

Proof. Exercise. □

Historical Remarks Formation sequences were introduced by Raymond Smullyan in his textbook *First-Order Logic* (Smullyan, 1968). Additional properties of formation sequences were established by Zuckerman (1973).

6.8 Free Variables and Sentences

Definition 6.33 (Free occurrences of a variable). The *free* occurrences of a variable in a formula are defined inductively as follows:

1. φ is atomic: all variable occurrences in φ are free.
2. $\varphi \equiv {\sim}\psi$: the free variable occurrences of φ are exactly those of ψ.
3. $\varphi \equiv (\psi * \chi)$: the free variable occurrences of φ are those in ψ together with those in χ.
4. $\varphi \equiv \forall x\, \psi$: the free variable occurrences in φ are all of those in ψ except for occurrences of x.
5. $\varphi \equiv \exists x\, \psi$: the free variable occurrences in φ are all of those in ψ except for occurrences of x.

Definition 6.34 (Bound Variables). An occurrence of a variable in a formula φ is *bound* if it is not free.

Definition 6.35 (Scope). If $\forall x\, \psi$ is an occurrence of a subformula in a formula φ, then the corresponding occurrence of ψ in φ is called the *scope* of the corresponding occurrence of $\forall x$. Similarly for $\exists x$.

If ψ is the scope of a quantifier occurrence $\forall x$ or $\exists x$ in φ, then the free occurrences of x in ψ are bound in $\forall x\, \psi$ and $\exists x\, \psi$. We say that these occurrences are *bound by* the mentioned quantifier occurrence.

Example 6.36. Consider the following formula:

$$\exists v_0 \underbrace{A_0^2(v_0, v_1)}_{\psi}$$

ψ represents the scope of $\exists v_0$. The quantifier binds the occurrence of v_0 in ψ, but does not bind the occurrence of v_1. So v_1 is a free variable in this case.

We can now see how this might work in a more complicated formula φ:

$$\forall v_0 \underbrace{(A_0^1(v_0) \supset A_0^2(v_0, v_1))}_{\psi} \supset \exists v_1 \underbrace{(A_1^2(v_0, v_1) \lor \forall v_0 \overbrace{\sim A_1^1(v_0)}^{\theta})}_{\chi}$$

ψ is the scope of the first $\forall v_0$, χ is the scope of $\exists v_1$, and θ is the scope of the second $\forall v_0$. The first $\forall v_0$ binds the occurrences of v_0 in ψ, $\exists v_1$ binds the occurrence of v_1 in χ, and the second $\forall v_0$ binds the occurrence of v_0 in θ. The first occurrence of v_1 and the fourth occurrence of v_0 are free in φ. The last occurrence of v_0 is free in θ, but bound in χ and φ.

Definition 6.37 (Sentence). A formula φ is a *sentence* iff it contains no free occurrences of variables.

6.9 Substitution

Definition 6.38 (Substitution in a term). We define $s[t/x]$, the result of *substituting* t for every occurrence of x in s, recursively:

1. $s \equiv c$: $s[t/x]$ is just s.
2. $s \equiv y$: $s[t/x]$ is also just s, provided y is a variable and $y \not\equiv x$.
3. $s \equiv x$: $s[t/x]$ is t.
4. $s \equiv f(t_1, \ldots, t_n)$: $s[t/x]$ is $f(t_1[t/x], \ldots, t_n[t/x])$.

Definition 6.39. A term t is *free for* x in φ if none of the free occurrences of x in φ occur in the scope of a quantifier that binds a variable in t.

Example 6.40.

1. v_8 is free for v_1 in $\exists v_3 A_4^2(v_3, v_1)$
2. $f_1^2(v_1, v_2)$ is *not* free for v_0 in $\forall v_2 A_4^2(v_0, v_2)$

Definition 6.41 (Substitution in a formula). If φ is a formula, x is a variable, and t is a term free for x in φ, then $\varphi[t/x]$ is the result of substituting t for all free occurrences of x in φ.

1. $\varphi \equiv \bot$: $\varphi[t/x]$ is $\bot$.
2. $\varphi \equiv P(t_1, \dots, t_n)$: $\varphi[t/x]$ is $P(t_1[t/x], \dots, t_n[t/x])$.
3. $\varphi \equiv t_1 = t_2$: $\varphi[t/x]$ is $t_1[t/x] = t_2[t/x]$.
4. $\varphi \equiv {\sim}\psi$: $\varphi[t/x]$ is ${\sim}\psi[t/x]$.
5. $\varphi \equiv (\psi \mathbin{\&} \chi)$: $\varphi[t/x]$ is $(\psi[t/x] \mathbin{\&} \chi[t/x])$.
6. $\varphi \equiv (\psi \lor \chi)$: $\varphi[t/x]$ is $(\psi[t/x] \lor \chi[t/x])$.
7. $\varphi \equiv (\psi \supset \chi)$: $\varphi[t/x]$ is $(\psi[t/x] \supset \chi[t/x])$.
8. $\varphi \equiv \forall y\, \psi$: $\varphi[t/x]$ is $\forall y\, \psi[t/x]$, provided y is a variable other than x; otherwise $\varphi[t/x]$ is just φ.
9. $\varphi \equiv \exists y\, \psi$: $\varphi[t/x]$ is $\exists y\, \psi[t/x]$, provided y is a variable other than x; otherwise $\varphi[t/x]$ is just φ.

Note that substitution may be vacuous: If x does not occur in φ at all, then $\varphi[t/x]$ is just φ.

The restriction that t must be free for x in φ is necessary to exclude cases like the following. If $\varphi \equiv \exists y\, x < y$ and $t \equiv y$, then $\varphi[t/x]$ would be $\exists y\, y < y$. In this case the free variable y is "captured" by the quantifier $\exists y$ upon substitution, and that is undesirable. For instance, we would like it to be the case that whenever $\forall x\, \psi$ holds, so does $\psi[t/x]$. But consider $\forall x\, \exists y\, x < y$ (here ψ is $\exists y\, x < y$). It is a sentence that is true about, e.g., the natural numbers: for every number x there is a number y greater than it. If we allowed y as a possible substitution for x, we would end up with $\psi[y/x] \equiv \exists y\, y < y$, which is false. We prevent this by requiring that none of the free variables in t would end up being bound by a quantifier in φ.

We often use the following convention to avoid cumbersome notation: If φ is a formula which may contain the variable x free, we also write $\varphi(x)$ to indicate this. When it is clear which φ and x we have in mind, and t is a term (assumed to be free for x in $\varphi(x)$), then we write $\varphi(t)$ as short for $\varphi[t/x]$. So for instance, we might say, "we call $\varphi(t)$ an instance of $\forall x\, \varphi(x)$." By this we mean that if φ is any formula, x a variable, and t a term that's free for x in φ, then $\varphi[t/x]$ is an instance of $\forall x\, \varphi$.

Problems

Problem 6.1. Prove Lemma 6.8.

Problem 6.2. Prove that for any term t, $l(t) = r(t)$.

Problem 6.3. Prove Lemma 6.12.

Problem 6.4. Prove Proposition 6.13 (Hint: Formulate and prove a version of Lemma 6.12 for terms.)

Problem 6.5. Prove Proposition 6.19.

Problem 6.6. Prove Proposition 6.20.

Problem 6.7. Prove Lemma 6.28.

Problem 6.8. Prove Proposition 6.30. Hint: use a similar strategy to that used in the proof of Theorem 6.29.

Problem 6.9. Prove Proposition 6.32.

Problem 6.10. Give an inductive definition of the bound variable occurrences along the lines of Definition 6.33.

Chapter 7

Semantics of First-Order Logic

7.1 Introduction

Giving the meaning of expressions is the domain of semantics. The central concept in semantics is that of satisfaction in a structure. A structure gives meaning to the building blocks of the language: a domain is a non-empty set of objects. The quantifiers are interpreted as ranging over this domain, constant symbols are assigned elements in the domain, function symbols are assigned functions from the domain to itself, and predicate symbols are assigned relations on the domain. The domain together with assignments to the basic vocabulary constitutes a structure. Variables may appear in formulae, and in order to give a semantics, we also have to assign elements of the domain to them—this is a variable assignment. The satisfaction relation, finally, brings these together. A formula may be satisfied in a structure $\mathfrak{M}$ relative to a variable assignment s, written as $\mathfrak{M}, s \vDash \varphi$. This relation is also defined by induction on the structure of φ, using the truth tables for the logical connectives to define, say, satisfaction of $(\varphi \mathbin{\&} \psi)$ in terms of satisfaction (or not) of φ and ψ. It then turns out that the variable assignment is irrelevant if the formula φ is a sentence, i.e., has no free variables, and so we can talk of sentences being simply satisfied (or not) in structures.

On the basis of the satisfaction relation $\mathfrak{M} \vDash \varphi$ for sentences we can then define the basic semantic notions of validity, entailment, and satisfiability. A sentence is valid, $\vDash \varphi$, if every structure satisfies it. It is entailed by a set of sentences, $\Gamma \vDash \varphi$, if every structure that satisfies all the sentences in Γ also satisfies φ. And a set of sentences is satisfiable if some structure satisfies all sentences in it at the same time. Because formulae are inductively defined, and satisfaction is in turn defined by induction on the structure of formulae, we can use induction to prove properties of our semantics and to relate the semantic notions defined.

7.2 Structures for First-order Languages

First-order languages are, by themselves, *uninterpreted:* the constant symbols, function symbols, and predicate symbols have no specific meaning attached to them. Meanings are given by specifying a *structure*. It specifies the *domain*, i.e., the objects which the constant symbols pick out, the function symbols operate on, and the quantifiers range over. In addition, it specifies which constant symbols pick out which objects, how a function symbol maps objects to objects, and which objects the predicate symbols apply to. Structures are the basis for *semantic* notions in logic, e.g., the notion of consequence, validity, satisfiability. They are variously called "structures," "interpretations," or "models" in the literature.

Definition 7.1 (Structures). A *structure* $\mathfrak{M}$, for a language $\mathcal{L}$ of first-order logic consists of the following elements:

1. *Domain:* a non-empty set, $|\mathfrak{M}|$
2. *Interpretation of constant symbols:* for each constant symbol c of $\mathcal{L}$, an element $c^{\mathfrak{M}} \in |\mathfrak{M}|$
3. *Interpretation of predicate symbols:* for each n-place predicate symbol R of $\mathcal{L}$ (other than $=$), an n-place relation $R^{\mathfrak{M}} \subseteq |\mathfrak{M}|^n$
4. *Interpretation of function symbols:* for each n-place function symbol f of $\mathcal{L}$, an n-place function $f^{\mathfrak{M}} \colon |\mathfrak{M}|^n \to |\mathfrak{M}|$

Example 7.2. A structure $\mathfrak{M}$ for the language of arithmetic consists of a set, an element of $|\mathfrak{M}|$, $\mathsf{o}^{\mathfrak{M}}$, as interpretation of the constant symbol o, a one-place function $\prime^{\mathfrak{M}} \colon |\mathfrak{M}| \to |\mathfrak{M}|$, two two-place functions $+^{\mathfrak{M}}$ and $\times^{\mathfrak{M}}$, both $|\mathfrak{M}|^2 \to |\mathfrak{M}|$, and a two-place relation $<^{\mathfrak{M}} \subseteq |\mathfrak{M}|^2$.

An obvious example of such a structure is the following:

1. $|\mathfrak{N}| = \mathbb{N}$
2. $\mathsf{o}^{\mathfrak{N}} = 0$
3. $\prime^{\mathfrak{N}}(n) = n + 1$ for all $n \in \mathbb{N}$
4. $+^{\mathfrak{N}}(n, m) = n + m$ for all $n, m \in \mathbb{N}$
5. $\times^{\mathfrak{N}}(n, m) = n \cdot m$ for all $n, m \in \mathbb{N}$
6. $<^{\mathfrak{N}} = \{\langle n, m \rangle \mid n \in \mathbb{N}, m \in \mathbb{N}, n < m\}$

The structure $\mathfrak{N}$ for $\mathcal{L}_A$ so defined is called the *standard model of arithmetic*, because it interprets the non-logical constants of $\mathcal{L}_A$ exactly how you would expect.

However, there are many other possible structures for $\mathcal{L}_A$. For instance, we might take as the domain the set $\mathbb{Z}$ of integers instead of $\mathbb{N}$, and define the interpretations of o, $\prime$, $+$, $\times$, $<$ accordingly. But we can also define structures for $\mathcal{L}_A$ which have nothing even remotely to do with numbers.

Example 7.3. A structure $\mathfrak{M}$ for the language $\mathcal{L}_Z$ of set theory requires just a set and a single-two place relation. So technically, e.g., the set of people plus the relation "x is older than y" could be used as a structure for $\mathcal{L}_Z$, as well as $\mathbb{N}$ together with $n \geq m$ for $n, m \in \mathbb{N}$.

A particularly interesting structure for $\mathcal{L}_Z$ in which the elements of the domain are actually sets, and the interpretation of $\in$ actually is the relation "x is an element of y" is the structure $\mathfrak{HF}$ of *hereditarily finite sets*:

1. $|\mathfrak{HF}| = \emptyset \cup \wp(\emptyset) \cup \wp(\wp(\emptyset)) \cup \wp(\wp(\wp(\emptyset))) \cup \dots$;
2. $\in^{\mathfrak{HF}} = \{\langle x, y \rangle \mid x, y \in |\mathfrak{HF}|, x \in y\}$.

The stipulations we make as to what counts as a structure impact our logic. For example, the choice to prevent empty domains ensures, given the usual account of satisfaction (or truth) for quantified sentences, that $\exists x\,(\varphi(x) \lor {\sim}\varphi(x))$ is valid—that is, a logical truth. And the stipulation that all constant symbols must refer to an object in the domain ensures that the existential generalization is a sound pattern of inference: $\varphi(a)$, therefore $\exists x\,\varphi(x)$. If we allowed names to refer outside the domain, or to not refer, then we would be on our way to a *free logic*, in which existential generalization requires an additional premise: $\varphi(a)$ and $\exists x\, x = a$, therefore $\exists x\,\varphi(x)$.

7.3 Covered Structures for First-order Languages

Recall that a term is *closed* if it contains no variables.

Definition 7.4 (Value of closed terms). If t is a closed term of the language $\mathcal{L}$ and $\mathfrak{M}$ is a structure for $\mathcal{L}$, the *value* $\mathrm{Val}^{\mathfrak{M}}(t)$ is defined as follows:

1. If t is just the constant symbol c, then $\mathrm{Val}^{\mathfrak{M}}(c) = c^{\mathfrak{M}}$.
2. If t is of the form $f(t_1, \dots, t_n)$, then
$$\mathrm{Val}^{\mathfrak{M}}(t) = f^{\mathfrak{M}}(\mathrm{Val}^{\mathfrak{M}}(t_1), \dots, \mathrm{Val}^{\mathfrak{M}}(t_n)).$$

Definition 7.5 (Covered structure). A structure is *covered* if every element of the domain is the value of some closed term.

Example 7.6. Let $\mathcal{L}$ be the language with constant symbols zero, one, two, $\dots$, the binary predicate symbol $<$, and the binary function symbols $+$ and $\times$. Then a structure $\mathfrak{M}$ for $\mathcal{L}$ is the one with domain $|\mathfrak{M}| = \{0, 1, 2, \dots\}$ and assignments $zero^{\mathfrak{M}} = 0$, $one^{\mathfrak{M}} = 1$, $two^{\mathfrak{M}} = 2$, and so forth. For the binary relation symbol $<$, the set $<^{\mathfrak{M}}$ is the set of all pairs $\langle c_1, c_2 \rangle \in |\mathfrak{M}|^2$ such that c_1 is less than c_2: for example, $\langle 1, 3 \rangle \in <^{\mathfrak{M}}$ but $\langle 2, 2 \rangle \notin <^{\mathfrak{M}}$. For the binary function symbol $+$, define $+^{\mathfrak{M}}$ in the usual way—for example, $+^{\mathfrak{M}}(2, 3)$ maps to 5, and similarly for the binary function symbol $\times$. Hence, the value of $four$ is just 4, and the value of $\times(two, +(three, zero))$ (or in infix notation, $two \times (three + zero)$) is

$$\begin{aligned}
\mathrm{Val}^{\mathfrak{M}}(\times(two, +(three, zero)) = \\
&= \times^{\mathfrak{M}}(\mathrm{Val}^{\mathfrak{M}}(two), \mathrm{Val}^{\mathfrak{M}}(+(three, zero))) \\
&= \times^{\mathfrak{M}}(\mathrm{Val}^{\mathfrak{M}}(two), +^{\mathfrak{M}}(\mathrm{Val}^{\mathfrak{M}}(three), \mathrm{Val}^{\mathfrak{M}}(zero))) \\
&= \times^{\mathfrak{M}}(two^{\mathfrak{M}}, +^{\mathfrak{M}}(three^{\mathfrak{M}}, zero^{\mathfrak{M}})) \\
&= \times^{\mathfrak{M}}(2, +^{\mathfrak{M}}(3, 0)) \\
&= \times^{\mathfrak{M}}(2, 3) \\
&= 6
\end{aligned}$$

7.4 Satisfaction of a Formula in a Structure

The basic notion that relates expressions such as terms and formulae, on the one hand, and structures on the other, are those of *value* of a term and *satisfaction* of a formula. Informally, the value of a term is an element of a structure—if the term is just a constant, its value is the object assigned to the constant by the structure, and if it is built up using function symbols, the value is computed from the values of constants and the functions assigned to the functions in the term. A formula is *satisfied* in a structure if the interpretation given to the predicates makes the formula true in the domain of the structure. This notion of satisfaction is specified inductively: the specification of the structure directly states when atomic formulae are satisfied, and we define when a complex formula is satisfied depending on the main connective or quantifier and whether or not the immediate subformulae are satisfied.

The case of the quantifiers here is a bit tricky, as the immediate subformula of a quantified formula has a free variable, and structures don't specify the values of variables. In order to deal with this difficulty, we also introduce *variable assignments* and define satisfaction not with respect to a structure alone, but with respect to a structure plus a variable assignment.

Definition 7.7 (Variable Assignment). A *variable assignment s* for a structure $\mathfrak{M}$ is a function which maps each variable to an element of $|\mathfrak{M}|$, i.e., $s\colon \mathrm{Var} \to |\mathfrak{M}|$.

A structure assigns a value to each constant symbol, and a variable assignment to each variable. But we want to use terms built up from them to also name elements of the domain. For this we define the value of terms inductively. For constant symbols and variables the value is just as the structure or the variable assignment specifies it; for more complex terms it is computed recursively using the functions the structure assigns to the function symbols.

Definition 7.8 (Value of Terms). If t is a term of the language $\mathcal{L}$, $\mathfrak{M}$ is a structure for $\mathcal{L}$, and s is a variable assignment for $\mathfrak{M}$, the *value* $\mathrm{Val}_s^{\mathfrak{M}}(t)$ is defined as follows:

1. $t \equiv c$: $\mathrm{Val}_s^{\mathfrak{M}}(t) = c^{\mathfrak{M}}$.
2. $t \equiv x$: $\mathrm{Val}_s^{\mathfrak{M}}(t) = s(x)$.
3. $t \equiv f(t_1, \dots, t_n)$:
$$\mathrm{Val}_s^{\mathfrak{M}}(t) = f^{\mathfrak{M}}(\mathrm{Val}_s^{\mathfrak{M}}(t_1), \dots, \mathrm{Val}_s^{\mathfrak{M}}(t_n)).$$

Definition 7.9 (x-Variant). If s is a variable assignment for a structure $\mathfrak{M}$, then any variable assignment s' for $\mathfrak{M}$ which differs from s at most in what it assigns to x is called an *x-variant* of s. If s' is an x-variant of s we write $s' \sim_x s$.

Note that an x-variant of an assignment s does not *have* to assign something different to x. In fact, every assignment counts as an x-variant of itself.

Definition 7.10. If s is a variable assignment for a structure $\mathfrak{M}$ and $m \in |\mathfrak{M}|$, then the assignment $s[m/x]$ is the variable assignment defined by
$$s[m/x](y) = \begin{cases} m & \text{if } y \equiv x \\ s(y) & \text{otherwise.} \end{cases}$$

In other words, $s[m/x]$ is the particular x-variant of s which assigns the domain element m to x, and assigns the same things to variables other than x that s does.

Definition 7.11 (Satisfaction). Satisfaction of a formula φ in a structure $\mathfrak{M}$ relative to a variable assignment s, in symbols: $\mathfrak{M}, s \vDash \varphi$, is defined recursively as follows. (We write $\mathfrak{M}, s \nvDash \varphi$ to mean "not $\mathfrak{M}, s \vDash \varphi$.")

1. $\varphi \equiv \bot$: $\mathfrak{M}, s \nvDash \varphi$.
2. $\varphi \equiv R(t_1, \dots, t_n)$: $\mathfrak{M}, s \vDash \varphi$ iff $\langle \mathrm{Val}_s^{\mathfrak{M}}(t_1), \dots, \mathrm{Val}_s^{\mathfrak{M}}(t_n) \rangle \in R^{\mathfrak{M}}$.
3. $\varphi \equiv t_1 = t_2$: $\mathfrak{M}, s \vDash \varphi$ iff $\mathrm{Val}_s^{\mathfrak{M}}(t_1) = \mathrm{Val}_s^{\mathfrak{M}}(t_2)$.
4. $\varphi \equiv {\sim}\psi$: $\mathfrak{M}, s \vDash \varphi$ iff $\mathfrak{M}, s \nvDash \psi$.
5. $\varphi \equiv (\psi \,\&\, \chi)$: $\mathfrak{M}, s \vDash \varphi$ iff $\mathfrak{M}, s \vDash \psi$ and $\mathfrak{M}, s \vDash \chi$.
6. $\varphi \equiv (\psi \lor \chi)$: $\mathfrak{M}, s \vDash \varphi$ iff $\mathfrak{M}, s \vDash \psi$ or $\mathfrak{M}, s \vDash \chi$ (or both).

7. $\varphi \equiv (\psi \supset \chi)$: $\mathfrak{M}, s \vDash \varphi$ iff $\mathfrak{M}, s \nvDash \psi$ or $\mathfrak{M}, s \vDash \chi$ (or both).

8. $\varphi \equiv \forall x\, \psi$: $\mathfrak{M}, s \vDash \varphi$ iff for every element $m \in |\mathfrak{M}|$, $\mathfrak{M}, s[m/x] \vDash \psi$.

9. $\varphi \equiv \exists x\, \psi$: $\mathfrak{M}, s \vDash \varphi$ iff for at least one element $m \in |\mathfrak{M}|$, $\mathfrak{M}, s[m/x] \vDash \psi$.

The variable assignments are important in the last two clauses. We cannot define satisfaction of $\forall x\, \psi(x)$ by "for all $m \in |\mathfrak{M}|$, $\mathfrak{M} \vDash \psi(m)$." We cannot define satisfaction of $\exists x\, \psi(x)$ by "for at least one $m \in |\mathfrak{M}|$, $\mathfrak{M} \vDash \psi(m)$." The reason is that if $m \in |\mathfrak{M}|$, it is not a symbol of the language, and so $\psi(m)$ is not a formula (that is, $\psi[m/x]$ is undefined). We also cannot assume that we have constant symbols or terms available that name every element of $\mathfrak{M}$, since there is nothing in the definition of structures that requires it. In the standard language, the set of constant symbols is countably infinite, so if $|\mathfrak{M}|$ is not countable there aren't even enough constant symbols to name every object.

We solve this problem by introducing variable assignments, which allow us to link variables directly with elements of the domain. Then instead of saying that, e.g., $\exists x\, \psi(x)$ is satisfied in $\mathfrak{M}$ iff for at least one $m \in |\mathfrak{M}|$, we say it is satisfied in $\mathfrak{M}$ *relative to s* iff $\psi(x)$ is satisfied relative to $s[m/x]$ for at least one $m \in |\mathfrak{M}|$.

Example 7.12. Let $\mathcal{L} = \{a, b, f, R\}$ where a and b are constant symbols, f is a two-place function symbol, and R is a two-place predicate symbol. Consider the structure $\mathfrak{M}$ defined by:

1. $|\mathfrak{M}| = \{1, 2, 3, 4\}$
2. $a^{\mathfrak{M}} = 1$
3. $b^{\mathfrak{M}} = 2$
4. $f^{\mathfrak{M}}(x, y) = x + y$ if $x + y \leq 3$ and $= 3$ otherwise.
5. $R^{\mathfrak{M}} = \{\langle 1, 1\rangle, \langle 1, 2\rangle, \langle 2, 3\rangle, \langle 2, 4\rangle\}$

The function $s(x) = 1$ that assigns $1 \in |\mathfrak{M}|$ to every variable is a variable assignment for $\mathfrak{M}$.

Then

$$\mathrm{Val}_s^{\mathfrak{M}}(f(a, b)) = f^{\mathfrak{M}}(\mathrm{Val}_s^{\mathfrak{M}}(a), \mathrm{Val}_s^{\mathfrak{M}}(b)).$$

Since a and b are constant symbols, $\mathrm{Val}_s^{\mathfrak{M}}(a) = a^{\mathfrak{M}} = 1$ and $\mathrm{Val}_s^{\mathfrak{M}}(b) = b^{\mathfrak{M}} = 2$. So

$$\mathrm{Val}_s^{\mathfrak{M}}(f(a, b)) = f^{\mathfrak{M}}(1, 2) = 1 + 2 = 3.$$

To compute the value of $f(f(a,b),a)$ we have to consider

$$\mathrm{Val}_s^{\mathfrak{M}}(f(f(a,b),a)) = f^{\mathfrak{M}}(\mathrm{Val}_s^{\mathfrak{M}}(f(a,b)), \mathrm{Val}_s^{\mathfrak{M}}(a)) = f^{\mathfrak{M}}(3,1) = 3,$$

since $3+1>3$. Since $s(x)=1$ and $\mathrm{Val}_s^{\mathfrak{M}}(x) = s(x)$, we also have

$$\mathrm{Val}_s^{\mathfrak{M}}(f(f(a,b),x)) = f^{\mathfrak{M}}(\mathrm{Val}_s^{\mathfrak{M}}(f(a,b)), \mathrm{Val}_s^{\mathfrak{M}}(x)) = f^{\mathfrak{M}}(3,1) = 3,$$

An atomic formula $R(t_1,t_2)$ is satisfied if the tuple of values of its arguments, i.e., $\langle \mathrm{Val}_s^{\mathfrak{M}}(t_1), \mathrm{Val}_s^{\mathfrak{M}}(t_2)\rangle$, is an element of $R^{\mathfrak{M}}$. So, e.g., we have $\mathfrak{M}, s \vDash R(b, f(a,b))$ since $\langle \mathrm{Val}^{\mathfrak{M}}(b), \mathrm{Val}^{\mathfrak{M}}(f(a,b))\rangle = \langle 2,3\rangle \in R^{\mathfrak{M}}$, but $\mathfrak{M}, s \nvDash R(x, f(a,b))$ since $\langle 1,3\rangle \notin R^{\mathfrak{M}}[s]$.

To determine if a non-atomic formula φ is satisfied, you apply the clauses in the inductive definition that applies to the main connective. For instance, the main connective in $R(a,a) \supset (R(b,x) \lor R(x,b))$ is the $\supset$, and

$$\begin{aligned}
&\mathfrak{M}, s \vDash R(a,a) \supset (R(b,x) \lor R(x,b)) \text{ iff}\\
&\qquad \mathfrak{M}, s \nvDash R(a,a) \text{ or } \mathfrak{M}, s \vDash R(b,x) \lor R(x,b)
\end{aligned}$$

Since $\mathfrak{M}, s \vDash R(a,a)$ (because $\langle 1,1\rangle \in R^{\mathfrak{M}}$) we can't yet determine the answer and must first figure out if $\mathfrak{M}, s \vDash R(b,x) \lor R(x,b)$:

$$\begin{aligned}
&\mathfrak{M}, s \vDash R(b,x) \lor R(x,b) \text{ iff}\\
&\qquad \mathfrak{M}, s \vDash R(b,x) \text{ or } \mathfrak{M}, s \vDash R(x,b)
\end{aligned}$$

And this is the case, since $\mathfrak{M}, s \vDash R(x,b)$ (because $\langle 1,2\rangle \in R^{\mathfrak{M}}$).

Recall that an x-variant of s is a variable assignment that differs from s at most in what it assigns to x. For every element of $|\mathfrak{M}|$, there is an x-variant of s:

$$\begin{aligned}
s_1 &= s[1/x], & s_2 &= s[2/x],\\
s_3 &= s[3/x], & s_4 &= s[4/x].
\end{aligned}$$

So, e.g., $s_2(x) = 2$ and $s_2(y) = s(y) = 1$ for all variables y other than x. These are all the x-variants of s for the structure $\mathfrak{M}$, since $|\mathfrak{M}| = \{1,2,3,4\}$. Note, in particular, that $s_1 = s$ (s is always an x-variant of itself).

To determine if an existentially quantified formula $\exists x\, \varphi(x)$ is satisfied, we have to determine if $\mathfrak{M}, s[m/x] \vDash \varphi(x)$ for at least one $m \in |\mathfrak{M}|$. So,

$$\mathfrak{M}, s \vDash \exists x\,(R(b,x) \lor R(x,b)),$$

since $\mathfrak{M}, s[1/x] \vDash R(b,x) \lor R(x,b)$ ($s[3/x]$ would also fit the bill). But,

$$\mathfrak{M}, s \nvDash \exists x\,(R(b,x) \mathbin{\&} R(x,b))$$

since, whichever $m \in |\mathfrak{M}|$ we pick, $\mathfrak{M}, s[m/x] \nvDash R(b,x) \mathbin{\&} R(x,b)$.

To determine if a universally quantified formula $\forall x\,\varphi(x)$ is satisfied, we have to determine if $\mathfrak{M}, s[m/x] \vDash \varphi(x)$ for all $m \in |\mathfrak{M}|$. So,

$$\mathfrak{M}, s \vDash \forall x\,(R(x,a) \supset R(a,x)),$$

since $\mathfrak{M}, s[m/x] \vDash R(x,a) \supset R(a,x)$ for all $m \in |\mathfrak{M}|$. For $m = 1$, we have $\mathfrak{M}, s[1/x] \vDash R(a,x)$ so the consequent is true; for $m = 2$, 3, and 4, we have $\mathfrak{M}, s[m/x] \nvDash R(x,a)$, so the antecedent is false. But,

$$\mathfrak{M}, s \nvDash \forall x\,(R(a,x) \supset R(x,a))$$

since $\mathfrak{M}, s[2/x] \nvDash R(a,x) \supset R(x,a)$ (because $\mathfrak{M}, s[2/x] \vDash R(a,x)$ and $\mathfrak{M}, s[2/x] \nvDash R(x,a)$).

For a more complicated case, consider

$$\forall x\,(R(a,x) \supset \exists y\,R(x,y)).$$

Since $\mathfrak{M}, s[3/x] \nvDash R(a,x)$ and $\mathfrak{M}, s[4/x] \nvDash R(a,x)$, the interesting cases where we have to worry about the consequent of the conditional are only $m = 1$ and $= 2$. Does $\mathfrak{M}, s[1/x] \vDash \exists y\,R(x,y)$ hold? It does if there is at least one $n \in |\mathfrak{M}|$ so that $\mathfrak{M}, s[1/x][n/y] \vDash R(x,y)$. In fact, if we take $n = 1$, we have $s[1/x][n/y] = s[1/y] = s$. Since $s(x) = 1$, $s(y) = 1$, and $\langle 1,1 \rangle \in R^{\mathfrak{M}}$, the answer is yes.

To determine if $\mathfrak{M}, s[2/x] \vDash \exists y\,R(x,y)$, we have to look at the variable assignments $s[2/x][n/y]$. Here, for $n = 1$, this assignment is $s_2 = s[2/x]$, which does not satisfy $R(x,y)$ ($s_2(x) = 2$, $s_2(y) = 1$, and $\langle 2,1 \rangle \notin R^{\mathfrak{M}}$). However, consider $s[2/x][3/y] = s_2[3/y]$. $\mathfrak{M}, s_2[3/y] \vDash R(x,y)$ since $\langle 2,3 \rangle \in R^{\mathfrak{M}}$, and so $\mathfrak{M}, s_2 \vDash \exists y\,R(x,y)$.

So, for all $n \in |\mathfrak{M}|$, either $\mathfrak{M}, s[m/x] \nvDash R(a,x)$ (if $m = 3, 4$) or $\mathfrak{M}, s[m/x] \vDash \exists y\,R(x,y)$ (if $m = 1, 2$), and so

$$\mathfrak{M}, s \vDash \forall x\,(R(a,x) \supset \exists y\,R(x,y)).$$

On the other hand,

$$\mathfrak{M}, s \nvDash \exists x\,(R(a,x) \mathbin{\&} \forall y\,R(x,y)).$$

We have $\mathfrak{M}, s[m/x] \vDash R(a,x)$ only for $m = 1$ and $m = 2$. But for both of these values of m, there is in turn an $n \in |\mathfrak{M}|$, namely $n = 4$, so that $\mathfrak{M}, s[m/x][n/y] \nvDash R(x,y)$ and so $\mathfrak{M}, s[m/x] \nvDash \forall y\,R(x,y)$ for $m = 1$ and $m = 2$. In sum, there is no $m \in |\mathfrak{M}|$ such that $\mathfrak{M}, s[m/x] \vDash R(a,x) \mathbin{\&} \forall y\,R(x,y)$.

7.5 Variable Assignments

A variable assignment s provides a value for *every* variable—and there are infinitely many of them. This is of course not necessary. We require variable assignments to assign values to all variables simply because it makes things a lot easier. The value of a term t, and whether or not a formula φ is satisfied in a structure with respect to s, only depend on the assignments s makes to the variables in t and the free variables of φ. This is the content of the next two propositions. To make the idea of "depends on" precise, we show that any two variable assignments that agree on all the variables in t give the same value, and that φ is satisfied relative to one iff it is satisfied relative to the other if two variable assignments agree on all free variables of φ.

Proposition 7.13. *If the variables in a term t are among $x_1, \dots, x_n$, and $s_1(x_i) = s_2(x_i)$ for $i = 1, \dots, n$, then $\mathrm{Val}^{\mathfrak{M}}_{s_1}(t) = \mathrm{Val}^{\mathfrak{M}}_{s_2}(t)$.*

Proof. By induction on the complexity of t. For the base case, t can be a constant symbol or one of the variables $x_1, \dots, x_n$. If $t = c$, then $\mathrm{Val}^{\mathfrak{M}}_{s_1}(t) = c^{\mathfrak{M}} = \mathrm{Val}^{\mathfrak{M}}_{s_2}(t)$. If $t = x_i$, $s_1(x_i) = s_2(x_i)$ by the hypothesis of the proposition, and so $\mathrm{Val}^{\mathfrak{M}}_{s_1}(t) = s_1(x_i) = s_2(x_i) = \mathrm{Val}^{\mathfrak{M}}_{s_2}(t)$.

For the inductive step, assume that $t = f(t_1, \dots, t_k)$ and that the claim holds for $t_1, \dots, t_k$. Then

$$\begin{aligned}\mathrm{Val}^{\mathfrak{M}}_{s_1}(t) &= \mathrm{Val}^{\mathfrak{M}}_{s_1}(f(t_1, \dots, t_k)) = \\ &= f^{\mathfrak{M}}(\mathrm{Val}^{\mathfrak{M}}_{s_1}(t_1), \dots, \mathrm{Val}^{\mathfrak{M}}_{s_1}(t_k))\end{aligned}$$

For $j = 1, \dots, k$, the variables of t_j are among $x_1, \dots, x_n$. By induction hypothesis, $\mathrm{Val}^{\mathfrak{M}}_{s_1}(t_j) = \mathrm{Val}^{\mathfrak{M}}_{s_2}(t_j)$. So,

$$\begin{aligned}\mathrm{Val}^{\mathfrak{M}}_{s_1}(t) &= \mathrm{Val}^{\mathfrak{M}}_{s_1}(f(t_1, \dots, t_k)) = \\ &= f^{\mathfrak{M}}(\mathrm{Val}^{\mathfrak{M}}_{s_1}(t_1), \dots, \mathrm{Val}^{\mathfrak{M}}_{s_1}(t_k)) = \\ &= f^{\mathfrak{M}}(\mathrm{Val}^{\mathfrak{M}}_{s_2}(t_1), \dots, \mathrm{Val}^{\mathfrak{M}}_{s_2}(t_k)) = \\ &= \mathrm{Val}^{\mathfrak{M}}_{s_2}(f(t_1, \dots, t_k)) = \mathrm{Val}^{\mathfrak{M}}_{s_2}(t).\end{aligned}$$

□

Proposition 7.14. *If the free variables in φ are among $x_1, \dots, x_n$, and $s_1(x_i) = s_2(x_i)$ for $i = 1, \dots, n$, then $\mathfrak{M}, s_1 \vDash \varphi$ iff $\mathfrak{M}, s_2 \vDash \varphi$.*

Proof. We use induction on the complexity of φ. For the base case, where φ is atomic, φ can be: $\bot$, $R(t_1, \dots, t_k)$ for a k-place predicate R and terms $t_1, \dots, t_k$, or $t_1 = t_2$ for terms t_1 and t_2.

1. $\varphi \equiv \bot$: both $\mathfrak{M}, s_1 \nvDash \varphi$ and $\mathfrak{M}, s_2 \nvDash \varphi$.

2. $\varphi \equiv R(t_1, \ldots, t_k)$: let $\mathfrak{M}, s_1 \vDash \varphi$. Then
$$\langle \mathrm{Val}^{\mathfrak{M}}_{s_1}(t_1), \ldots, \mathrm{Val}^{\mathfrak{M}}_{s_1}(t_k)\rangle \in R^{\mathfrak{M}}.$$
For $i = 1, \ldots, k$, $\mathrm{Val}^{\mathfrak{M}}_{s_1}(t_i) = \mathrm{Val}^{\mathfrak{M}}_{s_2}(t_i)$ by Proposition 7.13. So we also have $\langle \mathrm{Val}^{\mathfrak{M}}_{s_2}(t_i), \ldots, \mathrm{Val}^{\mathfrak{M}}_{s_2}(t_k)\rangle \in R^{\mathfrak{M}}$.

3. $\varphi \equiv t_1 = t_2$: suppose $\mathfrak{M}, s_1 \vDash \varphi$. Then $\mathrm{Val}^{\mathfrak{M}}_{s_1}(t_1) = \mathrm{Val}^{\mathfrak{M}}_{s_1}(t_2)$. So,
$$\begin{aligned} \mathrm{Val}^{\mathfrak{M}}_{s_2}(t_1) &= \mathrm{Val}^{\mathfrak{M}}_{s_1}(t_1) && \text{(by Proposition 7.13)} \\ &= \mathrm{Val}^{\mathfrak{M}}_{s_1}(t_2) && \text{(since } \mathfrak{M}, s_1 \vDash t_1 = t_2\text{)} \\ &= \mathrm{Val}^{\mathfrak{M}}_{s_2}(t_2) && \text{(by Proposition 7.13),} \end{aligned}$$
so $\mathfrak{M}, s_2 \vDash t_1 = t_2$.

Now assume $\mathfrak{M}, s_1 \vDash \psi$ iff $\mathfrak{M}, s_2 \vDash \psi$ for all formulae ψ less complex than φ. The induction step proceeds by cases determined by the main operator of φ. In each case, we only demonstrate the forward direction of the biconditional; the proof of the reverse direction is symmetrical. In all cases except those for the quantifiers, we apply the induction hypothesis to sub-formulae ψ of φ. The free variables of ψ are among those of φ. Thus, if s_1 and s_2 agree on the free variables of φ, they also agree on those of ψ, and the induction hypothesis applies to ψ.

1. $\varphi \equiv {\sim}\psi$: if $\mathfrak{M}, s_1 \vDash \varphi$, then $\mathfrak{M}, s_1 \nvDash \psi$, so by the induction hypothesis, $\mathfrak{M}, s_2 \nvDash \psi$, hence $\mathfrak{M}, s_2 \vDash \varphi$.

2. $\varphi \equiv \psi \,\&\, \chi$: if $\mathfrak{M}, s_1 \vDash \varphi$, then $\mathfrak{M}, s_1 \vDash \psi$ and $\mathfrak{M}, s_1 \vDash \chi$, so by induction hypothesis, $\mathfrak{M}, s_2 \vDash \psi$ and $\mathfrak{M}, s_2 \vDash \chi$. Hence, $\mathfrak{M}, s_2 \vDash \varphi$.

3. $\varphi \equiv \psi \lor \chi$: if $\mathfrak{M}, s_1 \vDash \varphi$, then $\mathfrak{M}, s_1 \vDash \psi$ or $\mathfrak{M}, s_1 \vDash \chi$. By induction hypothesis, $\mathfrak{M}, s_2 \vDash \psi$ or $\mathfrak{M}, s_2 \vDash \chi$, so $\mathfrak{M}, s_2 \vDash \varphi$.

4. $\varphi \equiv \psi \supset \chi$: exercise.

5. $\varphi \equiv \exists x\, \psi$: if $\mathfrak{M}, s_1 \vDash \varphi$, there is an $m \in |\mathfrak{M}|$ so that $\mathfrak{M}, s_1[m/x] \vDash \psi$. Let $s'_1 = s_1[m/x]$ and $s'_2 = s_2[m/x]$. The free variables of ψ are among x_1, …, x_n, and x. $s'_1(x_i) = s'_2(x_i)$, since s'_1 and s'_2 are x-variants of s_1 and s_2, respectively, and by hypothesis $s_1(x_i) = s_2(x_i)$. $s'_1(x) = s'_2(x) = m$ by the way we have defined s'_1 and s'_2. Then the induction hypothesis applies to ψ and s'_1, s'_2, so $\mathfrak{M}, s'_2 \vDash \psi$. Hence, since $s'_2 = s_2[m/x]$, there is an $m \in |\mathfrak{M}|$ such that $\mathfrak{M}, s_2[m/x] \vDash \psi$, and so $\mathfrak{M}, s_2 \vDash \varphi$.

6. $\varphi \equiv \forall x\, \psi$: exercise.

By induction, we get that $\mathfrak{M}, s_1 \vDash \varphi$ iff $\mathfrak{M}, s_2 \vDash \varphi$ whenever the free variables in φ are among $x_1, \ldots, x_n$ and $s_1(x_i) = s_2(x_i)$ for $i = 1, \ldots, n$. □

Sentences have no free variables, so any two variable assignments assign the same things to all the (zero) free variables of any sentence. The proposition just proved then means that whether or not a sentence is satisfied in a structure relative to a variable assignment is completely independent of the assignment. We'll record this fact. It justifies the definition of satisfaction of a sentence in a structure (without mentioning a variable assignment) that follows.

Corollary 7.15. *If φ is a sentence and s a variable assignment, then $\mathfrak{M}, s \vDash \varphi$ iff $\mathfrak{M}, s' \vDash \varphi$ for every variable assignment s'.*

Proof. Let s' be any variable assignment. Since φ is a sentence, it has no free variables, and so every variable assignment s' trivially assigns the same things to all free variables of φ as does s. So the condition of Proposition 7.14 is satisfied, and we have $\mathfrak{M}, s \vDash \varphi$ iff $\mathfrak{M}, s' \vDash \varphi$. □

Definition 7.16. If φ is a sentence, we say that a structure $\mathfrak{M}$ *satisfies* φ, $\mathfrak{M} \vDash \varphi$, iff $\mathfrak{M}, s \vDash \varphi$ for all variable assignments s.

If $\mathfrak{M} \vDash \varphi$, we also simply say that φ *is true in* $\mathfrak{M}$.

Proposition 7.17. *Let $\mathfrak{M}$ be a structure, φ be a sentence, and s a variable assignment. $\mathfrak{M} \vDash \varphi$ iff $\mathfrak{M}, s \vDash \varphi$.*

Proof. Exercise. □

Proposition 7.18. *Suppose $\varphi(x)$ only contains x free, and $\mathfrak{M}$ is a structure. Then:*

1. *$\mathfrak{M} \vDash \exists x\, \varphi(x)$ iff $\mathfrak{M}, s \vDash \varphi(x)$ for at least one variable assignment s.*

2. *$\mathfrak{M} \vDash \forall x\, \varphi(x)$ iff $\mathfrak{M}, s \vDash \varphi(x)$ for all variable assignments s.*

Proof. Exercise. □

7.6 Extensionality

Extensionality, sometimes called relevance, can be expressed informally as follows: the only factors that bear upon the satisfaction of formula φ in a structure $\mathfrak{M}$ relative to a variable assignment s, are the size of the domain and the assignments made by $\mathfrak{M}$ and s to the elements of the language that actually appear in φ.

One immediate consequence of extensionality is that where two structures $\mathfrak{M}$ and $\mathfrak{M}'$ agree on all the elements of the language appearing in a sentence φ and have the same domain, $\mathfrak{M}$ and $\mathfrak{M}'$ must also agree on whether or not φ itself is true.

Proposition 7.19 (Extensionality). *Let φ be a formula, and $\mathfrak{M}_1$ and $\mathfrak{M}_2$ be structures with $|\mathfrak{M}_1| = |\mathfrak{M}_2|$, and s a variable assignment on $|\mathfrak{M}_1| = |\mathfrak{M}_2|$. If $c^{\mathfrak{M}_1} = c^{\mathfrak{M}_2}$, $R^{\mathfrak{M}_1} = R^{\mathfrak{M}_2}$, and $f^{\mathfrak{M}_1} = f^{\mathfrak{M}_2}$ for every constant symbol c, relation symbol R, and function symbol f occurring in φ, then $\mathfrak{M}_1, s \vDash \varphi$ iff $\mathfrak{M}_2, s \vDash \varphi$.*

Proof. First prove (by induction on t) that for every term, $\mathrm{Val}_s^{\mathfrak{M}_1}(t) = \mathrm{Val}_s^{\mathfrak{M}_2}(t)$. Then prove the proposition by induction on φ, making use of the claim just proved for the induction basis (where φ is atomic). □

Corollary 7.20 (Extensionality for Sentences). *Let φ be a sentence and $\mathfrak{M}_1$, $\mathfrak{M}_2$ as in Proposition 7.19. Then $\mathfrak{M}_1 \vDash \varphi$ iff $\mathfrak{M}_2 \vDash \varphi$.*

Proof. Follows from Proposition 7.19 by Corollary 7.15. □

Moreover, the value of a term, and whether or not a structure satisfies a formula, only depend on the values of its subterms.

Proposition 7.21. *Let $\mathfrak{M}$ be a structure, t and t' terms, and s a variable assignment. Then $\mathrm{Val}_s^{\mathfrak{M}}(t[t'/x]) = \mathrm{Val}^{\mathfrak{M}}_{s[\mathrm{Val}_s^{\mathfrak{M}}(t')/x]}(t)$.*

Proof. By induction on t.

1. If t is a constant, say, $t \equiv c$, then $t[t'/x] = c$, and $\mathrm{Val}_s^{\mathfrak{M}}(c) = c^{\mathfrak{M}} = \mathrm{Val}^{\mathfrak{M}}_{s[\mathrm{Val}_s^{\mathfrak{M}}(t')/x]}(c)$.

2. If t is a variable other than x, say, $t \equiv y$, then $t[t'/x] = y$, and $\mathrm{Val}_s^{\mathfrak{M}}(y) = \mathrm{Val}^{\mathfrak{M}}_{s[\mathrm{Val}_s^{\mathfrak{M}}(t')/x]}(y)$ since $s \sim_x s[\mathrm{Val}_s^{\mathfrak{M}}(t')/x]$.

3. If $t \equiv x$, then $t[t'/x] = t'$. But $\mathrm{Val}^{\mathfrak{M}}_{s[\mathrm{Val}_s^{\mathfrak{M}}(t')/x]}(x) = \mathrm{Val}_s^{\mathfrak{M}}(t')$ by definition of $s[\mathrm{Val}_s^{\mathfrak{M}}(t')/x]$.

4. If $t \equiv f(t_1, \dots, t_n)$ then we have:

$$\begin{aligned}
&\mathrm{Val}_s^{\mathfrak{M}}(t[t'/x]) = \\
&\quad = \mathrm{Val}_s^{\mathfrak{M}}(f(t_1[t'/x], \dots, t_n[t'/x])) \\
&\qquad \text{by definition of } t[t'/x] \\
&\quad = f^{\mathfrak{M}}(\mathrm{Val}_s^{\mathfrak{M}}(t_1[t'/x]), \dots, \mathrm{Val}_s^{\mathfrak{M}}(t_n[t'/x])) \\
&\qquad \text{by definition of } \mathrm{Val}_s^{\mathfrak{M}}(f(\dots)) \\
&\quad = f^{\mathfrak{M}}(\mathrm{Val}^{\mathfrak{M}}_{s[\mathrm{Val}_s^{\mathfrak{M}}(t')/x]}(t_1), \dots, \mathrm{Val}^{\mathfrak{M}}_{s[\mathrm{Val}_s^{\mathfrak{M}}(t')/x]}(t_n)) \\
&\qquad \text{by induction hypothesis} \\
&\quad = \mathrm{Val}^{\mathfrak{M}}_{s[\mathrm{Val}_s^{\mathfrak{M}}(t')/x]}(t) \text{ by definition of } \mathrm{Val}^{\mathfrak{M}}_{s[\mathrm{Val}_s^{\mathfrak{M}}(t')/x]}(f(\dots))
\end{aligned}$$ □

Proposition 7.22. *Let $\mathfrak{M}$ be a structure, φ a formula, t' a term, and s a variable assignment. Then $\mathfrak{M}, s \vDash \varphi[t'/x]$ iff $\mathfrak{M}, s[\mathrm{Val}^{\mathfrak{M}}_s(t')/x] \vDash \varphi$.*

Proof. Exercise. $\square$

The point of Propositions 7.21 and 7.22 is the following. Suppose we have a term t or a formula φ and some term t', and we want to know the value of $t[t'/x]$ or whether or not $\varphi[t'/x]$ is satisfied in a structure $\mathfrak{M}$ relative to a variable assignment s. Then we can either perform the substitution first and then consider the value or satisfaction relative to $\mathfrak{M}$ and s, or we can first determine the value $m = \mathrm{Val}^{\mathfrak{M}}_s(t')$ of t' in $\mathfrak{M}$ relative to s, change the variable assignment to $s[m/x]$ and then consider the value of t in $\mathfrak{M}$ and $s[m/x]$, or whether $\mathfrak{M}, s[m/x] \vDash \varphi$. Propositions 7.21 and 7.22 guarantee that the answer will be the same, whichever way we do it.

7.7 Semantic Notions

Given the definition of structures for first-order languages, we can define some basic semantic properties of and relationships between sentences. The simplest of these is the notion of *validity* of a sentence. A sentence is valid if it is satisfied in every structure. Valid sentences are those that are satisfied regardless of how the non-logical symbols in it are interpreted. Valid sentences are therefore also called *logical truths*—they are true, i.e., satisfied, in any structure and hence their truth depends only on the logical symbols occurring in them and their syntactic structure, but not on the non-logical symbols or their interpretation.

Definition 7.23 (Validity). A sentence φ is *valid*, $\vDash \varphi$, iff $\mathfrak{M} \vDash \varphi$ for every structure $\mathfrak{M}$.

Definition 7.24 (Entailment). A set of sentences Γ *entails* a sentence φ, $\Gamma \vDash \varphi$, iff for every structure $\mathfrak{M}$ with $\mathfrak{M} \vDash \Gamma$, $\mathfrak{M} \vDash \varphi$.

Definition 7.25 (Satisfiability). A set of sentences Γ is *satisfiable* if $\mathfrak{M} \vDash \Gamma$ for some structure $\mathfrak{M}$. If Γ is not satisfiable it is called *unsatisfiable*.

Proposition 7.26. *A sentence φ is valid iff $\Gamma \vDash \varphi$ for every set of sentences Γ.*

Proof. For the forward direction, let φ be valid, and let Γ be a set of sentences. Let $\mathfrak{M}$ be a structure so that $\mathfrak{M} \vDash \Gamma$. Since φ is valid, $\mathfrak{M} \vDash \varphi$, hence $\Gamma \vDash \varphi$.

For the contrapositive of the reverse direction, let φ be invalid, so there is a structure $\mathfrak{M}$ with $\mathfrak{M} \nvDash \varphi$. When $\Gamma = \{\top\}$, since $\top$ is valid, $\mathfrak{M} \vDash \Gamma$. Hence, there is a structure $\mathfrak{M}$ so that $\mathfrak{M} \vDash \Gamma$ but $\mathfrak{M} \nvDash \varphi$, hence Γ does not entail φ. $\square$

Proposition 7.27. $\Gamma \vDash \varphi$ *iff* $\Gamma \cup \{{\sim}\varphi\}$ *is unsatisfiable.*

Proof. For the forward direction, suppose $\Gamma \vDash \varphi$ and suppose to the contrary that there is a structure $\mathfrak{M}$ so that $\mathfrak{M} \vDash \Gamma \cup \{\sim\varphi\}$. Since $\mathfrak{M} \vDash \Gamma$ and $\Gamma \vDash \varphi$, $\mathfrak{M} \vDash \varphi$. Also, since $\mathfrak{M} \vDash \Gamma \cup \{\sim\varphi\}$, $\mathfrak{M} \vDash \sim\varphi$, so we have both $\mathfrak{M} \vDash \varphi$ and $\mathfrak{M} \nvDash \varphi$, a contradiction. Hence, there can be no such structure $\mathfrak{M}$, so $\Gamma \cup \{\sim\varphi\}$ is unsatisfiable.

For the reverse direction, suppose $\Gamma \cup \{\sim\varphi\}$ is unsatisfiable. So for every structure $\mathfrak{M}$, either $\mathfrak{M} \nvDash \Gamma$ or $\mathfrak{M} \vDash \varphi$. Hence, for every structure $\mathfrak{M}$ with $\mathfrak{M} \vDash \Gamma$, $\mathfrak{M} \vDash \varphi$, so $\Gamma \vDash \varphi$. □

Proposition 7.28. *If $\Gamma \subseteq \Gamma'$ and $\Gamma \vDash \varphi$, then $\Gamma' \vDash \varphi$.*

Proof. Suppose that $\Gamma \subseteq \Gamma'$ and $\Gamma \vDash \varphi$. Let $\mathfrak{M}$ be a structure such that $\mathfrak{M} \vDash \Gamma'$; then $\mathfrak{M} \vDash \Gamma$, and since $\Gamma \vDash \varphi$, we get that $\mathfrak{M} \vDash \varphi$. Hence, whenever $\mathfrak{M} \vDash \Gamma'$, $\mathfrak{M} \vDash \varphi$, so $\Gamma' \vDash \varphi$. □

Theorem 7.29 (Semantic Deduction Theorem). *$\Gamma \cup \{\varphi\} \vDash \psi$ iff $\Gamma \vDash \varphi \supset \psi$.*

Proof. For the forward direction, let $\Gamma \cup \{\varphi\} \vDash \psi$ and let $\mathfrak{M}$ be a structure so that $\mathfrak{M} \vDash \Gamma$. If $\mathfrak{M} \vDash \varphi$, then $\mathfrak{M} \vDash \Gamma \cup \{\varphi\}$, so since $\Gamma \cup \{\varphi\}$ entails ψ, we get $\mathfrak{M} \vDash \psi$. Therefore, $\mathfrak{M} \vDash \varphi \supset \psi$, so $\Gamma \vDash \varphi \supset \psi$.

For the reverse direction, let $\Gamma \vDash \varphi \supset \psi$ and $\mathfrak{M}$ be a structure so that $\mathfrak{M} \vDash \Gamma \cup \{\varphi\}$. Then $\mathfrak{M} \vDash \Gamma$, so $\mathfrak{M} \vDash \varphi \supset \psi$, and since $\mathfrak{M} \vDash \varphi$, $\mathfrak{M} \vDash \psi$. Hence, whenever $\mathfrak{M} \vDash \Gamma \cup \{\varphi\}$, $\mathfrak{M} \vDash \psi$, so $\Gamma \cup \{\varphi\} \vDash \psi$. □

Proposition 7.30. *Let $\mathfrak{M}$ be a structure, and $\varphi(x)$ a formula with one free variable x, and t a closed term. Then:*

1. $\varphi(t) \vDash \exists x\, \varphi(x)$
2. $\forall x\, \varphi(x) \vDash \varphi(t)$

Proof.
1. Suppose $\mathfrak{M} \vDash \varphi(t)$. Let s be a variable assignment with $s(x) = \mathrm{Val}^{\mathfrak{M}}(t)$. Then $\mathfrak{M}, s \vDash \varphi(t)$ since $\varphi(t)$ is a sentence. By Proposition 7.22, $\mathfrak{M}, s \vDash \varphi(x)$. By Proposition 7.18, $\mathfrak{M} \vDash \exists x\, \varphi(x)$.
2. Exercise. □

Problems

Problem 7.1. Is $\mathfrak{N}$, the standard model of arithmetic, covered? Explain.

Problem 7.2. Let $\mathcal{L} = \{c, f, A\}$ with one constant symbol, one one-place function symbol and one two-place predicate symbol, and let the structure $\mathfrak{M}$ be given by

1. $|\mathfrak{M}| = \{1, 2, 3\}$

2. $c^{\mathfrak{M}} = 3$
3. $f^{\mathfrak{M}}(1) = 2, f^{\mathfrak{M}}(2) = 3, f^{\mathfrak{M}}(3) = 2$
4. $A^{\mathfrak{M}} = \{\langle 1,2\rangle, \langle 2,3\rangle, \langle 3,3\rangle\}$

(a) Let $s(v) = 1$ for all variables v. Find out whether

$$\mathfrak{M}, s \vDash \exists x\,(A(f(z), c) \supset \forall y\,(A(y,x) \lor A(f(y), x)))$$

Explain why or why not.

(b) Give a different structure and variable assignment in which the formula is not satisfied.

Problem 7.3. Complete the proof of Proposition 7.14.

Problem 7.4. Prove Proposition 7.17

Problem 7.5. Prove Proposition 7.18.

Problem 7.6. Suppose $\mathcal{L}$ is a language without function symbols. Given a structure $\mathfrak{M}$, c a constant symbol and $a \in |\mathfrak{M}|$, define $\mathfrak{M}[a/c]$ to be the structure that is just like $\mathfrak{M}$, except that $c^{\mathfrak{M}[a/c]} = a$. Define $\mathfrak{M} \mid\!\vDash \varphi$ for sentences φ by:

1. $\varphi \equiv \bot$: not $\mathfrak{M} \mid\!\vDash \varphi$.
2. $\varphi \equiv R(d_1, \dots, d_n)$: $\mathfrak{M} \mid\!\vDash \varphi$ iff $\langle d_1^{\mathfrak{M}}, \dots, d_n^{\mathfrak{M}}\rangle \in R^{\mathfrak{M}}$.
3. $\varphi \equiv d_1 = d_2$: $\mathfrak{M} \mid\!\vDash \varphi$ iff $d_1^{\mathfrak{M}} = d_2^{\mathfrak{M}}$.
4. $\varphi \equiv {\sim}\psi$: $\mathfrak{M} \mid\!\vDash \varphi$ iff not $\mathfrak{M} \mid\!\vDash \psi$.
5. $\varphi \equiv (\psi \,\&\, \chi)$: $\mathfrak{M} \mid\!\vDash \varphi$ iff $\mathfrak{M} \mid\!\vDash \psi$ and $\mathfrak{M} \mid\!\vDash \chi$.
6. $\varphi \equiv (\psi \lor \chi)$: $\mathfrak{M} \mid\!\vDash \varphi$ iff $\mathfrak{M} \mid\!\vDash \psi$ or $\mathfrak{M} \mid\!\vDash \chi$ (or both).
7. $\varphi \equiv (\psi \supset \chi)$: $\mathfrak{M} \mid\!\vDash \varphi$ iff not $\mathfrak{M} \mid\!\vDash \psi$ or $\mathfrak{M} \mid\!\vDash \chi$ (or both).
8. $\varphi \equiv \forall x\,\psi$: $\mathfrak{M} \mid\!\vDash \varphi$ iff for all $a \in |\mathfrak{M}|$, $\mathfrak{M}[a/c] \mid\!\vDash \psi[c/x]$, if c does not occur in ψ.
9. $\varphi \equiv \exists x\,\psi$: $\mathfrak{M} \mid\!\vDash \varphi$ iff there is an $a \in |\mathfrak{M}|$ such that $\mathfrak{M}[a/c] \mid\!\vDash \psi[c/x]$, if c does not occur in ψ.

Let $x_1, \dots, x_n$ be all free variables in φ, $c_1, \dots, c_n$ constant symbols not in φ, $a_1, \dots, a_n \in |\mathfrak{M}|$, and $s(x_i) = a_i$.

Show that $\mathfrak{M}, s \vDash \varphi$ iff $\mathfrak{M}[a_1/c_1, \dots, a_n/c_n] \mid\!\vDash \varphi[c_1/x_1] \dots [c_n/x_n]$.

(This problem shows that it is possible to give a semantics for first-order logic that makes do without variable assignments.)

Problem 7.7. Suppose that f is a function symbol not in $\varphi(x, y)$. Show that there is a structure $\mathfrak{M}$ such that $\mathfrak{M} \vDash \forall x\, \exists y\, \varphi(x, y)$ iff there is an $\mathfrak{M}'$ such that $\mathfrak{M}' \vDash \forall x\, \varphi(x, f(x))$.

(This problem is a special case of what's known as Skolem's Theorem; $\forall x\, \varphi(x, f(x))$ is called a *Skolem normal form* of $\forall x\, \exists y\, \varphi(x, y)$.)

Problem 7.8. Carry out the proof of Proposition 7.19 in detail.

Problem 7.9. Prove Proposition 7.22

Problem 7.10.

1. Show that $\Gamma \vDash \bot$ iff Γ is unsatisfiable.
2. Show that $\Gamma \cup \{\varphi\} \vDash \bot$ iff $\Gamma \vDash {\sim}\varphi$.
3. Suppose c does not occur in φ or Γ. Show that $\Gamma \vDash \forall x\, \varphi$ iff $\Gamma \vDash \varphi[c/x]$.

Problem 7.11. Complete the proof of Proposition 7.30.

Chapter 8

Theories and Their Models

8.1 Introduction

The development of the axiomatic method is a significant achievement in the history of science, and is of special importance in the history of mathematics. An axiomatic development of a field involves the clarification of many questions: What is the field about? What are the most fundamental concepts? How are they related? Can all the concepts of the field be defined in terms of these fundamental concepts? What laws do, and must, these concepts obey?

The axiomatic method and logic were made for each other. Formal logic provides the tools for formulating axiomatic theories, for proving theorems from the axioms of the theory in a precisely specified way, for studying the properties of all systems satisfying the axioms in a systematic way.

Definition 8.1. A set of sentences Γ is *closed* iff, whenever $\Gamma \vDash \varphi$ then $\varphi \in \Gamma$. The *closure* of a set of sentences Γ is $\{\varphi \mid \Gamma \vDash \varphi\}$.

We say that Γ is *axiomatized by* a set of sentences Δ if Γ is the closure of Δ.

We can think of an axiomatic theory as the set of sentences that is axiomatized by its set of axioms Δ. In other words, when we have a first-order language which contains non-logical symbols for the primitives of the axiomatically developed science we wish to study, together with a set of sentences that express the fundamental laws of the science, we can think of the theory as represented by all the sentences in this language that are entailed by the axioms. This ranges from simple examples with only a single primitive and simple axioms, such as the theory of partial orders, to complex theories such as Newtonian mechanics.

The important logical facts that make this formal approach to the axiomatic method so important are the following. Suppose Γ is an axiom system for a theory, i.e., a set of sentences.

1. We can state precisely when an axiom system captures an intended class of structures. That is, if we are interested in a certain class of structures, we will successfully capture that class by an axiom system Γ iff the structures are exactly those $\mathfrak{M}$ such that $\mathfrak{M} \vDash \Gamma$.

2. We may fail in this respect because there are $\mathfrak{M}$ such that $\mathfrak{M} \vDash \Gamma$, but $\mathfrak{M}$ is not one of the structures we intend. This may lead us to add axioms which are not true in $\mathfrak{M}$.

3. If we are successful at least in the respect that Γ is true in all the intended structures, then a sentence φ is true in all intended structures whenever $\Gamma \vDash \varphi$. Thus we can use logical tools (such as derivation methods) to show that sentences are true in all intended structures simply by showing that they are entailed by the axioms.

4. Sometimes we don't have intended structures in mind, but instead start from the axioms themselves: we begin with some primitives that we want to satisfy certain laws which we codify in an axiom system. One thing that we would like to verify right away is that the axioms do not contradict each other: if they do, there can be no concepts that obey these laws, and we have tried to set up an incoherent theory. We can verify that this doesn't happen by finding a model of Γ. And if there are models of our theory, we can use logical methods to investigate them, and we can also use logical methods to construct models.

5. The independence of the axioms is likewise an important question. It may happen that one of the axioms is actually a consequence of the others, and so is redundant. We can prove that an axiom φ in Γ is redundant by proving $\Gamma \setminus \{\varphi\} \vDash \varphi$. We can also prove that an axiom is not redundant by showing that $(\Gamma \setminus \{\varphi\}) \cup \{\sim\varphi\}$ is satisfiable. For instance, this is how it was shown that the parallel postulate is independent of the other axioms of geometry.

6. Another important question is that of definability of concepts in a theory: The choice of the language determines what the models of a theory consist of. But not every aspect of a theory must be represented separately in its models. For instance, every ordering $\leq$ determines a corresponding strict ordering $<$—given one, we can define the other. So it is not necessary that a model of a theory involving such an order must *also* contain the corresponding strict ordering. When is it the case, in general, that one relation can be defined in terms of others? When is it impossible to define a relation in terms of others (and hence must add it to the primitives of the language)?

8.2 Expressing Properties of Structures

It is often useful and important to express conditions on functions and relations, or more generally, that the functions and relations in a structure satisfy these conditions. For instance, we would like to have ways of distinguishing those structures for a language which "capture" what we want the predicate symbols to "mean" from those that do not. Of course we're completely free to specify which structures we "intend," e.g., we can specify that the interpretation of the predicate symbol $\leq$ must be an ordering, or that we are only interested in interpretations of $\mathcal{L}$ in which the domain consists of sets and $\in$ is interpreted by the "is an element of" relation. But can we do this with sentences of the language? In other words, which conditions on a structure $\mathfrak{M}$ can we express by a sentence (or perhaps a set of sentences) in the language of $\mathfrak{M}$? There are some conditions that we will not be able to express. For instance, there is no sentence of $\mathcal{L}_A$ which is only true in a structure $\mathfrak{M}$ if $|\mathfrak{M}| = \mathbb{N}$. We cannot express "the domain contains only natural numbers." But there are "structural properties" of structures that we perhaps can express. Which properties of structures can we express by sentences? Or, to put it another way, which collections of structures can we describe as those making a sentence (or set of sentences) true?

Definition 8.2 (Model of a set). Let Γ be a set of sentences in a language $\mathcal{L}$. We say that a structure $\mathfrak{M}$ *is a model of* Γ if $\mathfrak{M} \vDash \varphi$ for all $\varphi \in \Gamma$.

Example 8.3. The sentence $\forall x\, x \leq x$ is true in $\mathfrak{M}$ iff $\leq^{\mathfrak{M}}$ is a reflexive relation. The sentence $\forall x\, \forall y\, ((x \leq y \,\&\, y \leq x) \supset x = y)$ is true in $\mathfrak{M}$ iff $\leq^{\mathfrak{M}}$ is anti-symmetric. The sentence $\forall x\, \forall y\, \forall z\, ((x \leq y \,\&\, y \leq z) \supset x \leq z)$ is true in $\mathfrak{M}$ iff $\leq^{\mathfrak{M}}$ is transitive. Thus, the models of

$$\begin{aligned} \{ \quad & \forall x\, x \leq x, \\ & \forall x\, \forall y\, ((x \leq y \,\&\, y \leq x) \supset x = y), \\ & \forall x\, \forall y\, \forall z\, ((x \leq y \,\&\, y \leq z) \supset x \leq z) \quad \} \end{aligned}$$

are exactly those structures in which $\leq^{\mathfrak{M}}$ is reflexive, anti-symmetric, and transitive, i.e., a partial order. Hence, we can take them as axioms for the *first-order theory of partial orders*.

8.3 Examples of First-Order Theories

Example 8.4. The theory of strict linear orders in the language $\mathcal{L}_<$ is axiomatized by the set

$$\begin{aligned} \{ \quad & \forall x\, {\sim} x < x, \\ & \forall x\, \forall y\, ((x < y \lor y < x) \lor x = y), \\ & \forall x\, \forall y\, \forall z\, ((x < y \,\&\, y < z) \supset x < z) \quad \} \end{aligned}$$

It completely captures the intended structures: every strict linear order is a model of this axiom system, and vice versa, if R is a linear order on a set X, then the structure $\mathfrak{M}$ with $|\mathfrak{M}| = X$ and $<^{\mathfrak{M}} = R$ is a model of this theory.

Example 8.5. The theory of groups in the language 1 (constant symbol), $\cdot$ (two-place function symbol) is axiomatized by

$$\forall x\,(x \cdot 1) = x$$
$$\forall x\,\forall y\,\forall z\,(x \cdot (y \cdot z)) = ((x \cdot y) \cdot z)$$
$$\forall x\,\exists y\,(x \cdot y) = 1$$

Example 8.6. The theory of Peano arithmetic is axiomatized by the following sentences in the language of arithmetic $\mathcal{L}_A$.

$$\forall x\,\forall y\,(x' = y' \supset x = y)$$
$$\forall x\, 0 \neq x'$$
$$\forall x\,(x + 0) = x$$
$$\forall x\,\forall y\,(x + y') = (x + y)'$$
$$\forall x\,(x \times 0) = 0$$
$$\forall x\,\forall y\,(x \times y') = ((x \times y) + x)$$
$$\forall x\,\forall y\,(x < y \equiv \exists z\,(z' + x) = y)$$

plus all sentences of the form

$$(\varphi(0)\ \&\ \forall x\,(\varphi(x) \supset \varphi(x'))) \supset \forall x\,\varphi(x)$$

Since there are infinitely many sentences of the latter form, this axiom system is infinite. The latter form is called the *induction schema*. (Actually, the induction schema is a bit more complicated than we let on here.)

The last axiom is an *explicit definition* of $<$.

Example 8.7. The theory of pure sets plays an important role in the foundations (and in the philosophy) of mathematics. A set is pure if all its elements are also pure sets. The empty set counts therefore as pure, but a set that has something as an element that is not a set would not be pure. So the pure sets are those that are formed just from the empty set and no "urelements," i.e., objects that are not themselves sets.

The following might be considered as an axiom system for a theory of pure sets:

$$\exists x\,{\sim}\exists y\, y \in x$$
$$\forall x\,\forall y\,(\forall z(z \in x \equiv z \in y) \supset x = y)$$
$$\forall x\,\forall y\,\exists z\,\forall u\,(u \in z \equiv (u = x \lor u = y))$$
$$\forall x\,\exists y\,\forall z\,(z \in y \equiv \exists u\,(z \in u\ \&\ u \in x))$$

plus all sentences of the form

$$\exists x\, \forall y\, (y \in x \equiv \varphi(y))$$

The first axiom says that there is a set with no elements (i.e., $\emptyset$ exists); the second says that sets are extensional; the third that for any sets X and Y, the set $\{X, Y\}$ exists; the fourth that for any set X, the set $\cup X$ exists, where $\cup X$ is the union of all the elements of X.

The sentences mentioned last are collectively called the *naive comprehension scheme*. It essentially says that for every $\varphi(x)$, the set $\{x \mid \varphi(x)\}$ exists—so at first glance a true, useful, and perhaps even necessary axiom. It is called "naive" because, as it turns out, it makes this theory unsatisfiable: if you take $\varphi(y)$ to be $\sim y \in y$, you get the sentence

$$\exists x\, \forall y\, (y \in x \equiv \sim y \in y)$$

and this sentence is not satisfied in any structure.

Example 8.8. In the area of *mereology*, the relation of *parthood* is a fundamental relation. Just like theories of sets, there are theories of parthood that axiomatize various conceptions (sometimes conflicting) of this relation.

The language of mereology contains a single two-place predicate symbol P, and $P(x, y)$ "means" that x is a part of y. When we have this interpretation in mind, a structure for this language is called a *parthood structure*. Of course, not every structure for a single two-place predicate will really deserve this name. To have a chance of capturing "parthood," $P^{\mathfrak{M}}$ must satisfy some conditions, which we can lay down as axioms for a theory of parthood. For instance, parthood is a partial order on objects: every object is a part (albeit an *improper* part) of itself; no two different objects can be parts of each other; a part of a part of an object is itself part of that object. Note that in this sense "is a part of" resembles "is a subset of," but does not resemble "is an element of" which is neither reflexive nor transitive.

$$\begin{aligned}
&\forall x\, P(x, x) \\
&\forall x\, \forall y\, ((P(x, y) \,\&\, P(y, x)) \supset x = y) \\
&\forall x\, \forall y\, \forall z\, ((P(x, y) \,\&\, P(y, z)) \supset P(x, z))
\end{aligned}$$

Moreover, any two objects have a mereological sum (an object that has these two objects as parts, and is minimal in this respect).

$$\forall x\, \forall y\, \exists z\, \forall u\, (P(z, u) \equiv (P(x, u) \,\&\, P(y, u)))$$

These are only some of the basic principles of parthood considered by metaphysicians. Further principles, however, quickly become hard to formulate or write down without first introducing some defined relations. For instance, most metaphysicians interested in mereology also view the following as a valid principle: whenever an object x has a proper part y, it also has a part z that has no parts in common with y, and so that the fusion of y and z is x.

8.4 Expressing Relations in a Structure

One main use formulae can be put to is to express properties and relations in a structure $\mathfrak{M}$ in terms of the primitives of the language $\mathcal{L}$ of $\mathfrak{M}$. By this we mean the following: the domain of $\mathfrak{M}$ is a set of objects. The constant symbols, function symbols, and predicate symbols are interpreted in $\mathfrak{M}$ by some objects in $|\mathfrak{M}|$, functions on $|\mathfrak{M}|$, and relations on $|\mathfrak{M}|$. For instance, if A^2_0 is in $\mathcal{L}$, then $\mathfrak{M}$ assigns to it a relation $R = A^{2\,\mathfrak{M}}_0$. Then the formula $A^2_0(v_1, v_2)$ *expresses* that very relation, in the following sense: if a variable assignment s maps v_1 to $a \in |\mathfrak{M}|$ and v_2 to $b \in |\mathfrak{M}|$, then

$$Rab \quad \text{iff} \quad \mathfrak{M}, s \vDash A^2_0(v_1, v_2).$$

Note that we have to involve variable assignments here: we can't just say "Rab iff $\mathfrak{M} \vDash A^2_0(a, b)$" because a and b are not symbols of our language: they are elements of $|\mathfrak{M}|$.

Since we don't just have atomic formulae, but can combine them using the logical connectives and the quantifiers, more complex formulae can define other relations which aren't directly built into $\mathfrak{M}$. We're interested in how to do that, and specifically, which relations we can define in a structure.

Definition 8.9. Let $\varphi(v_1, \dots, v_n)$ be a formula of $\mathcal{L}$ in which only $v_1, \dots, v_n$ occur free, and let $\mathfrak{M}$ be a structure for $\mathcal{L}$. $\varphi(v_1, \dots, v_n)$ *expresses the relation* $R \subseteq |\mathfrak{M}|^n$ iff

$$Ra_1 \dots a_n \quad \text{iff} \quad \mathfrak{M}, s \vDash \varphi(v_1, \dots, v_n)$$

for any variable assignment s with $s(v_i) = a_i$ ($i = 1, \dots, n$).

Example 8.10. In the standard model of arithmetic $\mathfrak{N}$, the formula $v_1 < v_2 \lor v_1 = v_2$ expresses the $\leq$ relation on $\mathbb{N}$. The formula $v_2 = v_1'$ expresses the successor relation, i.e., the relation $R \subseteq \mathbb{N}^2$ where Rnm holds if m is the successor of n. The formula $v_1 = v_2'$ expresses the predecessor relation. The formulae $\exists v_3\,(v_3 \neq 0 \,\&\, v_2 = (v_1 + v_3))$ and $\exists v_3\,(v_1 + v_3') = v_2$ both express the $<$ relation. This means that the predicate symbol $<$ is actually superfluous in the language of arithmetic; it can be defined.

This idea is not just interesting in specific structures, but generally whenever we use a language to describe an intended model or models, i.e., when we consider theories. These theories often only contain a few predicate symbols as basic symbols, but in the domain they are used to describe often many other relations play an important role. If these other relations can be systematically expressed by the relations that interpret the basic predicate symbols of the language, we say we can *define* them in the language.

8.5 The Theory of Sets

Almost all of mathematics can be developed in the theory of sets. Developing mathematics in this theory involves a number of things. First, it requires a set of axioms for the relation $\in$. A number of different axiom systems have been developed, sometimes with conflicting properties of $\in$. The axiom system known as **ZFC**, Zermelo–Fraenkel set theory with the axiom of choice stands out: it is by far the most widely used and studied, because it turns out that its axioms suffice to prove almost all the things mathematicians expect to be able to prove. But before that can be established, it first is necessary to make clear how we can even *express* all the things mathematicians would like to express. For starters, the language contains no constant symbols or function symbols, so it seems at first glance unclear that we can talk about particular sets (such as $\emptyset$ or $\mathbb{N}$), can talk about operations on sets (such as $X \cup Y$ and $\wp(X)$), let alone other constructions which involve things other than sets, such as relations and functions.

To begin with, "is an element of" is not the only relation we are interested in: "is a subset of" seems almost as important. But we can *define* "is a subset of" in terms of "is an element of." To do this, we have to find a formula $\varphi(x, y)$ in the language of set theory which is satisfied by a pair of sets $\langle X, Y\rangle$ iff $X \subseteq Y$. But X is a subset of Y just in case all elements of X are also elements of Y. So we can define $\subseteq$ by the formula

$$\forall z\,(z \in x \supset z \in y)$$

Now, whenever we want to use the relation $\subseteq$ in a formula, we could instead use that formula (with x and y suitably replaced, and the bound variable z renamed if necessary). For instance, extensionality of sets means that if any sets x and y are contained in each other, then x and y must be the same set. This can be expressed by $\forall x\,\forall y\,((x \subseteq y \,\&\, y \subseteq x) \supset x = y)$, or, if we replace $\subseteq$ by the above definition, by

$$\forall x\,\forall y\,((\forall z\,(z \in x \supset z \in y) \,\&\, \forall z\,(z \in y \supset z \in x)) \supset x = y).$$

This is in fact one of the axioms of **ZFC**, the "axiom of extensionality."

There is no constant symbol for $\emptyset$, but we can express "x is empty" by $\sim\exists y\, y \in x$. Then "$\emptyset$ exists" becomes the sentence $\exists x \sim\exists y\, y \in x$. This is another axiom of **ZFC**. (Note that the axiom of extensionality implies that there is only one empty set.) Whenever we want to talk about $\emptyset$ in the language of set theory, we would write this as "there is a set that's empty and ..." As an example, to express the fact that $\emptyset$ is a subset of every set, we could write

$$\exists x\,({\sim}\exists y\, y \in x \,\&\, \forall z\, x \subseteq z)$$

where, of course, $x \subseteq z$ would in turn have to be replaced by its definition.

To talk about operations on sets, such as $X \cup Y$ and $\wp(X)$, we have to use a similar trick. There are no function symbols in the language of set theory, but we can express the functional relations $X \cup Y = Z$ and $\wp(X) = Y$ by

$$\forall u\,((u \in x \lor u \in y) \equiv u \in z)$$
$$\forall u\,(u \subseteq x \equiv u \in y)$$

since the elements of $X \cup Y$ are exactly the sets that are either elements of X or elements of Y, and the elements of $\wp(X)$ are exactly the subsets of X. However, this doesn't allow us to use $x \cup y$ or $\wp(x)$ as if they were terms: we can only use the entire formulae that define the relations $X \cup Y = Z$ and $\wp(X) = Y$. In fact, we do not know that these relations are ever satisfied, i.e., we do not know that unions and power sets always exist. For instance, the sentence $\forall x\, \exists y\, \wp(x) = y$ is another axiom of **ZFC** (the power set axiom).

Now what about talk of ordered pairs or functions? Here we have to explain how we can think of ordered pairs and functions as special kinds of sets. One way to define the ordered pair $\langle x, y\rangle$ is as the set $\{\{x\}, \{x, y\}\}$. But like before, we cannot introduce a function symbol that names this set; we can only define the relation $\langle x, y\rangle = z$, i.e., $\{\{x\}, \{x, y\}\} = z$:

$$\forall u\,(u \in z \equiv (\forall v\,(v \in u \equiv v = x) \lor \forall v\,(v \in u \equiv (v = x \lor v = y))))$$

This says that the elements u of z are exactly those sets which either have x as its only element or have x and y as its only elements (in other words, those sets that are either identical to $\{x\}$ or identical to $\{x, y\}$). Once we have this, we can say further things, e.g., that $X \times Y = Z$:

$$\forall z\,(z \in Z \equiv \exists x\, \exists y\,(x \in X \,\&\, y \in Y \,\&\, \langle x, y\rangle = z))$$

A function $f\colon X \to Y$ can be thought of as the relation $f(x) = y$, i.e., as the set of pairs $\{\langle x, y\rangle \mid f(x) = y\}$. We can then say that a set f is a function from X to Y if (a) it is a relation $\subseteq X \times Y$, (b) it is total, i.e., for all $x \in X$ there is some $y \in Y$ such that $\langle x, y\rangle \in f$ and (c) it is functional, i.e., whenever $\langle x, y\rangle, \langle x, y'\rangle \in f$, $y = y'$ (because values of functions must be unique). So "f is a function from X to Y" can be written as:

$$\begin{aligned}
&\forall u\,(u \in f \supset \exists x\, \exists y\,(x \in X \,\&\, y \in Y \,\&\, \langle x, y\rangle = u)) \,\&\\
&\forall x\,(x \in X \supset (\exists y\,(y \in Y \,\&\, \mathrm{maps}(f, x, y)) \,\&\\
&\qquad\qquad (\forall y\, \forall y'\,((\mathrm{maps}(f, x, y) \,\&\, \mathrm{maps}(f, x, y')) \supset y = y')))
\end{aligned}$$

where $\mathrm{maps}(f, x, y)$ abbreviates $\exists v\,(v \in f \,\&\, \langle x, y\rangle = v)$ (this formula expresses "$f(x) = y$").

It is now also not hard to express that $f\colon X \to Y$ is injective, for instance:

$$f\colon X \to Y \,\&\, \forall x\,\forall x'\,((x \in X \,\&\, x' \in X \,\&\, \exists y\,(\mathrm{maps}(f, x, y) \,\&\, \mathrm{maps}(f, x', y))) \supset x = x')$$

A function $f\colon X \to Y$ is injective iff, whenever f maps $x, x' \in X$ to a single y, $x = x'$. If we abbreviate this formula as $\mathrm{inj}(f, X, Y)$, we're already in a position to state in the language of set theory something as non-trivial as Cantor's theorem: there is no injective function from $\wp(X)$ to X:

$$\forall X\,\forall Y\,(\wp(X) = Y \supset {\sim}\exists f\,\mathrm{inj}(f, Y, X))$$

One might think that set theory requires another axiom that guarantees the existence of a set for every defining property. If $\varphi(x)$ is a formula of set theory with the variable x free, we can consider the sentence

$$\exists y\,\forall x\,(x \in y \equiv \varphi(x)).$$

This sentence states that there is a set y whose elements are all and only those x that satisfy $\varphi(x)$. This schema is called the "comprehension principle." It looks very useful; unfortunately it is inconsistent. Take $\varphi(x) \equiv {\sim}x \in x$, then the comprehension principle states

$$\exists y\,\forall x\,(x \in y \equiv x \notin x),$$

i.e., it states the existence of a set of all sets that are not elements of themselves. No such set can exist—this is Russell's Paradox. **ZFC**, in fact, contains a restricted—and consistent—version of this principle, the separation principle:

$$\forall z\,\exists y\,\forall x\,(x \in y \equiv (x \in z \,\&\, \varphi(x)).$$

8.6 Expressing the Size of Structures

There are some properties of structures we can express even without using the non-logical symbols of a language. For instance, there are sentences which are true in a structure iff the domain of the structure has at least, at most, or exactly a certain number n of elements.

Proposition 8.11. *The sentence*

$$\begin{aligned}\varphi_{\geq n} \equiv \exists x_1\,\exists x_2\,\ldots\,\exists x_n\\ (x_1 \neq x_2 \,\&\, x_1 \neq x_3 \,\&\, x_1 \neq x_4 \,\&\, \cdots \,\&\, x_1 \neq x_n \,\&\\ x_2 \neq x_3 \,\&\, x_2 \neq x_4 \,\&\, \cdots \,\&\, x_2 \neq x_n \,\&\\ \vdots\\ x_{n-1} \neq x_n)\end{aligned}$$

is true in a structure $\mathfrak{M}$ iff $|\mathfrak{M}|$ contains at least n elements. Consequently, $\mathfrak{M} \vDash {\sim}\varphi_{\geq n+1}$ iff $|\mathfrak{M}|$ contains at most n elements.

Proposition 8.12. *The sentence*

$$\begin{aligned}\varphi_{=n} \equiv \exists x_1\, \exists x_2 \,\ldots \exists x_n \\ (x_1 \neq x_2 \,\&\, x_1 \neq x_3 \,\&\, x_1 \neq x_4 \,\&\, \cdots \,\&\, x_1 \neq x_n \,\& \\ x_2 \neq x_3 \,\&\, x_2 \neq x_4 \,\&\, \cdots \,\&\, x_2 \neq x_n \,\& \\ \vdots \\ x_{n-1} \neq x_n \,\& \\ \forall y\, (y = x_1 \lor \cdots \lor y = x_n))\end{aligned}$$

is true in a structure $\mathfrak{M}$ iff $|\mathfrak{M}|$ contains exactly n elements.

Proposition 8.13. *A structure is infinite iff it is a model of*

$$\{\varphi_{\geq 1}, \varphi_{\geq 2}, \varphi_{\geq 3}, \ldots\}.$$

There is no single purely logical sentence which is true in $\mathfrak{M}$ iff $|\mathfrak{M}|$ is infinite. However, one can give sentences with non-logical predicate symbols which only have infinite models (although not every infinite structure is a model of them). The property of being a finite structure, and the property of being a uncountable structure cannot even be expressed with an infinite set of sentences. These facts follow from the compactness and Löwenheim–Skolem theorems.

Problems

Problem 8.1. Find formulae in $\mathcal{L}_A$ which define the following relations:

1. n is between i and j;
2. n evenly divides m (i.e., m is a multiple of n);
3. n is a prime number (i.e., no number other than 1 and n evenly divides n).

Problem 8.2. Suppose the formula $\varphi(v_1, v_2)$ expresses the relation $R \subseteq |\mathfrak{M}|^2$ in a structure $\mathfrak{M}$. Find formulas that express the following relations:

1. the inverse R^{-1} of R;
2. the relative product $R \mid R$;

Can you find a way to express R^+, the transitive closure of R?

Problem 8.3. Let $\mathcal{L}$ be the language containing a 2-place predicate symbol $<$ only (no other constant symbols, function symbols or predicate symbols— except of course $=$). Let $\mathfrak{N}$ be the structure such that $|\mathfrak{N}| = \mathbb{N}$, and $<^{\mathfrak{N}} = \{\langle n, m \rangle \mid n < m\}$. Prove the following:

1. $\{0\}$ is definable in $\mathfrak{N}$;
2. $\{1\}$ is definable in $\mathfrak{N}$;
3. $\{2\}$ is definable in $\mathfrak{N}$;
4. for each $n \in \mathbb{N}$, the set $\{n\}$ is definable in $\mathfrak{N}$;
5. every finite subset of $|\mathfrak{N}|$ is definable in $\mathfrak{N}$;
6. every co-finite subset of $|\mathfrak{N}|$ is definable in $\mathfrak{N}$ (where $X \subseteq \mathbb{N}$ is co-finite iff $\mathbb{N} \setminus X$ is finite).

Problem 8.4. Show that the comprehension principle is inconsistent by giving a derivation that shows

$$\exists y \, \forall x \, (x \in y \equiv x \notin x) \vdash \bot.$$

It may help to first show $(A \supset {\sim}A) \mathbin{\&} ({\sim}A \supset A) \vdash \bot$.

Chapter 9

Tableaux

9.1 Introduction

Logics commonly have both a semantics and a derivation system. The semantics concerns concepts such as truth, satisfiability, validity, and entailment. The purpose of derivation systems is to provide a purely syntactic method of establishing entailment and validity. They are purely syntactic in the sense that a derivation in such a system is a finite syntactic object, usually a sequence (or other finite arrangement) of sentences or formulae. Good derivation systems have the property that any given sequence or arrangement of sentences or formulae can be verified mechanically to be "correct."

The simplest (and historically first) derivation systems for first-order logic were *axiomatic*. A sequence of formulae counts as a derivation in such a system if each individual formula in it is either among a fixed set of "axioms" or follows from formulae coming before it in the sequence by one of a fixed number of "inference rules"—and it can be mechanically verified if a formula is an axiom and whether it follows correctly from other formulae by one of the inference rules. Axiomatic derivation systems are easy to describe—and also easy to handle meta-theoretically—but derivations in them are hard to read and understand, and are also hard to produce.

Other derivation systems have been developed with the aim of making it easier to construct derivations or easier to understand derivations once they are complete. Examples are natural deduction, truth trees, also known as tableaux proofs, and the sequent calculus. Some derivation systems are designed especially with mechanization in mind, e.g., the resolution method is easy to implement in software (but its derivations are essentially impossible to understand). Most of these other derivation systems represent derivations as trees of formulae rather than sequences. This makes it easier to see which parts of a derivation depend on which other parts.

So for a given logic, such as first-order logic, the different derivation systems will give different explications of what it is for a sentence to be a *theorem* and what it means for a sentence to be derivable from some others. However that is done (via axiomatic derivations, natural deductions, sequent derivations, truth trees, resolution refutations), we want these relations to match the semantic notions of validity and entailment. Let's write $\vdash \varphi$ for "φ is a theorem" and "$\Gamma \vdash \varphi$" for "φ is derivable from Γ." However $\vdash$ is defined, we want it to match up with $\vDash$, that is:

1. $\vdash \varphi$ if and only if $\vDash \varphi$

2. $\Gamma \vdash \varphi$ if and only if $\Gamma \vDash \varphi$

The "only if" direction of the above is called *soundness*. A derivation system is sound if derivability guarantees entailment (or validity). Every decent derivation system has to be sound; unsound derivation systems are not useful at all. After all, the entire purpose of a derivation is to provide a syntactic guarantee of validity or entailment. We'll prove soundness for the derivation systems we present.

The converse "if" direction is also important: it is called *completeness*. A complete derivation system is strong enough to show that φ is a theorem whenever φ is valid, and that $\Gamma \vdash \varphi$ whenever $\Gamma \vDash \varphi$. Completeness is harder to establish, and some logics have no complete derivation systems. First-order logic does. Kurt Gödel was the first one to prove completeness for a derivation system of first-order logic in his 1929 dissertation.

Another concept that is connected to derivation systems is that of *consistency*. A set of sentences is called inconsistent if anything whatsoever can be derived from it, and consistent otherwise. Inconsistency is the syntactic counterpart to unsatisfiablity: like unsatisfiable sets, inconsistent sets of sentences do not make good theories, they are defective in a fundamental way. Consistent sets of sentences may not be true or useful, but at least they pass that minimal threshold of logical usefulness. For different derivation systems the specific definition of consistency of sets of sentences might differ, but like $\vdash$, we want consistency to coincide with its semantic counterpart, satisfiability. We want it to always be the case that Γ is consistent if and only if it is satisfiable. Here, the "if" direction amounts to completeness (consistency guarantees satisfiability), and the "only if" direction amounts to soundness (satisfiability guarantees consistency). In fact, for classical first-order logic, the two versions of soundness and completeness are equivalent.

9.2 Rules and Tableaux

A tableau is a systematic survey of the possible ways a sentence can be true or false in a structure. The building blocks of a tableau are signed formulas: sentences plus a truth value "sign," either $\mathbb{T}$ or $\mathbb{F}$. These signed formulae are arranged in a (downward growing) tree.

Definition 9.1. A *signed formula* is a pair consisting of a truth value and a sentence, i.e., either:

$$\mathbb{T}\,\varphi \text{ or } \mathbb{F}\,\varphi.$$

Intuitively, we might read $\mathbb{T}\,\varphi$ as "φ might be true" and $\mathbb{F}\,\varphi$ as "φ might be false" (in some structure).

Each signed formula in the tree is either an *assumption* (which are listed at the very top of the tree), or it is obtained from a signed formula above it by one of a number of rules of inference. There are two rules for each possible main operator of the preceding formula, one for the case where the sign is $\mathbb{T}$, and one for the case where the sign is $\mathbb{F}$. Some rules allow the tree to branch, and some only add signed formulas to the branch. A rule may be (and often must be) applied not to the immediately preceding signed formula, but to any signed formula in the branch from the root to the place the rule is applied.

A branch is *closed* when it contains both $\mathbb{T}\,\varphi$ and $\mathbb{F}\,\varphi$. A closed tableau is one where every branch is closed. Under the intuitive interpretation, any branch describes a joint possibility, but $\mathbb{T}\,\varphi$ and $\mathbb{F}\,\varphi$ are not jointly possible. In other words, if a branch is closed, the possibility it describes has been ruled out. In particular, that means that a closed tableau rules out all possibilities of simultaneously making every assumption of the form $\mathbb{T}\,\varphi$ true and every assumption of the form $\mathbb{F}\,\varphi$ false.

A closed tableau *for* φ is a closed tableau with root $\mathbb{F}\,\varphi$. If such a closed tableau exists, all possibilities for φ being false have been ruled out; i.e., φ must be true in every structure.

9.3 Propositional Rules

Rules for $\sim$

$$\frac{\mathbb{T}\sim\varphi}{\mathbb{F}\,\varphi}\,\sim\mathbb{T} \qquad\qquad \frac{\mathbb{F}\sim\varphi}{\mathbb{T}\,\varphi}\,\sim\mathbb{F}$$

Rules for &

$$\frac{\mathbb{T}\,\varphi\,\&\,\psi}{\begin{array}{c}\mathbb{T}\,\varphi\\ \mathbb{T}\,\psi\end{array}}\,\&\mathbb{T} \qquad\qquad \frac{\mathbb{F}\,\varphi\,\&\,\psi}{\mathbb{F}\,\varphi \quad\mid\quad \mathbb{F}\,\psi}\,\&\mathbb{F}$$

Rules for $\lor$

$$\frac{\mathbb{T}\,\varphi \lor \psi}{\mathbb{T}\,\varphi \quad \mid \quad \mathbb{T}\,\psi}\,\lor\mathbb{T} \qquad\qquad \frac{\mathbb{F}\,\varphi \lor \psi}{\begin{array}{c}\mathbb{F}\,\varphi\\ \mathbb{F}\,\psi\end{array}}\,\lor\mathbb{F}$$

Rules for $\supset$

$$\frac{\mathbb{T}\,\varphi \supset \psi}{\mathbb{F}\,\varphi \quad \mid \quad \mathbb{T}\,\psi}\,\supset\mathbb{T} \qquad\qquad \frac{\mathbb{F}\,\varphi \supset \psi}{\begin{array}{c}\mathbb{T}\,\varphi\\ \mathbb{F}\,\psi\end{array}}\,\supset\mathbb{F}$$

The Cut Rule

$$\frac{}{\mathbb{T}\,\varphi \quad \mid \quad \mathbb{F}\,\varphi}\,\text{Cut}$$

The Cut rule is not applied "to" a previous signed formula; rather, it allows every branch in a tableau to be split in two, one branch containing $\mathbb{T}\,\varphi$, the other $\mathbb{F}\,\varphi$. It is not necessary—any set of signed formulas with a closed tableau has one not using Cut—but it allows us to combine tableaux in a convenient way.

9.4 Quantifier Rules

Rules for $\forall$

$$\frac{\mathbb{T}\forall x\,\varphi(x)}{\mathbb{T}\,\varphi(t)}\,\forall\mathbb{T} \qquad\qquad \frac{\mathbb{F}\forall x\,\varphi(x)}{\mathbb{F}\,\varphi(a)}\,\forall\mathbb{F}$$

In $\forall\mathbb{T}$, t is a closed term (i.e., one without variables). In $\forall\mathbb{F}$, a is a constant symbol which must not occur anywhere in the branch above $\forall\mathbb{F}$ rule. We call a the *eigenvariable* of the $\forall\mathbb{F}$ inference.[1]

[1] We use the term "eigenvariable" even though a in the above rule is a constant symbol. This has historical reasons.

Rules for $\exists$

$$\frac{\mathbb{T}\,\exists x\,\varphi(x)}{\mathbb{T}\,\varphi(a)}\;\exists\mathbb{T} \qquad\qquad \frac{\mathbb{F}\,\exists x\,\varphi(x)}{\mathbb{F}\,\varphi(t)}\;\exists\mathbb{F}$$

Again, t is a closed term, and a is a constant symbol which does not occur in the branch above the $\exists\mathbb{T}$ rule. We call a the *eigenvariable* of the $\exists\mathbb{T}$ inference.

The condition that an eigenvariable not occur in the branch above the $\forall\mathbb{F}$ or $\exists\mathbb{T}$ inference is called the *eigenvariable condition*.

Recall the convention that when φ is a formula with the variable x free, we indicate this by writing $\varphi(x)$. In the same context, $\varphi(t)$ then is short for $\varphi[t/x]$. So we could also write the $\exists\mathbb{F}$ rule as:

$$\frac{\mathbb{F}\,\exists x\,\varphi}{\mathbb{F}\,\varphi[t/x]}\;\exists\mathbb{F}$$

Note that t may already occur in φ, e.g., φ might be $P(t, x)$. Thus, inferring $\mathbb{F}\,P(t, t)$ from $\mathbb{F}\,\exists x\,P(t, x)$ is a correct application of $\exists\mathbb{F}$. However, the eigenvariable conditions in $\forall\mathbb{F}$ and $\exists\mathbb{T}$ require that the constant symbol a does not occur in φ. So, you cannot correctly infer $\mathbb{F}\,P(a, a)$ from $\mathbb{F}\,\forall x\,P(a, x)$ using $\forall\mathbb{F}$.

In $\forall\mathbb{T}$ and $\exists\mathbb{F}$ there are no restrictions on the term t. On the other hand, in the $\exists\mathbb{T}$ and $\forall\mathbb{F}$ rules, the eigenvariable condition requires that the constant symbol a does not occur anywhere in the branches above the respective inference. It is necessary to ensure that the system is sound. Without this condition, the following would be a closed tableau for $\exists x\,\varphi(x) \supset \forall x\,\varphi(x)$:

1.	$\mathbb{F}\,\exists x\,\varphi(x) \supset \forall x\,\varphi(x)$	Assumption
2.	$\mathbb{T}\,\exists x\,\varphi(x)$	$\supset\mathbb{F}\,1$
3.	$\mathbb{F}\,\forall x\,\varphi(x)$	$\supset\mathbb{F}\,1$
4.	$\mathbb{T}\,\varphi(a)$	$\exists\mathbb{T}\,2$
5.	$\mathbb{F}\,\varphi(a)$	$\forall\mathbb{F}\,3$
	$\otimes$	

However, $\exists x\,\varphi(x) \supset \forall x\,\varphi(x)$ is not valid.

9.5 Tableaux

We've said what an assumption is, and we've given the rules of inference. Tableaux are inductively generated from these: each tableau either is a single branch consisting of one or more assumptions, or it results from a tableau by applying one of the rules of inference on a branch.

Definition 9.2 (Tableau). A tableau for assumptions $S_1\varphi_1, \ldots, S_n\varphi_n$ (where each S_i is either $\mathbb{T}$ or $\mathbb{F}$) is a finite tree of signed formulas satisfying the following conditions:

1. The n topmost signed formulas of the tree are $S_i\varphi_i$, one below the other.

2. Every signed formula in the tree that is not one of the assumptions results from a correct application of an inference rule to a signed formula in the branch above it.

A branch of a tableau is *closed* iff it contains both $\mathbb{T}\varphi$ and $\mathbb{F}\varphi$, and *open* otherwise. A tableau in which every branch is closed is a *closed tableau* (for its set of assumptions). If a tableau is not closed, i.e., if it contains at least one open branch, it is *open*.

Example 9.3. Every set of assumptions on its own is a tableau, but it will generally not be closed. (Obviously, it is closed only if the assumptions already contain a pair of signed formulas $\mathbb{T}\varphi$ and $\mathbb{F}\varphi$.)

From a tableau (open or closed) we can obtain a new, larger one by applying one of the rules of inference to a signed formula φ in it. The rule will append one or more signed formulas to the end of any branch containing the occurrence of φ to which we apply the rule.

For instance, consider the assumption $\mathbb{T}\varphi \mathbin{\&} {\sim}\varphi$. Here is the (open) tableau consisting of just that assumption:

1.	$\mathbb{T}\varphi \mathbin{\&} {\sim}\varphi$	Assumption

We obtain a new tableau from it by applying the $\&\mathbb{T}$ rule to the assumption. That rule allows us to add two new lines to the tableau, $\mathbb{T}\varphi$ and $\mathbb{T}{\sim}\varphi$:

1.	$\mathbb{T}\varphi \mathbin{\&} {\sim}\varphi$	Assumption
2.	$\mathbb{T}\varphi$	$\&\mathbb{T}\,1$
3.	$\mathbb{T}{\sim}\varphi$	$\&\mathbb{T}\,1$

When we write down tableaux, we record the rules we've applied on the right (e.g., $\&\mathbb{T}1$ means that the signed formula on that line is the result of applying the $\&\mathbb{T}$ rule to the signed formula on line 1). This new tableau now contains additional signed formulas, but to only one ($\mathbb{T}{\sim}\varphi$) can we apply a rule (in this case, the ${\sim}\mathbb{T}$ rule). This results in the closed tableau

1.	$\mathbb{T}\varphi \mathbin{\&} {\sim}\varphi$	Assumption
2.	$\mathbb{T}\varphi$	$\&\mathbb{T}\,1$
3.	$\mathbb{T}{\sim}\varphi$	$\&\mathbb{T}\,1$
4.	$\mathbb{F}\varphi$	${\sim}\mathbb{T}\,3$
	$\otimes$	

9.6 Examples of Tableaux

Example 9.4. Let's find a closed tableau for the sentence $(\varphi \mathbin{\&} \psi) \supset \varphi$.

We begin by writing the corresponding assumption at the top of the tableau.

1.	$\mathbb{F}\,(\varphi \mathbin{\&} \psi) \supset \varphi$	Assumption

There is only one assumption, so only one signed formula to which we can apply a rule. (For every signed formula, there is always at most one rule that can be applied: it's the rule for the corresponding sign and main operator of the sentence.) In this case, this means, we must apply $\supset\mathbb{F}$.

1.	$\mathbb{F}\,(\varphi \mathbin{\&} \psi) \supset \varphi$ ✓	Assumption
2.	$\mathbb{T}\,\varphi \mathbin{\&} \psi$	$\supset\mathbb{F}\,1$
3.	$\mathbb{F}\,\varphi$	$\supset\mathbb{F}\,1$

To keep track of which signed formulas we have applied their corresponding rules to, we write a checkmark next to the sentence. However, *only* write a checkmark if the rule has been applied to all open branches. Once a signed formula has had the corresponding rule applied in every open branch, we will not have to return to it and apply the rule again. In this case, there is only one branch, so the rule only has to be applied once. (Note that checkmarks are only a convenience for constructing tableaux and are not officially part of the syntax of tableaux.)

There is one new signed formula to which we can apply a rule: the $\mathbb{T}\,\varphi \mathbin{\&} \psi$ on line 2. Applying the $\&\mathbb{T}$ rule results in:

1.	$\mathbb{F}\,(\varphi \mathbin{\&} \psi) \supset \varphi$ ✓	Assumption
2.	$\mathbb{T}\,\varphi \mathbin{\&} \psi$ ✓	$\supset\mathbb{F}\,1$
3.	$\mathbb{F}\,\varphi$	$\supset\mathbb{F}\,1$
4.	$\mathbb{T}\,\varphi$	$\&\mathbb{T}\,2$
5.	$\mathbb{T}\,\psi$	$\&\mathbb{T}\,2$
	$\otimes$	

Since the branch now contains both $\mathbb{T}\,\varphi$ (on line 4) and $\mathbb{F}\,\varphi$ (on line 3), the branch is closed. Since it is the only branch, the tableau is closed. We have found a closed tableau for $(\varphi \mathbin{\&} \psi) \supset \varphi$.

Example 9.5. Now let's find a closed tableau for $({\sim}\varphi \lor \psi) \supset (\varphi \supset \psi)$.

We begin with the corresponding assumption:

1.	$\mathbb{F}\,({\sim}\varphi \lor \psi) \supset (\varphi \supset \psi)$	Assumption

The one signed formula in this tableau has main operator $\supset$ and sign $\mathbb{F}$, so we apply the $\supset\mathbb{F}$ rule to it to obtain:

1.	$\mathbb{F}\,(\sim\varphi \lor \psi) \supset (\varphi \supset \psi)$ ✓	Assumption
2.	$\mathbb{T}\sim\varphi \lor \psi$	$\supset\mathbb{F}\,1$
3.	$\mathbb{F}\,(\varphi \supset \psi)$	$\supset\mathbb{F}\,1$

We now have a choice as to whether to apply $\lor\mathbb{T}$ to line 2 or $\supset\mathbb{F}$ to line 3. It actually doesn't matter which order we pick, as long as each signed formula has its corresponding rule applied in every branch. So let's pick the first one. The $\lor\mathbb{T}$ rule allows the tableau to branch, and the two conclusions of the rule will be the new signed formulas added to the two new branches. This results in:

1.	$\mathbb{F}\,(\sim\varphi \lor \psi) \supset (\varphi \supset \psi)$ ✓		Assumption
2.	$\mathbb{T}\sim\varphi \lor \psi$ ✓		$\supset\mathbb{F}\,1$
3.	$\mathbb{F}\,(\varphi \supset \psi)$		$\supset\mathbb{F}\,1$
4.	$\mathbb{T}\sim\varphi$	$\mathbb{T}\,\psi$	$\lor\mathbb{T}\,2$

We have not applied the $\supset\mathbb{F}$ rule to line 3 yet: let's do that now. To save time, we apply it to both branches. Recall that we write a checkmark next to a signed formula only if we have applied the corresponding rule in every open branch. So it's a good idea to apply a rule at the end of every branch that contains the signed formula the rule applies to. That way we won't have to return to that signed formula lower down in the various branches.

1.	$\mathbb{F}\,(\sim\varphi \lor \psi) \supset (\varphi \supset \psi)$ ✓		Assumption
2.	$\mathbb{T}\sim\varphi \lor \psi$ ✓		$\supset\mathbb{F}\,1$
3.	$\mathbb{F}\,(\varphi \supset \psi)$ ✓		$\supset\mathbb{F}\,1$
4.	$\mathbb{T}\sim\varphi$	$\mathbb{T}\,\psi$	$\lor\mathbb{T}\,2$
5.	$\mathbb{T}\,\varphi$	$\mathbb{T}\,\varphi$	$\supset\mathbb{F}\,3$
6.	$\mathbb{F}\,\psi$	$\mathbb{F}\,\psi$	$\supset\mathbb{F}\,3$
		$\otimes$	

The right branch is now closed. On the left branch, we can still apply the $\sim\mathbb{T}$ rule to line 4. This results in $\mathbb{F}\,\varphi$ and closes the left branch:

1.	$\mathbb{F}\,(\sim\varphi \lor \psi) \supset (\varphi \supset \psi)$ ✓		Assumption
2.	$\mathbb{T}\sim\varphi \lor \psi$ ✓		$\supset\mathbb{F}\,1$
3.	$\mathbb{F}\,(\varphi \supset \psi)$ ✓		$\supset\mathbb{F}\,1$
4.	$\mathbb{T}\sim\varphi$	$\mathbb{T}\psi$	$\lor\mathbb{T}\,2$
5.	$\mathbb{T}\varphi$	$\mathbb{T}\varphi$	$\supset\mathbb{F}\,3$
6.	$\mathbb{F}\psi$	$\mathbb{F}\psi$	$\supset\mathbb{F}\,3$
7.	$\mathbb{F}\varphi$	$\otimes$	$\sim\mathbb{T}\,4$
	$\otimes$		

Example 9.6. We can give tableaux for any number of signed formulas as assumptions. Often it is also necessary to apply more than one rule that allows branching; and in general a tableau can have any number of branches. For instance, consider a tableau for $\{\mathbb{T}\varphi \lor (\psi \,\&\, \chi), \mathbb{F}\,(\varphi \lor \psi) \,\&\, (\varphi \lor \chi)\}$. We start by applying the $\lor\mathbb{T}$ to the first assumption:

1.	$\mathbb{T}\varphi \lor (\psi \,\&\, \chi)$ ✓		Assumption
2.	$\mathbb{F}\,(\varphi \lor \psi) \,\&\, (\varphi \lor \chi)$		Assumption
3.	$\mathbb{T}\varphi$	$\mathbb{T}\psi \,\&\, \chi$	$\lor\mathbb{T}\,1$

Now we can apply the $\&\mathbb{F}$ rule to line 2. We do this on both branches simultaneously, and can therefore check off line 2:

1.	$\mathbb{T}\varphi \lor (\psi \,\&\, \chi)$ ✓				Assumption
2.	$\mathbb{F}\,(\varphi \lor \psi) \,\&\, (\varphi \lor \chi)$ ✓				Assumption
3.	$\mathbb{T}\varphi$		$\mathbb{T}\psi \,\&\, \chi$		$\lor\mathbb{T}\,1$
4.	$\mathbb{F}\varphi \lor \psi$	$\mathbb{F}\varphi \lor \chi$	$\mathbb{F}\varphi \lor \psi$	$\mathbb{F}\varphi \lor \chi$	$\&\mathbb{F}\,2$

Now we can apply $\lor\mathbb{F}$ to all the branches containing $\varphi \lor \psi$:

1.	$\mathbb{T}\varphi \lor (\psi \,\&\, \chi)$ ✓				Assumption
2.	$\mathbb{F}\,(\varphi \lor \psi) \,\&\, (\varphi \lor \chi)$ ✓				Assumption
3.	$\mathbb{T}\varphi$		$\mathbb{T}\psi \,\&\, \chi$		$\lor\mathbb{T}\,1$
4.	$\mathbb{F}\varphi \lor \psi$ ✓	$\mathbb{F}\varphi \lor \chi$	$\mathbb{F}\varphi \lor \psi$ ✓	$\mathbb{F}\varphi \lor \chi$	$\&\mathbb{F}\,2$
5.	$\mathbb{F}\varphi$		$\mathbb{F}\varphi$		$\lor\mathbb{F}\,4$
6.	$\mathbb{F}\psi$		$\mathbb{F}\psi$		$\lor\mathbb{F}\,4$
	$\otimes$				

The leftmost branch is now closed. Let's now apply $\lor\mathbb{F}$ to $\varphi \lor \chi$:

1.	$\mathbb{T}\,\varphi \lor (\psi \mathbin{\&} \chi)$ ✓				Assumption
2.	$\mathbb{F}\,(\varphi \lor \psi) \mathbin{\&} (\varphi \lor \chi)$ ✓				Assumption
3.	$\mathbb{T}\,\varphi$		$\mathbb{T}\,\psi \mathbin{\&} \chi$		$\lor\mathbb{T}\,1$
4.	$\mathbb{F}\,\varphi \lor \psi$ ✓	$\mathbb{F}\,\varphi \lor \chi$ ✓	$\mathbb{F}\,\varphi \lor \psi$ ✓	$\mathbb{F}\,\varphi \lor \chi$ ✓	$\&\mathbb{F}\,2$
5.	$\mathbb{F}\,\varphi$		$\mathbb{F}\,\varphi$		$\lor\mathbb{F}\,4$
6.	$\mathbb{F}\,\psi$		$\mathbb{F}\,\psi$		$\lor\mathbb{F}\,4$
7.	$\otimes$	$\mathbb{F}\,\varphi$		$\mathbb{F}\,\varphi$	$\lor\mathbb{F}\,4$
8.		$\mathbb{F}\,\chi$		$\mathbb{F}\,\chi$	$\lor\mathbb{F}\,4$
		$\otimes$			

Note that we moved the result of applying $\lor\mathbb{F}$ a second time below for clarity. In this instance it would not have been needed, since the justifications would have been the same.

Two branches remain open, and $\mathbb{T}\,\psi \mathbin{\&} \chi$ on line 3 remains unchecked. We apply $\&\mathbb{T}$ to it to obtain a closed tableau:

1.	$\mathbb{T}\,\varphi \lor (\psi \mathbin{\&} \chi)$ ✓				Assumption
2.	$\mathbb{F}\,(\varphi \lor \psi) \mathbin{\&} (\varphi \lor \chi)$ ✓				Assumption
3.	$\mathbb{T}\,\varphi$		$\mathbb{T}\,\psi \mathbin{\&} \chi$ ✓		$\lor\mathbb{T}\,1$
4.	$\mathbb{F}\,\varphi \lor \psi$ ✓	$\mathbb{F}\,\varphi \lor \chi$ ✓	$\mathbb{F}\,\varphi \lor \psi$ ✓	$\mathbb{F}\,\varphi \lor \chi$ ✓	$\&\mathbb{F}\,2$
5.	$\mathbb{F}\,\varphi$	$\mathbb{F}\,\varphi$	$\mathbb{F}\,\varphi$	$\mathbb{F}\,\varphi$	$\lor\mathbb{F}\,4$
6.	$\mathbb{F}\,\psi$	$\mathbb{F}\,\chi$	$\mathbb{F}\,\psi$	$\mathbb{F}\,\chi$	$\lor\mathbb{F}\,4$
7.	$\otimes$	$\otimes$	$\mathbb{T}\,\psi$	$\mathbb{T}\,\psi$	$\&\mathbb{T}\,3$
8.			$\mathbb{T}\,\chi$	$\mathbb{T}\,\chi$	$\&\mathbb{T}\,3$
			$\otimes$	$\otimes$	

For comparison, here's a closed tableau for the same set of assumptions in which the rules are applied in a different order:

1.	$\mathbb{T}\,\varphi \lor (\psi \,\&\, \chi)$ ✓				Assumption
2.	$\mathbb{F}\,(\varphi \lor \psi) \,\&\, (\varphi \lor \chi)$ ✓				Assumption
3.	$\mathbb{F}\,\varphi \lor \psi$ ✓		$\mathbb{F}\,\varphi \lor \chi$ ✓		$\&\mathbb{F}\,2$
4.	$\mathbb{F}\,\varphi$		$\mathbb{F}\,\varphi$		$\lor\mathbb{F}\,3$
5.	$\mathbb{F}\,\psi$		$\mathbb{F}\,\chi$		$\lor\mathbb{F}\,3$
6.	$\mathbb{T}\,\varphi$	$\mathbb{T}\,\psi \,\&\, \chi$ ✓	$\mathbb{T}\,\varphi$	$\mathbb{T}\,\psi \,\&\, \chi$ ✓	$\lor\mathbb{T}\,1$
7.	$\otimes$	$\mathbb{T}\,\psi$	$\otimes$	$\mathbb{T}\,\psi$	$\&\mathbb{T}\,6$
8.		$\mathbb{T}\,\chi$		$\mathbb{T}\,\chi$	$\&\mathbb{T}\,6$
		$\otimes$		$\otimes$	

9.7 Tableaux with Quantifiers

Example 9.7. When dealing with quantifiers, we have to make sure not to violate the eigenvariable condition, and sometimes this requires us to play around with the order of carrying out certain inferences. In general, it helps to try and take care of rules subject to the eigenvariable condition first (they will be higher up in the finished tableau).

Let's see how we'd give a tableau for the sentence $\exists x \sim\varphi(x) \supset \sim\forall x\, \varphi(x)$. Starting as usual, we start by recording the assumption,

1.	$\mathbb{F}\,\exists x \sim\varphi(x) \supset \sim\forall x\, \varphi(x)$	Assumption

Since the main operator is $\supset$, we apply the $\supset\mathbb{F}$:

1.	$\mathbb{F}\,\exists x \sim\varphi(x) \supset \sim\forall x\, \varphi(x)$ ✓	Assumption
2.	$\mathbb{T}\,\exists x \sim\varphi(x)$	$\supset\mathbb{F}\,1$
3.	$\mathbb{F}\sim\forall x\, \varphi(x)$	$\supset\mathbb{F}\,1$

The next line to deal with is 2. We use $\exists\mathbb{T}$. This requires a new constant symbol; since no constant symbols yet occur, we can pick any one, say, a.

1.	$\mathbb{F}\,\exists x \sim\varphi(x) \supset \sim\forall x\, \varphi(x)$ ✓	Assumption
2.	$\mathbb{T}\,\exists x \sim\varphi(x)$ ✓	$\supset\mathbb{F}\,1$
3.	$\mathbb{F}\sim\forall x\, \varphi(x)$	$\supset\mathbb{F}\,1$
4.	$\mathbb{T}\sim\varphi(a)$	$\exists\mathbb{T}\,2$

Now we apply $\sim\mathbb{F}$ to line 3:

1.	$\mathbb{F}\,\exists x \sim\varphi(x) \supset \sim\forall x\, \varphi(x)$ ✓	Assumption
2.	$\mathbb{T}\,\exists x \sim\varphi(x)$ ✓	$\supset\mathbb{F}\,1$
3.	$\mathbb{F}\sim\forall x\, \varphi(x)$ ✓	$\supset\mathbb{F}\,1$
4.	$\mathbb{T}\sim\varphi(a)$	$\exists\mathbb{T}\,2$
5.	$\mathbb{T}\forall x\, \varphi(x)$	$\sim\mathbb{F}\,3$

We obtain a closed tableau by applying $\sim\mathbb{T}$ to line 4, followed by $\forall\mathbb{T}$ to line 5.

1.	$\mathbb{F}\,\exists x\,{\sim}\varphi(x) \supset {\sim}\forall x\,\varphi(x)$ ✓	Assumption
2.	$\mathbb{T}\exists x\,{\sim}\varphi(x)$ ✓	$\supset\mathbb{F}\,1$
3.	$\mathbb{F}\,{\sim}\forall x\,\varphi(x)$ ✓	$\supset\mathbb{F}\,1$
4.	$\mathbb{T}{\sim}\varphi(a)$	$\exists\mathbb{T}\,2$
5.	$\mathbb{T}\forall x\,\varphi(x)$	${\sim}\mathbb{F}\,3$
6.	$\mathbb{F}\,\varphi(a)$	${\sim}\mathbb{T}\,4$
7.	$\mathbb{T}\,\varphi(a)$	$\forall\mathbb{T}\,5$
	$\otimes$	

Example 9.8. Let's see how we'd give a tableau for the set

$$\mathbb{F}\,\exists x\,\chi(x,b), \mathbb{T}\exists x\,(\varphi(x)\;\&\;\psi(x)), \mathbb{T}\forall x\,(\psi(x) \supset \chi(x,b)).$$

Starting as usual, we start with the assumptions:

1.	$\mathbb{F}\,\exists x\,\chi(x,b)$	Assumption
2.	$\mathbb{T}\exists x\,(\varphi(x)\;\&\;\psi(x))$	Assumption
3.	$\mathbb{T}\forall x\,(\psi(x) \supset \chi(x,b))$	Assumption

We should always apply a rule with the eigenvariable condition first; in this case that would be $\exists\mathbb{T}$ to line 2. Since the assumptions contain the constant symbol b, we have to use a different one; let's pick a again.

1.	$\mathbb{F}\,\exists x\,\chi(x,b)$	Assumption
2.	$\mathbb{T}\exists x\,(\varphi(x)\;\&\;\psi(x))$ ✓	Assumption
3.	$\mathbb{T}\forall x\,(\psi(x) \supset \chi(x,b))$	Assumption
4.	$\mathbb{T}\,\varphi(a)\;\&\;\psi(a)$	$\exists\mathbb{T}\,2$

If we now apply $\exists\mathbb{F}$ to line 1 or $\forall\mathbb{T}$ to line 3, we have to decide which term t to substitute for x. Since there is no eigenvariable condition for these rules, we can pick any term we like. In some cases we may even have to apply the rule several times with different ts. But as a general rule, it pays to pick one of the terms already occurring in the tableau—in this case, a and b—and in this case we can guess that a will be more likely to result in a closed branch.

1.	$\mathbb{F}\,\exists x\,\chi(x,b)$	Assumption
2.	$\mathbb{T}\exists x\,(\varphi(x)\;\&\;\psi(x))$ ✓	Assumption
3.	$\mathbb{T}\forall x\,(\psi(x) \supset \chi(x,b))$	Assumption
4.	$\mathbb{T}\,\varphi(a)\;\&\;\psi(a)$	$\exists\mathbb{T}\,2$
5.	$\mathbb{F}\,\chi(a,b)$	$\exists\mathbb{F}\,1$
6.	$\mathbb{T}\,\psi(a) \supset \chi(a,b)$	$\forall\mathbb{T}\,3$

We don't check the signed formulas in lines 1 and 3, since we may have to use them again. Now apply $\&\mathbb{T}$ to line 4:

$$
\begin{array}{lcl}
1. & \mathbb{F}\,\exists x\,\chi(x,b) & \text{Assumption} \\
2. & \mathbb{T}\exists x\,(\varphi(x)\mathbin{\&}\psi(x)) \checkmark & \text{Assumption} \\
3. & \mathbb{T}\forall x\,(\psi(x)\supset\chi(x,b)) & \text{Assumption} \\
4. & \mathbb{T}\,\varphi(a)\mathbin{\&}\psi(a) \checkmark & \exists\mathbb{T}\,2 \\
5. & \mathbb{F}\,\chi(a,b) & \exists\mathbb{F}\,1 \\
6. & \mathbb{T}\,\psi(a)\supset\chi(a,b) & \forall\mathbb{T}\,3 \\
7. & \mathbb{T}\,\varphi(a) & \&\mathbb{T}\,4 \\
8. & \mathbb{T}\,\psi(a) & \&\mathbb{T}\,4
\end{array}
$$

If we now apply $\supset\mathbb{T}$ to line 6, the tableau closes:

$$
\begin{array}{lcl}
1. & \mathbb{F}\,\exists x\,\chi(x,b) & \text{Assumption} \\
2. & \mathbb{T}\exists x\,(\varphi(x)\mathbin{\&}\psi(x)) \checkmark & \text{Assumption} \\
3. & \mathbb{T}\forall x\,(\psi(x)\supset\chi(x,b)) & \text{Assumption} \\
4. & \mathbb{T}\,\varphi(a)\mathbin{\&}\psi(a) \checkmark & \exists\mathbb{T}\,2 \\
5. & \mathbb{F}\,\chi(a,b) & \exists\mathbb{F}\,1 \\
6. & \mathbb{T}\,\psi(a)\supset\chi(a,b) \checkmark & \forall\mathbb{T}\,3 \\
7. & \mathbb{T}\,\varphi(a) & \&\mathbb{T}\,4 \\
8. & \mathbb{T}\,\psi(a) & \&\mathbb{T}\,4 \\
9. & \begin{array}{cc} \mathbb{F}\,\psi(a) & \mathbb{T}\,\chi(a,b) \\ \otimes & \otimes \end{array} & \supset\mathbb{T}\,6
\end{array}
$$

Example 9.9. We construct a tableau for the set

$$\mathbb{T}\forall x\,\varphi(x), \mathbb{T}\forall x\,\varphi(x)\supset\exists y\,\psi(y), \mathbb{T}{\sim}\exists y\,\psi(y).$$

Starting as usual, we write down the assumptions:

$$
\begin{array}{lcl}
1. & \mathbb{T}\forall x\,\varphi(x) & \text{Assumption} \\
2. & \mathbb{T}\forall x\,\varphi(x)\supset\exists y\,\psi(y) & \text{Assumption} \\
3. & \mathbb{T}{\sim}\exists y\,\psi(y) & \text{Assumption}
\end{array}
$$

We begin by applying the ${\sim}\mathbb{T}$ rule to line 3. A corollary to the rule "always apply rules with eigenvariable conditions first" is "defer applying quantifier rules without eigenvariable conditions until needed." Also, defer rules that result in a split.

$$
\begin{array}{lcl}
1. & \mathbb{T}\forall x\,\varphi(x) & \text{Assumption} \\
2. & \mathbb{T}\forall x\,\varphi(x)\supset\exists y\,\psi(y) & \text{Assumption} \\
3. & \mathbb{T}{\sim}\exists y\,\psi(y) \checkmark & \text{Assumption} \\
4. & \mathbb{F}\,\exists y\,\psi(y) & {\sim}\mathbb{T}\,3
\end{array}
$$

The new line 4 requires $\exists\mathbb{F}$, a quantifier rule without the eigenvariable condition. So we defer this in favor of using $\supset\mathbb{T}$ on line 2.

$$
\begin{array}{llll}
1. & & \mathbb{T}\forall x\, \varphi(x) & \text{Assumption} \\
2. & & \mathbb{T}\forall x\, \varphi(x) \supset \exists y\, \psi(y) \checkmark & \text{Assumption} \\
3. & & \mathbb{T}{\sim}\exists y\, \psi(y) \checkmark & \text{Assumption} \\
4. & & \mathbb{F}\,\exists y\, \psi(y) & {\sim}\mathbb{T}\,3 \\
5. & \mathbb{F}\forall x\, \varphi(x) & \mathbb{T}\exists y\, \psi(y) & \supset\mathbb{T}\,2
\end{array}
$$

Both new signed formulas require rules with eigenvariable conditions, so these should be next:

$$
\begin{array}{llll}
1. & & \mathbb{T}\forall x\, \varphi(x) & \text{Assumption} \\
2. & & \mathbb{T}\forall x\, \varphi(x) \supset \exists y\, \psi(y) \checkmark & \text{Assumption} \\
3. & & \mathbb{T}{\sim}\exists y\, \psi(y) \checkmark & \text{Assumption} \\
4. & & \mathbb{F}\,\exists y\, \psi(y) & {\sim}\mathbb{T}\,3 \\
5. & \mathbb{F}\forall x\, \varphi(x) \checkmark & \mathbb{T}\exists y\, \psi(y) \checkmark & \supset\mathbb{T}\,2 \\
6. & \mathbb{F}\,\varphi(b) & \mathbb{T}\,\psi(c) & \forall\mathbb{F}\,5;\ \exists\mathbb{T}\,5
\end{array}
$$

To close the branches, we have to use the signed formulas on lines 1 and 3. The corresponding rules ($\forall\mathbb{T}$ and $\exists\mathbb{F}$) don't have eigenvariable conditions, so we are free to pick whichever terms are suitable. In this case, that's b and c, respectively.

$$
\begin{array}{llll}
1. & & \mathbb{T}\forall x\, \varphi(x) & \text{Assumption} \\
2. & & \mathbb{T}\forall x\, \varphi(x) \supset \exists y\, \psi(y) \checkmark & \text{Assumption} \\
3. & & \mathbb{T}{\sim}\exists y\, \psi(y) \checkmark & \text{Assumption} \\
4. & & \mathbb{F}\,\exists y\, \psi(y) & {\sim}\mathbb{T}\,3 \\
5. & \mathbb{F}\forall x\, \varphi(x) \checkmark & \mathbb{T}\exists y\, \psi(y) \checkmark & \supset\mathbb{T}\,2 \\
6. & \mathbb{F}\,\varphi(b) & \mathbb{T}\,\psi(c) & \forall\mathbb{F}\,5;\ \exists\mathbb{T}\,5 \\
7. & \mathbb{T}\,\varphi(b) & \mathbb{F}\,\psi(c) & \forall\mathbb{T}\,1;\ \exists\mathbb{F}\,4 \\
 & \otimes & \otimes &
\end{array}
$$

9.8 Proof-Theoretic Notions

Just as we've defined a number of important semantic notions (validity, entailment, satisfiability), we now define corresponding *proof-theoretic notions*. These are not defined by appeal to satisfaction of sentences in structures, but by appeal to the existence of certain closed tableaux. It was an important discovery that these notions coincide. That they do is the content of the *soundness* and *completeness theorems*.

Definition 9.10 (Theorems). A sentence φ is a *theorem* if there is a closed tableau for $\mathbb{F}\,\varphi$. We write $\vdash \varphi$ if φ is a theorem and $\nvdash \varphi$ if it is not.

Definition 9.11 (Derivability). A sentence φ is *derivable from* a set of sentences Γ, $\Gamma \vdash \varphi$ iff there is a finite set $\{\psi_1, \dots, \psi_n\} \subseteq \Gamma$ and a closed tableau for the set

$$\{\mathbb{F}\,\varphi, \mathbb{T}\,\psi_1, \dots, \mathbb{T}\,\psi_n\}.$$

If φ is not derivable from Γ we write $\Gamma \nvdash \varphi$.

Definition 9.12 (Consistency). A set of sentences Γ is *inconsistent* iff there is a finite set $\{\psi_1, \dots, \psi_n\} \subseteq \Gamma$ and a closed tableau for the set

$$\{\mathbb{T}\,\psi_1, \dots, \mathbb{T}\,\psi_n\}.$$

If Γ is not inconsistent, we say it is *consistent*.

Proposition 9.13 (Reflexivity). *If $\varphi \in \Gamma$, then $\Gamma \vdash \varphi$.*

Proof. If $\varphi \in \Gamma$, $\{\varphi\}$ is a finite subset of Γ and the tableau

$$\begin{array}{lll} 1. & \mathbb{F}\,\varphi & \text{Assumption} \\ 2. & \mathbb{T}\,\varphi & \text{Assumption} \\ & \otimes & \end{array}$$

is closed. □

Proposition 9.14 (Monotonicity). *If $\Gamma \subseteq \Delta$ and $\Gamma \vdash \varphi$, then $\Delta \vdash \varphi$.*

Proof. Any finite subset of Γ is also a finite subset of Δ. □

Proposition 9.15 (Transitivity). *If $\Gamma \vdash \varphi$ and $\{\varphi\} \cup \Delta \vdash \psi$, then $\Gamma \cup \Delta \vdash \psi$.*

Proof. If $\{\varphi\} \cup \Delta \vdash \psi$, then there is a finite subset $\Delta_0 = \{\chi_1, \dots, \chi_n\} \subseteq \Delta$ such that

$$\{\mathbb{F}\,\psi, \mathbb{T}\,\varphi, \mathbb{T}\,\chi_1, \dots, \mathbb{T}\,\chi_n\}$$

has a closed tableau. If $\Gamma \vdash \varphi$ then there are $\theta_1, \dots, \theta_m$ such that

$$\{\mathbb{F}\,\varphi, \mathbb{T}\,\theta_1, \dots, \mathbb{T}\,\theta_m\}$$

has a closed tableau.

Now consider the tableau with assumptions

$$\mathbb{F}\,\psi, \mathbb{T}\,\chi_1, \dots, \mathbb{T}\,\chi_n, \mathbb{T}\,\theta_1, \dots, \mathbb{T}\,\theta_m.$$

Apply the Cut rule on φ. This generates two branches, one has $\mathbb{T}\,\varphi$ in it, the other $\mathbb{F}\,\varphi$. Thus, on the one branch, all of

$$\{\mathbb{F}\,\psi, \mathbb{T}\,\varphi, \mathbb{T}\,\chi_1, \dots, \mathbb{T}\,\chi_n\}$$

are available. Since there is a closed tableau for these assumptions, we can attach it to that branch; every branch through $\mathbb{T}\varphi$ closes. On the other branch, all of

$$\{\mathbb{F}\varphi, \mathbb{T}\theta_1, \dots, \mathbb{T}\theta_m\}$$

are available, so we can also complete the other side to obtain a closed tableau. This shows $\Gamma \cup \Delta \vdash \psi$. □

Note that this means that in particular if $\Gamma \vdash \varphi$ and $\varphi \vdash \psi$, then $\Gamma \vdash \psi$. It follows also that if $\varphi_1, \dots, \varphi_n \vdash \psi$ and $\Gamma \vdash \varphi_i$ for each i, then $\Gamma \vdash \psi$.

Proposition 9.16. *Γ is inconsistent iff $\Gamma \vdash \varphi$ for every sentence φ.*

Proof. Exercise. □

Proposition 9.17 (Compactness).

1. *If $\Gamma \vdash \varphi$ then there is a finite subset $\Gamma_0 \subseteq \Gamma$ such that $\Gamma_0 \vdash \varphi$.*
2. *If every finite subset of Γ is consistent, then Γ is consistent.*

Proof.

1. If $\Gamma \vdash \varphi$, then there is a finite subset $\Gamma_0 = \{\psi_1, \dots, \psi_n\}$ and a closed tableau for
$$\{\mathbb{F}\varphi, \mathbb{T}\psi_1, \dots, \mathbb{T}\psi_n\}$$
This tableau also shows $\Gamma_0 \vdash \varphi$.
2. If Γ is inconsistent, then for some finite subset $\Gamma_0 = \{\psi_1, \dots, \psi_n\}$ there is a closed tableau for
$$\{\mathbb{T}\psi_1, \dots, \mathbb{T}\psi_n\}$$
This closed tableau shows that Γ_0 is inconsistent. □

9.9 Derivability and Consistency

We will now establish a number of properties of the derivability relation. They are independently interesting, but each will play a role in the proof of the completeness theorem.

Proposition 9.18. *If $\Gamma \vdash \varphi$ and $\Gamma \cup \{\varphi\}$ is inconsistent, then Γ is inconsistent.*

Proof. There are finite $\Gamma_0 = \{\psi_1, \dots, \psi_n\}$ and $\Gamma_1 = \{\chi_1, \dots, \chi_n\} \subseteq \Gamma$ such that

$$\{\mathbb{F}\varphi, \mathbb{T}\psi_1, \dots, \mathbb{T}\psi_n\}$$
$$\{\mathbb{T}\varphi, \mathbb{T}\chi_1, \dots, \mathbb{T}\chi_m\}$$

have closed tableaux. Using the Cut rule on φ we can combine these into a single closed tableau that shows $\Gamma_0 \cup \Gamma_1$ is inconsistent. Since $\Gamma_0 \subseteq \Gamma$ and $\Gamma_1 \subseteq \Gamma$, $\Gamma_0 \cup \Gamma_1 \subseteq \Gamma$, hence Γ is inconsistent. □

Proposition 9.19. *$\Gamma \vdash \varphi$ iff $\Gamma \cup \{\sim\varphi\}$ is inconsistent.*

Proof. First suppose $\Gamma \vdash \varphi$, i.e., there is a closed tableau for

$$\{\mathbb{F}\,\varphi, \mathbb{T}\,\psi_1, \dots, \mathbb{T}\,\psi_n\}$$

Using the $\sim\mathbb{T}$ rule, this can be turned into a closed tableau for

$$\{\mathbb{T}\sim\varphi, \mathbb{T}\,\psi_1, \dots, \mathbb{T}\,\psi_n\}.$$

On the other hand, if there is a closed tableau for the latter, we can turn it into a closed tableau of the former by removing every formula that results from $\sim\mathbb{T}$ applied to the first assumption $\mathbb{T}\sim\varphi$ as well as that assumption, and adding the assumption $\mathbb{F}\,\varphi$. For if a branch was closed before because it contained the conclusion of $\sim\mathbb{T}$ applied to $\mathbb{T}\sim\varphi$, i.e., $\mathbb{F}\,\varphi$, the corresponding branch in the new tableau is also closed. If a branch in the old tableau was closed because it contained the assumption $\mathbb{T}\sim\varphi$ as well as $\mathbb{F}\sim\varphi$ we can turn it into a closed branch by applying $\sim\mathbb{F}$ to $\mathbb{F}\sim\varphi$ to obtain $\mathbb{T}\,\varphi$. This closes the branch since we added $\mathbb{F}\,\varphi$ as an assumption. □

Proposition 9.20. *If $\Gamma \vdash \varphi$ and $\sim\varphi \in \Gamma$, then Γ is inconsistent.*

Proof. Suppose $\Gamma \vdash \varphi$ and $\sim\varphi \in \Gamma$. Then there are $\psi_1, \dots, \psi_n \in \Gamma$ such that

$$\{\mathbb{F}\,\varphi, \mathbb{T}\,\psi_1, \dots, \mathbb{T}\,\psi_n\}$$

has a closed tableau. Replace the assumption $\mathbb{F}\,\varphi$ by $\mathbb{T}\sim\varphi$, and insert the conclusion of $\sim\mathbb{T}$ applied to $\mathbb{F}\,\varphi$ after the assumptions. Any sentence in the tableau justified by appeal to line 1 in the old tableau is now justified by appeal to line $n+1$. So if the old tableau was closed, the new one is. It shows that Γ is inconsistent, since all assumptions are in Γ. □

Proposition 9.21. *If $\Gamma \cup \{\varphi\}$ and $\Gamma \cup \{\sim\varphi\}$ are both inconsistent, then Γ is inconsistent.*

Proof. If there are $\psi_1, \dots, \psi_n \in \Gamma$ and $\chi_1, \dots, \chi_m \in \Gamma$ such that

$$\{\mathbb{T}\,\varphi, \mathbb{T}\,\psi_1, \dots, \mathbb{T}\,\psi_n\} \text{ and}$$
$$\{\mathbb{T}\sim\varphi, \mathbb{T}\,\chi_1, \dots, \mathbb{T}\,\chi_m\}$$

both have closed tableaux, we can construct a single, combined tableau that shows that Γ is inconsistent by using as assumptions $\mathbb{T}\,\psi_1, \dots, \mathbb{T}\,\psi_n$ together with $\mathbb{T}\,\chi_1, \dots, \mathbb{T}\,\chi_m$, followed by an application of the Cut rule. This yields two branches, one starting with $\mathbb{T}\,\varphi$, the other with $\mathbb{F}\,\varphi$.

On the left left side, add the part of the first tableau below its assumptions. Here, every rule application is still correct, since each of the assumptions of the first tableau, including $\mathbb{T}\varphi$, is available. Thus, every branch below $\mathbb{T}\varphi$ closes.

On the right side, add the part of the second tableau below its assumption, with the results of any applications of $\sim\mathbb{T}$ to $\mathbb{T}\sim\varphi$ removed. The conclusion of $\sim\mathbb{T}$ to $\mathbb{T}\sim\varphi$ is $\mathbb{F}\varphi$, which is nevertheless available, as it is the conclusion of the Cut rule on the right side of the combined tableau.

If a branch in the second tableau was closed because it contained the assumption $\mathbb{T}\sim\varphi$ (which no longer appears as an assumption in the combined tableau) as well as $\mathbb{F}\sim\varphi$, we can applying $\sim\mathbb{F}$ to $\mathbb{F}\sim\varphi$ to obtain $\mathbb{T}\varphi$. Now the corresponding branch in the combined tableau also closes, because it contains the right-hand conclusion of the Cut rule, $\mathbb{F}\varphi$. If a branch in the second tableau closed for any other reason, the corresponding branch in the combined tableau also closes, since any signed formulas other than $\mathbb{T}\sim\varphi$ occurring on the branch in the old, second tableau also occur on the corresponding branch in the combined tableau. □

9.10 Derivability and the Propositional Connectives

We establish that the derivability relation $\vdash$ of tableaux is strong enough to establish some basic facts involving the propositional connectives, such as that $\varphi \mathbin{\&} \psi \vdash \varphi$ and $\varphi, \varphi \supset \psi \vdash \psi$ (modus ponens). These facts are needed for the proof of the completeness theorem.

Proposition 9.22. *1. Both $\varphi \mathbin{\&} \psi \vdash \varphi$ and $\varphi \mathbin{\&} \psi \vdash \psi$.*

2. $\varphi, \psi \vdash \varphi \mathbin{\&} \psi$.

Proof. 1. Both $\{\mathbb{F}\varphi, \mathbb{T}\varphi \mathbin{\&} \psi\}$ and $\{\mathbb{F}\psi, \mathbb{T}\varphi \mathbin{\&} \psi\}$ have closed tableaux

1.	$\mathbb{F}\varphi$	Assumption
2.	$\mathbb{T}\varphi \mathbin{\&} \psi$	Assumption
3.	$\mathbb{T}\varphi$	$\&\mathbb{T}$ 2
4.	$\mathbb{T}\psi$	$\&\mathbb{T}$ 2
	$\otimes$	

1.	$\mathbb{F}\psi$	Assumption
2.	$\mathbb{T}\varphi \mathbin{\&} \psi$	Assumption
3.	$\mathbb{T}\varphi$	$\&\mathbb{T}$ 2
4.	$\mathbb{T}\psi$	$\&\mathbb{T}$ 2
	$\otimes$	

2. Here is a closed tableau for $\{\mathbb{T}\varphi, \mathbb{T}\psi, \mathbb{F}\varphi \mathbin{\&} \psi\}$:

1.	$\mathbb{F}\,\varphi \,\&\, \psi$		Assumption
2.	$\mathbb{T}\,\varphi$		Assumption
3.	$\mathbb{T}\,\psi$		Assumption
4.	$\mathbb{F}\,\varphi$	$\mathbb{F}\,\psi$	$\&\mathbb{F}\,1$
	$\otimes$	$\otimes$	

Proposition 9.23. *1. $\{\varphi \lor \psi, \sim\varphi, \sim\psi\}$ is inconsistent.*

2. Both $\varphi \vdash \varphi \lor \psi$ and $\psi \vdash \varphi \lor \psi$.

Proof. 1. We give a closed tableau of $\{\mathbb{T}\,\varphi \lor \psi, \mathbb{T}\sim\varphi, \mathbb{T}\sim\psi\}$:

1.	$\mathbb{T}\,\varphi \lor \psi$		Assumption
2.	$\mathbb{T}\sim\varphi$		Assumption
3.	$\mathbb{T}\sim\psi$		Assumption
4.	$\mathbb{F}\,\varphi$		$\sim\mathbb{T}\,2$
5.	$\mathbb{F}\,\psi$		$\sim\mathbb{T}\,3$
6.	$\mathbb{T}\varphi$	$\mathbb{T}\psi$	$\lor\mathbb{T}\,1$
	$\otimes$	$\otimes$	

2. Both $\{\mathbb{F}\,\varphi \lor \psi, \mathbb{T}\,\varphi\}$ and $\{\mathbb{F}\,\varphi \lor \psi, \mathbb{T}\,\psi\}$ have closed tableaux:

1.	$\mathbb{F}\,\varphi \lor \psi$	Assumption
2.	$\mathbb{T}\,\varphi$	Assumption
3.	$\mathbb{F}\varphi$	$\lor\mathbb{F}\,1$
4.	$\mathbb{F}\psi$	$\lor\mathbb{F}\,1$
	$\otimes$	

1.	$\mathbb{F}\,\varphi \lor \psi$	Assumption
2.	$\mathbb{T}\,\psi$	Assumption
3.	$\mathbb{F}\varphi$	$\lor\mathbb{F}\,1$
4.	$\mathbb{F}\psi$	$\lor\mathbb{F}\,1$
	$\otimes$	

Proposition 9.24. *1. $\varphi, \varphi \supset \psi \vdash \psi$.*

2. Both $\sim\varphi \vdash \varphi \supset \psi$ and $\psi \vdash \varphi \supset \psi$.

Proof. 1. $\{\mathbb{F}\,\psi, \mathbb{T}\,\varphi \supset \psi, \mathbb{T}\,\varphi\}$ has a closed tableau:

$$
\begin{array}{rcl}
1. & \mathbb{F}\,\psi & \text{Assumption} \\
2. & \mathbb{T}\,\varphi \supset \psi & \text{Assumption} \\
3. & \mathbb{T}\,\varphi & \text{Assumption} \\
4. & \mathbb{F}\,\varphi \qquad \mathbb{T}\,\psi & {\supset}\mathbb{T}\,2 \\
 & \otimes \qquad\quad \otimes &
\end{array}
$$

2. Both $\{\mathbb{F}\,\varphi \supset \psi, \mathbb{T}{\sim}\varphi\}$ and $\{\mathbb{F}\,\varphi \supset \psi, \mathbb{T}\,\psi\}$ have closed tableaux:

$$
\begin{array}{rcl}
1. & \mathbb{F}\,\varphi \supset \psi & \text{Assumption} \\
2. & \mathbb{T}{\sim}\varphi & \text{Assumption} \\
3. & \mathbb{T}\,\varphi & {\supset}\mathbb{F}\,1 \\
4. & \mathbb{F}\,\psi & {\supset}\mathbb{F}\,1 \\
5. & \mathbb{F}\,\varphi & {\sim}\mathbb{T}\,2 \\
 & \otimes &
\end{array}
$$

$$
\begin{array}{rcl}
1. & \mathbb{F}\,\varphi \supset \psi & \text{Assumption} \\
2. & \mathbb{T}\,\psi & \text{Assumption} \\
3. & \mathbb{T}\,\varphi & {\supset}\mathbb{F}\,1 \\
4. & \mathbb{F}\,\psi & {\supset}\mathbb{F}\,1 \\
 & \otimes &
\end{array}
$$

9.11 Derivability and the Quantifiers

The completeness theorem also requires that the tableaux rules yield the facts about $\vdash$ established in this section.

Theorem 9.25. *If c is a constant not occurring in Γ or $\varphi(x)$ and $\Gamma \vdash \varphi(c)$, then $\Gamma \vdash \forall x\, \varphi(x)$.*

Proof. Suppose $\Gamma \vdash \varphi(c)$, i.e., there are $\psi_1, \ldots, \psi_n \in \Gamma$ and a closed tableau for

$$\{\mathbb{F}\,\varphi(c), \mathbb{T}\,\psi_1, \ldots, \mathbb{T}\,\psi_n\}.$$

We have to show that there is also a closed tableau for

$$\{\mathbb{F}\forall x\, \varphi(x), \mathbb{T}\,\psi_1, \ldots, \mathbb{T}\,\psi_n\}.$$

Take the closed tableau and replace the first assumption with $\mathbb{F}\forall x\, \varphi(x)$, and insert $\mathbb{F}\,\varphi(c)$ after the assumptions.

$$
\begin{array}{c}
\mathbb{F}\,\varphi(c) \\
\mathbb{T}\,\psi_1 \\
\vdots \\
\mathbb{T}\,\psi_n
\end{array}
\qquad
\begin{array}{c}
\mathbb{F}\,\forall x\,\varphi(x) \\
\mathbb{T}\,\psi_1 \\
\vdots \\
\mathbb{T}\,\psi_n \\
\mathbb{F}\,\varphi(c)
\end{array}
$$

The tableau is still closed, since all sentences available as assumptions before are still available at the top of the tableau. The inserted line is the result of a correct application of $\forall\mathbb{F}$, since the constant symbol c does not occur in ψ_1, $\dots$, ψ_n or $\forall x\,\varphi(x)$, i.e., it does not occur above the inserted line in the new tableau. □

Proposition 9.26. *1.* $\varphi(t) \vdash \exists x\,\varphi(x)$*.*

2. $\forall x\,\varphi(x) \vdash \varphi(t)$*.*

Proof. 1. A closed tableau for $\mathbb{F}\,\exists x\,\varphi(x), \mathbb{T}\,\varphi(t)$ is:

1.	$\mathbb{F}\,\exists x\,\varphi(x)$	Assumption
2.	$\mathbb{T}\,\varphi(t)$	Assumption
3.	$\mathbb{F}\,\varphi(t)$	$\exists\mathbb{F}\,1$
	$\otimes$	

2. A closed tableau for $\mathbb{F}\,\varphi(t), \mathbb{T}\forall x\,\varphi(x)$, is:

1.	$\mathbb{F}\,\varphi(t)$	Assumption
2.	$\mathbb{T}\forall x\,\varphi(x)$	Assumption
3.	$\mathbb{T}\,\varphi(t)$	$\forall\mathbb{T}\,2$
	$\otimes$	

9.12 Soundness

A derivation system, such as tableaux, is *sound* if it cannot derive things that do not actually hold. Soundness is thus a kind of guaranteed safety property for derivation systems. Depending on which proof theoretic property is in question, we would like to know for instance, that

1. every derivable φ is valid;
2. if a sentence is derivable from some others, it is also a consequence of them;

3. if a set of sentences is inconsistent, it is unsatisfiable.

These are important properties of a derivation system. If any of them do not hold, the derivation system is deficient—it would derive too much. Consequently, establishing the soundness of a derivation system is of the utmost importance.

Because all these proof-theoretic properties are defined via closed tableaux of some kind or other, proving (1)–(3) above requires proving something about the semantic properties of closed tableaux. We will first define what it means for a signed formula to be satisfied in a structure, and then show that if a tableau is closed, no structure satisfies all its assumptions. (1)–(3) then follow as corollaries from this result.

Definition 9.27. A structure $\mathfrak{M}$ *satisfies* a signed formula $\mathbb{T}\,\varphi$ iff $\mathfrak{M} \vDash \varphi$, and it satisfies $\mathbb{F}\,\varphi$ iff $\mathfrak{M} \nvDash \varphi$. $\mathfrak{M}$ satisfies a set of signed formulas Γ iff it satisfies every $S\,\varphi \in \Gamma$. Γ is *satisfiable* if there is a structure that satisfies it, and *unsatisfiable* otherwise.

Theorem 9.28 (Soundness). *If Γ has a closed tableau, Γ is unsatisfiable.*

Proof. Let's call a branch of a tableau satisfiable iff the set of signed formulas on it is satisfiable, and let's call a tableau satisfiable if it contains at least one satisfiable branch.

We show the following: Extending a satisfiable tableau by one of the rules of inference always results in a satisfiable tableau. This will prove the theorem: any closed tableau results by applying rules of inference to the tableau consisting only of assumptions from Γ. So if Γ were satisfiable, any tableau for it would be satisfiable. A closed tableau, however, is clearly not satisfiable: every branch contains both $\mathbb{T}\,\varphi$ and $\mathbb{F}\,\varphi$, and no structure can both satisfy and not satisfy φ.

Suppose we have a satisfiable tableau, i.e., a tableau with at least one satisfiable branch. Applying a rule of inference either adds signed formulas to a branch, or splits a branch in two. If the tableau has a satisfiable branch which is not extended by the rule application in question, it remains a satisfiable branch in the extended tableau, so the extended tableau is satisfiable. So we only have to consider the case where a rule is applied to a satisfiable branch.

Let Γ be the set of signed formulas on that branch, and let $S\,\varphi \in \Gamma$ be the signed formula to which the rule is applied. If the rule does not result in a split branch, we have to show that the extended branch, i.e., Γ together with the conclusions of the rule, is still satisfiable. If the rule results in a split branch, we have to show that at least one of the two resulting branches is satisfiable.

First, we consider the possible inferences that do not result in a split branch.

1. The branch is expanded by applying $\sim\mathbb{T}$ to $\mathbb{T}{\sim}\psi \in \Gamma$. Then the extended branch contains the signed formulas $\Gamma \cup \{\mathbb{F}\,\psi\}$. Suppose $\mathfrak{M} \vDash \Gamma$. In particular, $\mathfrak{M} \vDash {\sim}\psi$. Thus, $\mathfrak{M} \nvDash \psi$, i.e., $\mathfrak{M}$ satisfies $\mathbb{F}\,\psi$.

2. The branch is expanded by applying $\sim\mathbb{F}$ to $\mathbb{F}\sim\psi \in \Gamma$: Exercise.

3. The branch is expanded by applying $\&\mathbb{T}$ to $\mathbb{T}\,\psi \,\&\, \chi \in \Gamma$, which results in two new signed formulas on the branch: $\mathbb{T}\,\psi$ and $\mathbb{T}\,\chi$. Suppose $\mathfrak{M} \vDash \Gamma$, in particular $\mathfrak{M} \vDash \psi \,\&\, \chi$. Then $\mathfrak{M} \vDash \psi$ and $\mathfrak{M} \vDash \chi$. This means that $\mathfrak{M}$ satisfies both $\mathbb{T}\,\psi$ and $\mathbb{T}\,\chi$.

4. The branch is expanded by applying $\vee\mathbb{F}$ to $\mathbb{F}\,\psi \vee \chi \in \Gamma$: Exercise.

5. The branch is expanded by applying $\supset\mathbb{F}$ to $\mathbb{F}\,\psi \supset \chi \in \Gamma$: This results in two new signed formulas on the branch: $\mathbb{T}\,\psi$ and $\mathbb{F}\,\chi$. Suppose $\mathfrak{M} \vDash \Gamma$, in particular $\mathfrak{M} \nvDash \psi \supset \chi$. Then $\mathfrak{M} \vDash \psi$ and $\mathfrak{M} \nvDash \chi$. This means that $\mathfrak{M}$ satisfies both $\mathbb{T}\,\psi$ and $\mathbb{F}\,\chi$.

6. The branch is expanded by applying $\forall\mathbb{T}$ to $\mathbb{T}\,\forall x\,\psi(x) \in \Gamma$: This results in a new signed formula $\mathbb{T}\,\varphi(t)$ on the branch. Suppose $\mathfrak{M} \vDash \Gamma$, in particular, $\mathfrak{M} \vDash \forall x\,\varphi(x)$. By Proposition 7.30, $\mathfrak{M} \vDash \varphi(t)$. Consequently, $\mathfrak{M}$ satisfies $\mathbb{T}\,\varphi(t)$.

7. The branch is expanded by applying $\forall\mathbb{F}$ to $\mathbb{F}\,\forall x\,\psi(x) \in \Gamma$: This results in a new signed formula $\mathbb{F}\,\varphi(a)$ where a is a constant symbol not occurring in Γ. Since Γ is satisfiable, there is a $\mathfrak{M}$ such that $\mathfrak{M} \vDash \Gamma$, in particular $\mathfrak{M} \nvDash \forall x\,\psi(x)$. We have to show that $\Gamma \cup \{\mathbb{F}\,\varphi(a)\}$ is satisfiable. To do this, we define a suitable $\mathfrak{M}'$ as follows.

   By Proposition 7.18, $\mathfrak{M} \nvDash \forall x\,\psi(x)$ iff for some s, $\mathfrak{M}, s \nvDash \psi(x)$. Now let $\mathfrak{M}'$ be just like $\mathfrak{M}$, except $a^{\mathfrak{M}'} = s(x)$. By Corollary 7.20, for any $\mathbb{T}\,\chi \in \Gamma$, $\mathfrak{M}' \vDash \chi$, and for any $\mathbb{F}\,\chi \in \Gamma$, $\mathfrak{M}' \nvDash \chi$, since a does not occur in Γ.

   By Proposition 7.19, $\mathfrak{M}', s \nvDash \varphi(x)$. By Proposition 7.22, $\mathfrak{M}', s \nvDash \varphi(a)$. Since $\varphi(a)$ is a sentence, by Proposition 7.17, $\mathfrak{M}' \nvDash \varphi(a)$, i.e., $\mathfrak{M}'$ satisfies $\mathbb{F}\,\varphi(a)$.

8. The branch is expanded by applying $\exists\mathbb{T}$ to $\mathbb{T}\,\exists x\,\psi(x) \in \Gamma$: Exercise.

9. The branch is expanded by applying $\exists\mathbb{F}$ to $\mathbb{F}\,\exists x\,\psi(x) \in \Gamma$: Exercise.

Now let's consider the possible inferences that result in a split branch.

1. The branch is expanded by applying $\&\mathbb{F}$ to $\mathbb{F}\,\psi \,\&\, \chi \in \Gamma$, which results in two branches, a left one continuing through $\mathbb{F}\,\psi$ and a right one through $\mathbb{F}\,\chi$. Suppose $\mathfrak{M} \vDash \Gamma$, in particular $\mathfrak{M} \nvDash \psi \,\&\, \chi$. Then $\mathfrak{M} \nvDash \psi$ or $\mathfrak{M} \nvDash \chi$. In the former case, $\mathfrak{M}$ satisfies $\mathbb{F}\,\psi$, i.e., $\mathfrak{M}$ satisfies the formulas on the left branch. In the latter, $\mathfrak{M}$ satisfies $\mathbb{F}\,\chi$, i.e., $\mathfrak{M}$ satisfies the formulas on the right branch.

2. The branch is expanded by applying $\vee\mathbb{T}$ to $\mathbb{T}\,\psi \vee \chi \in \Gamma$: Exercise.

3. The branch is expanded by applying $\supset\mathbb{T}$ to $\mathbb{T}\,\psi \supset \chi \in \Gamma$: Exercise.

4. The branch is expanded by Cut: This results in two branches, one containing $\mathbb{T}\,\psi$, the other containing $\mathbb{F}\,\psi$. Since $\mathfrak{M} \vDash \Gamma$ and either $\mathfrak{M} \vDash \psi$ or $\mathfrak{M} \nvDash \psi$, $\mathfrak{M}$ satisfies either the left or the right branch. □

Corollary 9.29. *If* $\vdash \varphi$ *then* φ *is valid.*

Corollary 9.30. *If* $\Gamma \vdash \varphi$ *then* $\Gamma \vDash \varphi$.

Proof. If $\Gamma \vdash \varphi$ then for some $\psi_1, \dots, \psi_n \in \Gamma$, $\{\mathbb{F}\,\varphi, \mathbb{T}\,\psi_1, \dots, \mathbb{T}\,\psi_n\}$ has a closed tableau. By Theorem 9.28, every structure $\mathfrak{M}$ either makes some ψ_i false or makes φ true. Hence, if $\mathfrak{M} \vDash \Gamma$ then also $\mathfrak{M} \vDash \varphi$. □

Corollary 9.31. *If* Γ *is satisfiable, then it is consistent.*

Proof. We prove the contrapositive. Suppose that Γ is not consistent. Then there are $\psi_1, \dots, \psi_n \in \Gamma$ and a closed tableau for $\{\mathbb{T}\,\psi_1, \dots, \mathbb{T}\,\psi_n\}$. By Theorem 9.28, there is no $\mathfrak{M}$ such that $\mathfrak{M} \vDash \psi_i$ for all $i = 1, \dots, n$. But then Γ is not satisfiable. □

9.13 Tableaux with Identity predicate

Tableaux with identity predicate require additional inference rules. The rules for $=$ are (t, t_1, and t_2 are closed terms):

$$\frac{}{\mathbb{T}\,t = t} = \qquad \frac{\begin{array}{c}\mathbb{T}\,t_1 = t_2\\ \mathbb{T}\,\varphi(t_1)\end{array}}{\mathbb{T}\,\varphi(t_2)} =\mathbb{T} \qquad \frac{\begin{array}{c}\mathbb{T}\,t_1 = t_2\\ \mathbb{F}\,\varphi(t_1)\end{array}}{\mathbb{F}\,\varphi(t_2)} =\mathbb{F}$$

Note that in contrast to all the other rules, $=\mathbb{T}$ and $=\mathbb{F}$ require that *two* signed formulae already appear on the branch, namely both $\mathbb{T}\,t_1 = t_2$ and $S\,\varphi(t_1)$.

Example 9.32. If s and t are closed terms, then $s = t, \varphi(s) \vdash \varphi(t)$:

1.	$\mathbb{F}\,\varphi(t)$	Assumption
2.	$\mathbb{T}\,s = t$	Assumption
3.	$\mathbb{T}\,\varphi(s)$	Assumption
4.	$\mathbb{T}\,\varphi(t)$	$=\mathbb{T}$ 2, 3
	$\otimes$	

This may be familiar as the principle of substitutability of identicals, or Leibniz' Law.

Tableaux prove that $=$ is symmetric, i.e., that $s_1 = s_2 \vdash s_2 = s_1$:

1.	$\mathbb{F}\, s_2 = s_1$	Assumption
2.	$\mathbb{T}\, s_1 = s_2$	Assumption
3.	$\mathbb{T}\, s_1 = s_1$	$=$
4.	$\mathbb{T}\, s_2 = s_1$	$=\mathbb{T}\, 2, 3$
	$\otimes$	

Here, line 2 is the first prerequisite formula $\mathbb{T}\, s_1 = s_2$ of $=\mathbb{T}$. Line 3 is the second one, of the form $\mathbb{T}\, \varphi(s_2)$—think of $\varphi(x)$ as $x = s_1$, then $\varphi(s_1)$ is $s_1 = s_1$ and $\varphi(s_2)$ is $s_2 = s_1$.

They also prove that $=$ is transitive, i.e., that $s_1 = s_2, s_2 = s_3 \vdash s_1 = s_3$:

1.	$\mathbb{F}\, s_1 = s_3$	Assumption
2.	$\mathbb{T}\, s_1 = s_2$	Assumption
3.	$\mathbb{T}\, s_2 = s_3$	Assumption
4.	$\mathbb{T}\, s_1 = s_3$	$=\mathbb{T}\, 3, 2$
	$\otimes$	

In this tableau, the first prerequisite formula of $=\mathbb{T}$ is line 3, $\mathbb{T}\, s_2 = s_3$ (s_2 plays the role of t_1, and s_3 the role of t_2). The second prerequisite, of the form $\mathbb{T}\, \varphi(s_2)$ is line 2. Here, think of $\varphi(x)$ as $s_1 = x$; that makes $\varphi(s_2)$ into $t_1 = t_2$ (i.e., line 2) and $\varphi(s_3)$ into the formula $s_1 = s_3$ in the conclusion.

9.14 Soundness with Identity predicate

Proposition 9.33. *Tableaux with rules for identity are sound: no closed tableau is satisfiable.*

Proof. We just have to show as before that if a tableau has a satisfiable branch, the branch resulting from applying one of the rules for $=$ to it is also satisfiable. Let Γ be the set of signed formulae on the branch, and let $\mathfrak{M}$ be a structure satisfying Γ.

Suppose the branch is expanded using $=$, i.e., by adding the signed formula $\mathbb{T}\, t = t$. Trivially, $\mathfrak{M} \models t = t$, so $\mathfrak{M}$ also satisfies $\Gamma \cup \{\mathbb{T}\, t = t\}$.

If the branch is expanded using $=\mathbb{T}$, we add a signed formula $S\, \varphi(t_2)$, but Γ contains both $\mathbb{T}\, t_1 = t_2$ and $\mathbb{T}\, \varphi(t_1)$. Thus we have $\mathfrak{M} \models t_1 = t_2$ and $\mathfrak{M} \models \varphi(t_1)$. Let s be a variable assignment with $s(x) = \mathrm{Val}^{\mathfrak{M}}(t_1)$. By Proposition 7.17, $\mathfrak{M}, s \models \varphi(t_1)$. Since $s \sim_x s$, by Proposition 7.22, $\mathfrak{M}, s \models \varphi(x)$. since $\mathfrak{M} \models t_1 = t_2$, we have $\mathrm{Val}^{\mathfrak{M}}(t_1) = \mathrm{Val}^{\mathfrak{M}}(t_2)$, and hence $s(x) = \mathrm{Val}^{\mathfrak{M}}(t_2)$. By applying Proposition 7.22 again, we also have $\mathfrak{M}, s \models \varphi(t_2)$. By Proposition 7.17, $\mathfrak{M} \models \varphi(t_2)$. The case of $=\mathbb{F}$ is treated similarly. □

Problems

Problem 9.1. Give closed tableaux of the following:

1. $\mathbb{T}\,\varphi \mathbin{\&} (\psi \mathbin{\&} \chi), \mathbb{F}\,(\varphi \mathbin{\&} \psi) \mathbin{\&} \chi$.
2. $\mathbb{T}\,\varphi \lor (\psi \lor \chi), \mathbb{F}\,(\varphi \lor \psi) \lor \chi$.
3. $\mathbb{T}\,\varphi \supset (\psi \supset \chi), \mathbb{F}\,\psi \supset (\varphi \supset \chi)$.
4. $\mathbb{T}\,\varphi, \mathbb{F}\,{\sim}{\sim}\varphi$.

Problem 9.2. Give closed tableaux of the following:

1. $\mathbb{T}\,(\varphi \lor \psi) \supset \chi, \mathbb{F}\,\varphi \supset \chi$.
2. $\mathbb{T}\,(\varphi \supset \chi) \mathbin{\&} (\psi \supset \chi), \mathbb{F}\,(\varphi \lor \psi) \supset \chi$.
3. $\mathbb{F}\,{\sim}(\varphi \mathbin{\&} {\sim}\varphi)$.
4. $\mathbb{T}\,\psi \supset \varphi, \mathbb{F}\,{\sim}\varphi \supset {\sim}\psi$.
5. $\mathbb{F}\,(\varphi \supset {\sim}\varphi) \supset {\sim}\varphi$.
6. $\mathbb{F}\,{\sim}(\varphi \supset \psi) \supset {\sim}\psi$.
7. $\mathbb{T}\,\varphi \supset \chi, \mathbb{F}\,{\sim}(\varphi \mathbin{\&} {\sim}\chi)$.
8. $\mathbb{T}\,\varphi \mathbin{\&} {\sim}\chi, \mathbb{F}\,{\sim}(\varphi \supset \chi)$.
9. $\mathbb{T}\,\varphi \lor \psi, {\sim}\psi, \mathbb{F}\,\varphi$.
10. $\mathbb{T}\,{\sim}\varphi \lor {\sim}\psi, \mathbb{F}\,{\sim}(\varphi \mathbin{\&} \psi)$.
11. $\mathbb{F}\,({\sim}\varphi \mathbin{\&} {\sim}\psi) \supset {\sim}(\varphi \lor \psi)$.
12. $\mathbb{F}\,{\sim}(\varphi \lor \psi) \supset ({\sim}\varphi \mathbin{\&} {\sim}\psi)$.

Problem 9.3. Give closed tableaux of the following:

1. $\mathbb{T}\,{\sim}(\varphi \supset \psi), \mathbb{F}\,\varphi$.
2. $\mathbb{T}\,{\sim}(\varphi \mathbin{\&} \psi), \mathbb{F}\,{\sim}\varphi \lor {\sim}\psi$.
3. $\mathbb{T}\,\varphi \supset \psi, \mathbb{F}\,{\sim}\varphi \lor \psi$.
4. $\mathbb{F}\,{\sim}{\sim}\varphi \supset \varphi$.
5. $\mathbb{T}\,\varphi \supset \psi, \mathbb{T}\,{\sim}\varphi \supset \psi, \mathbb{F}\,\psi$.
6. $\mathbb{T}\,(\varphi \mathbin{\&} \psi) \supset \chi, \mathbb{F}\,(\varphi \supset \chi) \lor (\psi \supset \chi)$.
7. $\mathbb{T}\,(\varphi \supset \psi) \supset \varphi, \mathbb{F}\,\varphi$.
8. $\mathbb{F}\,(\varphi \supset \psi) \lor (\psi \supset \chi)$.

Problem 9.4. Give closed tableaux of the following:

1. $\mathbb{F}\,(\forall x\,\varphi(x)\,\&\,\forall y\,\psi(y)) \supset \forall z\,(\varphi(z)\,\&\,\psi(z))$.
2. $\mathbb{F}\,(\exists x\,\varphi(x) \lor \exists y\,\psi(y)) \supset \exists z\,(\varphi(z) \lor \psi(z))$.
3. $\mathbb{T}\forall x\,(\varphi(x) \supset \psi), \mathbb{F}\,\exists y\,\varphi(y) \supset \psi$.
4. $\mathbb{T}\forall x\,{\sim}\varphi(x), \mathbb{F}\,{\sim}\exists x\,\varphi(x)$.
5. $\mathbb{F}\,{\sim}\exists x\,\varphi(x) \supset \forall x\,{\sim}\varphi(x)$.
6. $\mathbb{F}\,{\sim}\exists x\,\forall y\,((\varphi(x,y) \supset {\sim}\varphi(y,y))\,\&\,({\sim}\varphi(y,y) \supset \varphi(x,y)))$.

Problem 9.5. Give closed tableaux of the following:

1. $\mathbb{F}\,{\sim}\forall x\,\varphi(x) \supset \exists x\,{\sim}\varphi(x)$.
2. $\mathbb{T}\,(\forall x\,\varphi(x) \supset \psi), \mathbb{F}\,\exists y\,(\varphi(y) \supset \psi)$.
3. $\mathbb{F}\,\exists x\,(\varphi(x) \supset \forall y\,\varphi(y))$.

Problem 9.6. Prove Proposition 9.16

Problem 9.7. Prove that $\Gamma \vdash {\sim}\varphi$ iff $\Gamma \cup \{\varphi\}$ is inconsistent.

Problem 9.8. Complete the proof of Theorem 9.28.

Problem 9.9. Give closed tableaux for the following:

1. $\mathbb{F}\forall x\,\forall y\,((x = y\,\&\,\varphi(x)) \supset \varphi(y))$
2. $\mathbb{F}\,\exists x\,(\varphi(x)\,\&\,\forall y\,(\varphi(y) \supset y = x))$,
   $\mathbb{T}\,\exists x\,\varphi(x)\,\&\,\forall y\,\forall z\,((\varphi(y)\,\&\,\varphi(z)) \supset y = z)$

Chapter 10

The Completeness Theorem

10.1 Introduction

The completeness theorem is one of the most fundamental results about logic. It comes in two formulations, the equivalence of which we'll prove. In its first formulation it says something fundamental about the relationship between semantic consequence and our derivation system: if a sentence φ follows from some sentences Γ, then there is also a derivation that establishes $\Gamma \vdash \varphi$. Thus, the derivation system is as strong as it can possibly be without proving things that don't actually follow.

In its second formulation, it can be stated as a model existence result: every consistent set of sentences is satisfiable. Consistency is a proof-theoretic notion: it says that our derivation system is unable to produce certain derivations. But who's to say that just because there are no derivations of a certain sort from Γ, it's guaranteed that there is a structure $\mathfrak{M}$? Before the completeness theorem was first proved—in fact before we had the derivation systems we now do—the great German mathematician David Hilbert held the view that consistency of mathematical theories guarantees the existence of the objects they are about. He put it as follows in a letter to Gottlob Frege:

> If the arbitrarily given axioms do not contradict one another with all their consequences, then they are true and the things defined by the axioms exist. This is for me the criterion of truth and existence.

Frege vehemently disagreed. The second formulation of the completeness theorem shows that Hilbert was right in at least the sense that if the axioms are consistent, then *some* structure exists that makes them all true.

These aren't the only reasons the completeness theorem—or rather, its proof—is important. It has a number of important consequences, some of which we'll discuss separately. For instance, since any derivation that shows $\Gamma \vdash \varphi$ is finite and so can only use finitely many of the sentences in Γ, it follows by the completeness theorem that if φ is a consequence of Γ, it is already a consequence of a finite subset of Γ. This is called *compactness*. Equivalently, if every finite subset of Γ is consistent, then Γ itself must be consistent.

Although the compactness theorem follows from the completeness theorem via the detour through derivations, it is also possible to use the *the proof of* the completeness theorem to establish it directly. For what the proof does is take a set of sentences with a certain property—consistency—and constructs a structure out of this set that has certain properties (in this case, that it satisfies the set). Almost the very same construction can be used to directly establish compactness, by starting from "finitely satisfiable" sets of sentences instead of consistent ones. The construction also yields other consequences, e.g., that any satisfiable set of sentences has a finite or countably infinite model. (This result is called the Löwenheim–Skolem theorem.) In general, the construction of structures from sets of sentences is used often in logic, and sometimes even in philosophy.

10.2 Outline of the Proof

The proof of the completeness theorem is a bit complex, and upon first reading it, it is easy to get lost. So let us outline the proof. The first step is a shift of perspective, that allows us to see a route to a proof. When completeness is thought of as "whenever $\Gamma \vDash \varphi$ then $\Gamma \vdash \varphi$," it may be hard to even come up with an idea: for to show that $\Gamma \vdash \varphi$ we have to find a derivation, and it does not look like the hypothesis that $\Gamma \vDash \varphi$ helps us for this in any way. For some proof systems it is possible to directly construct a derivation, but we will take a slightly different approach. The shift in perspective required is this: completeness can also be formulated as: "if Γ is consistent, it is satisfiable." Perhaps we can use the information in Γ together with the hypothesis that it is consistent to construct a structure that satisfies every sentence in Γ. After all, we know what kind of structure we are looking for: one that is as Γ describes it!

If Γ contains only atomic sentences, it is easy to construct a model for it. Suppose the atomic sentences are all of the form $P(a_1, \dots, a_n)$ where the a_i are constant symbols. All we have to do is come up with a domain $|\mathfrak{M}|$ and an assignment for P so that $\mathfrak{M} \vDash P(a_1, \dots, a_n)$. But that's not very hard: put $|\mathfrak{M}| = \mathbb{N}$, $c_i^{\mathfrak{M}} = i$, and for every $P(a_1, \dots, a_n) \in \Gamma$, put the tuple $\langle k_1, \dots, k_n \rangle$ into $P^{\mathfrak{M}}$, where k_i is the index of the constant symbol a_i (i.e., $a_i \equiv c_{k_i}$).

Now suppose Γ contains some formula $\sim\psi$, with ψ atomic. We might worry that the construction of $\mathfrak{M}$ interferes with the possibility of making $\sim\psi$ true. But here's where the consistency of Γ comes in: if $\sim\psi \in \Gamma$, then $\psi \notin \Gamma$, or else Γ would be inconsistent. And if $\psi \notin \Gamma$, then according to our construction of $\mathfrak{M}$, $\mathfrak{M} \nvDash \psi$, so $\mathfrak{M} \vDash \sim\psi$. So far so good.

What if Γ contains complex, non-atomic formulas? Say it contains $\varphi \mathbin{\&} \psi$. To make that true, we should proceed as if both φ and ψ were in Γ. And if

$\varphi \lor \psi \in \Gamma$, then we will have to make at least one of them true, i.e., proceed as if one of them was in Γ.

This suggests the following idea: we add additional formulae to Γ so as to (a) keep the resulting set consistent and (b) make sure that for every possible atomic sentence φ, either φ is in the resulting set, or $\sim\varphi$ is, and (c) such that, whenever $\varphi \,\&\, \psi$ is in the set, so are both φ and ψ, if $\varphi \lor \psi$ is in the set, at least one of φ or ψ is also, etc. We keep doing this (potentially forever). Call the set of all formulae so added Γ^*. Then our construction above would provide us with a structure $\mathfrak{M}$ for which we could prove, by induction, that it satisfies all sentences in Γ^*, and hence also all sentence in Γ since $\Gamma \subseteq \Gamma^*$. It turns out that guaranteeing (a) and (b) is enough. A set of sentences for which (b) holds is called *complete*. So our task will be to extend the consistent set Γ to a consistent and complete set Γ^*.

There is one wrinkle in this plan: if $\exists x\, \varphi(x) \in \Gamma$ we would hope to be able to pick some constant symbol c and add $\varphi(c)$ in this process. But how do we know we can always do that? Perhaps we only have a few constant symbols in our language, and for each one of them we have $\sim\varphi(c) \in \Gamma$. We can't also add $\varphi(c)$, since this would make the set inconsistent, and we wouldn't know whether $\mathfrak{M}$ has to make $\varphi(c)$ or $\sim\varphi(c)$ true. Moreover, it might happen that Γ contains only sentences in a language that has no constant symbols at all (e.g., the language of set theory).

The solution to this problem is to simply add infinitely many constants at the beginning, plus sentences that connect them with the quantifiers in the right way. (Of course, we have to verify that this cannot introduce an inconsistency.)

Our original construction works well if we only have constant symbols in the atomic sentences. But the language might also contain function symbols. In that case, it might be tricky to find the right functions on $\mathbb{N}$ to assign to these function symbols to make everything work. So here's another trick: instead of using i to interpret c_i, just take the set of constant symbols itself as the domain. Then $\mathfrak{M}$ can assign every constant symbol to itself: $c_i^{\mathfrak{M}} = c_i$. But why not go all the way: let $|\mathfrak{M}|$ be all *terms* of the language! If we do this, there is an obvious assignment of functions (that take terms as arguments and have terms as values) to function symbols: we assign to the function symbol f_i^n the function which, given n terms $t_1, \dots, t_n$ as input, produces the term $f_i^n(t_1, \dots, t_n)$ as value.

The last piece of the puzzle is what to do with $=$. The predicate symbol $=$ has a fixed interpretation: $\mathfrak{M} \vDash t = t'$ iff $\mathrm{Val}^{\mathfrak{M}}(t) = \mathrm{Val}^{\mathfrak{M}}(t')$. Now if we set things up so that the value of a term t is t itself, then this structure will make *no* sentence of the form $t = t'$ true unless t and t' are one and the same term. And of course this is a problem, since basically every interesting theory in a language with function symbols will have as theorems sentences $t = t'$ where t and t' are not the same term (e.g., in theories of arithmetic: $(\mathsf{0} + \mathsf{0}) = \mathsf{0}$). To solve this problem, we change the domain of $\mathfrak{M}$: instead of using terms as the objects in $|\mathfrak{M}|$, we use sets of terms, and each set is so that it contains all those terms which the sentences in Γ require to be equal. So, e.g., if Γ is a theory of arithmetic, one of these sets will contain: 0, $(0 + 0)$, (0×0), etc. This will be the set we assign to 0, and it will turn out that this set is also the value of all the terms in it, e.g., also of $(0 + 0)$. Therefore, the sentence $(0 + 0) = 0$ will be true in this revised structure.

So here's what we'll do. First we investigate the properties of complete consistent sets, in particular we prove that a complete consistent set contains $\varphi \mathbin{\&} \psi$ iff it contains both φ and ψ, $\varphi \lor \psi$ iff it contains at least one of them, etc. (Proposition 10.2). Then we define and investigate "saturated" sets of sentences. A saturated set is one which contains conditionals that link each quantified sentence to instances of it (Definition 10.5). We show that any consistent set Γ can always be extended to a saturated set Γ' (Lemma 10.6). If a set is consistent, saturated, and complete it also has the property that it contains $\exists x\, \varphi(x)$ iff it contains $\varphi(t)$ for some closed term t and $\forall x\, \varphi(x)$ iff it contains $\varphi(t)$ for all closed terms t (Proposition 10.7). We'll then take the saturated consistent set Γ' and show that it can be extended to a saturated, consistent, and complete set Γ^* (Lemma 10.8). This set Γ^* is what we'll use to define our term model $\mathfrak{M}(\Gamma^*)$. The term model has the set of closed terms as its domain, and the interpretation of its predicate symbols is given by the atomic sentences in Γ^* (Definition 10.9). We'll use the properties of saturated, complete consistent sets to show that indeed $\mathfrak{M}(\Gamma^*) \vDash \varphi$ iff $\varphi \in \Gamma^*$ (Lemma 10.12), and thus in particular, $\mathfrak{M}(\Gamma^*) \vDash \Gamma$. Finally, we'll consider how to define a term model if Γ contains $=$ as well (Definition 10.16) and show that it satisfies Γ^* (Lemma 10.19).

10.3 Complete Consistent Sets of Sentences

Definition 10.1 (Complete set). A set Γ of sentences is *complete* iff for any sentence φ, either $\varphi \in \Gamma$ or $\sim\varphi \in \Gamma$.

Complete sets of sentences leave no questions unanswered. For any sentence φ, Γ "says" if φ is true or false. The importance of complete sets extends beyond the proof of the completeness theorem. A theory which is complete and axiomatizable, for instance, is always decidable.

Complete consistent sets are important in the completeness proof since we can guarantee that every consistent set of sentences Γ is contained in a complete consistent set Γ^*. A complete consistent set contains, for each sentence φ, either φ or its negation $\sim\varphi$, but not both. This is true in particular for atomic sentences, so from a complete consistent set in a language suitably expanded by constant symbols, we can construct a structure where the interpretation of predicate symbols is defined according to which atomic sentences are in Γ^*. This structure can then be shown to make all sentences in Γ^* (and hence also all those in Γ) true. The proof of this latter fact requires that $\sim\varphi \in \Gamma^*$ iff $\varphi \notin \Gamma^*$, $(\varphi \lor \psi) \in \Gamma^*$ iff $\varphi \in \Gamma^*$ or $\psi \in \Gamma^*$, etc.

In what follows, we will often tacitly use the properties of reflexivity, monotonicity, and transitivity of $\vdash$ (see section 9.8).

Proposition 10.2. *Suppose Γ is complete and consistent. Then:*

1. *If $\Gamma \vdash \varphi$, then $\varphi \in \Gamma$.*
2. *$\varphi \mathbin{\&} \psi \in \Gamma$ iff both $\varphi \in \Gamma$ and $\psi \in \Gamma$.*
3. *$\varphi \lor \psi \in \Gamma$ iff either $\varphi \in \Gamma$ or $\psi \in \Gamma$.*
4. *$\varphi \supset \psi \in \Gamma$ iff either $\varphi \notin \Gamma$ or $\psi \in \Gamma$.*

Proof. Let us suppose for all of the following that Γ is complete and consistent.

1. If $\Gamma \vdash \varphi$, then $\varphi \in \Gamma$.

   Suppose that $\Gamma \vdash \varphi$. Suppose to the contrary that $\varphi \notin \Gamma$. Since Γ is complete, $\sim\varphi \in \Gamma$. By Proposition 9.20, Γ is inconsistent. This contradicts the assumption that Γ is consistent. Hence, it cannot be the case that $\varphi \notin \Gamma$, so $\varphi \in \Gamma$.

2. $\varphi \mathbin{\&} \psi \in \Gamma$ iff both $\varphi \in \Gamma$ and $\psi \in \Gamma$:

   For the forward direction, suppose $\varphi \mathbin{\&} \psi \in \Gamma$. Then by Proposition 9.22, item (1), $\Gamma \vdash \varphi$ and $\Gamma \vdash \psi$. By (1), $\varphi \in \Gamma$ and $\psi \in \Gamma$, as required.

   For the reverse direction, let $\varphi \in \Gamma$ and $\psi \in \Gamma$. By Proposition 9.22, item (2), $\Gamma \vdash \varphi \mathbin{\&} \psi$. By (1), $\varphi \mathbin{\&} \psi \in \Gamma$.

3. First we show that if $\varphi \lor \psi \in \Gamma$, then either $\varphi \in \Gamma$ or $\psi \in \Gamma$. Suppose $\varphi \lor \psi \in \Gamma$ but $\varphi \notin \Gamma$ and $\psi \notin \Gamma$. Since Γ is complete, $\sim\varphi \in \Gamma$ and $\sim\psi \in \Gamma$. By Proposition 9.23, item (1), Γ is inconsistent, a contradiction. Hence, either $\varphi \in \Gamma$ or $\psi \in \Gamma$.

   For the reverse direction, suppose that $\varphi \in \Gamma$ or $\psi \in \Gamma$. By Proposition 9.23, item (2), $\Gamma \vdash \varphi \lor \psi$. By (1), $\varphi \lor \psi \in \Gamma$, as required.

4. Exercise. □

10.4 Henkin Expansion

Part of the challenge in proving the completeness theorem is that the model we construct from a complete consistent set Γ must make all the quantified formulae in Γ true. In order to guarantee this, we use a trick due to Leon Henkin. In essence, the trick consists in expanding the language by infinitely many constant symbols and adding, for each formula with one free variable

$\varphi(x)$ a formula of the form $\exists x\, \varphi(x) \supset \varphi(c)$, where c is one of the new constant symbols. When we construct the structure satisfying Γ, this will guarantee that each true existential sentence has a witness among the new constants.

Proposition 10.3. *If Γ is consistent in $\mathcal{L}$ and $\mathcal{L}'$ is obtained from $\mathcal{L}$ by adding a countably infinite set of new constant symbols d_0, d_1, …, then Γ is consistent in $\mathcal{L}'$.*

Definition 10.4 (Saturated set). A set Γ of formulae of a language $\mathcal{L}$ is *saturated* iff for each formula $\varphi(x) \in \mathrm{Frm}(\mathcal{L})$ with one free variable x there is a constant symbol $c \in \mathcal{L}$ such that $\exists x\, \varphi(x) \supset \varphi(c) \in \Gamma$.

The following definition will be used in the proof of the next theorem.

Definition 10.5. Let $\mathcal{L}'$ be as in Proposition 10.3. Fix an enumeration $\varphi_0(x_0)$, $\varphi_1(x_1)$, … of all formulae $\varphi_i(x_i)$ of $\mathcal{L}'$ in which one variable (x_i) occurs free. We define the sentences θ_n by induction on n.

Let c_0 be the first constant symbol among the d_i we added to $\mathcal{L}$ which does not occur in $\varphi_0(x_0)$. Assuming that θ_0, …, θ_{n-1} have already been defined, let c_n be the first among the new constant symbols d_i that occurs neither in θ_0, …, θ_{n-1} nor in $\varphi_n(x_n)$.

Now let θ_n be the formula $\exists x_n\, \varphi_n(x_n) \supset \varphi_n(c_n)$.

Lemma 10.6. *Every consistent set Γ can be extended to a saturated consistent set Γ'.*

Proof. Given a consistent set of sentences Γ in a language $\mathcal{L}$, expand the language by adding a countably infinite set of new constant symbols to form $\mathcal{L}'$. By Proposition 10.3, Γ is still consistent in the richer language. Further, let θ_i be as in Definition 10.5. Let

$$\Gamma_0 = \Gamma$$
$$\Gamma_{n+1} = \Gamma_n \cup \{\theta_n\}$$

i.e., $\Gamma_{n+1} = \Gamma \cup \{\theta_0, \ldots, \theta_n\}$, and let $\Gamma' = \bigcup_n \Gamma_n$. Γ' is clearly saturated.

If Γ' were inconsistent, then for some n, Γ_n would be inconsistent (Exercise: explain why). So to show that Γ' is consistent it suffices to show, by induction on n, that each set Γ_n is consistent.

The induction basis is simply the claim that $\Gamma_0 = \Gamma$ is consistent, which is the hypothesis of the theorem. For the induction step, suppose that Γ_n is consistent but $\Gamma_{n+1} = \Gamma_n \cup \{\theta_n\}$ is inconsistent. Recall that θ_n is $\exists x_n\, \varphi_n(x_n) \supset \varphi_n(c_n)$, where $\varphi_n(x_n)$ is a formula of $\mathcal{L}'$ with only the variable x_n free. By the way we've chosen the c_n (see Definition 10.5), c_n does not occur in $\varphi_n(x_n)$ nor in Γ_n.

If $\Gamma_n \cup \{\theta_n\}$ is inconsistent, then $\Gamma_n \vdash {\sim}\theta_n$, and hence both of the following hold:

$$\Gamma_n \vdash \exists x_n\, \varphi_n(x_n) \qquad \Gamma_n \vdash {\sim}\varphi_n(c_n)$$

Since c_n does not occur in Γ_n or in $\varphi_n(x_n)$, Theorem 9.25 applies. From $\Gamma_n \vdash \sim\varphi_n(c_n)$, we obtain $\Gamma_n \vdash \forall x_n \sim\varphi_n(x_n)$. Thus we have that both $\Gamma_n \vdash \exists x_n\, \varphi_n(x_n)$ and $\Gamma_n \vdash \forall x_n \sim\varphi_n(x_n)$, so Γ_n itself is inconsistent. (Note that $\forall x_n \sim\varphi_n(x_n) \vdash \sim\exists x_n\, \varphi_n(x_n)$.) Contradiction: Γ_n was supposed to be consistent. Hence $\Gamma_n \cup \{\theta_n\}$ is consistent. □

We'll now show that *complete*, consistent sets which are saturated have the property that it contains a universally quantified sentence iff it contains all its instances and it contains an existentially quantified sentence iff it contains at least one instance. We'll use this to show that the structure we'll generate from a complete, consistent, saturated set makes all its quantified sentences true.

Proposition 10.7. *Suppose Γ is complete, consistent, and saturated.*

1. $\exists x\, \varphi(x) \in \Gamma$ *iff* $\varphi(t) \in \Gamma$ *for at least one closed term t.*
2. $\forall x\, \varphi(x) \in \Gamma$ *iff* $\varphi(t) \in \Gamma$ *for all closed terms t.*

Proof. 1. First suppose that $\exists x\, \varphi(x) \in \Gamma$. Because Γ is saturated, $(\exists x\, \varphi(x) \supset \varphi(c)) \in \Gamma$ for some constant symbol c. By Proposition 9.24, item (1), and Proposition 10.2(1), $\varphi(c) \in \Gamma$.

For the other direction, saturation is not necessary: Suppose $\varphi(t) \in \Gamma$. Then $\Gamma \vdash \exists x\, \varphi(x)$ by Proposition 9.26, item (1). By Proposition 10.2(1), $\exists x\, \varphi(x) \in \Gamma$.

2. Exercise. □

10.5 Lindenbaum's Lemma

We now prove a lemma that shows that any consistent set of sentences is contained in some set of sentences which is not just consistent, but also complete. The proof works by adding one sentence at a time, guaranteeing at each step that the set remains consistent. We do this so that for every φ, either φ or $\sim\varphi$ gets added at some stage. The union of all stages in that construction then contains either φ or its negation $\sim\varphi$ and is thus complete. It is also consistent, since we make sure at each stage not to introduce an inconsistency.

Lemma 10.8 (Lindenbaum's Lemma). *Every consistent set Γ in a language $\mathcal{L}$ can be extended to a complete and consistent set Γ^*.*

Proof. Let Γ be consistent. Let $\varphi_0, \varphi_1, \ldots$ be an enumeration of all the sentences of $\mathcal{L}$. Define $\Gamma_0 = \Gamma$, and

$$\Gamma_{n+1} = \begin{cases} \Gamma_n \cup \{\varphi_n\} & \text{if } \Gamma_n \cup \{\varphi_n\} \text{ is consistent;} \\ \Gamma_n \cup \{\sim\varphi_n\} & \text{otherwise.} \end{cases}$$

Let $\Gamma^* = \bigcup_{n \geq 0} \Gamma_n$.

Each Γ_n is consistent: Γ_0 is consistent by definition. If $\Gamma_{n+1} = \Gamma_n \cup \{\varphi_n\}$, this is because the latter is consistent. If it isn't, $\Gamma_{n+1} = \Gamma_n \cup \{\sim\varphi_n\}$. We have to verify that $\Gamma_n \cup \{\sim\varphi_n\}$ is consistent. Suppose it's not. Then *both* $\Gamma_n \cup \{\varphi_n\}$ and $\Gamma_n \cup \{\sim\varphi_n\}$ are inconsistent. This means that Γ_n would be inconsistent by Proposition 9.21, contrary to the induction hypothesis.

For every n and every $i < n$, $\Gamma_i \subseteq \Gamma_n$. This follows by a simple induction on n. For $n = 0$, there are no $i < 0$, so the claim holds automatically. For the inductive step, suppose it is true for n. We have $\Gamma_{n+1} = \Gamma_n \cup \{\varphi_n\}$ or $= \Gamma_n \cup \{\sim\varphi_n\}$ by construction. So $\Gamma_n \subseteq \Gamma_{n+1}$. If $i < n$, then $\Gamma_i \subseteq \Gamma_n$ by inductive hypothesis, and so $\subseteq \Gamma_{n+1}$ by transitivity of $\subseteq$.

From this it follows that every finite subset of Γ^* is a subset of Γ_n for some n, since each $\psi \in \Gamma^*$ not already in Γ_0 is added at some stage i. If n is the last one of these, then all ψ in the finite subset are in Γ_n. So, every finite subset of Γ^* is consistent. By Proposition 9.17, Γ^* is consistent.

Every sentence of $\mathrm{Frm}(\mathcal{L})$ appears on the list used to define Γ^*. If $\varphi_n \notin \Gamma^*$, then that is because $\Gamma_n \cup \{\varphi_n\}$ was inconsistent. But then $\sim\varphi_n \in \Gamma^*$, so Γ^* is complete. □

10.6 Construction of a Model

Right now we are not concerned about $=$, i.e., we only want to show that a consistent set Γ of sentences not containing $=$ is satisfiable. We first extend Γ to a consistent, complete, and saturated set Γ^*. In this case, the definition of a model $\mathfrak{M}(\Gamma^*)$ is simple: We take the set of closed terms of $\mathcal{L}'$ as the domain. We assign every constant symbol to itself, and make sure that more generally, for every closed term t, $\mathrm{Val}^{\mathfrak{M}(\Gamma^*)}(t) = t$. The predicate symbols are assigned extensions in such a way that an atomic sentence is true in $\mathfrak{M}(\Gamma^*)$ iff it is in Γ^*. This will obviously make all the atomic sentences in Γ^* true in $\mathfrak{M}(\Gamma^*)$. The rest are true provided the Γ^* we start with is consistent, complete, and saturated.

Definition 10.9 (Term model). Let Γ^* be a complete and consistent, saturated set of sentences in a language $\mathcal{L}$. The *term model* $\mathfrak{M}(\Gamma^*)$ of Γ^* is the structure defined as follows:

1. The domain $|\mathfrak{M}(\Gamma^*)|$ is the set of all closed terms of $\mathcal{L}$.

2. The interpretation of a constant symbol c is c itself: $c^{\mathfrak{M}(\Gamma^*)} = c$.

3. The function symbol f is assigned the function which, given as arguments the closed terms $t_1, \ldots, t_n$, has as value the closed term $f(t_1, \ldots, t_n)$:

$$f^{\mathfrak{M}(\Gamma^*)}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$$

4. If R is an n-place predicate symbol, then

$$\langle t_1, \dots, t_n \rangle \in R^{\mathfrak{M}(\Gamma^*)} \text{ iff } R(t_1, \dots, t_n) \in \Gamma^*.$$

We will now check that we indeed have $\mathrm{Val}^{\mathfrak{M}(\Gamma^*)}(t) = t$.

Lemma 10.10. *Let $\mathfrak{M}(\Gamma^*)$ be the term model of Definition 10.9, then $\mathrm{Val}^{\mathfrak{M}(\Gamma^*)}(t) = t$.*

Proof. The proof is by induction on t, where the base case, when t is a constant symbol, follows directly from the definition of the term model. For the induction step assume $t_1, \dots, t_n$ are closed terms such that $\mathrm{Val}^{\mathfrak{M}(\Gamma^*)}(t_i) = t_i$ and that f is an n-ary function symbol. Then

$$\begin{aligned}
\mathrm{Val}^{\mathfrak{M}(\Gamma^*)}(f(t_1, \dots, t_n)) &= f^{\mathfrak{M}(\Gamma^*)}(\mathrm{Val}^{\mathfrak{M}(\Gamma^*)}(t_1), \dots, \mathrm{Val}^{\mathfrak{M}(\Gamma^*)}(t_n)) \\
&= f^{\mathfrak{M}(\Gamma^*)}(t_1, \dots, t_n) \\
&= f(t_1, \dots, t_n),
\end{aligned}$$

and so by induction this holds for every closed term t. □

A structure $\mathfrak{M}$ may make an existentially quantified sentence $\exists x\, \varphi(x)$ true without there being an instance $\varphi(t)$ that it makes true. A structure $\mathfrak{M}$ may make all instances $\varphi(t)$ of a universally quantified sentence $\forall x\, \varphi(x)$ true, without making $\forall x\, \varphi(x)$ true. This is because in general not every element of $|\mathfrak{M}|$ is the value of a closed term ($\mathfrak{M}$ may not be covered). This is the reason the satisfaction relation is defined via variable assignments. However, for our term model $\mathfrak{M}(\Gamma^*)$ this wouldn't be necessary—because it is covered. This is the content of the next result.

Proposition 10.11. *Let $\mathfrak{M}(\Gamma^*)$ be the term model of Definition 10.9.*

1. *$\mathfrak{M}(\Gamma^*) \vDash \exists x\, \varphi(x)$ iff $\mathfrak{M}(\Gamma^*) \vDash \varphi(t)$ for at least one closed term t.*
2. *$\mathfrak{M}(\Gamma^*) \vDash \forall x\, \varphi(x)$ iff $\mathfrak{M}(\Gamma^*) \vDash \varphi(t)$ for all closed terms t.*

Proof.

1. By Proposition 7.18, $\mathfrak{M}(\Gamma^*) \vDash \exists x\, \varphi(x)$ iff for at least one variable assignment s, $\mathfrak{M}(\Gamma^*), s \vDash \varphi(x)$. As $|\mathfrak{M}(\Gamma^*)|$ consists of the closed terms of $\mathcal{L}$, this is the case iff there is at least one closed term t such that $s(x) = t$ and $\mathfrak{M}(\Gamma^*), s \vDash \varphi(x)$. By Proposition 7.22, $\mathfrak{M}(\Gamma^*), s \vDash \varphi(x)$ iff $\mathfrak{M}(\Gamma^*), s \vDash \varphi(t)$, where $s(x) = t$. By Proposition 7.17, $\mathfrak{M}(\Gamma^*), s \vDash \varphi(t)$ iff $\mathfrak{M}(\Gamma^*) \vDash \varphi(t)$, since $\varphi(t)$ is a sentence.
2. Exercise. □

Lemma 10.12 (Truth Lemma). *Suppose φ does not contain $=$. Then $\mathfrak{M}(\Gamma^*) \vDash \varphi$ iff $\varphi \in \Gamma^*$.*

Proof. We prove both directions simultaneously, and by induction on φ.

1. $\varphi \equiv \bot$: $\mathfrak{M}(\Gamma^*) \nvDash \bot$ by definition of satisfaction. On the other hand, $\bot \notin \Gamma^*$ since Γ^* is consistent.

2. $\varphi \equiv R(t_1, \ldots, t_n)$: $\mathfrak{M}(\Gamma^*) \vDash R(t_1, \ldots, t_n)$ iff $\langle t_1, \ldots, t_n \rangle \in R^{\mathfrak{M}(\Gamma^*)}$ (by the definition of satisfaction) iff $R(t_1, \ldots, t_n) \in \Gamma^*$ (by the construction of $\mathfrak{M}(\Gamma^*)$).

3. $\varphi \equiv \sim\psi$: $\mathfrak{M}(\Gamma^*) \vDash \varphi$ iff $\mathfrak{M}(\Gamma^*) \nvDash \psi$ (by definition of satisfaction). By induction hypothesis, $\mathfrak{M}(\Gamma^*) \nvDash \psi$ iff $\psi \notin \Gamma^*$. Since Γ^* is consistent and complete, $\psi \notin \Gamma^*$ iff $\sim\psi \in \Gamma^*$.

4. $\varphi \equiv \psi \,\&\, \chi$: $\mathfrak{M}(\Gamma^*) \vDash \varphi$ iff we have both $\mathfrak{M}(\Gamma^*) \vDash \psi$ and $\mathfrak{M}(\Gamma^*) \vDash \chi$ (by definition of satisfaction) iff both $\psi \in \Gamma^*$ and $\chi \in \Gamma^*$ (by the induction hypothesis). By Proposition 10.2(2), this is the case iff $(\psi \,\&\, \chi) \in \Gamma^*$.

5. $\varphi \equiv \psi \vee \chi$: $\mathfrak{M}(\Gamma^*) \vDash \varphi$ iff $\mathfrak{M}(\Gamma^*) \vDash \psi$ or $\mathfrak{M}(\Gamma^*) \vDash \chi$ (by definition of satisfaction) iff $\psi \in \Gamma^*$ or $\chi \in \Gamma^*$ (by induction hypothesis). This is the case iff $(\psi \vee \chi) \in \Gamma^*$ (by Proposition 10.2(3)).

6. $\varphi \equiv \psi \supset \chi$: exercise.

7. $\varphi \equiv \forall x\, \psi(x)$: exercise.

8. $\varphi \equiv \exists x\, \psi(x)$: $\mathfrak{M}(\Gamma^*) \vDash \varphi$ iff $\mathfrak{M}(\Gamma^*) \vDash \psi(t)$ for at least one term t (Proposition 10.11). By induction hypothesis, this is the case iff $\psi(t) \in \Gamma^*$ for at least one term t. By Proposition 10.7, this in turn is the case iff $\exists x\, \psi(x) \in \Gamma^*$. □

10.7 Identity

The construction of the term model given in the preceding section is enough to establish completeness for first-order logic for sets Γ that do not contain $=$. The term model satisfies every $\varphi \in \Gamma^*$ which does not contain $=$ (and hence all $\varphi \in \Gamma$). It does not work, however, if $=$ is present. The reason is that Γ^* then may contain a sentence $t = t'$, but in the term model the value of any term is that term itself. Hence, if t and t' are different terms, their values in the term model—i.e., t and t', respectively—are different, and so $t = t'$ is false. We can fix this, however, using a construction known as "factoring."

Definition 10.13. Let Γ^* be a consistent and complete set of sentences in $\mathcal{L}$. We define the relation $\approx$ on the set of closed terms of $\mathcal{L}$ by

$$t \approx t' \quad \text{iff} \quad t = t' \in \Gamma^*$$

Proposition 10.14. *The relation $\approx$ has the following properties:*

1. *$\approx$ is reflexive.*
2. *$\approx$ is symmetric.*
3. *$\approx$ is transitive.*
4. *If $t \approx t'$, f is a function symbol, and $t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n$ are closed terms, then*

$$f(t_1, \ldots, t_{i-1}, t, t_{i+1}, \ldots, t_n) \approx f(t_1, \ldots, t_{i-1}, t', t_{i+1}, \ldots, t_n).$$

5. *If $t \approx t'$, R is a predicate symbol, and $t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n$ are closed terms, then*

$$R(t_1, \ldots, t_{i-1}, t, t_{i+1}, \ldots, t_n) \in \Gamma^* \text{ iff } R(t_1, \ldots, t_{i-1}, t', t_{i+1}, \ldots, t_n) \in \Gamma^*.$$

Proof. Since Γ^* is consistent and complete, $t = t' \in \Gamma^*$ iff $\Gamma^* \vdash t = t'$. Thus it is enough to show the following:

1. $\Gamma^* \vdash t = t$ for all closed terms t.
2. If $\Gamma^* \vdash t = t'$ then $\Gamma^* \vdash t' = t$.
3. If $\Gamma^* \vdash t = t'$ and $\Gamma^* \vdash t' = t''$, then $\Gamma^* \vdash t = t''$.
4. If $\Gamma^* \vdash t = t'$, then

$$\Gamma^* \vdash f(t_1, \ldots, t_{i-1}, t, t_{i+1}, , \ldots, t_n) = f(t_1, \ldots, t_{i-1}, t', t_{i+1}, \ldots, t_n)$$

   for every n-place function symbol f and closed terms $t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n$.
5. If $\Gamma^* \vdash t = t'$ and $\Gamma^* \vdash R(t_1, \ldots, t_{i-1}, t, t_{i+1}, \ldots, t_n)$, then $\Gamma^* \vdash R(t_1, \ldots, t_{i-1}, t', t_{i+1}, \ldots, t_n)$ for every n-place predicate symbol R and closed terms $t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n$. □

Definition 10.15. Suppose Γ^* is a consistent and complete set in a language $\mathcal{L}$, t is a closed term, and $\approx$ as in the previous definition. Then:

$$[t]_\approx = \{t' \mid t' \in \mathrm{Trm}(\mathcal{L}), t \approx t'\}$$

and $\mathrm{Trm}(\mathcal{L})/_\approx = \{[t]_\approx \mid t \in \mathrm{Trm}(\mathcal{L})\}$.

Definition 10.16. Let $\mathfrak{M} = \mathfrak{M}(\Gamma^*)$ be the term model for Γ^* from Definition 10.9. Then $\mathfrak{M}/_\approx$ is the following structure:

1. $|\mathfrak{M}/_\approx| = \mathrm{Trm}(\mathcal{L})/_\approx$.

2. $c^{\mathfrak{M}/\approx} = [c]_\approx$
3. $f^{\mathfrak{M}/\approx}([t_1]_\approx, \dots, [t_n]_\approx) = [f(t_1, \dots, t_n)]_\approx$
4. $\langle [t_1]_\approx, \dots, [t_n]_\approx \rangle \in R^{\mathfrak{M}/\approx}$ iff $\mathfrak{M} \vDash R(t_1, \dots, t_n)$, i.e., iff $R(t_1, \dots, t_n) \in \Gamma^*$.

Note that we have defined $f^{\mathfrak{M}/\approx}$ and $R^{\mathfrak{M}/\approx}$ for elements of $\mathrm{Trm}(\mathcal{L})/_\approx$ by referring to them as $[t]_\approx$, i.e., via *representatives* $t \in [t]_\approx$. We have to make sure that these definitions do not depend on the choice of these representatives, i.e., that for some other choices t' which determine the same equivalence classes ($[t]_\approx = [t']_\approx$), the definitions yield the same result. For instance, if R is a one-place predicate symbol, the last clause of the definition says that $[t]_\approx \in R^{\mathfrak{M}/\approx}$ iff $\mathfrak{M} \vDash R(t)$. If for some other term t' with $t \approx t'$, $\mathfrak{M} \nvDash R(t)$, then the definition would require $[t']_\approx \notin R^{\mathfrak{M}/\approx}$. If $t \approx t'$, then $[t]_\approx = [t']_\approx$, but we can't have both $[t]_\approx \in R^{\mathfrak{M}/\approx}$ and $[t]_\approx \notin R^{\mathfrak{M}/\approx}$. However, Proposition 10.14 guarantees that this cannot happen.

Proposition 10.17. *$\mathfrak{M}/_\approx$ is well defined, i.e., if $t_1, \dots, t_n, t'_1, \dots, t'_n$ are closed terms, and $t_i \approx t'_i$ then*

1. *$[f(t_1, \dots, t_n)]_\approx = [f(t'_1, \dots, t'_n)]_\approx$, i.e.,*
$$f(t_1, \dots, t_n) \approx f(t'_1, \dots, t'_n)$$
and
2. *$\mathfrak{M} \vDash R(t_1, \dots, t_n)$ iff $\mathfrak{M} \vDash R(t'_1, \dots, t'_n)$, i.e.,*
$$R(t_1, \dots, t_n) \in \Gamma^* \text{ iff } R(t'_1, \dots, t'_n) \in \Gamma^*.$$

Proof. Follows from Proposition 10.14 by induction on n. □

As in the case of the term model, before proving the truth lemma we need the following lemma.

Lemma 10.18. *Let $\mathfrak{M} = \mathfrak{M}(\Gamma^*)$, then $\mathrm{Val}^{\mathfrak{M}/\approx}(t) = [t]_\approx$.*

Proof. The proof is similar to that of Lemma 10.10. □

Lemma 10.19. *$\mathfrak{M}/_\approx \vDash \varphi$ iff $\varphi \in \Gamma^*$ for all sentences φ.*

Proof. By induction on φ, just as in the proof of Lemma 10.12. The only case that needs additional attention is when $\varphi \equiv t = t'$.

$$\begin{aligned} \mathfrak{M}/_\approx \vDash t = t' &\text{ iff } [t]_\approx = [t']_\approx \text{ (by definition of } \mathfrak{M}/_\approx) \\ &\text{ iff } t \approx t' \text{ (by definition of } [t]_\approx) \\ &\text{ iff } t = t' \in \Gamma^* \text{ (by definition of } \approx). \end{aligned}$$
□

Note that while $\mathfrak{M}(\Gamma^*)$ is always countable and infinite, $\mathfrak{M}/_{\approx}$ may be finite, since it may turn out that there are only finitely many classes $[t]_{\approx}$. This is to be expected, since Γ may contain sentences which require any structure in which they are true to be finite. For instance, $\forall x\, \forall y\, x = y$ is a consistent sentence, but is satisfied only in structures with a domain that contains exactly one element.

10.8 The Completeness Theorem

Let's combine our results: we arrive at the completeness theorem.

Theorem 10.20 (Completeness Theorem). *Let Γ be a set of sentences. If Γ is consistent, it is satisfiable.*

Proof. Suppose Γ is consistent. By Lemma 10.6, there is a saturated consistent set $\Gamma' \supseteq \Gamma$. By Lemma 10.8, there is a $\Gamma^* \supseteq \Gamma'$ which is consistent and complete. Since $\Gamma' \subseteq \Gamma^*$, for each formula $\varphi(x)$, Γ^* contains a sentence of the form $\exists x\, \varphi(x) \supset \varphi(c)$ and so Γ^* is saturated. If Γ does not contain $=$, then by Lemma 10.12, $\mathfrak{M}(\Gamma^*) \vDash \varphi$ iff $\varphi \in \Gamma^*$. From this it follows in particular that for all $\varphi \in \Gamma$, $\mathfrak{M}(\Gamma^*) \vDash \varphi$, so Γ is satisfiable. If Γ does contain $=$, then by Lemma 10.19, for all sentences φ, $\mathfrak{M}/_{\approx} \vDash \varphi$ iff $\varphi \in \Gamma^*$. In particular, $\mathfrak{M}/_{\approx} \vDash \varphi$ for all $\varphi \in \Gamma$, so Γ is satisfiable. □

Corollary 10.21 (Completeness Theorem, Second Version). *For all Γ and sentences φ: if $\Gamma \vDash \varphi$ then $\Gamma \vdash \varphi$.*

Proof. Note that the Γ's in Corollary 10.21 and Theorem 10.20 are universally quantified. To make sure we do not confuse ourselves, let us restate Theorem 10.20 using a different variable: for any set of sentences Δ, if Δ is consistent, it is satisfiable. By contraposition, if Δ is not satisfiable, then Δ is inconsistent. We will use this to prove the corollary.

Suppose that $\Gamma \vDash \varphi$. Then $\Gamma \cup \{{\sim}\varphi\}$ is unsatisfiable by Proposition 7.27. Taking $\Gamma \cup \{{\sim}\varphi\}$ as our Δ, the previous version of Theorem 10.20 gives us that $\Gamma \cup \{{\sim}\varphi\}$ is inconsistent. By Proposition 9.19, $\Gamma \vdash \varphi$. □

10.9 The Compactness Theorem

One important consequence of the completeness theorem is the compactness theorem. The compactness theorem states that if each *finite* subset of a set of sentences is satisfiable, the entire set is satisfiable—even if the set itself is infinite. This is far from obvious. There is nothing that seems to rule out, at first glance at least, the possibility of there being infinite sets of sentences which are contradictory, but the contradiction only arises, so to speak, from the infinite number. The compactness theorem says that such a scenario can be ruled out: there are no unsatisfiable infinite sets of sentences each finite subset of which is satisfiable. Like the completeness theorem, it has a version related to entailment: if an infinite set of sentences entails something, already a finite subset does.

Definition 10.22. A set Γ of formulae is *finitely satisfiable* iff every finite $\Gamma_0 \subseteq \Gamma$ is satisfiable.

Theorem 10.23 (Compactness Theorem). *The following hold for any sentences Γ and φ:*

1. *$\Gamma \vDash \varphi$ iff there is a finite $\Gamma_0 \subseteq \Gamma$ such that $\Gamma_0 \vDash \varphi$.*
2. *Γ is satisfiable iff it is finitely satisfiable.*

Proof. We prove (2). If Γ is satisfiable, then there is a structure $\mathfrak{M}$ such that $\mathfrak{M} \vDash \varphi$ for all $\varphi \in \Gamma$. Of course, this $\mathfrak{M}$ also satisfies every finite subset of Γ, so Γ is finitely satisfiable.

Now suppose that Γ is finitely satisfiable. Then every finite subset $\Gamma_0 \subseteq \Gamma$ is satisfiable. By soundness (Corollary 9.31), every finite subset is consistent. Then Γ itself must be consistent by Proposition 9.17. By completeness (Theorem 10.20), since Γ is consistent, it is satisfiable. □

Example 10.24. In every model $\mathfrak{M}$ of a theory Γ, each term t of course picks out an element of $|\mathfrak{M}|$. Can we guarantee that it is also true that every element of $|\mathfrak{M}|$ is picked out by some term or other? In other words, are there theories Γ all models of which are covered? The compactness theorem shows that this is not the case if Γ has infinite models. Here's how to see this: Let $\mathfrak{M}$ be an infinite model of Γ, and let c be a constant symbol not in the language of Γ. Let Δ be the set of all sentences $c \neq t$ for t a term in the language $\mathcal{L}$ of Γ, i.e.,

$$\Delta = \{c \neq t \mid t \in \mathrm{Trm}(\mathcal{L})\}.$$

A finite subset of $\Gamma \cup \Delta$ can be written as $\Gamma' \cup \Delta'$, with $\Gamma' \subseteq \Gamma$ and $\Delta' \subseteq \Delta$. Since Δ' is finite, it can contain only finitely many terms. Let $a \in |\mathfrak{M}|$ be an element of $|\mathfrak{M}|$ not picked out by any of them, and let $\mathfrak{M}'$ be the structure that is just like $\mathfrak{M}$, but also $c^{\mathfrak{M}'} = a$. Since $a \neq \mathrm{Val}^{\mathfrak{M}}(t)$ for all t occurring in Δ', $\mathfrak{M}' \vDash \Delta'$. Since $\mathfrak{M} \vDash \Gamma$, $\Gamma' \subseteq \Gamma$, and c does not occur in Γ, also $\mathfrak{M}' \vDash \Gamma'$. Together, $\mathfrak{M}' \vDash \Gamma' \cup \Delta'$ for every finite subset $\Gamma' \cup \Delta'$ of $\Gamma \cup \Delta$. So every finite subset of $\Gamma \cup \Delta$ is satisfiable. By compactness, $\Gamma \cup \Delta$ itself is satisfiable. So there are models $\mathfrak{M} \vDash \Gamma \cup \Delta$. Every such $\mathfrak{M}$ is a model of Γ, but is not covered, since $\mathrm{Val}^{\mathfrak{M}}(c) \neq \mathrm{Val}^{\mathfrak{M}}(t)$ for all terms t of $\mathcal{L}$.

Example 10.25. Consider a language $\mathcal{L}$ containing the predicate symbol $<$, constant symbols $\mathtt{0}$, $\mathtt{1}$, and function symbols $+$, $\times$, and $-$. Let Γ be the set of all sentences in this language true in the structure $\mathfrak{Q}$ with domain $\mathbb{Q}$ and the obvious interpretations. Γ is the set of all sentences of $\mathcal{L}$ true about the rational numbers. Of course, in $\mathbb{Q}$ (and even in $\mathbb{R}$), there are no numbers r which are greater than 0 but less than $1/k$ for all $k \in \mathbb{Z}^+$. Such a number, if it existed, would be an *infinitesimal:* non-zero, but infinitely small. The compactness theorem can be used to show that there are models of Γ in which infinitesimals exist. We do not have a function symbol for division in our language (division by zero is undefined, and function symbols have to be interpreted by total functions). However, we can still express that $r < 1/k$, since this is the case iff $r \cdot k < 1$. Now let c be a new constant symbol and let Δ be

$$\{0 < c\} \cup \{c \times \overline{k} < 1 \mid k \in \mathbb{Z}^+\}$$

(where $\overline{k} = (1 + (1 + \cdots + (1 + 1) \ldots))$ with k 1's). For any finite subset Δ_0 of Δ there is a K such that for all the sentences $c \times \overline{k} < 1$ in Δ_0 have $k < K$. If we expand $\mathfrak{Q}$ to $\mathfrak{Q}'$ with $c^{\mathfrak{Q}'} = 1/K$ we have that $\mathfrak{Q}' \vDash \Gamma \cup \Delta_0$, and so $\Gamma \cup \Delta$ is finitely satisfiable (Exercise: prove this in detail). By compactness, $\Gamma \cup \Delta$ is satisfiable. Any model $\mathfrak{S}$ of $\Gamma \cup \Delta$ contains an infinitesimal, namely $c^{\mathfrak{S}}$.

Example 10.26. We know that first-order logic with identity predicate can express that the size of the domain must have some minimal size: The sentence $\varphi_{\geq n}$ (which says "there are at least n distinct objects") is true only in structures where $|\mathfrak{M}|$ has at least n objects. So if we take

$$\Delta = \{\varphi_{\geq n} \mid n \geq 1\}$$

then any model of Δ must be infinite. Thus, we can guarantee that a theory only has infinite models by adding Δ to it: the models of $\Gamma \cup \Delta$ are all and only the infinite models of Γ.

So first-order logic can express infinitude. The compactness theorem shows that it cannot express finitude, however. For suppose some set of sentences Λ were satisfied in all and only finite structures. Then $\Delta \cup \Lambda$ is finitely satisfiable. Why? Suppose $\Delta' \cup \Lambda' \subseteq \Delta \cup \Lambda$ is finite with $\Delta' \subseteq \Delta$ and $\Lambda' \subseteq \Lambda$. Let n be the largest number such that $\varphi_{\geq n} \in \Delta'$. Λ, being satisfied in all finite structures, has a model $\mathfrak{M}$ with finitely many but $\geq n$ elements. But then $\mathfrak{M} \vDash \Delta' \cup \Lambda'$. By compactness, $\Delta \cup \Lambda$ has an infinite model, contradicting the assumption that Λ is satisfied only in finite structures.

10.10 A Direct Proof of the Compactness Theorem

We can prove the Compactness Theorem directly, without appealing to the Completeness Theorem, using the same ideas as in the proof of the completeness theorem. In the proof of the Completeness Theorem we started with a consistent set Γ of sentences, expanded it to a consistent, saturated, and complete set Γ^* of sentences, and then showed that in the term model $\mathfrak{M}(\Gamma^*)$ constructed from Γ^*, all sentences of Γ are true, so Γ is satisfiable.

We can use the same method to show that a finitely satisfiable set of sentences is satisfiable. We just have to prove the corresponding versions of the results leading to the truth lemma where we replace "consistent" with "finitely satisfiable."

Proposition 10.27. *Suppose Γ is complete and finitely satisfiable. Then:*

1. $(\varphi \mathbin{\&} \psi) \in \Gamma$ *iff both* $\varphi \in \Gamma$ *and* $\psi \in \Gamma$.
2. $(\varphi \lor \psi) \in \Gamma$ *iff either* $\varphi \in \Gamma$ *or* $\psi \in \Gamma$.
3. $(\varphi \supset \psi) \in \Gamma$ *iff either* $\varphi \notin \Gamma$ *or* $\psi \in \Gamma$.

Lemma 10.28. *Every finitely satisfiable set Γ can be extended to a saturated finitely satisfiable set Γ'.*

Proposition 10.29. *Suppose Γ is complete, finitely satisfiable, and saturated.*

1. $\exists x\, \varphi(x) \in \Gamma$ *iff* $\varphi(t) \in \Gamma$ *for at least one closed term t.*
2. $\forall x\, \varphi(x) \in \Gamma$ *iff* $\varphi(t) \in \Gamma$ *for all closed terms t.*

Lemma 10.30. *Every finitely satisfiable set Γ can be extended to a complete and finitely satisfiable set Γ^*.*

Theorem 10.31 (Compactness). *Γ is satisfiable if and only if it is finitely satisfiable.*

Proof. If Γ is satisfiable, then there is a structure $\mathfrak{M}$ such that $\mathfrak{M} \models \varphi$ for all $\varphi \in \Gamma$. Of course, this $\mathfrak{M}$ also satisfies every finite subset of Γ, so Γ is finitely satisfiable.

Now suppose that Γ is finitely satisfiable. By Lemma 10.28, there is a finitely satisfiable, saturated set $\Gamma' \supseteq \Gamma$. By Lemma 10.30, Γ' can be extended to a complete and finitely satisfiable set Γ^*, and Γ^* is still saturated. Construct the term model $\mathfrak{M}(\Gamma^*)$ as in Definition 10.9. Note that Proposition 10.11 did not rely on the fact that Γ^* is consistent (or complete or saturated, for that matter), but just on the fact that $\mathfrak{M}(\Gamma^*)$ is covered. The proof of the Truth Lemma (Lemma 10.12) goes through if we replace references to Proposition 10.2 and Proposition 10.7 by references to Proposition 10.27 and Proposition 10.29 □

10.11 The Löwenheim–Skolem Theorem

The Löwenheim–Skolem Theorem says that if a theory has an infinite model, then it also has a model that is at most countably infinite. An immediate consequence of this fact is that first-order logic cannot express that the size of a structure is uncountable: any sentence or set of sentences satisfied in all uncountable structures is also satisfied in some countable structure.

Theorem 10.32. *If Γ is consistent then it has a countable model, i.e., it is satisfiable in a structure whose domain is either finite or countably infinite.*

Proof. If Γ is consistent, the structure $\mathfrak{M}$ delivered by the proof of the completeness theorem has a domain $|\mathfrak{M}|$ that is no larger than the set of the terms of the language $\mathcal{L}$. So $\mathfrak{M}$ is at most countably infinite. □

Theorem 10.33. *If Γ is a consistent set of sentences in the language of first-order logic without identity, then it has a countably infinite model, i.e., it is satisfiable in a structure whose domain is infinite and countable.*

Proof. If Γ is consistent and contains no sentences in which identity appears, then the structure $\mathfrak{M}$ delivered by the proof of the completeness theorem has a domain $|\mathfrak{M}|$ identical to the set of terms of the language $\mathcal{L}'$. So $\mathfrak{M}$ is countably infinite, since $\mathrm{Trm}(\mathcal{L}')$ is. □

Example 10.34 (Skolem's Paradox). Zermelo–Fraenkel set theory **ZFC** is a very powerful framework in which practically all mathematical statements can be expressed, including facts about the sizes of sets. So for instance, **ZFC** can prove that the set $\mathbb{R}$ of real numbers is uncountable, it can prove Cantor's Theorem that the power set of any set is larger than the set itself, etc. If **ZFC** is consistent, its models are all infinite, and moreover, they all contain elements about which the theory says that they are uncountable, such as the element that makes true the theorem of **ZFC** that the power set of the natural numbers exists. By the Löwenheim–Skolem Theorem, **ZFC** also has countable models—models that contain "uncountable" sets but which themselves are countable.

Problems

Problem 10.1. Complete the proof of Proposition 10.2.

Problem 10.2. Complete the proof of Proposition 10.11.

Problem 10.3. Complete the proof of Lemma 10.12.

Problem 10.4. Complete the proof of Proposition 10.14.

Problem 10.5. Complete the proof of Lemma 10.18.

Problem 10.6. Use Corollary 10.21 to prove Theorem 10.20, thus showing that the two formulations of the completeness theorem are equivalent.

Problem 10.7. In order for a derivation system to be complete, its rules must be strong enough to prove every unsatisfiable set inconsistent. Which of the rules of derivation were necessary to prove completeness? Are any of these rules not used anywhere in the proof? In order to answer these questions, make a list or diagram that shows which of the rules of derivation were used in which results that lead up to the proof of Theorem 10.20. Be sure to note any tacit uses of rules in these proofs.

Problem 10.8. Prove (1) of Theorem 10.23.

Problem 10.9. In the standard model of arithmetic $\mathfrak{N}$, there is no element $k \in |\mathfrak{N}|$ which satisfies every formula $\overline{n} < x$ (where $\overline{n}$ is $0'^{\dots\prime}$ with n $\prime$'s). Use the compactness theorem to show that the set of sentences in the language of arithmetic which are true in the standard model of arithmetic $\mathfrak{N}$ are also true in a structure $\mathfrak{N}'$ that contains an element which *does* satisfy every formula $\overline{n} < x$.

Problem 10.10. Prove Proposition 10.27. Avoid the use of $\vdash$.

Problem 10.11. Prove Lemma 10.28. (Hint: The crucial step is to show that if Γ_n is finitely satisfiable, so is $\Gamma_n \cup \{\theta_n\}$, without any appeal to derivations or consistency.)

Problem 10.12. Prove Proposition 10.29.

Problem 10.13. Prove Lemma 10.30. (Hint: the crucial step is to show that if Γ_n is finitely satisfiable, then either $\Gamma_n \cup \{\varphi_n\}$ or $\Gamma_n \cup \{{\sim}\varphi_n\}$ is finitely satisfiable.)

Problem 10.14. Write out the complete proof of the Truth Lemma (Lemma 10.12) in the version required for the proof of Theorem 10.31.

Chapter 11

Beyond First-order Logic

11.1 Overview

First-order logic is not the only system of logic of interest: there are many extensions and variations of first-order logic. A logic typically consists of the formal specification of a language, usually, but not always, a deductive system, and usually, but not always, an intended semantics. But the technical use of the term raises an obvious question: what do logics that are not first-order logic have to do with the word "logic," used in the intuitive or philosophical sense? All of the systems described below are designed to model reasoning of some form or another; can we say what makes them logical?

No easy answers are forthcoming. The word "logic" is used in different ways and in different contexts, and the notion, like that of "truth," has been analyzed from numerous philosophical stances. For example, one might take the goal of logical reasoning to be the determination of which statements are necessarily true, true a priori, true independent of the interpretation of the nonlogical terms, true by virtue of their form, or true by linguistic convention; and each of these conceptions requires a good deal of clarification. Even if one restricts one's attention to the kind of logic used in mathematics, there is little agreement as to its scope. For example, in the *Principia Mathematica*, Russell and Whitehead tried to develop mathematics on the basis of logic, in the *logicist* tradition begun by Frege. Their system of logic was a form of higher-type logic similar to the one described below. In the end they were forced to introduce axioms which, by most standards, do not seem purely logical (notably, the axiom of infinity, and the axiom of reducibility), but one might nonetheless hold that some forms of higher-order reasoning should be accepted as logical. In contrast, Quine, whose ontology does not admit "propositions" as legitimate objects of discourse, argues that second-order and higher-order logic are really manifestations of set theory in sheep's clothing; in other words, systems involving quantification over predicates are not purely logical.

For now, it is best to leave such philosophical issues for a rainy day, and simply think of the systems below as formal idealizations of various kinds of reasoning, logical or otherwise.

11.2 Many-Sorted Logic

In first-order logic, variables and quantifiers range over a single domain. But it is often useful to have multiple (disjoint) domains: for example, you might want to have a domain of numbers, a domain of geometric objects, a domain of functions from numbers to numbers, a domain of abelian groups, and so on.

Many-sorted logic provides this kind of framework. One starts with a list of "sorts"—the "sort" of an object indicates the "domain" it is supposed to inhabit. One then has variables and quantifiers for each sort, and (usually) an identity predicate for each sort. Functions and relations are also "typed" by the sorts of objects they can take as arguments. Otherwise, one keeps the usual rules of first-order logic, with versions of the quantifier-rules repeated for each sort.

For example, to study international relations we might choose a language with two sorts of objects, French citizens and German citizens. We might have a unary relation, "drinks wine," for objects of the first sort; another unary relation, "eats wurst," for objects of the second sort; and a binary relation, "forms a multinational married couple," which takes two arguments, where the first argument is of the first sort and the second argument is of the second sort. If we use variables a, b, c to range over French citizens and x, y, z to range over German citizens, then

$$\forall a\, \forall x[(MarriedTo(a, x) \supset (DrinksWine(a) \lor \sim EatsWurst(x))]]$$

asserts that if any French person is married to a German, either the French person drinks wine or the German doesn't eat wurst.

Many-sorted logic can be embedded in first-order logic in a natural way, by lumping all the objects of the many-sorted domains together into one first-order domain, using unary predicate symbols to keep track of the sorts, and relativizing quantifiers. For example, the first-order language corresponding to the example above would have unary predicate symbols "$German$" and "$French$," in addition to the other relations described, with the sort requirements erased. A sorted quantifier $\forall x\, \varphi$, where x is a variable of the German sort, translates to

$$\forall x\, (German(x) \supset \varphi).$$

We need to add axioms that insure that the sorts are separate—e.g., $\forall x \sim (German(x)\, \&\, French(x))$—as well as axioms that guarantee that "drinks wine" only holds of objects satisfying the predicate $French(x)$, etc. With these conventions and axioms, it is not difficult to show that many-sorted sentences translate to first-order sentences, and many-sorted derivations translate to first-order derivations. Also, many-sorted structures "translate" to corresponding first-order structures and vice-versa, so we also have a completeness theorem for many-sorted logic.

11.3 Second-Order logic

The language of second-order logic allows one to quantify not just over a domain of individuals, but over relations on that domain as well. Given a first-order language $\mathcal{L}$, for each k one adds variables R which range over k-ary relations, and allows quantification over those variables. If R is a variable for a k-ary relation, and $t_1, \ldots, t_k$ are ordinary (first-order) terms, $R(t_1, \ldots, t_k)$ is an atomic formula. Otherwise, the set of formulae is defined just as in the case of first-order logic, with additional clauses for second-order quantification. Note that we only have the identity predicate for first-order terms: if R and S are relation variables of the same arity k, we can define $R = S$ to be an abbreviation for

$$\forall x_1 \ldots \forall x_k \, (R(x_1, \ldots, x_k) \equiv S(x_1, \ldots, x_k)).$$

The rules for second-order logic simply extend the quantifier rules to the new second order variables. Here, however, one has to be a little bit careful to explain how these variables interact with the predicate symbols of $\mathcal{L}$, and with formulae of $\mathcal{L}$ more generally. At the bare minimum, relation variables count as terms, so one has inferences of the form

$$\varphi(R) \vdash \exists R \, \varphi(R)$$

But if $\mathcal{L}$ is the language of arithmetic with a constant relation symbol $<$, one would also expect the following inference to be valid:

$$x < y \vdash \exists R \, R(x, y)$$

or for a given formula φ,

$$\varphi(x_1, \ldots, x_k) \vdash \exists R \, R(x_1, \ldots, x_k)$$

More generally, we might want to allow inferences of the form

$$\varphi[\lambda \vec{x}. \, \psi(\vec{x})/R] \vdash \exists R \, \varphi$$

where $\varphi[\lambda \vec{x}. \, \psi(\vec{x})/R]$ denotes the result of replacing every atomic formula of the form $Rt_1, \ldots, t_k$ in φ by $\psi(t_1, \ldots, t_k)$. This last rule is equivalent to having a *comprehension schema*, i.e., an axiom of the form

$$\exists R \, \forall x_1, \ldots, x_k \, (\varphi(x_1, \ldots, x_k) \equiv R(x_1, \ldots, x_k)),$$

one for each formula φ in the second-order language, in which R is not a free variable. (Exercise: show that if R is allowed to occur in φ, this schema is inconsistent!)

When logicians refer to the "axioms of second-order logic" they usually mean the minimal extension of first-order logic by second-order quantifier rules together with the comprehension schema. But it is often interesting to study weaker subsystems of these axioms and rules. For example, note that in its full generality the axiom schema of comprehension is *impredicative*: it allows one to assert the existence of a relation $R(x_1, \ldots, x_k)$ that is "defined" by a formula with second-order quantifiers; and these quantifiers range over the set of all such relations—a set which includes R itself! Around the turn of the twentieth century, a common reaction to Russell's paradox was to lay the blame on such definitions, and to avoid them in developing the foundations of mathematics. If one prohibits the use of second-order quantifiers in the formula φ, one has a *predicative* form of comprehension, which is somewhat weaker.

From the semantic point of view, one can think of a second-order structure as consisting of a first-order structure for the language, coupled with a set of relations on the domain over which the second-order quantifiers range (more precisely, for each k there is a set of relations of arity k). Of course, if comprehension is included in the derivation system, then we have the added requirement that there are enough relations in the "second-order part" to satisfy the comprehension axioms—otherwise the derivation system is not sound! One easy way to insure that there are enough relations around is to take the second-order part to consist of *all* the relations on the first-order part. Such a structure is called *full*, and, in a sense, is really the "intended structure" for the language. If we restrict our attention to full structures we have what is known as the *full* second-order semantics. In that case, specifying a structure boils down to specifying the first-order part, since the contents of the second-order part follow from that implicitly.

To summarize, there is some ambiguity when talking about second-order logic. In terms of the derivation system, one might have in mind either

1. A "minimal" second-order derivation system, together with some comprehension axioms.

2. The "standard" second-order derivation system, with full comprehension.

In terms of the semantics, one might be interested in either

1. The "weak" semantics, where a structure consists of a first-order part, together with a second-order part big enough to satisfy the comprehension axioms.

2. The "standard" second-order semantics, in which one considers full structures only.

When logicians do not specify the derivation system or the semantics they have in mind, they are usually referring to the second item on each list. The advantage to using this semantics is that, as we will see, it gives us categorical descriptions of many natural mathematical structures; at the same time, the derivation system is quite strong, and sound for this semantics. The drawback is that the derivation system is *not* complete for the semantics; in fact, *no* effectively given derivation system is complete for the full second-order semantics. On the other hand, we will see that the derivation system *is* complete for the weakened semantics; this implies that if a sentence is not provable, then there is *some* structure, not necessarily the full one, in which it is false.

The language of second-order logic is quite rich. One can identify unary relations with subsets of the domain, and so in particular you can quantify over these sets; for example, one can express induction for the natural numbers with a single axiom

$$\forall R\, ((R(\mathrm{o}) \,\&\, \forall x\, (R(x) \supset R(x'))) \supset \forall x\, R(x)).$$

If one takes the language of arithmetic to have symbols $\mathrm{o}, {\prime}, +, \times$ and $<$, one can add the following axioms to describe their behavior:

1. $\forall x \sim x' = \mathrm{o}$
2. $\forall x\, \forall y\, (s(x) = s(y) \supset x = y)$
3. $\forall x\, (x + \mathrm{o}) = x$
4. $\forall x\, \forall y\, (x + y') = (x + y)'$
5. $\forall x\, (x \times \mathrm{o}) = \mathrm{o}$
6. $\forall x\, \forall y\, (x \times y') = ((x \times y) + x)$
7. $\forall x\, \forall y\, (x < y \equiv \exists z\, y = (x + z'))$

It is not difficult to show that these axioms, together with the axiom of induction above, provide a categorical description of the structure $\mathfrak{N}$, the standard model of arithmetic, provided we are using the full second-order semantics. Given any structure $\mathfrak{M}$ in which these axioms are true, define a function f from $\mathbb{N}$ to the domain of $\mathfrak{M}$ using ordinary recursion on $\mathbb{N}$, so that $f(0) = \mathrm{o}^{\mathfrak{M}}$ and $f(x+1) = {\prime}^{\mathfrak{M}}(f(x))$. Using ordinary induction on $\mathbb{N}$ and the fact that axioms (1) and (2) hold in $\mathfrak{M}$, we see that f is injective. To see that f is surjective, let P be the set of elements of $|\mathfrak{M}|$ that are in the range of f. Since $\mathfrak{M}$ is full, P is in the second-order domain. By the construction of f, we know that $\mathrm{o}^{\mathfrak{M}}$ is in P, and that P is closed under ${\prime}^{\mathfrak{M}}$. The fact that the induction axiom holds in $\mathfrak{M}$

(in particular, for P) guarantees that P is equal to the entire first-order domain of $\mathfrak{M}$. This shows that f is a bijection. Showing that f is a homomorphism is no more difficult, using ordinary induction on $\mathbb{N}$ repeatedly.

In set-theoretic terms, a function is just a special kind of relation; for example, a unary function f can be identified with a binary relation R satisfying $\forall x\, \exists! y\, R(x,y)$. As a result, one can quantify over functions too. Using the full semantics, one can then define the class of infinite structures to be the class of structures $\mathfrak{M}$ for which there is an injective function from the domain of $\mathfrak{M}$ to a proper subset of itself:

$$\exists f\, (\forall x\, \forall y\, (f(x) = f(y) \supset x = y)\ \&\ \exists y\, \forall x\, f(x) \neq y).$$

The negation of this sentence then defines the class of finite structures.

In addition, one can define the class of well-orderings, by adding the following to the definition of a linear ordering:

$$\forall P\, (\exists x\, P(x) \supset \exists x\, (P(x)\ \&\ \forall y\, (y < x \supset {\sim}P(y)))).$$

This asserts that every non-empty set has a least element, modulo the identification of “set” with “one-place relation”. For another example, one can express the notion of connectedness for graphs, by saying that there is no nontrivial separation of the vertices into disconnected parts:

$${\sim}\exists A\, (\exists x\, A(x)\ \&\ \exists y\, {\sim}A(y)\ \&\ \forall w\, \forall z\, ((A(w)\ \&\ {\sim}A(z)) \supset {\sim}R(w,z))).$$

For yet another example, you might try as an exercise to define the class of finite structures whose domain has even size. More strikingly, one can provide a categorical description of the real numbers as a complete ordered field containing the rationals.

In short, second-order logic is much more expressive than first-order logic. That’s the good news; now for the bad. We have already mentioned that there is no effective derivation system that is complete for the full second-order semantics. For better or for worse, many of the properties of first-order logic are absent, including compactness and the Löwenheim–Skolem theorems.

On the other hand, if one is willing to give up the full second-order semantics in terms of the weaker one, then the minimal second-order derivation system is complete for this semantics. In other words, if we read $\vdash$ as “proves in the minimal system” and $\vDash$ as “logically implies in the weaker semantics”, we can show that whenever $\Gamma \vDash \varphi$ then $\Gamma \vdash \varphi$. If one wants to include specific comprehension axioms in the derivation system, one has to restrict the semantics to second-order structures that satisfy these axioms: for example, if Δ consists of a set of comprehension axioms (possibly all of them), we have that if $\Gamma \cup \Delta \vDash \varphi$, then $\Gamma \cup \Delta \vdash \varphi$. In particular, if φ is not provable using the comprehension axioms we are considering, then there is a model of ${\sim}\varphi$ in which these comprehension axioms nonetheless hold.

The easiest way to see that the completeness theorem holds for the weaker semantics is to think of second-order logic as a many-sorted logic, as follows. One sort is interpreted as the ordinary "first-order" domain, and then for each k we have a domain of "relations of arity k." We take the language to have built-in relation symbols "$true_k(R, x_1, \ldots, x_k)$" which is meant to assert that R holds of $x_1, \ldots, x_k$, where R is a variable of the sort "k-ary relation" and x_1, $\ldots$, x_k are objects of the first-order sort.

With this identification, the weak second-order semantics is essentially the usual semantics for many-sorted logic; and we have already observed that many-sorted logic can be embedded in first-order logic. Modulo the translations back and forth, then, the weaker conception of second-order logic is really a form of first-order logic in disguise, where the domain contains both "objects" and "relations" governed by the appropriate axioms.

11.4 Higher-Order logic

Passing from first-order logic to second-order logic enabled us to talk about sets of objects in the first-order domain, within the formal language. Why stop there? For example, third-order logic should enable us to deal with sets of sets of objects, or perhaps even sets which contain both objects and sets of objects. And fourth-order logic will let us talk about sets of objects of that kind. As you may have guessed, one can iterate this idea arbitrarily.

In practice, higher-order logic is often formulated in terms of functions instead of relations. (Modulo the natural identifications, this difference is inessential.) Given some basic "sorts" A, B, C, $\ldots$ (which we will now call "types"), we can create new ones by stipulating

> If σ and τ are finite types then so is $\sigma \to \tau$.

Think of types as syntactic "labels," which classify the objects we want in our domain; $\sigma \to \tau$ describes those objects that are functions which take objects of type σ to objects of type τ. For example, we might want to have a type Ω of truth values, "true" and "false," and a type $\mathbb{N}$ of natural numbers. In that case, you can think of objects of type $\mathbb{N} \to \Omega$ as unary relations, or subsets of $\mathbb{N}$; objects of type $\mathbb{N} \to \mathbb{N}$ are functions from natural numbers to natural numbers; and objects of type $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ are "functionals," that is, higher-type functions that take functions to numbers.

As in the case of second-order logic, one can think of higher-order logic as a kind of many-sorted logic, where there is a sort for each type of object we want to consider. But it is usually clearer just to define the syntax of higher-type logic from the ground up. For example, we can define a set of finite types inductively, as follows:

1. $\mathbb{N}$ is a finite type.

2. If σ and τ are finite types, then so is $\sigma \to \tau$.

3. If σ and τ are finite types, so is $\sigma \times \tau$.

Intuitively, $\mathbb{N}$ denotes the type of the natural numbers, $\sigma \to \tau$ denotes the type of functions from σ to τ, and $\sigma \times \tau$ denotes the type of pairs of objects, one from σ and one from τ. We can then define a set of terms inductively, as follows:

1. For each type σ, there is a stock of variables $x, y, z, \ldots$ of type σ

2. 0 is a term of type $\mathbb{N}$

3. S (successor) is a term of type $\mathbb{N} \to \mathbb{N}$

4. If s is a term of type σ, and t is a term of type $\mathbb{N} \to (\sigma \to \sigma)$, then R_{st} is a term of type $\mathbb{N} \to \sigma$

5. If s is a term of type $\tau \to \sigma$ and t is a term of type τ, then $s(t)$ is a term of type σ

6. If s is a term of type σ and x is a variable of type τ, then $\lambda x. s$ is a term of type $\tau \to \sigma$.

7. If s is a term of type σ and t is a term of type τ, then $\langle s, t \rangle$ is a term of type $\sigma \times \tau$.

8. If s is a term of type $\sigma \times \tau$ then $p_1(s)$ is a term of type σ and $p_2(s)$ is a term of type τ.

Intuitively, R_{st} denotes the function defined recursively by

$$R_{st}(0) = s$$
$$R_{st}(x+1) = t(x, R_{st}(x)),$$

$\langle s, t \rangle$ denotes the pair whose first component is s and whose second component is t, and $p_1(s)$ and $p_2(s)$ denote the first and second elements ("projections") of s. Finally, $\lambda x. s$ denotes the function f defined by

$$f(x) = s$$

for any x of type σ; so item (6) gives us a form of comprehension, enabling us to define functions using terms. Formulae are built up from identity predicate statements $s = t$ between terms of the same type, the usual propositional connectives, and higher-type quantification. One can then take the axioms of the system to be the basic equations governing the terms defined above, together with the usual rules of logic with quantifiers and identity predicate.

If one augments the finite type system with a type Ω of truth values, one has to include axioms which govern its use as well. In fact, if one is clever, one can get rid of complex formulae entirely, replacing them with terms of type Ω! The proof system can then be modified accordingly. The result is essentially the *simple theory of types* set forth by Alonzo Church in the 1930s.

As in the case of second-order logic, there are different versions of higher-type semantics that one might want to use. In the full version, variables of type $\sigma \to \tau$ range over the set of *all* functions from the objects of type σ to objects of type τ. As you might expect, this semantics is too strong to admit a complete, effective derivation system. But one can consider a weaker semantics, in which a structure consists of sets of elements T_τ for each type τ, together with appropriate operations for application, projection, etc. If the details are carried out correctly, one can obtain completeness theorems for the kinds of derivation systems described above.

Higher-type logic is attractive because it provides a framework in which we can embed a good deal of mathematics in a natural way: starting with $\mathbb{N}$, one can define real numbers, continuous functions, and so on. It is also particularly attractive in the context of intuitionistic logic, since the types have clear "constructive" interpretations. In fact, one can develop constructive versions of higher-type semantics (based on intuitionistic, rather than classical logic) that clarify these constructive interpretations quite nicely, and are, in many ways, more interesting than the classical counterparts.

11.5 Intuitionistic Logic

In contrast to second-order and higher-order logic, intuitionistic first-order logic represents a restriction of the classical version, intended to model a more "constructive" kind of reasoning. The following examples may serve to illustrate some of the underlying motivations.

Suppose someone came up to you one day and announced that they had determined a natural number x, with the property that if x is prime, the Riemann hypothesis is true, and if x is composite, the Riemann hypothesis is false. Great news! Whether the Riemann hypothesis is true or not is one of the big open questions of mathematics, and here they seem to have reduced the problem to one of calculation, that is, to the determination of whether a specific number is prime or not.

What is the magic value of x? They describe it as follows: x is the natural number that is equal to 7 if the Riemann hypothesis is true, and 9 otherwise.

Angrily, you demand your money back. From a classical point of view, the description above does in fact determine a unique value of x; but what you really want is a value of x that is given *explicitly*.

To take another, perhaps less contrived example, consider the following question. We know that it is possible to raise an irrational number to a rational power, and get a rational result. For example, $\sqrt{2}^2 = 2$. What is less clear is whether or not it is possible to raise an irrational number to an *irrational* power, and get a rational result. The following theorem answers this in the affirmative:

Theorem 11.1. *There are irrational numbers a and b such that a^b is rational.*

Proof. Consider $\sqrt{2}^{\sqrt{2}}$. If this is rational, we are done: we can let $a = b = \sqrt{2}$. Otherwise, it is irrational. Then we have

$$(\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = \sqrt{2}^{\sqrt{2}\cdot\sqrt{2}} = \sqrt{2}^{2} = 2,$$

which is certainly rational. So, in this case, let a be $\sqrt{2}^{\sqrt{2}}$, and let b be $\sqrt{2}$. □

Does this constitute a valid proof? Most mathematicians feel that it does. But again, there is something a little bit unsatisfying here: we have proved the existence of a pair of real numbers with a certain property, without being able to say *which* pair of numbers it is. It is possible to prove the same result, but in such a way that the pair a, b *is* given in the proof: take $a = \sqrt{3}$ and $b = \log_3 4$. Then

$$a^b = \sqrt{3}^{\log_3 4} = 3^{1/2\cdot\log_3 4} = (3^{\log_3 4})^{1/2} = 4^{1/2} = 2,$$

since $3^{\log_3 x} = x$.

Intuitionistic logic is designed to model a kind of reasoning where moves like the one in the first proof are disallowed. Proving the existence of an x satisfying $\varphi(x)$ means that you have to give a specific x, and a proof that it satisfies φ, like in the second proof. Proving that φ or ψ holds requires that you can prove one or the other.

Formally speaking, intuitionistic first-order logic is what you get if you restrict a derivation system for first-order logic in a certain way. Similarly, there are intuitionistic versions of second-order or higher-order logic. From the mathematical point of view, these are just formal deductive systems, but, as already noted, they are intended to model a kind of mathematical reasoning. One can take this to be the kind of reasoning that is justified on a certain philosophical view of mathematics (such as Brouwer's intuitionism); one can take it to be a kind of mathematical reasoning which is more "concrete" and satisfying (along the lines of Bishop's constructivism); and one can argue about whether or not the formal description captures the informal motivation. But whatever philosophical positions we may hold, we can study intuitionistic logic as a formally presented logic; and for whatever reasons, many mathematical logicians find it interesting to do so.

There is an informal constructive interpretation of the intuitionist connectives, usually known as the BHK interpretation (named after Brouwer, Heyting, and Kolmogorov). It runs as follows: a proof of $\varphi \,\&\, \psi$ consists of a proof of φ paired with a proof of ψ; a proof of $\varphi \lor \psi$ consists of either a proof of φ, or a proof of ψ, where we have explicit information as to which is the case;

a proof of $\varphi \supset \psi$ consists of a procedure, which transforms a proof of φ to a proof of ψ; a proof of $\forall x\, \varphi(x)$ consists of a procedure which returns a proof of $\varphi(x)$ for any value of x; and a proof of $\exists x\, \varphi(x)$ consists of a value of x, together with a proof that this value satisfies φ. One can describe the interpretation in computational terms known as the "Curry–Howard isomorphism" or the "formulae-as-types paradigm": think of a formula as specifying a certain kind of data type, and proofs as computational objects of these data types that enable us to see that the corresponding formula is true.

Intuitionistic logic is often thought of as being classical logic "minus" the law of the excluded middle. This following theorem makes this more precise.

Theorem 11.2. *Intuitionistically, the following axiom schemata are equivalent:*

1. $(\sim\varphi \supset \bot) \supset \varphi$.
2. $\varphi \vee \sim\varphi$
3. $\sim\sim\varphi \supset \varphi$

Obtaining instances of one schema from either of the others is a good exercise in intuitionistic logic.

The first deductive systems for intuitionistic propositional logic, put forth as formalizations of Brouwer's intuitionism, are due, independently, to Kolmogorov, Glivenko, and Heyting. The first formalization of intuitionistic first-order logic (and parts of intuitionist mathematics) is due to Heyting. Though a number of classically valid schemata are not intuitionistically valid, many are.

The *double-negation translation* describes an important relationship between classical and intuitionist logic. It is defined inductively follows (think of φ^N as the "intuitionist" translation of the classical formula φ):

$$\begin{aligned}
!A^N &\equiv \sim\sim\varphi \quad \text{for atomic formulae } \varphi\\
(\varphi \,\&\, \psi)^N &\equiv (\varphi^N \,\&\, \psi^N)\\
(\varphi \vee \psi)^N &\equiv \sim\sim(\varphi^N \vee \psi^N)\\
(\varphi \supset \psi)^N &\equiv (\varphi^N \supset \psi^N)\\
(\forall x\, \varphi)^N &\equiv \forall x\, \varphi^N\\
(\exists x\, \varphi)^N &\equiv \sim\sim\exists x\, \varphi^N
\end{aligned}$$

Kolmogorov and Glivenko had versions of this translation for propositional logic; for predicate logic, it is due to Gödel and Gentzen, independently. We have

Theorem 11.3. 1. *$\varphi \equiv \varphi^N$ is provable classically*

2. *If φ is provable classically, then φ^N is provable intuitionistically.*

We can now envision the following dialogue. Classical mathematician: "I've proved φ!" Intuitionist mathematician: "Your proof isn't valid. What you've really proved is φ^N." Classical mathematician: "Fine by me!" As far as the classical mathematician is concerned, the intuitionist is just splitting hairs, since the two are equivalent. But the intuitionist insists there is a difference.

Note that the above translation concerns pure logic only; it does not address the question as to what the appropriate *nonlogical* axioms are for classical and intuitionistic mathematics, or what the relationship is between them. But the following slight extension of the theorem above provides some useful information:

Theorem 11.4. *If Γ proves φ classically, Γ^N proves φ^N intuitionistically.*

In other words, if φ is provable from some hypotheses classically, then φ^N is provable from their double-negation translations.

To show that a sentence or propositional formula is intuitionistically valid, all you have to do is provide a proof. But how can you show that it is not valid? For that purpose, we need a semantics that is sound, and preferably complete. A semantics due to Kripke nicely fits the bill.

We can play the same game we did for classical logic: define the semantics, and prove soundness and completeness. It is worthwhile, however, to note the following distinction. In the case of classical logic, the semantics was the "obvious" one, in a sense implicit in the meaning of the connectives. Though one can provide some intuitive motivation for Kripke semantics, the latter does not offer the same feeling of inevitability. In addition, the notion of a classical structure is a natural mathematical one, so we can either take the notion of a structure to be a tool for studying classical first-order logic, or take classical first-order logic to be a tool for studying mathematical structures. In contrast, Kripke structures can only be viewed as a logical construct; they don't seem to have independent mathematical interest.

A Kripke structure $\mathfrak{M} = \langle W, R, V \rangle$ for a propositional language consists of a set W, partial order R on W with a least element, and an "monotone" assignment of propositional variables to the elements of W. The intuition is that the elements of W represent "worlds," or "states of knowledge"; an element $v \geq u$ represents a "possible future state" of u; and the propositional variables assigned to u are the propositions that are known to be true in state u. The forcing relation $\mathfrak{M}, w \Vdash \varphi$ then extends this relationship to arbitrary formulae in the language; read $\mathfrak{M}, w \Vdash \varphi$ as "φ is true in state w." The relationship is defined inductively, as follows:

1. $\mathfrak{M}, w \Vdash p_i$ iff p_i is one of the propositional variables assigned to w.

2. $\mathfrak{M}, w \nVdash \bot$.

3. $\mathfrak{M}, w \Vdash (\varphi \,\&\, \psi)$ iff $\mathfrak{M}, w \Vdash \varphi$ and $\mathfrak{M}, w \Vdash \psi$.

4. $\mathfrak{M}, w \Vdash (\varphi \vee \psi)$ iff $\mathfrak{M}, w \Vdash \varphi$ or $\mathfrak{M}, w \Vdash \psi$.

5. $\mathfrak{M}, w \Vdash (\varphi \supset \psi)$ iff, whenever $w' \geq w$ and $\mathfrak{M}, w' \Vdash \varphi$, then $\mathfrak{M}, w' \Vdash \psi$.

It is a good exercise to try to show that $\sim(p \,\&\, q) \supset (\sim p \vee \sim q)$ is not intuitionistically valid, by cooking up a Kripke structure that provides a counterexample.

11.6 Modal Logics

Consider the following example of a conditional sentence:

> If Jeremy is alone in that room, then he is drunk and naked and dancing on the chairs.

This is an example of a conditional assertion that may be materially true but nonetheless misleading, since it seems to suggest that there is a stronger link between the antecedent and conclusion other than simply that either the antecedent is false or the consequent true. That is, the wording suggests that the claim is not only true in this particular world (where it may be trivially true, because Jeremy is not alone in the room), but that, moreover, the conclusion *would have* been true *had* the antecedent been true. In other words, one can take the assertion to mean that the claim is true not just in this world, but in any "possible" world; or that it is *necessarily* true, as opposed to just true in this particular world.

Modal logic was designed to make sense of this kind of necessity. One obtains modal propositional logic from ordinary propositional logic by adding a box operator; which is to say, if φ is a formula, so is $\Box\varphi$. Intuitively, $\Box\varphi$ asserts that φ is *necessarily* true, or true in any possible world. $\Diamond\varphi$ is usually taken to be an abbreviation for $\sim\Box\sim\varphi$, and can be read as asserting that φ is *possibly* true. Of course, modality can be added to predicate logic as well.

Kripke structures can be used to provide a semantics for modal logic; in fact, Kripke first designed this semantics with modal logic in mind. Rather than restricting to partial orders, more generally one has a set of "possible worlds," P, and a binary "accessibility" relation $R(x, y)$ between worlds. Intuitively, $R(p, q)$ asserts that the world q is compatible with p; i.e., if we are "in" world p, we have to entertain the possibility that the world could have been like q.

Modal logic is sometimes called an "intensional" logic, as opposed to an "extensional" one. The intended semantics for an extensional logic, like classical logic, will only refer to a single world, the "actual" one; while the semantics for an "intensional" logic relies on a more elaborate ontology. In addition to structureing necessity, one can use modality to structure other linguistic constructions, reinterpreting $\Box$ and $\Diamond$ according to the application. For example:

1. In provability logic, $\Box\varphi$ is read "φ is provable" and $\Diamond\varphi$ is read "φ is consistent."

2. In epistemic logic, one might read $\Box\varphi$ as "I know φ" or "I believe φ."

3. In temporal logic, one can read $\Box\varphi$ as "φ is always true" and $\Diamond\varphi$ as "φ is sometimes true."

One would like to augment logic with rules and axioms dealing with modality. For example, the system **S4** consists of the ordinary axioms and rules of propositional logic, together with the following axioms:

$$\Box(\varphi \supset \psi) \supset (\Box\varphi \supset \Box\psi)$$
$$\Box\varphi \supset \varphi$$
$$\Box\varphi \supset \Box\Box\varphi$$

as well as a rule, "from φ conclude $\Box\varphi$." **S5** adds the following axiom:

$$\Diamond\varphi \supset \Box\Diamond\varphi$$

Variations of these axioms may be suitable for different applications; for example, S5 is usually taken to characterize the notion of logical necessity. And the nice thing is that one can usually find a semantics for which the derivation system is sound and complete by restricting the accessibility relation in the Kripke structures in natural ways. For example, **S4** corresponds to the class of Kripke structures in which the accessibility relation is reflexive and transitive. **S5** corresponds to the class of Kripke structures in which the accessibility relation is *universal*, which is to say that every world is accessible from every other; so $\Box\varphi$ holds if and only if φ holds in every world.

11.7 Other Logics

As you may have gathered by now, it is not hard to design a new logic. You too can create your own a syntax, make up a deductive system, and fashion a semantics to go with it. You might have to be a bit clever if you want the derivation system to be complete for the semantics, and it might take some effort to convince the world at large that your logic is truly interesting. But, in return, you can enjoy hours of good, clean fun, exploring your logic's mathematical and computational properties.

Recent decades have witnessed a veritable explosion of formal logics. Fuzzy logic is designed to model reasoning about vague properties. Probabilistic logic is designed to model reasoning about uncertainty. Default logics and nonmonotonic logics are designed to model defeasible forms of reasoning, which is to say, "reasonable" inferences that can later be overturned in the face of new information. There are epistemic logics, designed to model reasoning about knowledge; causal logics, designed to model reasoning about causal relationships; and even "deontic" logics, which are designed to model reasoning about moral and ethical obligations. Depending on whether the primary motivation for introducing these systems is philosophical, mathematical, or computational, you may find such creatures studies under the rubric of mathematical logic, philosophical logic, artificial intelligence, cognitive science, or elsewhere.

The list goes on and on, and the possibilities seem endless. We may never attain Leibniz' dream of reducing all of human reason to calculation—but that can't stop us from trying.

Part III

Methods

Appendix A

Proofs

A.1 Introduction

Based on your experiences in introductory logic, you might be comfortable with a derivation system—probably a natural deduction or Fitch style derivation system, or perhaps a proof-tree system. You probably remember doing proofs in these systems, either proving a formula or show that a given argument is valid. In order to do this, you applied the rules of the system until you got the desired end result. In reasoning *about* logic, we also prove things, but in most cases we are not using a derivation system. In fact, most of the proofs we consider are done in English (perhaps, with some symbolic language thrown in) rather than entirely in the language of first-order logic. When constructing such proofs, you might at first be at a loss—how do I prove something without a derivation system? How do I start? How do I know if my proof is correct?

Before attempting a proof, it's important to know what a proof is and how to construct one. As implied by the name, a *proof* is meant to show that something is true. You might think of this in terms of a dialogue—someone asks you if something is true, say, if every prime other than two is an odd number. To answer "yes" is not enough; they might want to know *why*. In this case, you'd give them a proof.

In everyday discourse, it might be enough to gesture at an answer, or give an incomplete answer. In logic and mathematics, however, we want rigorous proof—we want to show that something is true beyond *any* doubt. This means that every step in our proof must be justified, and the justification must be cogent (i.e., the assumption you're using is actually assumed in the statement of the theorem you're proving, the definitions you apply must be correctly applied, the justifications appealed to must be correct inferences, etc.).

Usually, we're proving some statement. We call the statements we're proving by various names: propositions, theorems, lemmas, or corollaries. A proposition is a basic proof-worthy statement: important enough to record, but perhaps not particularly deep nor applied often. A theorem is a significant, important proposition. Its proof often is broken into several steps, and sometimes it is named after the person who first proved it (e.g., Cantor's Theorem, the Löwenheim–Skolem theorem) or after the fact it concerns (e.g., the completeness theorem). A lemma is a proposition or theorem that is used in the proof of a more important result. Confusingly, sometimes lemmas are important results in themselves, and also named after the person who introduced them (e.g., Zorn's Lemma). A corollary is a result that easily follows from another one.

A statement to be proved often contains assumptions that clarify which kinds of things we're proving something about. It might begin with "Let φ be a formula of the form $\psi \supset \chi$" or "Suppose $\Gamma \vdash \varphi$" or something of the sort. These are *hypotheses* of the proposition, theorem, or lemma, and you may assume these to be true in your proof. They restrict what we're proving, and also introduce some names for the objects we're talking about. For instance, if your proposition begins with "Let φ be a formula of the form $\psi \supset \chi$," you're proving something about all formulas of a certain sort only (namely, conditionals), and it's understood that $\psi \supset \chi$ is an arbitrary conditional that your proof will talk about.

A.2 Starting a Proof

But where do you even start?

You've been given something to prove, so this should be the last thing that is mentioned in the proof (you can, obviously, *announce* that you're going to prove it at the beginning, but you don't want to use it as an assumption). Write what you are trying to prove at the bottom of a fresh sheet of paper—this way you don't lose sight of your goal.

Next, you may have some assumptions that you are able to use (this will be made clearer when we talk about the *type* of proof you are doing in the next section). Write these at the top of the page and make sure to flag that they are assumptions (i.e., if you are assuming p, write "assume that p," or "suppose that p"). Finally, there might be some definitions in the question that you need to know. You might be told to use a specific definition, or there might be various definitions in the assumptions or conclusion that you are working towards. *Write these down and ensure that you understand what they mean.*

How you set up your proof will also be dependent upon the form of the question. The next section provides details on how to set up your proof based on the type of sentence.

A.3 Using Definitions

We mentioned that you must be familiar with all definitions that may be used in the proof, and that you can properly apply them. This is a really important point, and it is worth looking at in a bit more detail. Definitions are used to abbreviate properties and relations so we can talk about them more succinctly. The introduced abbreviation is called the *definiendum*, and what it abbreviates is the *definiens*. In proofs, we often have to go back to how the definiendum was introduced, because we have to exploit the logical structure of the definiens (the long version of which the defined term is the abbreviation) to get through our proof. By unpacking definitions, you're ensuring that you're getting to the heart of where the logical action is.

We'll start with an example. Suppose you want to prove the following:

Proposition A.1. *For any sets A and B, $A \cup B = B \cup A$.*

In order to even start the proof, we need to know what it means for two sets to be identical; i.e., we need to know what the "$=$" in that equation means for sets. Sets are defined to be identical whenever they have the same elements. So the definition we have to unpack is:

Definition A.2. Sets A and B are *identical*, $A = B$, iff every element of A is an element of B, and vice versa.

This definition uses A and B as placeholders for arbitrary sets. What it defines—the *definiendum*—is the expression "$A = B$" by giving the condition under which $A = B$ is true. This condition—"every element of A is an element of B, and vice versa"—is the *definiens*.[1] The definition specifies that $A = B$ is true if, and only if (we abbreviate this to "iff") the condition holds.

When you apply the definition, you have to match the A and B in the definition to the case you're dealing with. In our case, it means that in order for $A \cup B = B \cup A$ to be true, each $z \in A \cup B$ must also be in $B \cup A$, and vice versa. The expression $A \cup B$ in the proposition plays the role of A in the definition, and $B \cup A$ that of B. Since A and B are used both in the definition and in the statement of the proposition we're proving, but in different uses, you have to be careful to make sure you don't mix up the two. For instance, it would be a mistake to think that you could prove the proposition by showing that every element of A is an element of B, and vice versa—that would show that $A = B$, not that $A \cup B = B \cup A$. (Also, since A and B may be any two sets, you won't get very far, because if nothing is assumed about A and B they may well be different sets.)

[1] In this particular case—and very confusingly!—when $A = B$, the sets A and B are just one and the same set, even though we use different letters for it on the left and the right side. But the ways in which that set is picked out may be different, and that makes the definition non-trivial.

Within the proof we are dealing with set-theoretic notions such as union, and so we must also know the meanings of the symbol $\cup$ in order to understand how the proof should proceed. And sometimes, unpacking the definition gives rise to further definitions to unpack. For instance, $A \cup B$ is defined as $\{z \mid z \in A \text{ or } z \in B\}$. So if you want to prove that $x \in A \cup B$, unpacking the definition of $\cup$ tells you that you have to prove $x \in \{z \mid z \in A \text{ or } z \in B\}$. Now you also have to remember that $x \in \{z \mid \ldots z \ldots\}$ iff $\ldots x \ldots$. So, further unpacking the definition of the $\{z \mid \ldots z \ldots\}$ notation, what you have to show is: $x \in A$ or $x \in B$. So, "every element of $A \cup B$ is also an element of $B \cup A$" really means: "for every x, if $x \in A$ or $x \in B$, then $x \in B$ or $x \in A$." If we fully unpack the definitions in the proposition, we see that what we have to show is this:

Proposition A.3. *For any sets A and B: (a) for every x, if $x \in A$ or $x \in B$, then $x \in B$ or $x \in A$, and (b) for every x, if $x \in B$ or $x \in A$, then $x \in A$ or $x \in B$.*

What's important is that unpacking definitions is a necessary part of constructing a proof. Properly doing it is sometimes difficult: you must be careful to distinguish and match the variables in the definition and the terms in the claim you're proving. In order to be successful, you must know what the question is asking and what all the terms used in the question mean—you will often need to unpack more than one definition. In simple proofs such as the ones below, the solution follows almost immediately from the definitions themselves. Of course, it won't always be this simple.

Suppose you are asked to prove that $A \cap B \neq \emptyset$. Unpack all the definitions occurring here, i.e., restate this in a way that does not mention "$\cap$", "$=$", or "$\emptyset$".

A.4 Inference Patterns

Proofs are composed of individual inferences. When we make an inference, we typically indicate that by using a word like "so," "thus," or "therefore." The inference often relies on one or two facts we already have available in our proof—it may be something we have assumed, or something that we've concluded by an inference already. To be clear, we may label these things, and in the inference we indicate what other statements we're using in the inference. An inference will often also contain an explanation of *why* our new conclusion follows from the things that come before it. There are some common patterns of inference that are used very often in proofs; we'll go through some below. Some patterns of inference, like proofs by induction, are more involved (and will be discussed later).

We've already discussed one pattern of inference: unpacking, or applying, a definition. When we unpack a definition, we just restate something that involves the definiendum by using the definiens. For instance, suppose that we have already established in the course of a proof that $D = E$ (a). Then we may apply the definition of $=$ for sets and infer: "Thus, by definition from (a), every element of D is an element of E and vice versa."

Somewhat confusingly, we often do not write the justification of an inference when we actually make it, but before. Suppose we haven't already proved that $D = E$, but we want to. If $D = E$ is the conclusion we aim for, then we can restate this aim also by applying the definition: to prove $D = E$ we have to prove that every element of D is an element of E and vice versa. So our proof will have the form: (a) prove that every element of D is an element of E; (b) every element of E is an element of D; (c) therefore, from (a) and (b) by definition of $=$, $D = E$. But we would usually not write it this way. Instead we might write something like,

> We want to show $D = E$. By definition of $=$, this amounts to showing that every element of D is an element of E and vice versa.
>
> (a) ... (a proof that every element of D is an element of E) ...
>
> (b) ... (a proof that every element of E is an element of D) ...

Using a Conjunction

Perhaps the simplest inference pattern is that of drawing as conclusion one of the conjuncts of a conjunction. In other words: if we have assumed or already proved that p and q, then we're entitled to infer that p (and also that q). This is such a basic inference that it is often not mentioned. For instance, once we've unpacked the definition of $D = E$ we've established that every element of D is an element of E and vice versa. From this we can conclude that every element of E is an element of D (that's the "vice versa" part).

Proving a Conjunction

Sometimes what you'll be asked to prove will have the form of a conjunction; you will be asked to "prove p and q." In this case, you simply have to do two things: prove p, and then prove q. You could divide your proof into two sections, and for clarity, label them. When you're making your first notes, you might write "(1) Prove p" at the top of the page, and "(2) Prove q" in the middle of the page. (Of course, you might not be explicitly asked to prove a conjunction but find that your proof requires that you prove a conjunction. For instance, if you're asked to prove that $D = E$ you will find that, after unpacking the definition of $=$, you have to prove: every element of D is an element of E *and* every element of E is an element of D).

Proving a Disjunction

When what you are proving takes the form of a disjunction (i.e., it is an statement of the form "p or q"), it is enough to show that one of the disjuncts is true. However, it basically never happens that either disjunct just follows from the assumptions of your theorem. More often, the assumptions of your theorem are themselves disjunctive, or you're showing that all things of a certain kind have one of two properties, but some of the things have the one and others have the other property. This is where proof by cases is useful (see below).

Conditional Proof

Many theorems you will encounter are in conditional form (i.e., show that if p holds, then q is also true). These cases are nice and easy to set up—simply assume the antecedent of the conditional (in this case, p) and prove the conclusion q from it. So if your theorem reads, "If p then q," you start your proof with "assume p" and at the end you should have proved q.

Conditionals may be stated in different ways. So instead of "If p then q," a theorem may state that "p only if q," "q if p," or "q, provided p." These all mean the same and require assuming p and proving q from that assumption. Recall that a biconditional ("p if and only if (iff) q") is really two conditionals put together: if p then q, and if q then p. All you have to do, then, is two instances of conditional proof: one for the first conditional and another one for the second. Sometimes, however, it is possible to prove an "iff" statement by chaining together a bunch of other "iff" statements so that you start with "p" an end with "q"—but in that case you have to make sure that each step really is an "iff."

Universal Claims

Using a universal claim is simple: if something is true for anything, it's true for each particular thing. So if, say, the hypothesis of your proof is $A \subseteq B$, that means (unpacking the definition of $\subseteq$), that, for every $x \in A$, $x \in B$. Thus, if you already know that $z \in A$, you can conclude $z \in B$.

Proving a universal claim may seem a little bit tricky. Usually these statements take the following form: "If x has P, then it has Q" or "All Ps are Qs." Of course, it might not fit this form perfectly, and it takes a bit of practice to figure out what you're asked to prove exactly. But: we often have to prove that all objects with some property have a certain other property.

The way to prove a universal claim is to introduce names or variables, for the things that have the one property and then show that they also have the other property. We might put this by saying that to prove something for *all* Ps you have to prove it for an *arbitrary* P. And the name introduced is a name for an arbitrary P. We typically use single letters as these names for arbitrary things, and the letters usually follow conventions: e.g., we use n for natural numbers, φ for formulae, A for sets, f for functions, etc.

The trick is to maintain generality throughout the proof. You start by assuming that an arbitrary object ("x") has the property P, and show (based only on definitions or what you are allowed to assume) that x has the property Q. Because you have not stipulated what x is specifically, other that it has the property P, then you can assert that all every P has the property Q. In short, x is a stand-in for *all* things with property P.

Proposition A.4. *For all sets A and B, $A \subseteq A \cup B$.*

Proof. Let A and B be arbitrary sets. We want to show that $A \subseteq A \cup B$. By definition of $\subseteq$, this amounts to: for every x, if $x \in A$ then $x \in A \cup B$. So let $x \in A$ be an arbitrary element of A. We have to show that $x \in A \cup B$. Since $x \in A$, $x \in A$ or $x \in B$. Thus, $x \in \{x \mid x \in A \lor x \in B\}$. But that, by definition of $\cup$, means $x \in A \cup B$. □

Proof by Cases

Suppose you have a disjunction as an assumption or as an already established conclusion—you have assumed or proved that p or q is true. You want to prove r. You do this in two steps: first you assume that p is true, and prove r, then you assume that q is true and prove r again. This works because we assume or know that one of the two alternatives holds. The two steps establish that either one is sufficient for the truth of r. (If both are true, we have not one but two reasons for why r is true. It is not necessary to separately prove that r is true assuming both p and q.) To indicate what we're doing, we announce that we "distinguish cases." For instance, suppose we know that $x \in B \cup C$. $B \cup C$ is defined as $\{x \mid x \in B \text{ or } x \in C\}$. In other words, by definition, $x \in B$ or $x \in C$. We would prove that $x \in A$ from this by first assuming that $x \in B$, and proving $x \in A$ from this assumption, and then assume $x \in C$, and again prove $x \in A$ from this. You would write "We distinguish cases" under the assumption, then "Case (1): $x \in B$" underneath, and "Case (2): $x \in C$ halfway down the page. Then you'd proceed to fill in the top half and the bottom half of the page.

Proof by cases is especially useful if what you're proving is itself disjunctive. Here's a simple example:

Proposition A.5. *Suppose $B \subseteq D$ and $C \subseteq E$. Then $B \cup C \subseteq D \cup E$.*

Proof. Assume (a) that $B \subseteq D$ and (b) $C \subseteq E$. By definition, any $x \in B$ is also $\in D$ (c) and any $x \in C$ is also $\in E$ (d). To show that $B \cup C \subseteq D \cup E$, we have to show that if $x \in B \cup C$ then $x \in D \cup E$ (by definition of $\subseteq$). $x \in B \cup C$ iff $x \in B$ or $x \in C$ (by definition of $\cup$). Similarly, $x \in D \cup E$ iff $x \in D$ or $x \in E$. So, we have to show: for any x, if $x \in B$ or $x \in C$, then $x \in D$ or $x \in E$.

> So far we've only unpacked definitions! We've reformulated our proposition without $\subseteq$ and $\cup$ and are left with trying to prove a universal conditional claim. By what we've discussed above, this is done by assuming that x is something about which we assume the "if" part is true, and we'll go on to show that the "then" part is true as well. In other words, we'll assume that $x \in B$ or $x \in C$ and show that $x \in D$ or $x \in E$.[2]

Suppose that $x \in B$ or $x \in C$. We have to show that $x \in D$ or $x \in E$. We distinguish cases.

Case 1: $x \in B$. By (c), $x \in D$. Thus, $x \in D$ or $x \in E$. (Here we've made the inference discussed in the preceding subsection!)

Case 2: $x \in C$. By (d), $x \in E$. Thus, $x \in D$ or $x \in E$. □

Proving an Existence Claim

When asked to prove an existence claim, the question will usually be of the form "prove that there is an x such that $\dots x \dots$", i.e., that some object that has the property described by "$\dots x \dots$". In this case you'll have to identify a suitable object show that is has the required property. This sounds straightforward, but a proof of this kind can be tricky. Typically it involves *constructing* or *defining* an object and proving that the object so defined has the required property. Finding the right object may be hard, proving that it has the required property may be hard, and sometimes it's even tricky to show that you've succeeded in defining an object at all!

Generally, you'd write this out by specifying the object, e.g., "let x be $\dots$" (where $\dots$ specifies which object you have in mind), possibly proving that $\dots$ in fact describes an object that exists, and then go on to show that x has the property Q. Here's a simple example.

Proposition A.6. *Suppose that $x \in B$. Then there is an A such that $A \subseteq B$ and $A \neq \emptyset$.*

Proof. Assume $x \in B$. Let $A = \{x\}$.

> Here we've defined the set A by enumerating its elements. Since we assume that x is an object, and we can always form a set by enumerating its elements, we don't have to show that we've succeeded in defining a set A here. However, we still have to show that A has the properties required by the proposition. The proof isn't complete without that!

Since $x \in A$, $A \neq \emptyset$.

[2] This paragraph just explains what we're doing—it's not part of the proof, and you don't have to go into all this detail when you write down your own proofs.

> This relies on the definition of A as $\{x\}$ and the obvious facts that $x \in \{x\}$ and $x \notin \emptyset$.

Since x is the only element of $\{x\}$, and $x \in B$, every element of A is also an element of B. By definition of $\subseteq$, $A \subseteq B$. □

Using Existence Claims

Suppose you know that some existence claim is true (you've proved it, or it's a hypothesis you can use), say, "for some x, $x \in A$" or "there is an $x \in A$." If you want to use it in your proof, you can just pretend that you have a name for one of the things which your hypothesis says exist. Since A contains at least one thing, there are things to which that name might refer. You might of course not be able to pick one out or describe it further (other than that it is $\in A$). But for the purpose of the proof, you can pretend that you have picked it out and give a name to it. It's important to pick a name that you haven't already used (or that appears in your hypotheses), otherwise things can go wrong. In your proof, you indicate this by going from "for some x, $x \in A$" to "Let $a \in A$." Now you can reason about a, use some other hypotheses, etc., until you come to a conclusion, p. If p no longer mentions a, p is independent of the asusmption that $a \in A$, and you've shown that it follows just from the assumption "for some x, $x \in A$."

Proposition A.7. *If $A \neq \emptyset$, then $A \cup B \neq \emptyset$.*

Proof. Suppose $A \neq \emptyset$. So for some x, $x \in A$.

> Here we first just restated the hypothesis of the proposition. This hypothesis, i.e., $A \neq \emptyset$, hides an existential claim, which you get to only by unpacking a few definitions. The definition of $=$ tells us that $A = \emptyset$ iff every $x \in A$ is also $\in \emptyset$ and every $x \in \emptyset$ is also $\in A$. Negating both sides, we get: $A \neq \emptyset$ iff either some $x \in A$ is $\notin \emptyset$ or some $x \in \emptyset$ is $\notin A$. Since nothing is $\in \emptyset$, the second disjunct can never be true, and "$x \in A$ and $x \notin \emptyset$" reduces to just $x \in A$. So $x \neq \emptyset$ iff for some x, $x \in A$. That's an existence claim. Now we use that existence claim by introducing a name for one of the elements of A:

Let $a \in A$.

> Now we've introduced a name for one of the things $\in A$. We'll continue to argue about a, but we'll be careful to only assume that $a \in A$ and nothing else:

Since $a \in A$, $a \in A \cup B$, by definition of $\cup$. So for some x, $x \in A \cup B$, i.e., $A \cup B \neq \emptyset$.

> In that last step, we went from "$a \in A \cup B$" to "for some x, $x \in A \cup B$." That doesn't mention a anymore, so we know that "for some x, $x \in A \cup B$" follows from "for some x, $x \in A$ alone." But that means that $A \cup B \neq \emptyset$. □

It's maybe good practice to keep bound variables like "x" separate from hypothetical names like a, like we did. In practice, however, we often don't and just use x, like so:

> Suppose $A \neq \emptyset$, i.e., there is an $x \in A$. By definition of $\cup$, $x \in A \cup B$. So $A \cup B \neq \emptyset$.

However, when you do this, you have to be extra careful that you use different x's and y's for different existential claims. For instance, the following is *not* a correct proof of "If $A \neq \emptyset$ and $B \neq \emptyset$ then $A \cap B \neq \emptyset$" (which is not true).

> Suppose $A \neq \emptyset$ and $B \neq \emptyset$. So for some x, $x \in A$ and also for some x, $x \in B$. Since $x \in A$ and $x \in B$, $x \in A \cap B$, by definition of $\cap$. So $A \cap B \neq \emptyset$.

Can you spot where the incorrect step occurs and explain why the result does not hold?

A.5 An Example

Our first example is the following simple fact about unions and intersections of sets. It will illustrate unpacking definitions, proofs of conjunctions, of universal claims, and proof by cases.

Proposition A.8. *For any sets A, B, and C, $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$*

Let's prove it!

Proof. We want to show that for any sets A, B, and C, $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

> First we unpack the definition of "$=$" in the statement of the proposition. Recall that proving sets identical means showing that the sets have the same elements. That is, all elements of $A \cup (B \cap C)$ are also elements of $(A \cup B) \cap (A \cup C)$, and vice versa. The "vice versa" means that also every element of $(A \cup B) \cap (A \cup C)$ must be an element of $A \cup (B \cap C)$. So in unpacking the definition, we see that we have to prove a conjunction. Let's record this:

By definition, $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ iff every element of $A \cup (B \cap C)$ is also an element of $(A \cup B) \cap (A \cup C)$, and every element of $(A \cup B) \cap (A \cup C)$ is an element of $A \cup (B \cap C)$.

Since this is a conjunction, we must prove each conjunct separately. Lets start with the first: let's prove that every element of $A \cup (B \cap C)$ is also an element of $(A \cup B) \cap (A \cup C)$.

This is a universal claim, and so we consider an arbitrary element of $A \cup (B \cap C)$ and show that it must also be an element of $(A \cup B) \cap (A \cup C)$. We'll pick a variable to call this arbitrary element by, say, z. Our proof continues:

First, we prove that every element of $A \cup (B \cap C)$ is also an element of $(A \cup B) \cap (A \cup C)$. Let $z \in A \cup (B \cap C)$. We have to show that $z \in (A \cup B) \cap (A \cup C)$.

Now it is time to unpack the definition of $\cup$ and $\cap$. For instance, the definition of $\cup$ is: $A \cup B = \{z \mid z \in A \text{ or } z \in B\}$. When we apply the definition to "$A \cup (B \cap C)$," the role of the "B" in the definition is now played by "$B \cap C$," so $A \cup (B \cap C) = \{z \mid z \in A \text{ or } z \in B \cap C\}$. So our assumption that $z \in A \cup (B \cap C)$ amounts to: $z \in \{z \mid z \in A \text{ or } z \in B \cap C\}$. And $z \in \{z \mid \ldots z \ldots\}$ iff $\ldots z \ldots$, i.e., in this case, $z \in A$ or $z \in B \cap C$.

By the definition of $\cup$, either $z \in A$ or $z \in B \cap C$.

Since this is a disjunction, it will be useful to apply proof by cases. We take the two cases, and show that in each one, the conclusion we're aiming for (namely, "$z \in (A \cup B) \cap (A \cup C)$") obtains.

Case 1: Suppose that $z \in A$.

There's not much more to work from based on our assumptions. So let's look at what we have to work with in the conclusion. We want to show that $z \in (A \cup B) \cap (A \cup C)$. Based on the definition of $\cap$, if we want to show that $z \in (A \cup B) \cap (A \cup C)$, we have to show that it's in both $(A \cup B)$ and $(A \cup C)$. But $z \in A \cup B$ iff $z \in A$ or $z \in B$, and we already have (as the assumption of case 1) that $z \in A$. By the same reasoning—switching C for B—$z \in A \cup C$. This argument went in the reverse direction, so let's record our reasoning in the direction needed in our proof.

Since $z \in A$, $z \in A$ or $z \in B$, and hence, by definition of $\cup$, $z \in A \cup B$. Similarly, $z \in A \cup C$. But this means that $z \in (A \cup B) \cap (A \cup C)$, by definition of $\cap$.

This completes the first case of the proof by cases. Now we want to derive the conclusion in the second case, where $z \in B \cap C$.

Case 2: Suppose that $z \in B \cap C$.

Again, we are working with the intersection of two sets. Let's apply the definition of $\cap$:

Since $z \in B \cap C$, z must be an element of both B and C, by definition of $\cap$.

> It's time to look at our conclusion again. We have to show that z is in both $(A \cup B)$ and $(A \cup C)$. And again, the solution is immediate.

Since $z \in B$, $z \in (A \cup B)$. Since $z \in C$, also $z \in (A \cup C)$. So, $z \in (A \cup B) \cap (A \cup C)$.

> Here we applied the definitions of $\cup$ and $\cap$ again, but since we've already recalled those definitions, and already showed that if z is in one of two sets it is in their union, we don't have to be as explicit in what we've done.
>
> We've completed the second case of the proof by cases, so now we can assert our first conclusion.

So, if $z \in A \cup (B \cap C)$ then $z \in (A \cup B) \cap (A \cup C)$.

> Now we just want to show the other direction, that every element of $(A \cup B) \cap (A \cup C)$ is an element of $A \cup (B \cap C)$. As before, we prove this universal claim by assuming we have an arbitrary element of the first set and show it must be in the second set. Let's state what we're about to do.

Now, assume that $z \in (A \cup B) \cap (A \cup C)$. We want to show that $z \in A \cup (B \cap C)$.

> We are now working from the hypothesis that $z \in (A \cup B) \cap (A \cup C)$. It hopefully isn't too confusing that we're using the same z here as in the first part of the proof. When we finished that part, all the assumptions we've made there are no longer in effect, so now we can make new assumptions about what z is. If that is confusing to you, just replace z with a different variable in what follows.
>
> We know that z is in both $A \cup B$ and $A \cup C$, by definition of $\cap$. And by the definition of $\cup$, we can further unpack this to: either $z \in A$ or $z \in B$, and also either $z \in A$ or $z \in C$. This looks like a proof by cases again—except the "and" makes it confusing. You might think that this amounts to there being three possibilities: z is either in A, B or C. But that would be a mistake. We have to be careful, so let's consider each disjunction in turn.

By definition of $\cap$, $z \in A \cup B$ and $z \in A \cup C$. By definition of $\cup$, $z \in A$ or $z \in B$. We distinguish cases.

> Since we're focusing on the first disjunction, we haven't gotten our second disjunction (from unpacking $A \cup C$) yet. In fact, we don't need it yet. The first case is $z \in A$, and an element of a set is also an element of the union of that set with any other. So case 1 is easy:

Case 1: Suppose that $z \in A$. It follows that $z \in A \cup (B \cap C)$.

> Now for the second case, $z \in B$. Here we'll unpack the second $\cup$ and do another proof-by-cases:

Case 2: Suppose that $z \in B$. Since $z \in A \cup C$, either $z \in A$ or $z \in C$. We distinguish cases further:

Case 2a: $z \in A$. Then, again, $z \in A \cup (B \cap C)$.

> Ok, this was a bit weird. We didn't actually need the assumption that $z \in B$ for this case, but that's ok.

Case 2b: $z \in C$. Then $z \in B$ and $z \in C$, so $z \in B \cap C$, and consequently, $z \in A \cup (B \cap C)$.

> This concludes both proofs-by-cases and so we're done with the second half.

So, if $z \in (A \cup B) \cap (A \cup C)$ then $z \in A \cup (B \cap C)$. □

A.6 Another Example

Proposition A.9. *If $A \subseteq C$, then $A \cup (C \setminus A) = C$.*

Proof. Suppose that $A \subseteq C$. We want to show that $A \cup (C \setminus A) = C$.

> We begin by observing that this is a conditional statement. It is tacitly universally quantified: the proposition holds for all sets A and C. So A and C are variables for arbitrary sets. To prove such a statement, we assume the antecedent and prove the consequent.
>
> We continue by using the assumption that $A \subseteq C$. Let's unpack the definition of $\subseteq$: the assumption means that all elements of A are also elements of C. Let's write this down—it's an important fact that we'll use throughout the proof.

By the definition of $\subseteq$, since $A \subseteq C$, for all z, if $z \in A$, then $z \in C$.

> We've unpacked all the definitions that are given to us in the assumption. Now we can move onto the conclusion. We want to show that $A \cup (C \setminus A) = C$, and so we set up a proof similarly to the last example: we show that every element of $A \cup (C \setminus A)$ is also an element of C and, conversely, every element of C is an element of $A \cup (C \setminus A)$. We can shorten this to: $A \cup (C \setminus A) \subseteq C$ and $C \subseteq A \cup (C \setminus A)$. (Here we're doing the opposite of unpacking a definition, but it makes the proof a bit easier to read.) Since this is a conjunction, we have to prove both parts. To show the first part, i.e., that every element of $A \cup (C \setminus A)$ is also an element of C, we

> assume that $z \in A \cup (C \setminus A)$ for an arbitrary z and show that $z \in C$. By the definition of $\cup$, we can conclude that $z \in A$ or $z \in C \setminus A$ from $z \in A \cup (C \setminus A)$. You should now be getting the hang of this.

$A \cup (C \setminus A) = C$ iff $A \cup (C \setminus A) \subseteq C$ and $C \subseteq (A \cup (C \setminus A)$. First we prove that $A \cup (C \setminus A) \subseteq C$. Let $z \in A \cup (C \setminus A)$. So, either $z \in A$ or $z \in (C \setminus A)$.

> We've arrived at a disjunction, and from it we want to prove that $z \in C$. We do this using proof by cases.

Case 1: $z \in A$. Since for all z, if $z \in A$, $z \in C$, we have that $z \in C$.

> Here we've used the fact recorded earlier which followed from the hypothesis of the proposition that $A \subseteq C$. The first case is complete, and we turn to the second case, $z \in (C \setminus A)$. Recall that $C \setminus A$ denotes the *difference* of the two sets, i.e., the set of all elements of C which are not elements of A. But any element of C not in A is in particular an element of C.

Case 2: $z \in (C \setminus A)$. This means that $z \in C$ and $z \notin A$. So, in particular, $z \in C$.

> Great, we've proved the first direction. Now for the second direction. Here we prove that $C \subseteq A \cup (C \setminus A)$. So we assume that $z \in C$ and prove that $z \in A \cup (C \setminus A)$.

Now let $z \in C$. We want to show that $z \in A$ or $z \in C \setminus A$.

> Since all elements of A are also elements of C, and $C \setminus A$ is the set of all things that are elements of C but not A, it follows that z is either in A or in $C \setminus A$. This may be a bit unclear if you don't already know why the result is true. It would be better to prove it step-by-step. It will help to use a simple fact which we can state without proof: $z \in A$ or $z \notin A$. This is called the "principle of excluded middle:" for any statement p, either p is true or its negation is true. (Here, p is the statement that $z \in A$.) Since this is a disjunction, we can again use proof-by-cases.

Either $z \in A$ or $z \notin A$. In the former case, $z \in A \cup (C \setminus A)$. In the latter case, $z \in C$ and $z \notin A$, so $z \in C \setminus A$. But then $z \in A \cup (C \setminus A)$.

> Our proof is complete: we have shown that $A \cup (C \setminus A) = C$. □

A.7 Proof by Contradiction

In the first instance, proof by contradiction is an inference pattern that is used to prove negative claims. Suppose you want to show that some claim p is *false*, i.e., you want to show $\sim p$. The most promising strategy is to (a) suppose that p is true, and (b) show that this assumption leads to something you know to be false. "Something known to be false" may be a result that conflicts with—contradicts—p itself, or some other hypothesis of the overall claim you are considering. For instance, a proof of "if q then $\sim p$" involves assuming that q is true and proving $\sim p$ from it. If you prove $\sim p$ by contradiction, that means assuming p in addition to q. If you can prove $\sim q$ from p, you have shown that the assumption p leads to something that contradicts your other assumption q, since q and $\sim q$ cannot both be true. Of course, you have to use other inference patterns in your proof of the contradiction, as well as unpacking definitions. Let's consider an example.

Proposition A.10. *If $A \subseteq B$ and $B = \emptyset$, then A has no elements.*

Proof. Suppose $A \subseteq B$ and $B = \emptyset$. We want to show that A has no elements.

> Since this is a conditional claim, we assume the antecedent and want to prove the consequent. The consequent is: A has no elements. We can make that a bit more explicit: it's not the case that there is an $x \in A$.

A has no elements iff it's not the case that there is an x such that $x \in A$.

> So we've determined that what we want to prove is really a negative claim $\sim p$, namely: it's not the case that there is an $x \in A$. To use proof by contradiction, we have to assume the corresponding positive claim p, i.e., there is an $x \in A$, and prove a contradiction from it. We indicate that we're doing a proof by contradiction by writing "by way of contradiction, assume" or even just "suppose not," and then state the assumption p.

Suppose not: there is an $x \in A$.

> This is now the new assumption we'll use to obtain a contradiction. We have two more assumptions: that $A \subseteq B$ and that $B = \emptyset$. The first gives us that $x \in B$:

Since $A \subseteq B$, $x \in B$.

> But since $B = \emptyset$, every element of B (e.g., x) must also be an element of $\emptyset$.

Since $B = \emptyset$, $x \in \emptyset$. This is a contradiction, since by definition $\emptyset$ has no elements.

> This already completes the proof: we've arrived at what we need (a contradiction) from the assumptions we've set up, and this means that the assumptions can't all be true. Since the first two assumptions ($A \subseteq B$ and $B = \emptyset$) are not contested, it must be the last assumption introduced (there is an $x \in A$) that must be false. But if we want to be thorough, we can spell this out.

Thus, our assumption that there is an $x \in A$ must be false, hence, A has no elements by proof by contradiction. □

Every positive claim is trivially equivalent to a negative claim: p iff $\sim\sim p$. So proofs by contradiction can also be used to establish positive claims "indirectly," as follows: To prove p, read it as the negative claim $\sim\sim p$. If we can prove a contradiction from $\sim p$, we've established $\sim\sim p$ by proof by contradiction, and hence p.

In the last example, we aimed to prove a negative claim, namely that A has no elements, and so the assumption we made for the purpose of proof by contradiction (i.e., that there is an $x \in A$) was a positive claim. It gave us something to work with, namely the hypothetical $x \in A$ about which we continued to reason until we got to $x \in \emptyset$.

When proving a positive claim indirectly, the assumption you'd make for the purpose of proof by contradiction would be negative. But very often you can easily reformulate a positive claim as a negative claim, and a negative claim as a positive claim. Our previous proof would have been essentially the same had we proved "$A = \emptyset$" instead of the negative consequent "A has no elements." (By definition of $=$, "$A = \emptyset$" is a general claim, since it unpacks to "every element of A is an element of $\emptyset$ and vice versa".) But it is easily seen to be equivalent to the negative claim "not: there is an $x \in A$."

So it is sometimes easier to work with $\sim p$ as an assumption than it is to prove p directly. Even when a direct proof is just as simple or even simpler (as in the next examples), some people prefer to proceed indirectly. If the double negation confuses you, think of a proof by contradiction of some claim as a proof of a contradiction from the *opposite* claim. So, a proof by contradiction of $\sim p$ is a proof of a contradiction from the assumption p; and proof by contradiction of p is a proof of a contradiction from $\sim p$.

Proposition A.11. *$A \subseteq A \cup B$.*

Proof. We want to show that $A \subseteq A \cup B$.

> On the face of it, this is a positive claim: every $x \in A$ is also in $A \cup B$. The negation of that is: some $x \in A$ is $\notin A \cup B$. So we can prove the claim indirectly by assuming this negated claim, and showing that it leads to a contradiction.

Suppose not, i.e., $A \nsubseteq A \cup B$.

We have a definition of $A \subseteq A \cup B$: every $x \in A$ is also $\in A \cup B$. To understand what $A \nsubseteq A \cup B$ means, we have to use some elementary logical manipulation on the unpacked definition: it's false that every $x \in A$ is also $\in A \cup B$ iff there is *some* $x \in A$ that is $\notin C$. (This is a place where you want to be very careful: many students' attempted proofs by contradiction fail because they analyze the negation of a claim like "all As are Bs" incorrectly.) In other words, $A \nsubseteq A \cup B$ iff there is an x such that $x \in A$ and $x \notin A \cup B$. From then on, it's easy.

So, there is an $x \in A$ such that $x \notin A \cup B$. By definition of $\cup$, $x \in A \cup B$ iff $x \in A$ or $x \in B$. Since $x \in A$, we have $x \in A \cup B$. This contradicts the assumption that $x \notin A \cup B$. □

Prove *indirectly* that $A \cap B \subseteq A$.

Proposition A.12. *If $A \subseteq B$ and $B \subseteq C$ then $A \subseteq C$.*

Proof. Suppose $A \subseteq B$ and $B \subseteq C$. We want to show $A \subseteq C$.

Let's proceed indirectly: we assume the negation of what we want to etablish.

Suppose not, i.e., $A \nsubseteq C$.

As before, we reason that $A \nsubseteq C$ iff not every $x \in A$ is also $\in C$, i.e., some $x \in A$ is $\notin C$. Don't worry, with practice you won't have to think hard anymore to unpack negations like this.

In other words, there is an x such that $x \in A$ and $x \notin C$.

Now we can use this to get to our contradiction. Of course, we'll have to use the other two assumptions to do it.

Since $A \subseteq B$, $x \in B$. Since $B \subseteq C$, $x \in C$. But this contradicts $x \notin C$. □

Proposition A.13. *If $A \cup B = A \cap B$ then $A = B$.*

Proof. Suppose $A \cup B = A \cap B$. We want to show that $A = B$.

The beginning is now routine:

Assume, by way of contradiction, that $A \neq B$.

Our assumption for the proof by contradiction is that $A \neq B$. Since $A = B$ iff $A \subseteq B$ an $B \subseteq A$, we get that $A \neq B$ iff $A \nsubseteq B$ *or* $B \nsubseteq A$. (Note how important it is to be careful when manipulating negations!) To prove a contradiction from this disjunction, we use a proof by cases and show that in each case, a contradiction follows.

$A \neq B$ iff $A \nsubseteq B$ or $B \nsubseteq A$. We distinguish cases.

> In the first case, we assume $A \nsubseteq B$, i.e., for some x, $x \in A$ but $\notin B$. $A \cap B$ is defined as those elements that A and B have in common, so if something isn't in one of them, it's not in the intersection. $A \cup B$ is A together with B, so anything in either is also in the union. This tells us that $x \in A \cup B$ but $x \notin A \cap B$, and hence that $A \cap B \neq A \cup B$.

Case 1: $A \nsubseteq B$. Then for some x, $x \in A$ but $x \notin B$. Since $x \notin B$, then $x \notin A \cap B$. Since $x \in A$, $x \in A \cup B$. So, $A \cap B \neq A \cup B$, contradicting the assumption that $A \cap B = A \cup B$.

Case 2: $B \nsubseteq A$. Then for some y, $y \in B$ but $y \notin A$. As before, we have $y \in A \cup B$ but $y \notin A \cap B$, and so $A \cap B \neq A \cup B$, again contradicting $A \cap B = A \cup B$. □

A.8 Reading Proofs

Proofs you find in textbooks and articles very seldom give all the details we have so far included in our examples. Authors often do not draw attention to when they distinguish cases, when they give an indirect proof, or don't mention that they use a definition. So when you read a proof in a textbook, you will often have to fill in those details for yourself in order to understand the proof. Doing this is also good practice to get the hang of the various moves you have to make in a proof. Let's look at an example.

Proposition A.14 (Absorption). *For all sets A, B,*

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

Proof. If $z \in A \cap (A \cup B)$, then $z \in A$, so $A \cap (A \cup B) \subseteq A$. Now suppose $z \in A$. Then also $z \in A \cup B$, and therefore also $z \in A \cap (A \cup B)$. □

The preceding proof of the absorption law is very condensed. There is no mention of any definitions used, no "we have to prove that" before we prove it, etc. Let's unpack it. The proposition proved is a general claim about any sets A and B, and when the proof mentions A or B, these are variables for arbitrary sets. The general claims the proof establishes is what's required to prove identity of sets, i.e., that every element of the left side of the identity is an element of the right and vice versa.

> "If $z \in A \cap (A \cup B)$, then $z \in A$, so $A \cap (A \cup B) \subseteq A$."

This is the first half of the proof of the identity: it establishes that if an arbitrary z is an element of the left side, it is also an element of the right, i.e., $A \cap (A \cup B) \subseteq A$. Assume that $z \in A \cap (A \cup B)$. Since z is an element of the intersection of two sets iff it is an element of both sets, we can conclude that $z \in A$ and also $z \in A \cup B$. In particular, $z \in A$, which is what we wanted to show. Since that's all that has to be done for the first half, we know that the rest of the proof must be a proof of the second half, i.e., a proof that $A \subseteq A \cap (A \cup B)$.

> "Now suppose $z \in A$. Then also $z \in A \cup B$, and therefore also $z \in A \cap (A \cup B)$."

We start by assuming that $z \in A$, since we are showing that, for any z, if $z \in A$ then $z \in A \cap (A \cup B)$. To show that $z \in A \cap (A \cup B)$, we have to show (by definition of "$\cap$") that (i) $z \in A$ and also (ii) $z \in A \cup B$. Here (i) is just our assumption, so there is nothing further to prove, and that's why the proof does not mention it again. For (ii), recall that z is an element of a union of sets iff it is an element of at least one of those sets. Since $z \in A$, and $A \cup B$ is the union of A and B, this is the case here. So $z \in A \cup B$. We've shown both (i) $z \in A$ and (ii) $z \in A \cup B$, hence, by definition of "$\cap$," $z \in A \cap (A \cup B)$. The proof doesn't mention those definitions; it's assumed the reader has already internalized them. If you haven't, you'll have to go back and remind yourself what they are. Then you'll also have to recognize why it follows from $z \in A$ that $z \in A \cup B$, and from $z \in A$ and $z \in A \cup B$ that $z \in A \cap (A \cup B)$.

Here's another version of the proof above, with everything made explicit:

Proof. [By definition of $=$ for sets, $A \cap (A \cup B) = A$ we have to show (a) $A \cap (A \cup B) \subseteq A$ and (b) $A \cap (A \cup B) \subseteq A$. (a): By definition of $\subseteq$, we have to show that if $z \in A \cap (A \cup B)$, then $z \in A$.] If $z \in A \cap (A \cup B)$, then $z \in A$ [since by definition of $\cap$, $z \in A \cap (A \cup B)$ iff $z \in A$ and $z \in A \cup B$], so $A \cap (A \cup B) \subseteq A$. [(b): By definition of $\subseteq$, we have to show that if $z \in A$, then $z \in A \cap (A \cup B)$.] Now suppose [(1)] $z \in A$. Then also [(2)] $z \in A \cup B$ [since by (1) $z \in A$ or $z \in B$, which by definition of $\cup$ means $z \in A \cup B$], and therefore also $z \in A \cap (A \cup B)$ [since the definition of $\cap$ requires that $z \in A$, i.e., (1), and $z \in A \cup B$), i.e., (2)]. □

Expand the following proof of $A \cup (A \cap B) = A$, where you mention all the inference patterns used, why each step follows from assumptions or claims established before it, and where we have to appeal to which definitions.

Proof. If $z \in A \cup (A \cap B)$ then $z \in A$ or $z \in A \cap B$. If $z \in A \cap B$, $z \in A$. Any $z \in A$ is also $\in A \cup (A \cap B)$. □

A.9 I Can't Do It!

We all get to a point where we feel like giving up. But you *can* do it. Your instructor and teaching assistant, as well as your fellow students, can help.

Ask them for help! Here are a few tips to help you avoid a crisis, and what to do if you feel like giving up.

To make sure you can solve problems successfully, do the following:

1. *Start as far in advance as possible.* We get busy throughout the semester and many of us struggle with procrastination, one of the best things you can do is to start your homework assignments early. That way, if you're stuck, you have time to look for a solution (that isn't crying).

2. *Talk to your classmates*. You are not alone. Others in the class may also struggle—but they may struggle with different things. Talking it out with your peers can give you a different perspective on the problem that might lead to a breakthrough. Of course, don't just copy their solution: ask them for a hint, or explain where you get stuck and ask them for the next step. And when you do get it, reciprocate. Helping someone else along, and explaining things will help you understand better, too.

3. *Ask for help.* You have many resources available to you—your instructor and teaching assistant are there for you and *want* you to succeed. They should be able to help you work out a problem and identify where in the process you're struggling.

4. *Take a break.* If you're stuck, it *might* be because you've been staring at the problem for too long. Take a short break, have a cup of tea, or work on a different problem for a while, then return to the problem with a fresh mind. Sleep on it.

Notice how these strategies require that you've started to work on the proof well in advance? If you've started the proof at 2am the day before it's due, these might not be so helpful.

This might sound like doom and gloom, but solving a proof is a challenge that pays off in the end. Some people do this as a career—so there must be something to enjoy about it. Like basically everything, solving problems and doing proofs is something that requires practice. You might see classmates who find this easy: they've probably just had lots of practice already. Try not to give in too easily.

If you do run out of time (or patience) on a particular problem: that's ok. It doesn't mean you're stupid or that you will never get it. Find out (from your instructor or another student) how it is done, and identify where you went wrong or got stuck, so you can avoid doing that the next time you encounter a similar issue. Then try to do it without looking at the solution. And next time, start (and ask for help) earlier.

A.10 Other Resources

There are many books on how to do proofs in mathematics which may be useful. Check out *How to Read and do Proofs: An Introduction to Mathematical Thought Processes* (Solow, 2013) and *How to Prove It: A Structured Approach* (Velleman, 2019) in particular. The *Book of Proof* (Hammack, 2013) and *Mathematical Reasoning* (Sandstrum, 2019) are books on proof that are freely available online. Philosophers might find *More Precisely: The Math you need to do Philosophy* (Steinhart, 2018) to be a good primer on mathematical reasoning.

There are also various shorter guides to proofs available on the internet; e.g., “Introduction to Mathematical Arguments” (Hutchings, 2003) and “How to write proofs” (Cheng, 2004).

Motivational Videos

Feel like you have no motivation to do your homework? Feeling down? These videos might help!

- `https://www.youtube.com/watch?v=ZXsQAXx_ao0`
- `https://www.youtube.com/watch?v=BQ4yd2W50No`
- `https://www.youtube.com/watch?v=StTqXEQ2l-Y`

Problems

Appendix B

Induction

B.1 Introduction

Induction is an important proof technique which is used, in different forms, in almost all areas of logic, theoretical computer science, and mathematics. It is needed to prove many of the results in logic.

Induction is often contrasted with deduction, and characterized as the inference from the particular to the general. For instance, if we observe many green emeralds, and nothing that we would call an emerald that's not green, we might conclude that all emeralds are green. This is an inductive inference, in that it proceeds from many particular cases (this emerald is green, that emerald is green, etc.) to a general claim (all emeralds are green). *Mathematical* induction is also an inference that concludes a general claim, but it is of a very different kind than this "simple induction."

Very roughly, an inductive proof in mathematics concludes that all mathematical objects of a certain sort have a certain property. In the simplest case, the mathematical objects an inductive proof is concerned with are natural numbers. In that case an inductive proof is used to establish that all natural numbers have some property, and it does this by showing that

1. 0 has the property, and

2. whenever a number k has the property, so does $k + 1$.

Induction on natural numbers can then also often be used to prove general claims about mathematical objects that can be assigned numbers. For instance, finite sets each have a finite number n of elements, and if we can use induction to show that every number n has the property "all finite sets of size n are ..." then we will have shown something about all finite sets.

Induction can also be generalized to mathematical objects that are *inductively defined*. For instance, expressions of a formal language such as those of first-order logic are defined inductively. *Structural induction* is a way to prove results about all such expressions. Structural induction, in particular, is very useful—and widely used—in logic.

B.2 Induction on $\mathbb{N}$

In its simplest form, induction is a technique used to prove results for all natural numbers. It uses the fact that by starting from 0 and repeatedly adding 1 we eventually reach every natural number. So to prove that something is true for every number, we can (1) establish that it is true for 0 and (2) show that whenever it is true for a number n, it is also true for the next number $n+1$. If we abbreviate "number n has property P" by $P(n)$ (and "number k has property P" by $P(k)$, etc.), then a proof by induction that $P(n)$ for all $n \in \mathbb{N}$ consists of:

1. a proof of $P(0)$, and
2. a proof that, for any k, if $P(k)$ then $P(k+1)$.

To make this crystal clear, suppose we have both (1) and (2). Then (1) tells us that $P(0)$ is true. If we also have (2), we know in particular that if $P(0)$ then $P(0+1)$, i.e., $P(1)$. This follows from the general statement "for any k, if $P(k)$ then $P(k+1)$" by putting 0 for k. So by modus ponens, we have that $P(1)$. From (2) again, now taking 1 for n, we have: if $P(1)$ then $P(2)$. Since we've just established $P(1)$, by modus ponens, we have $P(2)$. And so on. For any number n, after doing this n times, we eventually arrive at $P(n)$. So (1) and (2) together establish $P(n)$ for any $n \in \mathbb{N}$.

Let's look at an example. Suppose we want to find out how many different sums we can throw with n dice. Although it might seem silly, let's start with 0 dice. If you have no dice there's only one possible sum you can "throw": no dots at all, which sums to 0. So the number of different possible throws is 1. If you have only one die, i.e., $n = 1$, there are six possible values, 1 through 6. With two dice, we can throw any sum from 2 through 12, that's 11 possibilities. With three dice, we can throw any number from 3 to 18, i.e., 16 different possibilities. 1, 6, 11, 16: looks like a pattern: maybe the answer is $5n+1$? Of course, $5n+1$ is the maximum possible, because there are only $5n+1$ numbers between n, the lowest value you can throw with n dice (all 1's) and $6n$, the highest you can throw (all 6's).

Theorem B.1. *With n dice one can throw all $5n+1$ possible values between n and $6n$.*

Proof. Let $P(n)$ be the claim: "It is possible to throw any number between n and $6n$ using n dice." To use induction, we prove:

1. The *induction basis* $P(1)$, i.e., with just one die, you can throw any number between 1 and 6.

2. The *induction step*, for all k, if $P(k)$ then $P(k+1)$.

(1) Is proved by inspecting a 6-sided die. It has all 6 sides, and every number between 1 and 6 shows up one on of the sides. So it is possible to throw any number between 1 and 6 using a single die.

To prove (2), we assume the antecedent of the conditional, i.e., $P(k)$. This assumption is called the *inductive hypothesis*. We use it to prove $P(k+1)$. The hard part is to find a way of thinking about the possible values of a throw of $k+1$ dice in terms of the possible values of throws of k dice plus of throws of the extra $k+1$-st die—this is what we have to do, though, if we want to use the inductive hypothesis.

The inductive hypothesis says we can get any number between k and $6k$ using k dice. If we throw a 1 with our $(k+1)$-st die, this adds 1 to the total. So we can throw any value between $k+1$ and $6k+1$ by throwing k dice and then rolling a 1 with the $(k+1)$-st die. What's left? The values $6k+2$ through $6k+6$. We can get these by rolling k 6s and then a number between 2 and 6 with our $(k+1)$-st die. Together, this means that with $k+1$ dice we can throw any of the numbers between $k+1$ and $6(k+1)$, i.e., we've proved $P(k+1)$ using the assumption $P(k)$, the inductive hypothesis. $\square$

Very often we use induction when we want to prove something about a series of objects (numbers, sets, etc.) that is itself defined "inductively," i.e., by defining the $(n+1)$-st object in terms of the n-th. For instance, we can define the sum s_n of the natural numbers up to n by

$$\begin{aligned} s_0 &= 0 \\ s_{n+1} &= s_n + (n+1) \end{aligned}$$

This definition gives:

$$\begin{aligned} s_0 &= 0, & & \\ s_1 &= s_0 + 1 & &= 1, \\ s_2 &= s_1 + 2 & &= 1+2 = 3 \\ s_3 &= s_2 + 3 & &= 1+2+3 = 6, \text{ etc.} \end{aligned}$$

Now we can prove, by induction, that $s_n = n(n+1)/2$.

Proposition B.2. $s_n = n(n+1)/2$.

Proof. We have to prove (1) that $s_0 = 0 \cdot (0+1)/2$ and (2) if $s_k = k(k+1)/2$ then $s_{k+1} = (k+1)(k+2)/2$. (1) is obvious. To prove (2), we assume the inductive hypothesis: $s_k = k(k+1)/2$. Using it, we have to show that $s_{k+1} = (k+1)(k+2)/2$.

What is s_{k+1}? By the definition, $s_{k+1} = s_k + (k+1)$. By inductive hypothesis, $s_k = k(k+1)/2$. We can substitute this into the previous equation, and then just need a bit of arithmetic of fractions:

$$\begin{aligned}
s_{k+1} &= \frac{k(k+1)}{2} + (k+1) = \\
&= \frac{k(k+1)}{2} + \frac{2(k+1)}{2} = \\
&= \frac{k(k+1) + 2(k+1)}{2} = \\
&= \frac{(k+2)(k+1)}{2}.
\end{aligned}$$ □

The important lesson here is that if you're proving something about some inductively defined sequence a_n, induction is the obvious way to go. And even if it isn't (as in the case of the possibilities of dice throws), you can use induction if you can somehow relate the case for $k+1$ to the case for k.

B.3 Strong Induction

In the principle of induction discussed above, we prove $P(0)$ and also if $P(k)$, then $P(k+1)$. In the second part, we assume that $P(k)$ is true and use this assumption to prove $P(k+1)$. Equivalently, of course, we could assume $P(k-1)$ and use it to prove $P(k)$—the important part is that we be able to carry out the inference from any number to its successor; that we can prove the claim in question for any number under the assumption it holds for its predecessor.

There is a variant of the principle of induction in which we don't just assume that the claim holds for the predecessor $k-1$ of k, but for all numbers smaller than k, and use this assumption to establish the claim for k. This also gives us the claim $P(n)$ for all $n \in \mathbb{N}$. For once we have established $P(0)$, we have thereby established that P holds for all numbers less than 1. And if we know that if $P(l)$ for all $l < k$, then $P(k)$, we know this in particular for $k = 1$. So we can conclude $P(1)$. With this we have proved $P(0)$ and $P(1)$, i.e., $P(l)$ for all $l < 2$, and since we have also the conditional, if $P(l)$ for all $l < 2$, then $P(2)$, we can conclude $P(2)$, and so on.

In fact, if we can establish the general conditional "for all k, if $P(l)$ for all $l < k$, then $P(k)$," we do not have to establish $P(0)$ anymore, since it follows from it. For remember that a general claim like "for all $l < k$, $P(l)$" is true if there are no $l < k$. This is a case of vacuous quantification: "all As are Bs" is true if there are no As, $\forall x\,(\varphi(x) \supset \psi(x))$ is true if no x satisfies $\varphi(x)$. In this case, the formalized version would be "$\forall l\,(l < k \supset P(l))$"—and that is true if there are no $l < k$. And if $k = 0$ that's exactly the case: no $l < 0$, hence "for all $l < 0$, $P(0)$" is true, whatever P is. A proof of "if $P(l)$ for all $l < k$, then $P(k)$" thus automatically establishes $P(0)$.

This variant is useful if establishing the claim for k can't be made to just rely on the claim for $k-1$ but may require the assumption that it is true for one or more $l < k$.

B.4 Inductive Definitions

In logic we very often define kinds of objects *inductively*, i.e., by specifying rules for what counts as an object of the kind to be defined which explain how to get new objects of that kind from old objects of that kind. For instance, we often define special kinds of sequences of symbols, such as the terms and formulae of a language, by induction. For a simple example, consider strings of consisting of letters a, b, c, d, the symbol $\circ$, and brackets [and], such as "[[c $\circ$ d][", "[a[]$\circ$]", "a" or "[[a $\circ$ b] $\circ$ d]". You probably feel that there's something "wrong" with the first two strings: the brackets don't "balance" at all in the first, and you might feel that the "$\circ$" should "connect" expressions that themselves make sense. The third and fourth string look better: for every "[" there's a closing "]" (if there are any at all), and for any $\circ$ we can find "nice" expressions on either side, surrounded by a pair of parentheses.

We would like to precisely specify what counts as a "nice term." First of all, every letter by itself is nice. Anything that's not just a letter by itself should be of the form "$[t \circ s]$" where s and t are themselves nice. Conversely, if t and s are nice, then we can form a new nice term by putting a $\circ$ between them and surround them by a pair of brackets. We might use these operations to *define* the set of nice terms. This is an *inductive definition*.

Definition B.3 (Nice terms). The set of *nice terms* is inductively defined as follows:

1. Any letter a, b, c, d is a nice term.
2. If s_1 and s_2 are nice terms, then so is $[s_1 \circ s_2]$.
3. Nothing else is a nice term.

This definition tells us that something counts as a nice term iff it can be constructed according to the two conditions (1) and (2) in some finite number of steps. In the first step, we construct all nice terms just consisting of letters by themselves, i.e.,

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

In the second step, we apply (2) to the terms we've constructed. We'll get

$$[\mathrm{a} \circ \mathrm{a}], [\mathrm{a} \circ \mathrm{b}], [\mathrm{b} \circ \mathrm{a}], \ldots, [\mathrm{d} \circ \mathrm{d}]$$

for all combinations of two letters. In the third step, we apply (2) again, to any two nice terms we've constructed so far. We get new nice term such as $[\mathrm{a} \circ [\mathrm{a} \circ$

a]]—where t is a from step 1 and s is [a ∘ a] from step 2—and [[b ∘ c] ∘ [d ∘ b]] constructed out of the two terms [b ∘ c] and [d ∘ b] from step 2. And so on. Clause (3) rules out that anything not constructed in this way sneaks into the set of nice terms.

Note that we have not yet proved that every sequence of symbols that "feels" nice is nice according to this definition. However, it should be clear that everything we can construct does in fact "feel nice": brackets are balanced, and ∘ connects parts that are themselves nice.

The key feature of inductive definitions is that if you want to prove something about all nice terms, the definition tells you which cases you must consider. For instance, if you are told that t is a nice term, the inductive definition tells you what t can look like: t can be a letter, or it can be $[s_1 \circ s_2]$ for some pair of nice terms s_1 and s_2. Because of clause (3), those are the only possibilities.

When proving claims about all of an inductively defined set, the strong form of induction becomes particularly important. For instance, suppose we want to prove that for every nice term of length n, the number of [in it is $< n/2$. This can be seen as a claim about all n: for every n, the number of [in any nice term of length n is $< n/2$.

Proposition B.4. *For any n, the number of [in a nice term of length n is $< n/2$.*

Proof. To prove this result by (strong) induction, we have to show that the following conditional claim is true:

> If for every $l < k$, any nice term of length l has $< l/2$ ['s, then any nice term of length k has $< k/2$ ['s.

To show this conditional, assume that its antecedent is true, i.e., assume that for any $l < k$, nice terms of length l contain $< l/2$ ['s. We call this assumption the inductive hypothesis. We want to show the same is true for nice terms of length k.

So suppose t is a nice term of length k. Because nice terms are inductively defined, we have two cases: (1) t is a letter by itself, or (2) t is $[s_1 \circ s_2]$ for some nice terms s_1 and s_2.

1. t is a letter. Then $k = 1$, and the number of [in t is 0. Since $0 < 1/2$, the claim holds.

2. t is $[s_1 \circ s_2]$ for some nice terms s_1 and s_2. Let's let l_1 be the length of s_1 and l_2 be the length of s_2. Then the length k of t is $l_1 + l_2 + 3$ (the lengths of s_1 and s_2 plus three symbols [, ∘,]). Since $l_1 + l_2 + 3$ is always greater than l_1, $l_1 < k$. Similarly, $l_2 < k$. That means that the induction hypothesis applies to the terms s_1 and s_2: the number m_1 of [in s_1 is $< l_1/2$, and the number m_2 of [in s_2 is $< l_2/2$.

The number of $[$ in t is the number of $[$ in s_1, plus the number of $[$ in s_2, plus 1, i.e., it is $m_1 + m_2 + 1$. Since $m_1 < l_1/2$ and $m_2 < l_2/2$ we have:

$$m_1 + m_2 + 1 < \frac{l_1}{2} + \frac{l_2}{2} + 1 = \frac{l_1 + l_2 + 2}{2} < \frac{l_1 + l_2 + 3}{2} = k/2.$$

In each case, we've shown that the number of $[$ in t is $< k/2$ (on the basis of the inductive hypothesis). By strong induction, the proposition follows. $\square$

Define the set of supernice terms by

1. Any letter a, b, c, d is a supernice term.
2. If s is a supernice term, then so is $[s]$.
3. If s_1 and s_2 are supernice terms, then so is $[s_1 \circ s_2]$.
4. Nothing else is a supernice term.

Show that the number of $[$ in a supernice term t of length n is $\leq n/2 + 1$.

B.5 Structural Induction

So far we have used induction to establish results about all natural numbers. But a corresponding principle can be used directly to prove results about all elements of an inductively defined set. This often called *structural* induction, because it depends on the structure of the inductively defined objects.

Generally, an inductive definition is given by (a) a list of "initial" elements of the set and (b) a list of operations which produce new elements of the set from old ones. In the case of nice terms, for instance, the initial objects are the letters. We only have one operation: the operations are

$$o(s_1, s_2) = [s_1 \circ s_2]$$

You can even think of the natural numbers $\mathbb{N}$ themselves as being given by an inductive definition: the initial object is 0, and the operation is the successor function $x + 1$.

In order to prove something about all elements of an inductively defined set, i.e., that every element of the set has a property P, we must:

1. Prove that the initial objects have P
2. Prove that for each operation o, if the arguments have P, so does the result.

For instance, in order to prove something about all nice terms, we would prove that it is true about all letters, and that it is true about $[s_1 \circ s_2]$ provided it is true of s_1 and s_2 individually.

Proposition B.5. *The number of* [*equals the number of*] *in any nice term* t*.*

Proof. We use structural induction. Nice terms are inductively defined, with letters as initial objects and the operation o for constructing new nice terms out of old ones.

1. The claim is true for every letter, since the number of [in a letter by itself is 0 and the number of] in it is also 0.

2. Suppose the number of [in s_1 equals the number of], and the same is true for s_2. The number of [in $o(s_1, s_2)$, i.e., in $[s_1 \circ s_2]$, is the sum of the number of [in s_1 and s_2 plus one. The number of] in $o(s_1, s_2)$ is the sum of the number of] in s_1 and s_2 plus one. Thus, the number of [in $o(s_1, s_2)$ equals the number of] in $o(s_1, s_2)$. □

Prove by structural induction that no nice term starts with].

Let's give another proof by structural induction: a proper initial segment of a string t of symbols is any string s that agrees with t symbol by symbol, read from the left, but t is longer. So, e.g., $[a \circ$ is a proper initial segment of $[a \circ b]$, but neither are $[b \circ$ (they disagree at the second symbol) nor $[a \circ b]$ (they are the same length).

Proposition B.6. *Every proper initial segment of a nice term* t *has more* ['s than]'s.

Proof. By induction on t:

1. t is a letter by itself: Then t has no proper initial segments.

2. $t = [s_1 \circ s_2]$ for some nice terms s_1 and s_2. If r is a proper initial segment of t, there are a number of possibilities:

   a) r is just [: Then r has one more [than it does].

   b) r is $[r_1$ where r_1 is a proper initial segment of s_1: Since s_1 is a nice term, by induction hypothesis, r_1 has more [than] and the same is true for $[r_1$.

   c) r is $[s_1$ or $[s_1 \circ$: By the previous result, the number of [and] in s_1 are equal; so the number of [in $[s_1$ or $[s_1 \circ$ is one more than the number of].

   d) r is $[s_1 \circ r_2$ where r_2 is a proper initial segment of s_2: By induction hypothesis, r_2 contains more [than]. By the previous result, the number of [and of] in s_1 are equal. So the number of [in $[s_1 \circ r_2$ is greater than the number of].

   e) r is $[s_1 \circ s_2$: By the previous result, the number of [and] in s_1 are equal, and the same for s_2. So there is one more [in $[s_1 \circ s_2$ than there are]. □

B.6 Relations and Functions

When we have defined a set of objects (such as the natural numbers or the nice terms) inductively, we can also define *relations on* these objects by induction. For instance, consider the following idea: a nice term t_1 is a subterm of a nice term t_2 if it occurs as a part of it. Let's use a symbol for it: $t_1 \sqsubseteq t_2$. Every nice term is a subterm of itself, of course: $t \sqsubseteq t$. We can give an inductive definition of this relation as follows:

Definition B.7. The relation of a nice term t_1 being a subterm of t_2, $t_1 \sqsubseteq t_2$, is defined by induction on t_2 as follows:

1. If t_2 is a letter, then $t_1 \sqsubseteq t_2$ iff $t_1 = t_2$.
2. If t_2 is $[s_1 \circ s_2]$, then $t_1 \sqsubseteq t_2$ iff $t_1 = t_2$, $t_1 \sqsubseteq s_1$, or $t_1 \sqsubseteq s_2$.

This definition, for instance, will tell us that $\mathrm{a} \sqsubseteq [\mathrm{b} \circ \mathrm{a}]$. For (2) says that $\mathrm{a} \sqsubseteq [\mathrm{b} \circ \mathrm{a}]$ iff $\mathrm{a} = [\mathrm{b} \circ \mathrm{a}]$, or $\mathrm{a} \sqsubseteq b$, or $\mathrm{a} \sqsubseteq \mathrm{a}$. The first two are false: a clearly isn't identical to $[\mathrm{b} \circ \mathrm{a}]$, and by (1), $\mathrm{a} \sqsubseteq \mathrm{b}$ iff $\mathrm{a} = \mathrm{b}$, which is also false. However, also by (1), $\mathrm{a} \sqsubseteq \mathrm{a}$ iff $\mathrm{a} = \mathrm{a}$, which is true.

It's important to note that the success of this definition depends on a fact that we haven't proved yet: every nice term t is either a letter by itself, or there are *uniquely determined* nice terms s_1 and s_2 such that $t = [s_1 \circ s_2]$. "Uniquely determined" here means that if $t = [s_1 \circ s_2]$ it isn't *also* $= [r_1 \circ r_2]$ with $s_1 \neq r_1$ or $s_2 \neq r_2$. If this were the case, then clause (2) may come in conflict with itself: reading t_2 as $[s_1 \circ s_2]$ we might get $t_1 \sqsubseteq t_2$, but if we read t_2 as $[r_1 \circ r_2]$ we might get not $t_1 \sqsubseteq t_2$. Before we prove that this can't happen, let's look at an example where it *can* happen.

Definition B.8. Define *bracketless terms* inductively by

1. Every letter is a bracketless term.
2. If s_1 and s_2 are bracketless terms, then $s_1 \circ s_2$ is a bracketless term.
3. Nothing else is a bracketless term.

Bracketless terms are, e.g., a, $\mathrm{b} \circ \mathrm{d}$, $\mathrm{b} \circ \mathrm{a} \circ \mathrm{b}$. Now if we defined "subterm" for bracketless terms the way we did above, the second clause would read

$$\text{If } t_2 = s_1 \circ s_2 \text{, then } t_1 \sqsubseteq t_2 \text{ iff } t_1 = t_2,\ t_1 \sqsubseteq s_1 \text{, or } t_1 \sqsubseteq s_2.$$

Now $\mathrm{b} \circ \mathrm{a} \circ \mathrm{b}$ is of the form $s_1 \circ s_2$ with

$$s_1 = \mathrm{b} \text{ and} \qquad\qquad s_2 = \mathrm{a} \circ \mathrm{b}.$$

It is also of the form $r_1 \circ r_2$ with

$$r_1 = \mathrm{b} \circ \mathrm{a} \text{ and} \qquad\qquad r_2 = \mathrm{b}.$$

Now is a ∘ b a subterm of b ∘ a ∘ b? The answer is yes if we go by the first reading, and no if we go by the second.

The property that the way a nice term is built up from other nice terms is unique is called *unique readability*. Since inductive definitions of relations for such inductively defined objects are important, we have to prove that it holds.

Proposition B.9. *Suppose t is a nice term. Then either t is a letter by itself, or there are uniquely determined nice terms s_1, s_2 such that $t = [s_1 \circ s_2]$.*

Proof. If t is a letter by itself, the condition is satisfied. So assume t isn't a letter by itself. We can tell from the inductive definition that then t must be of the form $[s_1 \circ s_2]$ for some nice terms s_1 and s_2. It remains to show that these are uniquely determined, i.e., if $t = [r_1 \circ r_2]$, then $s_1 = r_1$ and $s_2 = r_2$.

So suppose $t = [s_1 \circ s_2]$ and also $t = [r_1 \circ r_2]$ for nice terms s_1, s_2, r_1, r_2. We have to show that $s_1 = r_1$ and $s_2 = r_2$. First, s_1 and r_1 must be identical, for otherwise one is a proper initial segment of the other. But by Proposition B.6, that is impossible if s_1 and r_1 are both nice terms. But if $s_1 = r_1$, then clearly also $s_2 = r_2$. □

We can also define functions inductively: e.g., we can define the function f that maps any nice term to the maximum depth of nested $[\dots]$ in it as follows:

Definition B.10. The *depth* of a nice term, $f(t)$, is defined inductively as follows:

$$f(t) = \begin{cases} 0 & \text{if } t \text{ is a letter} \\ \max(f(s_1), f(s_2)) + 1 & \text{if } t = [s_1 \circ s_2]. \end{cases}$$

For instance

$$\begin{aligned} f([\mathrm{a} \circ \mathrm{b}]) &= \max(f(\mathrm{a}), f(\mathrm{b})) + 1 = \\ &= \max(0, 0) + 1 = 1, \text{ and} \\ f([[\mathrm{a} \circ \mathrm{b}] \circ \mathrm{c}]) &= \max(f([\mathrm{a} \circ \mathrm{b}]), f(\mathrm{c})) + 1 = \\ &= \max(1, 0) + 1 = 2. \end{aligned}$$

Here, of course, we assume that s_1 an s_2 are nice terms, and make use of the fact that every nice term is either a letter or of the form $[s_1 \circ s_2]$. It is again important that it can be of this form in only one way. To see why, consider again the bracketless terms we defined earlier. The corresponding "definition" would be:

$$g(t) = \begin{cases} 0 & \text{if } t \text{ is a letter} \\ \max(g(s_1), g(s_2)) + 1 & \text{if } t = s_1 \circ s_2. \end{cases}$$

Now consider the bracketless term a ∘ b ∘ c ∘ d. It can be read in more than one way, e.g., as $s_1 \circ s_2$ with

$$s_1 = \mathrm{a} \text{ and} \qquad\qquad s_2 = \mathrm{b} \circ \mathrm{c} \circ \mathrm{d},$$

or as $r_1 \circ r_2$ with

$$r_1 = \mathrm{a} \circ b \text{ and} \qquad\qquad r_2 = \mathrm{c} \circ \mathrm{d}.$$

Calculating g according to the first way of reading it would give

$$\begin{aligned} g(s_1 \circ s_2) &= \max(g(\mathrm{a}), g(\mathrm{b} \circ \mathrm{c} \circ \mathrm{d})) + 1 = \\ &= \max(0, 2) + 1 = 3 \end{aligned}$$

while according to the other reading we get

$$\begin{aligned} g(r_1 \circ r_2) &= \max(g(\mathrm{a} \circ \mathrm{b}), g(\mathrm{c} \circ \mathrm{d})) + 1 = \\ &= \max(1, 1) + 1 = 2 \end{aligned}$$

But a function must always yield a unique value; so our "definition" of g doesn't define a function at all.

Give an inductive definition of the function l, where $l(t)$ is the number of symbols in the nice term t.

Prove by structural induction on nice terms t that $f(t) < l(t)$ (where $l(t)$ is the number of symbols in t and $f(t)$ is the depth of t as defined in Definition B.10).

Problems

Bibliography

Cantor, Georg. 1892. Über eine elementare Frage der Mannigfaltigkeitslehre. *Jahresbericht der deutschen Mathematiker-Vereinigung* 1: 75–8.

Cheng, Eugenia. 2004. How to write proofs: A quick quide. URL http://http://eugeniacheng.com/wp-content/uploads/2017/02/cheng-proofguide.pdf.

Frege, Gottlob. 1884. *Die Grundlagen der Arithmetik: Eine logisch mathematische Untersuchung über den Begriff der Zahl*. Breslau: Wilhelm Koebner. Translation in Frege (1953).

Frege, Gottlob. 1953. *Foundations of Arithmetic*, ed. J. L. Austin. Oxford: Basil Blackwell & Mott, 2nd ed.

Hammack, Richard. 2013. *Book of Proof*. Richmond, VA: Virginia Commonwealth University. URL http://www.people.vcu.edu/~rhammack/BookOfProof/BookOfProof.pdf.

Hutchings, Michael. 2003. Introduction to mathematical arguments. URL https://math.berkeley.edu/~hutching/teach/proofs.pdf.

Magnus, P. D., Tim Button, J. Robert Loftis, Aaron Thomas-Bolduc, Robert Trueman, and Richard Zach. 2021. *Forall* x*: Calgary. An Introduction to Formal Logic*. Calgary: Open Logic Project, f21 ed. URL https://forallx.openlogicproject.org/.

Potter, Michael. 2004. *Set Theory and its Philosophy*. Oxford: Oxford University Press.

Sandstrum, Ted. 2019. *Mathematical Reasoning: Writing and Proof*. Allendale, MI: Grand Valley State University. URL https://scholarworks.gvsu.edu/books/7/.

Smullyan, Raymond M. 1968. *First-Order Logic*. New York, NY: Springer. Corrected reprint, New York, NY: Dover, 1995.

Solow, Daniel. 2013. *How to Read and Do Proofs*. Hoboken, NJ: Wiley.

Steinhart, Eric. 2018. *More Precisely: The Math You Need to Do Philosophy*. Peterborough, ON: Broadview, 2nd ed.

Velleman, Daniel J. 2019. *How to Prove It: A Structured Approach*. Cambridge: Cambridge University Press, 3rd ed.

Zuckerman, Martin M. 1973. Formation sequences for propositional formulas. *Notre Dame Journal of Formal Logic* 14(1): 134–138.