Logic II

Nicole Wyatt

Readings for Philosophy 379
Winter Term 2015
University of Calgary

This text is based on the
Open Logic Project
Version of January 12, 2015

Contents

Preface

A formal logic consists of a symbolic language together with a semantics, which captures the possible meanings or truth-conditions of the sentences of the language, and a deductive system, which aims to capture which inferences are correct. In this course we study the scope and limits of formal logic by examining the relationship between these three parts of a logic. The major results to be presented include soundness ("the deductive system captures only truths"), completeness ("the deductive system captures all the truths"), undecidability ("there is no mechanical procedure for establishing whether or not an argument is valid"), and the Löwenheim-Skolem theorems (which concern some of the limits on the expressive power of first-order logic). Along the way we will study some set theory, Turing machines, the limits of computation, and some of the philosophical motivations for the development of first-order logic in the early twentieth century. The course is fast-paced and students will need to supplement the lectures with independent study.

Week 1 (Jan 13). Introduction. Logic and mechanical procedures. **No class Jan 15**

Week 2 (Jan 20, 22). Sets, Relations, Functions. Enumerability. *Assignment 1 due Jan 22*

Week 3 (Jan 27, 29). Syntax and Semantics of FOL. *Assignment 2 due Jan 29*

Week 4 (Feb 3, 5). Syntax and Semantics of FOL continued. *Assignment 3 due Feb 5*

Week 5 (Feb 10, 12). Sequent Calculus and Proofs in FOL. *Assignment 4 due Feb 12*

Reading week No classes Feb 16-20.

Week 6 (Feb 24). Introduction to soundness and completeness. *Midterm exam Feb 26th*

Week 7 (Mar 3, 5). Completeness Proofs

Week 8 (Mar 10, 12). Compactness and Löwenheim-Skolem Theorems *Assignment 5 due Mar 12*

Week 9 (Mar 17, 19). Computability and Turing Machines *Assignment 6 due Mar 19*

Week 10 (Mar 24, 26). Turing machines continued. *Assignment 7 due Mar 26*

Week 11 (Mar 31). The Church-Turing Thesis. **No class April 2** *Assignment 8 due April 2*

Week 12 (Apr 7, 9). Undecidability

Week 13 (Apr 14). Undecidability continued. **Final exam will be scheduled by the registrar**

Part I

Sets, Relations, Functions

Chapter 1

Sets

1.1 Basics

Sets are the most fundamental building blocks of mathematical objects. In fact, almost every mathematical object can be seen as a set of some kind. In logic, as in other parts of mathematics, sets and set theoretical talk is ubiquitous. So it will be important to discuss what sets are, and introduce the notations necessary to talk about sets and operations on sets in a standard way.

Definition 1.1. A *set* is a collection of objects, considered independently of the way it is specified, of the order of its elements, or of their multiplicity. The objects making up the set are called *elements* or *members* of the set. If a is an element of a set X, we write $a \in X$ (otherwise, $a \notin X$). The set which has no elements is called the empty set and denoted $\emptyset$.

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 $S = \{\text{Ruth}\}$. In general, when we have some objects $a_1, \ldots, a_n$, then the set consisting of exactly those objects is written $\{a_1, \ldots, a_n\}$. Frequently we'll specify a set by some property that its elements share—as we just did, for instance, by specifying S as the set of Richard's siblings. We'll use the following shorthand notation for that: $\{x : \ldots x \ldots\}$, where the $\ldots x \ldots$ stands for the property that x has to have in order to be counted among the elements of the set. In our example, we could have specified S also as $S = \{x : \text{is a sibling of Richard}\}$.

When we say that a sets are independent of the way they are specified, we mean that the elements of a set are all that matters. For instance, it so happens that $\{\text{Nicole}, \text{Jacob}\}$, $\{x : \text{is a niece or nephew of Richard}\}$ and $\{x : \text{is a child of Ruth}\}$ are three ways of specifying one and the same set.

Saying that sets are considered independently of the order of their elements and their multiplicity is a fancy way of saying that $\{\text{Nicole}, \text{Jacob}\}$ and

{Jacob, Nicole} are two ways of specifying the same set; and that {Nicole, Jacob} and {Jacob, Nicole, Nicole} are two ways of specifying the same set.

1.2 Some Important Sets

Example 1.3. Mostly we'll be dealing with sets that have mathematical objects as members. You will remember the various sets of numbers: $\mathbb{N}$ is the set of *natural* numbers 0, 1, 2, 3, ...; $\mathbb{Z}$ the set of *integers* ..., -3, -2, -1, 0, 1, 2, 3, ...; $\mathbb{Q}$ the set of *rationals* ($\mathbb{Q} = \{z/n : z \in \mathbb{Z}, n \in \mathbb{N}, n \neq 0\}$); and $\mathbb{R}$ the set of *real* numbers. These are all *infinite* sets, that is, they each have infinitely many elements. As it turns out, $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ have the same number of elements, while $\mathbb{R}$ has a whole bunch more—$\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ are "countably infinite" whereas $\mathbb{R}$ is "uncountable".

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

Example 1.4 (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,

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

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

Example 1.5 (Infinite sequences)**.** For any set A we may also consider the set A^ω of infinite sequences of elements of A. An infinite sequence $a_1a_2a_3a_4\dots$ consists of a one-way infinite list of elements of A.

1.3 Subsets

Sets are made up of their elements, and every element of a set is a part of that set. But there is also a sense that some of the elements of a set *taken together* are a "part of" that set. For instance, the number 2 is part of the set of integers, but the set of even numbers is also a part of the set of integers. It's important to keep those two senses of being part of a set separate.

Definition 1.6. If every element of a set X is also an element of Y, then we say that X is a *subset* of Y, and write $X \subseteq Y$.

Example 1.7. First of all, 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 $\{a, b, c\}$.

Note that a set may contain other sets!In particular, a set may happen to *both* be an element and a subset of another, e.g., $\{0\} \in \{0, \{0\}\}$ and also $\{0\} \subseteq \{0, \{0\}\}$.

Definition 1.8. The set consisting of all subsets of a set X is called the *power set of* X, written $\wp(X)$.

$$\wp(X) = \{x : x \subseteq X\}$$

Example 1.9. What are all the possible subsets of $\{a, b, c\}$? They are: $\emptyset$, $\{a\}$, $\{b\}$, $\{c\}$, $\{a, b\}$, $\{a, c\}$, $\{a, c\}$, $\{a, b, c\}$. The set of all these subsets is $\wp(\{a, b, c\})$:

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

Problem 1.1. List all subsets of $\{a, b, c, d\}$.

1.4 Unions and Intersections

Definition 1.10. The *union* of two sets X and Y, written $A \cup B$, is the set of all things which are members of X, Y, or both.

$$X \cup Y = \{x : x \in X \vee x \in Y\}$$

Example 1.11. 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: $\{a, b, c\} \cup \emptyset = \{a, b, c\}$.

Definition 1.12. The *intersection* of two sets$X$ and Y, written $X \cap Y$, is the set of all things which are elements of both X and Y.

$$X \cap Y = \{x : x \in X \wedge x \in Y\}$$

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

Example 1.13. If two sets have no elements in common, their intersection is empty: $\{a, b, c\} \cap \{0, 1\} = \emptyset$.

If two sets have elements in comon, 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 obviously 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 or intersection of all our original sets as the set of all objects which belong to at least one, respectively, to all members of the set.

Definition 1.14. If C is a set of sets, then $\bigcup C$ is the set of elements of elements of C:

$$\bigcup C = \{x : x \text{ belongs to an element of } C\}$$

Definition 1.15. If C is a set of sets, then $\bigcap C$ is the set of objects which all elements of C have in common:

$$\bigcap C = \{x : x \text{ belongs to every element of } C\}$$

Example 1.16. Suppose $C = \{\{a, b\}, \{a, d, e\}, \{a, d\}\}$. Then $\bigcup C = \{a, b, d, e\}$ and $\bigcap C = \{a\}$.

We could also do the same for a sequence of sets $A_1, A_2, \ldots$
$\bigcup_i A_i = \{x : x \text{ belongs to one of the } A_i\}$.
$\bigcap_i A_i = \{x : x \text{ belongs to every } A_i\}$.

Definition 1.17. The *difference* $X \setminus Y$ is the set of all elements of X which are not also elements of Y, i.e.,

$$X \setminus Y = \{x : x \in X \text{ and } x \notin Y\}.$$

1.5 Proofs about sets

Sets and the notations we've introduced so far provide us with convenient shorthands for specifying sets and expressing relationships between them. Often it will also be necessary to prove claims about such relationships. If you're not familiar with mathematical proofs, this may be new to you. So we'll walk through a simple example. We'll prove that for any sets X and Y, it's always the case that $X \cap (X \cup Y) = X$. How do you prove an identity between sets like this? Recall that sets are determined solely by their elements, i.e., sets are identical if they have the same elements. So in this case we have to prove that (a) every element of $X \cap (X \cup Y))$ is also an element of X and, conversely, that (b) every element of X is also an element of $X \cap (X \cup Y)$. In other words, we show that both (a) $X \cap (X \cup Y) \subseteq X$ and (b) $X \subseteq X \cap (X \cup Y)$.

A proof of a general claim like "every element z of $X \cap (X \cup Y)$ is also an element of X" is proved by first assuming that an arbitrary $z \in X \cap (X \cup Y)$ is given, and proving from this assumtion that $Z \in X$. You may know this pattern as "general conditional proof." In this proof we'll also have to make use of the definitions involved in the assumtion and conclusion, e.g., in this case of "$\cap$" and "$\cup$. So case (a) would be argued as follows:

(a) We first want to show that $X \cap (X \cup Y) \subseteq X$, i.e., by definition of $\subseteq$, that if $z \in X \cap (X \cup Y)$ then $z \in X$, for any z. So assume that $z \in X \cap (X \cup Y)$. Since z is an element of the intersection of two sets iff it is an element of both sets, we can conclude that $z \in X$ and also $z \in X \cup Y$. In particular, $z \in X$. But this is what we wanted to show.

This completes the first half of the proof. Note that in the last step we used the fact that if a conjucntion ($z \in X$ and $z \in X \cup Y$) follows from an assumption, each conjunct follows from that same assumption. You may know this rule as "conjunction elimination," or $\wedge$Elim. Now let's prove (b):

(b) We now prove that $X \subseteq X \cap (X \cup Y)$, i.e., by definition of $\subseteq$, that if $z \in X$ then also $z \in X \cap (X \cup Y)$, for any z. Assume $z \in X$. To show that $z \in X \cap (X \cup Y)$, we have to show (by definition of "$\cap$") that (i) $z \in X$ and also (ii) $z \in X \cup Y$. Here (i) is just our assumption, so there is nothing further to prove. 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 $z \in X$, and $X \cup Y$ is the union of X and Y, this is the case here. So $z \in X \cup Y$. We've shown both (i) $z \in X$ and (ii) $z \in X \cup Y$, hence, by definition of "$\cap$," $z \in X \cap (X \cup Y)$.

This was somewhat long-winded, but it illustrates how we reason about sets and their relationships. We usually aren't this explicit; in particular, we might not repeat all the definitions. A "textbook" proof of our result would look something like this.

Proposition 1.18 (Absorption)**.** *For all sets X, Y,*

$$X \cap (X \cup Y) = X$$

Proof. (a) Suppose $z \in X \cap (X \cup Y)$. Then $z \in X$, so $X \cap (X \cup Y) \subseteq X$.

(b) Now suppose $z \in X$. Then also $z \in X \cup Y$, and therefore also $z \in X \cap (X \cup Y)$. Thus, $X \subseteq X \cap (X \cup Y)$. □

Problem 1.2. Prove in detail that $X \cup (X \cap Y) = X$. Then compress it into a "textbook proof." (Hint: for the $X \cup (X \cap Y) \subseteq X$ direction you will need proof by cases, aka $\vee$Elim.)

1.6 Pairs, Tuples, Cartesian Products

Sets have no order to their elements. We just think of them as an unordered collection. So if we want to represent order, we use *ordered pairs* $\langle x, y \rangle$, or more generally, *ordered n-tuples* $\langle x_1, \ldots, x_n \rangle$.

Definition 1.19. Given sets X and Y, their *Cartesian product* $X \times Y$ is $\{\langle x, y\rangle : x \in A \text{ and } y \in B\}$.

Example 1.20. If $X = \{0, 1\}$, and $Y = \{1, a, b\}$, then their product is

$$X \times Y = \{\langle 0, 1\rangle, \langle 0, a\rangle, \langle 0, b\rangle, \langle 1, 1\rangle, \langle 1, a\rangle, \langle 1, b\rangle\}.$$

Example 1.21. If X is a set, the product of X with itself, $X \times X$, is also written X^2. It is the set of *all* pairs $\langle x, y\rangle$ with $x, y \in X$.

Chapter 2

Relations

2.1 Relations as Sets

You will no doubt remember some interesting relations between objects of some of the sets we've mentioned. For instance, numbers come with an *order relation* $<$ and from the theory of whole numbers the relation of *divisibility without remainder* (usually written $n \mid m$) may be familar. There is also the relation *is identical with* that every object bears to itself and to no other thing. But there are many more interesting relations that we'll encounter, and even more possible relations. Before we review them, we'll just point out that we can look at relations as a special sort of set. For this, first recall what a *pair* is: if a and b are two objects, we can combine them into the *ordered pair* $\langle a, b\rangle$. Note that for ordered pairs the order *does* matter, e.g, $\langle a, b\rangle \neq \langle b, a\rangle$, in contrast to unordered pairs, i.e., 2-element sets, where $\{a, b\} = \{b, a\}$.

If X and Y are sets, then the *Cartesian product* $X \times Y$ of X and Y is the set of all pairs $\langle a, b\rangle$ with $a \in X$ and $b \in Y$. In particular, $X^2 = X \times X$ is the set of all pairs from X.

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

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

Then there is a close connection between the number n being less than a number m and the corresponding pair $\langle n, m\rangle$ being a member of R, namely, $n < m$ if and only if $\langle n, m\rangle \in R$. In a sense we can consider the set R to *be* the $<$-relation on the set $\mathbb{N}$. 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. A *binary relation* on a set X is a subset of X^2. If $R \subseteq X^2$ is a binary relation on X and $x, y \in X$, we 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:

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

The subset consisting of the pairs lying on the diagonal, $\{\langle 0,0\rangle, \langle 1,1\rangle, \langle 2,2\rangle, \dots\}$, is the *identity relation on* $\mathbb{N}$. (Since the identity relation is popular, let's define $\mathrm{Id}_X = \{\langle x, x\rangle : x \in X\}$ for any set X.) The subset of all pairs lying above the diagonal, $L = \{\langle 0,1\rangle, \langle 0,2\rangle, \dots, \langle 1,2\rangle, \langle 1,3\rangle, \dots, \langle 2,3\rangle, \langle 2,4\rangle, \dots\}$ is the *less than* relation, i.e., Lnm iff $n < m$. The subset of pairs below the diagonal, $G = \{\langle 1,0\rangle, \langle 2,0\rangle, \langle 2,1\rangle, \langle 3,0\rangle, \langle 3,1\rangle, \langle 3,2\rangle, \dots\}$ is the *greater than* relation, i.e., Gnm iff $n > m$. The union of L with I, $K = L \cup I$, is the *less than or equal to* relation: Kmn iff $n \leq m$. Similarly, $H = G \cup I$ is the *greater than or equal to relation.* 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 *antireflexive*, 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 X^2 is a relation on X, 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 X^2itself is a relation on X as well (one which every pair bears). But also something like $E = \{\langle n, m\rangle : n > 5 \text{ or } m \times n \geq 34\}$ counts as a relation.

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

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 $\wp(()X)$ 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. A relation $R \subset X^2X$ is *reflexive* iff, for every $x \in X$, Rxx.

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

Definition 2.5. A relation $R \subset X^2$ is *symmetric* iff, whenever both Rxy, then Ryx.

Definition 2.6. A relation $R \subset X^2$ is *anti-symmetric* iff, whenever both Rxy and Ryx, then $x = y$ (or, in other words: if $x \neq y$ then either $\not Rxy$ or $\not Ryx$).

Definition 2.7. A relation R is *total* if for all $x, y \in X$, either Rxy or Ryx.

Definition 2.8. A relation $R \subseteq X^2$ that is reflexive, transitive, and anti-symmetric is called a *partial order*. A partial order that is also total is called a *linear order*.

Definition 2.9. A relation $R \subseteq X^2$ that is reflexive, symmetric, and transitive is called an *equivalence relation*.

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.

2.3 Orders

Definition 2.10. A relation which is both reflexive and transitive is called a *preorder.* A preorder which is also anti-symmetric is called a *partial order*. A partial order which is also total is called a *total order* or *linear order.* (If we want to emphasize that the order is reflexive, we add the adjective "weak"—see below).

Example 2.11. Every linear order is also a partial order, and every partial order is also a preorder, but the converses don't hold. For instance, the identity relation and the full relation on X are preorders, but they are not partial orders, because they are not anti-symmetric (if X has more than one element). For a somewhat less silly example, consider the *no longer than* relation $\preccurlyeq$ on $\mathbb{B}^*$: $x \preccurlyeq y$ iff $\mathrm{len}(()x) \leq \mathrm{len}(()y)$. This is a preorder, even a total preorder, but not a partial order.

The relation of *divisibility without remainder* gives us an example of a partial order which isn't a total order: for integers n, m, we say n (evenly) divides m, in symbols: $n \mid m$, if there is some 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 anti-symmetry fails: $1 \mid -1$ and $-1 \mid 1$ but $1 \neq -1$. Another important partial order is the relation $\subseteq$ on a set of sets.

Notice that the examples L and G from Example 2.2, although we said there that they were called "strict orders" are not total orders even though they are total. But there is a close connection, as we will see momentarily.

Definition 2.12. A relation R on $X$is called *irreflexive* if, for all $x \in X$, $x \not R x$. R is called *asymmetric* if for no pair $x, y \in X$ we have xRy and yRx. A *strict partial order* is a relation which is irreflexive, asymmetric, and transitive. A strict partial order which is also linear is called a *strict linear order.*

A strict partial order R on X can be turned into a weak partial order $R'$by adding the identity relation on X: $R' = R \cup \mathrm{Id}_X$. Conversely, starting from a weak partial order, one can get a strict partial order by removing Id_X

Proposition 2.13. *R is a strict partial (linear) order on X iff $R'$is a weak partial (linear) order. Moreover, xRy iff $xR'y$ for all $x \neq y$.*

Example 2.14. $\leq$ is the weak linear order corresponding to the strict linear order $<$. $\subseteq$is the weak partial order corresponding to the strict partial order $\subsetneq$.

Problem 2.3. Show that if R is a weak partial order on X, then $R^- = R \setminus \mathrm{Id}_X$ is a strict partial order and xRy iff xR^-y for all $x \neq y$.

2.4 Operations on Relations

It is often useful to modify or combine relations. We've already used the union of relations above (which is just the union of two relations considered as sets of pairs). Here are some other ways:

Definition 2.15. Let R, S be relations and X a set.

1. The *inverse* R^{-1} of R is $R^-1 = \{\langle b, a\rangle : \langle a, b\rangle \in R\}$.
2. The *relative product* $R \mid S$ of R and S is
$$(R \mid S) = \{\langle a, c\rangle : \text{for some } b, Rab \text{ and } Rbc\}$$
3. The *restriction* $R \restriction X$ of R to X
4. The *application* $R[X]$ of R to X is
$$R[X] = \{b : \text{for some } a, Rab\}$$

Definition 2.16. The *transitive closure* R^+ of a relation R is $R^+ = \bigcup_{i=1}^{\infty} R^i$ where $R^1 = R$ and $R^{i+1} = R^i \mid R$.

The *reflexive transitive closure* of R is $R^* = R^+ \cup I_{\mathrm{dom}(R)}$.

Problem 2.4. Show that the transitive closure of R is in fact transitive.

Chapter 3

Functions

3.1 Basics

A function is a relation in which each object is related to a unique partner. 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. A function, more generally, is something that takes one or more things as input and returns some kind of output. 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. A *function* $f\colon X \to Y$ is a mapping of each elements of X to an elements of Y. We call X the *domain* of f and Y the *codomain* of f. The *range* of f is the subset of the codomain that is actually output by f for some input.

Example 3.2. Multiplication 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 $n \times 1$.

Multiplication is a function because it pairs each input—each pair of natural numbers—with a single output. In 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 $-\sqrt{n}$. 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—generally each student will have at least two parents.

Example 3.3. Let $f\colon \mathbb{N} \to \mathbb{N}$ be defined such that $f(x) = x + 1$. This tells us that f is a function which takes in natural numbers and outputs natural numbers. It then tells us that, given a natural number, f will output its successor.

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 3.4. Let $g\colon \mathbb{N} \to \mathbb{N}$ be defined such that $g(x) = x - 1 + 2$. This tells us that g is a function which takes in natural numbers and outputs natural numbers. It then tells us that, given a natural number, g will output the successor of the successor of its predecessor. Despite their different definitions, g and f are the same function.

Functions f and g defined above are the same because for any natural number x, $x - 1 + 2 = x + 1$. f and g pair each natural number with the same output. Functions, just like relations more generally, can be treated as just sets of pairs. The definitions for f and g specify the same set by means of different equations, and as we know, sets are independent of how they are specified.

Example 3.5. We can also define functions by cases. For instance, we could define $f\colon \mathbb{N} \to \mathbb{N}$ by

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

This is fine, since every natural number is either even or odd, and 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.

3.2 Kinds of functions

Definition 3.6. A function $f\colon X \to Y$ is *surjective* iff Y is also the range of f.

If you want to show that a function is surjective, then you need to show that every object in the codomain is the output of the function given some input or other.

Definition 3.7. A function $f\colon X \to Y$ is *injective* iff for each $x \in X$ there is at most one y in Y such $f(x) = y$.

Any function pairs each input with a unique output. An injective function has a unique input for each possible output. If you want to show that a function f is injective, you need to show that for any element y of the codomain, if $f(x) = y$ and $f(w) = y$, then $x = w$.

A function which is neither injective, nor surjective, is the constant function $f\colon \mathbb{N} \to \mathbb{N}$ where $f(x) = 1$.

A function which is both injective and surjective is the identity function $f\colon \mathbb{N} \to \mathbb{N}$ where $f(x) = x$.

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

The function

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

is surjective, but not injective.

Definition 3.8. A *bijection* between X to Y is a function $f\colon X \to Y$ which is both surjective and injective.

Problem 3.1. Show that if $f\colon X \to Y$ and $g\colon Y \to Z$ are both injective, then $g \circ f\colon X \to Z$ is injective.

Problem 3.2. Show that if $f\colon X \to Y$ and $g\colon Y \to Z$ are both surjective, then $g \circ f\colon X \to Z$ is surjective.

Problem 3.3. A function $g\colon Y \to X$ is the *inverse* of a function $f\colon Y \to X$ if $f(g(y)) = y$ and $g(f(x)) = x$ for all $x \in X$ and $y \in Y$.

Show that if f is bijective, such a function g exists, i.e., define such a function and show that it is a function. Then show that it is also bijective.

3.3 Operations on functions

We've already seen that the inverse of a one-to-one function f^{-1} is itself a function. It is also possible to compose functions.

Definition 3.9. Let $f : X \to Y$ and $g : Y \to W$. Then we *compose* g and f to form the function $g \circ f : X \to W$, where $g \circ f(x) = g(f(x))$.

The function $g \circ f : X \to W$ pairs each member of X with a member of W. We specify which member of W a member of X is paired with as follows—given an input $x \in X$, first apply the function f, to x which will output some $y \in Y$. Then apply the function g to y, which will out put some $w \in W$.

Example 3.10. Consider the functions $f(x) = x + 1$, and $g(x) = 2x$. What function do you get when you compose these two? $g \circ f(x) = g(f(x))$. So that means for every natural number you give this function, you first add one, and then you multiply the result by two. So their composition is $g \circ f(x) = 2(x + 1)$.

Problem 3.4. Show that if the functions $g : X \to Y$ and $f : Y \to W$ are both bijections then $g \circ f : X \to W$ is also a bijection.

3.4 Isomorphism

An *isomorphism* is a bijection that preserves the structure of the sets it relates, where structure is a matter of the relationships that obtain between the members of the sets. Consider the following two sets $X = \{1, 2, 3\}$ and $Y = \{4, 5, 6\}$. These sets are both structured by the relations successor, less than and greater than. An isomorphism between the two sets is a bijection that preserves those structures. So a function $f\colon X \to Y$ is an isomorphism if, among other things, $i < j$ iff $f(i) < f(j)$, and j is the successor of i iff $f(j)$ is the successor of $f(i)$.

Definition 3.11. Let U be the pair $\langle X, R\rangle$ and V be the pair $\langle Y, S\rangle$ such that X and Y are sets and R and S are relations on X and Y respectively. A bijection f from X to Y is an *isomorphism* from U to V iff it preserves the relational structure, that is, for any x and u in X, $\langle x, u\rangle \in R$ iff $\langle f(x), f(u)\rangle \in S$.

Example 3.12. Consider the following two sets $X = \{1, 2, 3\}$ and $Y = \{4, 5, 6\}$, and the relations successor, less than, and greater than. The function $f\colon X \to Y$ where $f(x) = x + 3$ is an isomorphism between X and Y.

3.5 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 3.13. A *partial function* $f\colon X \nrightarrow Y$ is a mapping which assigns to every element of X at most one element of Y. If f assigns an element of Y to $x \in X$, 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 X where it is defined, i.e., $\mathrm{dom}(f) = \{x : f(x)\downarrow\}$.

Example 3.14. Every function $f\colon X \to Y$ is also a partial function. Partial functions that are defined everywhere on X are called *total*.

Example 3.15. The partial function $f\colon \mathbb{R} \nrightarrow \mathbb{R}$ given by $f(x) = 1/x$ is undefined for $x = 0$, and defined everywhere else.

Chapter 4

The Size of Sets

4.1 Introduction

When Georg Cantor developed set theory in the 1870's, his interest was in part to make palatable the idea of an infinite collection — an actual infinity, as the medievals would say. Key to this rehabilitation of the notion of the infinite was the notion of countability.

4.2 Enumerable Sets

Definition 4.1. Informally, an *enumeration* of a set X is a list (possibly infinite) such that every element of X appears some finite number of places into the list. If X has an enumeration, then X is said to be *enumerable*.

A couple of points about enumerations:

1. The order of elements of X in the enumeration does not matter, as long as every element appears: 4, 1, 25, 16, 9 enumerates the (set of the) first five square numbers just as well as 1, 4, 9, 16, 25 does.

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

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

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

5. The following is not an enumeration of the natural numbers: 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.

6. Lists may be gappy: 2, −, 4, −, 6, −, ... enumerates the even natural numbers.

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

The following provides a more formal definition of an enumeration:

Definition 4.2. An *enumeration* of a set X is any surjective function $f : \mathbb{N} \to X$.

Let's convince ourselves that the formal definition and the informal definition using an possibly gappy, possibly infinite list are equivalent. A surjective function (partial or total) from $\mathbb{N}$ to a set X enumerates X. Such a function determines an enumeration as defined informally above. Then an enumeration for X is the list $f(1)$, $f(2)$, $f(3)$, ... Since f is surjective, every element of X is guaranteed to be the value of $f(n)$ for some $n \in \mathbb{N}$. Hence, every element of X appears at some finite place in the list. Since the function may be partial or not injective, the list may be gappy or redundant, but that is acceptable (as noted above). On the other hand, given a list that enumerates all elements of X, we can define an surjective function $f\colon \mathbb{N} \to X$ by letting $f(n)$ be the $(n-1)$st member of the list, or undefined if the list has a gap in the $(n-1)$st spot.

Example 4.3. A function enumerating the natural numbers ($\mathbb{N}$) is simply the identity function given by $f(n) = n$.

Example 4.4. The functions $f\colon \mathbb{N} \to \mathbb{N}$ and $g\colon \mathbb{N} \to \mathbb{N}$ given by

$$f(n) = 2n \text{ and} \tag{4.1}$$

$$g(n) = 2n + 1 \tag{4.2}$$

enumerate the even natural numbers and the odd natural numbers, respectively. However, neither function is an enumeration of $\mathbb{N}$, since neither is surjective.

Example 4.5. The function $f(n) = \lceil \frac{(-1)^n n}{2} \rceil$ (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:

$f(1)$	$f(2)$	$f(3)$	$f(4)$	$f(5)$	$f(6)$	...
$\lceil -\frac{1}{2} \rceil$	$\lceil \frac{2}{2} \rceil$	$\lceil -\frac{3}{2} \rceil$	$\lceil \frac{4}{2} \rceil$	$\lceil -\frac{5}{2} \rceil$	$\lceil \frac{6}{2} \rceil$	...
0	1	-1	2	-2	3	...

That is fine for "easy" sets. What about the set of, say, pairs of natural numbers?

$$\mathbb{N}^2 = \mathbb{N} \times \mathbb{N} = \{\langle n, m\rangle : n, m \in \mathbb{N}\}$$

Another method we can use to enumerate sets is to organize them in an *array*, such as the following:

	1	**2**	**3**	**4**	...
1	$\langle 1,1\rangle$	$\langle 1,2\rangle$	$\langle 1,3\rangle$	$\langle 1,4\rangle$	...
2	$\langle 2,1\rangle$	$\langle 2,2\rangle$	$\langle 2,3\rangle$	$\langle 2,4\rangle$	...
3	$\langle 3,1\rangle$	$\langle 3,2\rangle$	$\langle 3,3\rangle$	$\langle 3,4\rangle$	...
4	$\langle 4,1\rangle$	$\langle 4,2\rangle$	$\langle 4,3\rangle$	$\langle 4,4\rangle$	...
⋮	⋮	⋮	⋮	⋮	⋱

Clearly, every ordered pair in $\mathbb{N}^2$ will appear at least once in the array. In particular, $\langle n, m\rangle$ will appear in the nth column and mth row. But how do we organize the elements of an array into a list? The pattern in the array below demonstrates one way to do this:

	1	2	4	7	...
	3	5	8	...	...
	6	9	...	...	...
	10	...	...	...	...
	⋮	⋮	⋮	⋮	⋱

This pattern is called *Cantor's zig-zag method*. Other patterns are perfectly permissible, as long as they "zig-zag" through every cell of the array. By Cantor's zig-zag method, the enumeration for $\mathbb{N}^2$ according to this scheme would be:

$$\langle 1,1\rangle, \langle 1,2\rangle, \langle 2,1\rangle, \langle 1,3\rangle, \langle 2,2\rangle, \langle 3,1\rangle, \langle 1,4\rangle, \langle 2,3\rangle, \langle 3,2\rangle, \langle 4,1\rangle, \ldots$$

What ought we do about enumerating, say, the set of ordered triples of natural numbers?

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

We can think of $\mathbb{N}^3$ as the Cartesian product of $\mathbb{N}$ and $\mathbb{Z}^2$, that is,

$$\mathbb{N}^3 = \mathbb{N}^2 \times \mathbb{N} = \{(\vec{a}, k) : \vec{a} \in \mathbb{N}^2, k \in \mathbb{N}\}$$

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 $\mathbb{N}^2$:

	1	**2**	**3**	**4**	...
$\langle \mathbf{1},\mathbf{1}\rangle$	$\langle 1,1,1\rangle$	$\langle 1,1,2\rangle$	$\langle 1,1,3\rangle$	$\langle 1,1,4\rangle$	...
$\langle \mathbf{1},\mathbf{2}\rangle$	$\langle 1,2,1\rangle$	$\langle 1,2,2\rangle$	$\langle 1,2,3\rangle$	$\langle 1,2,4\rangle$	...
$\langle \mathbf{2},\mathbf{1}\rangle$	$\langle 2,1,1\rangle$	$\langle 2,1,2\rangle$	$\langle 2,1,3\rangle$	$\langle 2,1,4\rangle$	...
$\langle \mathbf{1},\mathbf{3}\rangle$	$\langle 1,3,1\rangle$	$\langle 1,3,2\rangle$	$\langle 1,3,3\rangle$	$\langle 1,3,4\rangle$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Thus, by using a method like Cantor's zig-zag method, we may similarly obtain an enumeration of $\mathbb{N}^3$.

Problem 4.1. Give an enumeration of the set of all ordered pairs of positive rational numbers.

Problem 4.2. Recall from your introductory logic course that each possible in a 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 4.3. Show that the set of all finite subsets of an arbitrary infinite enumerable set is enumerable.

Problem 4.4. Show that if X and Y are enumerable, so is $X \cup Y$.

Problem 4.5. A set of positive integers is said to be *cofinite* iff it is the complement of a finite set of positive integers. Let I be the set that contains all the finite and cofinite sets of positive integers. Show that I is enumerable.

Problem 4.6. Show that if X_1, X_2, X_3, ... are all enumerable, so is $\bigcup_{i=1}^{\infty} X_i$.

4.3 Non-enumerable Sets

Some sets, such as the set $\mathbb{N}$ of natural numbers, are infinite. So far we've seen examples of infinite sets which were all enumerable. However, there are also infinite sets which do not have this property. Such sets are called *non-enumerable*.

Cantor's method of diagonalization shows a set to be non-enumerable via a reductio proof. We start with the assumption that the set is enumerable, and show that a contradiction results from this assumption. Our first example is the set $\mathbb{B}^\omega$ of all infinite, non-gappy sequences of 0's and 1's.

Theorem 4.6. *$\mathbb{B}^\omega$ is non-enumerable.*

Proof. Suppose, for reductio, that $\mathbb{B}^\omega$ is enumerable, so that there is a list s_1, s_2, s_3, s_4, ... of all the elements of $\mathbb{B}^\omega$. We may arrange this list, and the elements of each sequence s_i in it vertically in an array with the positive integers on the horizontal axis, as so:

1	2	3	4	...
$s_1(1)$	$s_1(2)$	$s_1(3)$	$s_1(4)$	...
$s_2(1)$	$s_2(2)$	$s_2(3)$	$s_2(4)$	...
$s_3(1)$	$s_3(2)$	$s_3(3)$	$s_3(4)$	...
$s_4(1)$	$s_4(2)$	$s_4(3)$	$s_4(4)$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Here $s_1(1)$ is a name for whatever number, a 0 or a 1, is the first member in the sequence s_1, and so on.

Now define $\overline{s}$ as follows: The nth member $\overline{s}(n)$ of the sequence $\overline{s}$ is set to

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

In other words, $\overline{s}(n)$ has the opposite value to $s_n(n)$.

Clearly $\overline{s}$ is a non-gappy infinite sequence of 0s and 1s, since it is just the mirror sequence to the sequence of 0s and 1s that appear on the diagonal of our array. So $\overline{s}$ is an element of $\mathbb{B}^\omega$. Since it is an element of $\mathbb{B}^\omega$, it must appear somewhere in the enumeration of $\mathbb{B}^\omega$, that is, $\overline{s} = s_n$ for some n.

If $\overline{s} = s_n$, then for any m, $\overline{s}(m) = s_n(m)$. (This is just the criterion of identity for sequences—sequences are identical when they agree at every place.)

So in particular, $\overline{s}(n) = s_n(n)$. $\overline{s}(n)$ must be either an 0 or a 1. If it is a 0 then (given the definition of $\overline{s}$) $s_n(n)$ must be a 1. But if it is a 1 then $s_n(n)$ must be a 0. In either case $\overline{s}(n) \neq s_n(n)$. □

Diagonalization need not involve the presence of an array, though the array method is where it takes its name.

Theorem 4.7. *$\wp(\mathbb{Z}^+)$ is not enumerable.*

Proof. Suppose, for reductio, that $\wp(\mathbb{Z}^+)$ is enumerable, and so it has an enumeration, i.e., a list of all subsets of $\mathbb{Z}^+$:

$$Z_1, Z_2, Z_3, \ldots$$

We now define a set $\overline{Z}$ such that for any positive integer i, $i \in \overline{Z}$ iff $i \notin Z_i$:

$$\overline{Z} = \{i \in \mathbb{Z}^+ : i \notin Z_i\}$$

$\overline{Z}$ is clearly a set of positive integers, and thus $\overline{Z} \in \wp(()\mathbb{Z}^+)$. So $\overline{Z}$ must be $= Z_k$ for some $k \in \mathbb{Z}^+$. And if that is the case, i.e., $\overline{Z} = Z_k$, then $i \in \overline{Z}$ iff $i \in Z_k$ for all $i \in \mathbb{Z}^+$.

In particular, $k \in \overline{Z}$ iff $k \in Z_k$.

Now either $k \in Z_k$ or $k \notin Z_k$. In the first case, by the previous line, $k \in \overline{Z}$. But we've defined $\overline{Z}$ so that it contains exactly those $i \in \mathbb{Z}^+$ which are *not* elements of Z_i. So by that definition, we would have to also have $k \notin Z_k$. In the second case, $k \notin Z_k$. But now k satisfies the condition by which we have defined $\overline{Z}$, and that means that $k \in \overline{Z}$. And as $\overline{Z} = Z_k$, we get that $k \in Z_k$ after all. Either case leads to a contradiction. □

Problem 4.7. Show that $\wp(\mathbb{N})$ is non-enumerable.

Problem 4.8. Show that the set of functions $f : \mathbb{Z}^+ \to \mathbb{Z}^+$ is non-enumerable by a direct diagonal argument.

4.4 Reduction

We showed $\wp(()\mathbb{Z}^+)$ to be non-enumerable by a diagonalization argument. However, with the proof of the non-enumerability of $\mathbb{B}^\omega$, the set of all infinite sequences of 0s and 1s, in place, we could have instead showed $\wp(\mathbb{Z}^+)$ to be non-enumerable by showing that *if* $\wp(\mathbb{Z}^+)$ *is enumerable then* $\mathbb{B}^\omega$ *is also enumerable*. This called *reducing* one problem to another.

Proof of Theorem 4.7 by reduction. Suppose, for reductio, $\wp(\mathbb{Z}^+)$ is enumerable, and thus that there is an enumeration of it $Z_1, Z_2, Z_3, \ldots$

Define the function $f : \wp(\mathbb{Z}^+) \to \mathbb{B}^\omega$ by letting $f(Z)$ be the sequence s_k such that $s_k(j) = 1$ iff $j \in Z$.

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 the sequence has 1s. In other words, this is a surjective function.

Now consider the list

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

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. So this list must enumerate $\mathbb{B}^\omega$.

So if $\wp(\mathbb{Z}^+)$ were enumerable, $\mathbb{B}^\omega$ would be enumerable. But $\mathbb{B}^\omega$ is non-enumerable (Theorem 4.6). □

Problem 4.9. Show that the set of all sets of pairs of positive integers is non-enumerable.

Problem 4.10. Show that $\mathbb{N}^\omega$, the set of infinite sequences of natural numbers, is non-enumerable.

Problem 4.11. Let P be the set of total 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 enumerable and Q is not.

Problem 4.12. Let S be the set of all total surjective functions from the set of positive integers to the set $\{0,1\}$. Show that S is non-enumerable.

Problem 4.13. Show that the set $\mathbb{R}$ of all real numbers is non-enumerable.

4.5 Equinumerous Sets

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 with defining when sets are the same size. Let's say sets of the same size are *equinumerous*. We want the formal notion of equinumerosity to correspond with our intuitive notion of "same size," hence the formal notion ought to satisfy the following properties:

Reflexivity: Every set is equinumerous with itself.

Symmetry: For any sets X and Y, if X is equinumerous with Y, then Y is equinumerous with X.

Transitivity: For any sets X, Y, and Z, if X is equinumerous with Y and Y is equinumerous with Z, then X is equinumerous with Z.

In other words, we want equinumerosity to be an *equivalence relation*.

Definition 4.8. A set X is *equinumerous* with a set Y if and only if there is a total bijection f from X to Y (that is, $f\colon X \to Y$).

Proposition 4.9. *Equinumerosity defines an equivalence relation.*

Proof. Let X, Y, and Z be sets.

Reflexivity: Using the identity map $1_X\colon X \to X$, where $1_X(x) = x$ for all $x \in X$, we see that X is equinumerous with itself (clearly, 1_X is bijective).

Symmetry: Suppose that X is equinumerous with Y. Then there is a bijection $f\colon X \to Y$. Since f is bijective, its inverse f^{-1} is also a bijection. Since f is surjective, f^{-1} is total. Hence, $f^{-1}\colon Y \to X$ is a total bijection from Y to X, so Y is also equinumerous with X.

Transitivity: Suppose that X is equinumerous with Y via the total bijection f and that Y is equinumerous with Z via the total bijection g. Then the composition of $g \circ f\colon X \to Z$ is a total bijection, and X is thus equinumerous with Z.

Therefore, equinumerosity is an equivalence relation by the given definition. □

Theorem 4.10. *Suppose X and Y are equinumerous. Then X is enumerable if and only if Y is.*

Proof. Let X and Y be equinumerous. Suppose that X is enumerable. Then there is a possibly partial, surjective function $f\colon \mathbb{N} \to X$. Since X and Y are equinumerous, there is a total bijection $g\colon X \to Y$. Claim: $g \circ f\colon \mathbb{N} \to Y$ is surjective. Clearly, $g \circ f$ is a function (since functions are closed under composition). To see $g \circ f$ is surjective, let $y \in Y$. Since g is surjective, there is an $x \in X$ such that $g(x) = y$. Since f is surjective, there is an $n \in \mathbb{N}$ such that $f(n) = x$. Hence,

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

and thus $g \circ f$ is surjective. Since $g \circ f\colon \mathbb{N} \to Y$ is surjective, Y it is an enumeration of Y, and so Y is enumerable. □

Problem 4.14. Show that if X is equinumerous with U and and Y is equinumerous with V, and the intersections $X \cap Y$ and $U \cap V$ are empty, then the unions $X \cup Y$ and $U \cup V$ are equinumerous.

Problem 4.15. Given an enumeration of a set X, show that if X is not finite then it is equinumerous with the positive integers $\mathbb{Z}^+$.

4.6 Comparing Sizes of Sets

Just like we were able to make precise when two sets have the same size in a way that also accounts for the size of infinite sets, we can also compare the sizes of sets in a precise way. Our definition of "is smaller than (or equinumerous)" will require, instead of a bijection between the sets, a total injective function 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 injective function 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 4.11. $|X| \leq |Y|$ if and only if there is an injective function $f\colon X \to Y$.

Theorem 4.12 (Schröder-Bernstein)**.** *Let X and Y be sets. If $|X| \leq |Y|$ and $|Y| \leq |X|$, then $|X| = |Y|$.*

In other words, if there is a total injective function from X to Y, and if there is a total injective function from Y back to X, then there is a total bijection from X to Y. Sometimes, it can be difficult to think of a bijection between two equinumerous sets, so 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. The Schröder-Bernstein theorem, apart

from being convenient, justifies the act of discussing the "sizes" of sets, for it tells us that set cardinalities have the familiar anti-symmetric property that numbers have.

Part II

Computability

Chapter 5

Turing Machine Computations

5.1 Introduction

Even though the term "Turing machine" evokes the image of a physical machine with moving parts, strictly speaking a Turing machine is a purely mathematical construct. It is perhaps best to think of a Turing machine as a program for a special kind of imaginary mechanism. This mechanism consists of a *tape* and a *read-write head*. In our version of Turing machines, the tape is infinite in one direction (to the right), and it is divided into *squares*, each of which may contain a symbol from a finite *alphabet*. Such alphabets can contain any number of different symbols, but we will mainly make do with three: $\triangleright$, $\sqcup$, and $|$. When he mechanism is started, the tape is empty (i.e., each square contains the symbol $\sqcup$) except for the leftmost square, which contains $\triangleright$, and a finite number of squares which contain the *input*. At any time, the mechanism is in one of a finite number of *states*. At the outset, the head scans the leftmost square and in a specified *initial state*. At each step of the mechanism's run, the content of the square currently scanned together with the state the mechanism is in and the Turing machine program determine what happens next. The Turing machine program consists of a list of 5-tuples $\langle q_i, \sigma, q_j, \sigma', D\rangle$. Whenever the mechanism is in state q_i and reads symbol σ, it replaces the symbol on the current square with σ', the head moves left, right, or stays put according to whether D is L, R, or N, and the mechanism goes into state q_j. When the mechanism enters state h we say it *halts*, and the contents of the tape at that point is its *output*.

5.2 Turing Machines

The formal definition of what constitutes a Turing machine looks abstract, but is actually very simple: it merely packs into one mathematical structure all the information needed to specify the workings of a Turing machine. This includes (1) which states the machine can be in, (2) which symbols are allowed

to be on the tape, (3) which state the machine should start in, and (4) what the instruction set of the machine is.

Definition 5.1. A *Turing machine* $T = \langle Q, \Sigma, s, I \rangle$ consists of

1. a finite set of states Q which includes the *halting state* h,
2. a finite alphabet Σ which includes $\triangleright$ and $\sqcup$,
3. an initial state $s \in Q$,
4. a finite instruction set $I \subseteq Q \times \Sigma \times Q \times \Sigma \times \{L, R, N\}$.

We assume that the tape is infinite in one direction only. For this reason it is useful to designate a special symbol $\triangleright$ as a marker for the left end of the tape. This makes it easier for Turing machne programs to tell when they're "in danger" of running off the tape. Other definitions of Turing machines are possible, including one where the tape is infinite in both directions. In that case, marker for the left end of the tape is not necessary.

5.3 Configurations and Computations

The imaginary mechanism consisting of tape, read/write head, and Turing machine program is really just in intuitive way of visualizing what a Turing machine computation is. Formally, we can define the computation of a Turing machine on a given input as a sequence of *configurations*—and a configuration in turn is a sequence of symbols (corresponding to the contents of the tape at a given point in the computation), a number indicating the position of the read/write head, and a state. Using these, we can define what the Turing machine M computes on a given input.

Definition 5.2. A *configuration* of Turing machine $M = \langle Q, \Sigma, s, I \rangle$ is a triple $\langle C, n, q \rangle$ where

1. $C \in \Sigma^*$ is a finite sequence of symbols from Σ,
2. $n \in \mathbb{N}$ is a number $\leq \text{len}(C)$, and
3. $q \in Q$

The potential input for a Turing machine is a sequence of symbols, usually a sequence that encodes a number in some form. The initial configuration of the Turing machine is that configuration in which we start the Turing machine to work on that input: the tape contains the tape end marker immediately followed by the input written on the squares to the right, the read/write head is scanning the leftmost square of the tape (i.e., the left end marker), and the mechanism is in the designated start state s.

Definition 5.3. The *initial configuration* of M for input $I \in \Sigma^*$ is

$$\langle \triangleright \frown I, 0, s \rangle$$

Definition 5.4. We say that a configuration $\langle C, n, q \rangle$ *yields* $\langle C', n', q' \rangle$ *in one step* (according to M), iff

1. the n-th symbol of C is σ,
2. the instruction set of M contains a tuple $\langle q, \sigma, q', \sigma', D \rangle$,
3. the n-th symbol of C' is $sigma'$,
4. a) $D = L$ and $n' = n - 1$, or
   b) $D = R$ and $n' = n$, or
   c) $D = N$ and $n' = n$,
5. for all $i \neq n$, $C'(i) = C(i)$,
6. if $n' > \text{len}(C)$, then $\text{len}(C') = \text{len}(C) + 1$ and the n'-th symbol of C' is $\sqcup$.

Definition 5.5. A *run of M on input I* is a sequence C_i of configurations of M, where C_0 is the initial configuration of M for input I, and each C_i yields C_{i+1} in one step.

We say that *M halts on input I after k steps* if $C_k = \langle \triangleright \frown O, n, h \rangle$. In that case the *output* of M for input I is O.

5.4 Unary Representation of Numbers

Turing machines work on sequences of symbols written on their tape. Depending on the alphabet a Turing machine uses, these sequences of symbols can represent various inputs and outputs. Of particular interest, of course, are Turing machines which compute *arithmetical* functions, i.e., functions of natural numbers. One very simple way to represent positive integers is by coding them as sequences of a single symbol $|$.

Definition 5.6. A Turing machine M *computes* the function $f\colon \mathbb{N}^n \to \mathbb{N}$ iff M halts on input

$$|^{k_1} \sqcup |^{k_2} \sqcup \ldots \sqcup |^{k_n}$$

with output $|^{f(k_1,\ldots,k_n)}$.

Part III

First-order Logic

Chapter 6

Syntax and Semantics

6.1 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 *formulas* 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: $\neg$ (negation), $\vee$ (disjunction), $\exists$ (existential quantifier).
   b) The two-place identity predicate $=$.
   c) A denumerable set of variables: v_0, v_1, v_2, ...
2. Non-logical symbols, making up the *standard language* of first-order logic
   a) A denumerable set of n-place predicate symbols for each $n > 0$: A^n_0, A^n_1, A^n_2, ...
   b) A denumerable set of constant symbols: c_0, c_1, c_2, ...
   c) A denumerable set of n-place function symbols for each $n > 0$: f^n_0, f^n_1, f^n_2, ...
3. Punctuation marks: (,), and the comma.

In addition to the primitive connectives and quantifier introduced above, we also use the following *defined* symbols: $\wedge$ (conjunction), $\rightarrow$ (conditional), $\leftrightarrow$ (biconditional), $\forall$ (universal quantifier), falsity $\bot$, 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 either $\sim$, $\neg$, and ! for "negation", $\wedge$, $\cdot$, and & 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 very common 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 are treating 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 $\{\neg, \vee\}$, $\{\neg, \wedge\}$, and $\{\neg, \rightarrow\}$ — 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.

6.2 Terms and Formulas

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 *formulas*.

Definition 6.1 (Term)**.** The set of *terms* Trm of $\mathcal{L}$ is defined inductively:

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 $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term.

4. Nothing else is a term.

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

The constants 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 $f(t_1, \ldots, t_n)$ as just f by itself if $n = 0$.

Definition 6.2 (Formula)**.** The set of *formulas* $\mathrm{Frm}(\mathcal{L})$ of the language $\mathcal{L}$ is defined inductively as follows:

1. If R is an n-place predicate symbol of $\mathcal{L}$ and $t_1, \ldots, t_n$ are terms of $\mathcal{L}$, then $R(t_1, \ldots, t_n)$ is an (atomic) formula.

2. If t_1 and t_2 are terms of $\mathcal{L}$, then $t_1 = t_2$ is an atomic formula.

3. If φ is a formula, then $\neg\varphi$ is formula.

4. If φ and ψ are formulas, then $(\varphi \vee \psi)$ is a formula.

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

6. Nothing else is a formula.

By convention, $\neg t_1 = t_2$ is abbreviated as $t_1 \neq t_2$, and when writing a formula containing a two-place operator, we leave out the outermost parentheses.

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

Definition 6.3. Formulas constructed using the defined operators are to be understood as follows:

1. $\top$ abbreviates $(\varphi \vee \neg\varphi)$ for some fixed atomic formula φ.

2. $\bot$ abbreviates $(\varphi \wedge \neg\varphi)$ for some fixed atomic formula φ.

3. $\varphi \wedge \psi$ abbreviates $\neg(\neg\varphi \vee \neg\psi)$.

4. $\varphi \to \psi$ abbreviates $\neg\varphi \vee \psi)$.

5. $\varphi \leftrightarrow \psi$ abbreviates $(\varphi \to \psi) \wedge (\psi \to \varphi)$.

6. $\forall x\, \varphi$ abbreviates $\neg\exists x\, \neg\varphi$.

6.3 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 $\neg\varphi$ is $\neg$, the main operator of $(\varphi \vee \psi)$ is $\vee$, etc.

Definition 6.4 (Main operator)**.** The *main operator* of a formula φ is inductively defined as follows:

1. φ is atomic: φ has no main operator. $\varphi \equiv \varphi$: φ has no main operator.
2. $\varphi \equiv \neg\psi$: the main operator of φ is $\neg$.
3. $\varphi \equiv \psi \vee \chi$: the main operator of φ is $\vee$.
4. $\varphi \equiv \exists x\, \psi$: the main operator of φ is $\exists$.

Formulas with each type of main operator:

Main operator	Type of formula	Example
$\neg$	negation	$\neg\varphi$
$\wedge$	conjunction	$\varphi \wedge \psi$
$\vee$	disjunction	$\varphi \vee \psi$
$\rightarrow$	conditional	$\varphi \rightarrow \psi$
$\forall$	universal	$\forall x\, \varphi$
$\exists$	existential	$\exists x\, \varphi$

6.4 Subformulas

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

Definition 6.5 (Sub-formula)**.** The set of *sub-formulas* of a formula φ, $\mathrm{SFrm}(\varphi)$, is defined inductively as follows:

1. If φ is atomic, then $\mathrm{SFrm}(\varphi) = \{\varphi\}$.
2. $\mathrm{SFrm}(\neg\varphi) = \{\neg\varphi\} \cup \mathrm{SFrm}(\varphi)$.
3. If ψ is also a formula, then $\mathrm{SFrm}(\varphi \vee \psi) = \{\varphi \vee \psi\} \cup \mathrm{SFrm}(\varphi) \cup \mathrm{SFrm}(\psi)$.
4. If x is a variable, then $\mathrm{SFrm}(\exists x\, \varphi) = \{\exists x\, \varphi\} \cup \mathrm{SFrm}(\varphi)$.

The set of all *proper* sub-formulas of φ is $\mathrm{SFrm}(\varphi) \setminus \{\varphi\}$. That is, a formula ψ is a proper subformula of φ if and only if ψ is a sub-formula of φ but is distinct from φ.

Definition 6.6 (Immediate sub-formula)**.** Given formulas φ and ψ, define *immediate sub-formula* inductively as follows:

1. Atomic formulas have no immediate sub-formulas.
2. φ is the immediate sub-formula of $\neg\varphi$.
3. φ and ψ are the immediate sub-formulas of $(\varphi \vee \psi)$.
4. φ is the immediate sub-formula of $\exists x\, \varphi$.

Definition 6.7 (Sub-formula)**.** Define *sub-formula* as follows:

1. If ψ is equal to φ, then ψ is a sub-formula of φ.
2. If ψ is an immediate sub-formula of φ, then ψ is a sub-formula of φ.
3. If χ is a sub-formula of ψ and ψ is a sub-formula of φ, then χ is also a sub-formula of φ.

A formula ψ is a *proper* sub-formula of φ if and only if it is a sub-formula of φ and is not equal to φ.

Definition 6.8 (Proper sub-formula)**.** Define *proper sub-formula* as follows:

1. If ψ is an immediate sub-formula of φ, then ψ is a proper sub-formula of φ.
2. If χ is a proper sub-formula of ψ and ψ is a proper sub-formula of φ, then χ is also a proper sub-formula of φ.

Definition 6.9 (Sub-formula)**.** Define *sub-formula* as follows:

1. If ψ is equal to φ, then ψ is a sub-formula of φ.
2. If ψ is a proper sub-formula of φ, then ψ is a sub-formula of φ.

6.5 Free Variables and Sentences

Definition 6.10 (Free occurrences of a variable)**.** The *free* occurrences of a variable in a formula are defined inductively:

1. If φ is atomic, all variable occurrences in φ are free.
2. If φ is a formula, then the free variable occurrences of $\neg\varphi$ are exactly those of φ.
3. If φ and ψ are formulas, then the free variable occurrences in $(\varphi \vee \psi)$ are those in φ together with those in ψ.

4. If φ is a formula, then the free variable occurrences in $\exists x\, \varphi$ are all of those in φ except for occurrences of x.

Definition 6.11 (Sentence)**.** A formula φ is a *sentence* iff it contains no free occurrences of variables.

Definition 6.12. A term t is *free for x in φ* if x does not occur in φ within the scope of a quantifier $\forall z$ binding a variable z occurring in t.

6.6 Substitution

Definition 6.13 (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 other than 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 6.14 (Substitution in a formula)**.** If φ is a formula, x is a variable, and t is a term, $\varphi[t/x]$ is the result of substituting t for all free occurrences of x in φ.

1. $\varphi \equiv P(t_1, \ldots, t_n)$: $\varphi[t/x]$ is $P(t_1[t/x], \ldots, t_n[t/x])$.
2. $\varphi \equiv t_1 = t_2$: $\varphi[t/x]$ is $t_1[t/x] = t_2[t/x]$.
3. $\varphi \equiv \neg\psi$: $\varphi[t/x]$ is $\neg\psi[t/x]$.
4.
5. $\varphi \equiv \psi \vee \chi$: $\varphi[t/x]$ is $(\psi[t/x] \vee \chi[t/x])$.
6.
7.
8.
9. $\varphi \equiv \exists y\, \psi$: $\varphi[t/x]$ is $\exists y\, \psi[t/x]$, provided y is a variable other than x.
10. $\varphi \equiv \exists x\, \psi$: $\varphi[t/x]$ 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.

Definition 6.15. 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.

6.7 Structures for First-order Languages

First-order languages are, by themselves, *uninterpreted:* the constants, functions, and predicates have no specific meaning attached to them. Meanings are given by specifying a *structure* (sometimes called an *interpretation*). It specifies the *domain*, i.e., the objects which the constats pick out, the functions operate on, and the quantifiers range over. In addition, it specifies which constants pick out which objects, how a function maps objects to objects, and which objects the predicates apply to. Structures are the basis for *semantic* notions in logic, e.g., the notion of consequence, validity, satisfiablity.

Definition 6.16 (Structure)**.** 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. *Name assignment:* for each constant symbol c of $\mathcal{L}$, an element $c^{\mathfrak{M}} \in |\mathfrak{M}|$
3. *Relations:* for each n-place predicate symbol R of $\mathcal{L}$ (other than $=$), an n-ary relation $R^{\mathfrak{M}} \subseteq |\mathfrak{M}|^n$
4. *Functions:* for each n-place function symbol f of $\mathcal{L}$, an n-ary function $f^{\mathfrak{M}} \colon |\mathfrak{M}|^n \to |\mathfrak{M}|$

Recall that a term is *closed* if it is closed or if it is a function of constants only (without variables, whether free or bound).

Definition 6.17 (Denotation of closed terms)**.** If t is a closed term of the langage $\mathcal{L}$ and $\mathfrak{M}$ is a structure for $\mathcal{L}$, the *denotation* $\mathrm{Val}^{\mathfrak{M}}(t)$ is defined as follows:

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

Definition 6.18 (Covered structure)**.** A structure is *covered* if every element of the domain is the denotation of some closed term.

Example 6.19. Let $\mathcal{L}$ be the language with constant symbols Zero, One, Two, ..., the binary predicate symbols $=$ and $<$, and the binary function symbols $+$ and $\times$. Then a structure $\mathfrak{M}$ for $\mathcal{L}$ is the one with domain $|\mathfrak{M}| = \{0, 1, 2, \ldots\}$ and name assignment $\mathsf{Zero}^{\mathfrak{M}} = 0$, $\mathsf{One}^{\mathfrak{M}} = 1$, $\mathsf{Two}^{\mathfrak{M}} = 2$, 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 the integer c_1 is less than the integer c_2: for example, $\langle 1, 3 \rangle \in \mathrm{Val}^{\mathfrak{M}}(<)$ but $\langle 2, 2 \rangle \notin \mathrm{Val}^{\mathfrak{M}}(<)$. 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 denotation of Four is just 4, and the denotation of $\times(\mathsf{Two}, +(\mathsf{Three}, \mathsf{Zero}))$ (or in infix notation, $\mathsf{Two} \times (\mathsf{Three} + \mathsf{Zero})$) is

$$\begin{aligned}
\mathrm{Val}^{\mathfrak{M}}(\times(\mathsf{Two}, +(\mathsf{Three}, \mathsf{Zero}))) &= \times^{\mathfrak{M}}(\mathrm{Val}^{\mathfrak{M}}(\mathsf{Two}), \mathrm{Val}^{\mathfrak{M}}(\mathsf{Two}, +(\mathsf{Three}, \mathsf{Zero}))) \\
&= \times^{\mathfrak{M}}(\mathrm{Val}^{\mathfrak{M}}(\mathsf{Two}), +^{\mathfrak{M}}(\mathrm{Val}^{\mathfrak{M}}(\mathsf{Three}), \mathrm{Val}^{\mathfrak{M}}(\mathsf{Zero}))) \\
&= \times^{\mathfrak{M}}(\mathsf{Two}^{\mathfrak{M}}, +^{\mathfrak{M}}(\mathsf{Three}^{\mathfrak{M}}, \mathsf{Zero}^{\mathfrak{M}})) \\
&= \times^{\mathfrak{M}}(2, +^{\mathfrak{M}}(3, 0)) \\
&= \times^{\mathfrak{M}}(2, 3) \\
&= 5
\end{aligned}$$

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 $\exists x\, (\varphi(x) \vee \neg\varphi(x))$ is valid — that is, a logical truth. And the stipulation that all names 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 $\exists x\, \varphi(x)$.

6.8 Satisfaction of a Formula in a Structure

The basic notion that relates expressions such as terms and formulas, on the one hand, and structures on the other, are those of *denotation* and *satisfaction*. Informally, a term *denotes* an element of a structure—if the term is just a constant, it denotes the object assigned to the constant by the structure, and if it is built up using function symbols, the denotation 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 formulas are satisfied, and we define when a complex formula is satisfied depending on the main connective or quantifier and whether or not the immediate subformulas 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 what variables denote. 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 6.20 (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., $s\colon \mathrm{Var} \to |\mathfrak{M}|$.

Definition 6.21 (Denotation 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 *denotation* $\mathrm{Val}_s^{\mathfrak{M}}(t)$ 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. $t \equiv f(t_1, \ldots, t_n)$:

$$\mathrm{Val}_s^{\mathfrak{M}}(t) = f^{\mathfrak{M}}(\mathrm{Val}_s^{\mathfrak{M}}(t_1), \ldots, \mathrm{Val}_s^{\mathfrak{M}}(t_n)).$$

Definition 6.22 (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 only in what it assigns to x is called an *x-variant* of s. If s' is an x-variant of s we write $s \sim_x s'$.

Definition 6.23 (Satisfaction)**.** Satisfaction of a formula φ in a structure $\mathfrak{M}$ relative to a variable assignment s, $\mathfrak{M}, s \models \varphi$ is defined inductively as follows. If not $\mathfrak{M}, s \models \varphi$ we write $\mathfrak{M}, s \not\models \varphi$.

1. $\varphi \equiv R(t_1, \ldots, t_n)$: $\mathfrak{M}, s \models \varphi$ iff $\langle \mathrm{Val}_s^{\mathfrak{M}}(t_1), \ldots, \mathrm{Val}_s^{\mathfrak{M}}(t_n) \rangle \in R^{\mathfrak{M}}$.
2. $\varphi \equiv t_1 = t_2$: $\mathfrak{M}, s \models \varphi$ iff $\mathrm{Val}_s^{\mathfrak{M}}(t_1) = \mathrm{Val}_s^{\mathfrak{M}}(t_2)$.
3. $\varphi \equiv \neg\psi$: $\mathfrak{M}, s \models \varphi$ iff $\mathfrak{M}, s \not\models \psi$.
4. $\varphi \equiv \psi \vee \chi$: $\mathfrak{M}, s \models \varphi$ iff $\mathfrak{M}, s \models \varphi$ or $\mathfrak{M}, s \models \psi$ (or both).
5. $\varphi \equiv \exists x\, \psi$: $\mathfrak{M}, s \models \varphi$ iff there is an x-variant s' of s so that $\mathfrak{M}, s' \models \psi$.

A variable assignment s provide a value for *every* variable in the language. This is of course not necessary: whether or not a formula φ is satisfied in a structure with respect to s only depends on the assignments s makes to the free variables that actually occur in φ. This is the content of the next theorem. We require variable assignments to assign values to all variables simply because it makes things a lot easier.

Proposition 6.24. *If $x_1, \ldots, x_n$ are the only free variables in φ and $s(x_i) = s'(x_i)$ for $i = 1, \ldots, n$, then $\mathfrak{M}, s \models \varphi$ iff $\mathfrak{M}, s' \models \varphi$.*

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

1. $\varphi \equiv R(t_1, \ldots, t_k)$: let $\mathfrak{M}, s \models \varphi$. Then $\langle \mathrm{Val}_s^{\mathfrak{M}}(t_1), \ldots, \mathrm{Val}_s^{\mathfrak{M}}(t_k) \rangle \in R^{\mathfrak{M}}$. For $1 = i = k$, if t_i is a constant, then $\mathrm{Val}_s^{\mathfrak{M}}(t_i) = \mathrm{Val}^{\mathfrak{M}}(t_i) = \mathrm{Val}_{s'}^{\mathfrak{M}}(t_i)$. If t_i is a free variable, then since the mappings s and s' agree on all free variables, $\mathrm{Val}_s^{\mathfrak{M}}(t_i) = s(t_i) = s'(t_i) = \mathrm{Val}_{s'}^{\mathfrak{M}}(t_i)$. Similarly, if t_i is of

the form $f(t'_1, \ldots, t'_j)$, we will also get $\mathrm{Val}_s^{\mathfrak{M}}(t_i) = \mathrm{Val}_{s'}^{\mathfrak{M}}(t_i)$. Hence, $\mathrm{Val}_s^{\mathfrak{M}}(t_i) = \mathrm{Val}_{s'}^{\mathfrak{M}}(t_i)$ for any term t_i for $i = 1, \ldots, k$, so we also have $\langle \mathrm{Val}_{s'}^{\mathfrak{M}}(t_i), \ldots, \mathrm{Val}_{s'}^{\mathfrak{M}}(t_k) \rangle \in R^{\mathfrak{M}}$.

2. $\varphi \equiv t_1 = t_2$: if $\mathfrak{M}, s \models \varphi$, $\mathrm{Val}_{s'}^{\mathfrak{M}}(t_1) = \mathrm{Val}_s^{\mathfrak{M}}(t_1) = \mathrm{Val}_s^{\mathfrak{M}}(t_2) = \mathrm{Val}_{s'}^{\mathfrak{M}}(t_2)$, so $\mathfrak{M}, s' \models t_1 = t_2$.

Let $\mathfrak{M}, s \models \psi \Leftrightarrow \mathfrak{M}, s' \models \psi$ for all formulas 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.

1. $\varphi \equiv \neg\psi$: if $\mathfrak{M}, s \models \varphi$, then $\mathfrak{M}, s \not\models \psi$, so by the induction hypothesis, $\mathfrak{M}, s' \not\models \psi$, hence $\mathfrak{M}, s' \models \varphi$.

2. $\varphi \equiv \psi \vee \chi$: if $\mathfrak{M}, s \models \varphi$, then $\mathfrak{M}, s \models \psi$ or $\mathfrak{M}, s \models \chi$. By induction hypothesis, $\mathfrak{M}, s' \models \psi$ or $\mathfrak{M}, s' \models \chi$, so $\mathfrak{M}, s' \models \varphi$.

3. $\varphi \equiv \exists x\, \psi$: exercise.

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

Problem 6.1. Complete the proof of Proposition 6.24.

Definition 6.25. If φ is a sentence, we say that a structure $\mathfrak{M}$ *satisfies* φ, $\mathfrak{M} \models \varphi$, iff $\mathfrak{M}, s \models \varphi$ for all variable assignments s.

Corollary 6.26 (Extensionality)**.** *Let φ be a sentence, and $\mathfrak{M}$ and $\mathfrak{M}'$ be structures. If $c^{\mathfrak{M}} = c^{\mathfrak{M}'}$, $R^{\mathfrak{M}} = R^{\mathfrak{M}'}$, and $f^{\mathfrak{M}} = f^{\mathfrak{M}'}$ for every constant symbol c, relation symbol R, and function symbol f occurring in φ, then $\mathfrak{M} \models \varphi$ iff $\mathfrak{M}' \models \varphi$.*

Problem 6.2. Show that if φ is a sentence, $\mathfrak{M} \models \varphi$ iff there is a variable assignment s so that $\mathfrak{M}, s \models \varphi$.

6.9 Extensionality

Extensionality, sometimes called relevance, can be expressed informally as follows: the only thing that bears upon the satisfaction of formula φ in a structure $\mathfrak{M}$ relative to a variable assignment s, are 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 **Undefined token structures** $\mathfrak{M}$ and $\mathfrak{N}$ agree on all the elements of the language appearing in a sentence φ and have the same domain, $\mathfrak{M}$ and $\mathfrak{N}$ must also agree on φ itself.

We will now give a formal statement of the theorem and prove it.

6.10 Semantic Notions

Give 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., satisified, 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 6.27 (Validity)**.** A sentence φ is *valid*, $\models \varphi$, iff $\mathfrak{M} \models \varphi$ for every structure $\mathfrak{M}$.

Definition 6.28 (Entailment)**.** A set of sentences Γ *entails* a sentence φ, $\Gamma \models \varphi$, iff for every structure $\mathfrak{M}$ with $\mathfrak{M} \models \Gamma$, $\mathfrak{M} \models \varphi$.

Definition 6.29 (Satisfiability)**.** A set of sentences Γ is *satisfiable* if $\mathfrak{M} \models \Gamma$ for some structure $\mathfrak{M}$. If Γ is not satisfiable it is called *unsatisfiable*.

Proposition 6.30. *A sentence φ is valid iff $\Gamma \models \varphi$ 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 $\mathfrak{M} \models \Gamma$. Since φ is valid, $\mathfrak{M} \models \varphi$, hence $\Gamma \models \varphi$.

For the contrapositive of the reverse direction, let φ be invalid, so there is a structure $\mathfrak{M}$ with $\not\models M\varphi$. When $\Gamma = \{\top\}$, since $\top$ is valid, $\mathfrak{M} \models \Gamma$. Hence, there is a structure $\mathfrak{M}$ so that $\mathfrak{M} \models \Gamma$ but $\not\models M\varphi$, hence Γ does not entail φ. □

Proposition 6.31. *$\Gamma \models \varphi$ iff $\Gamma \cup \{\neg\varphi\}$ is unsatisfiable.*

Proof. For the forward direction, let Γ entail φ and suppose to the contrary that there is a structure $\mathfrak{M}$ so that $\mathfrak{M} \models \Gamma \cup \{\neg\varphi\}$. Since $\mathfrak{M} \models \Gamma$ and $\Gamma \models \varphi$, $\mathfrak{M} \models \varphi$. Also, since $\mathfrak{M} \models \Gamma \cup \{\neg\varphi\}$, $\mathfrak{M} \models \neg\varphi$, so we have both $\mathfrak{M} \models \varphi$ and $\not\models M\varphi$, a contradiction. Hence, there can be no such structure $\mathfrak{M}$, so $\Gamma \cup \{\varphi\}$ is unsatisfiable.

For the reverse direction, let $\Gamma \cup \{\neg\varphi\}$. So for every structure $\mathfrak{M}$, either $\not\models M\Gamma$ or $\mathfrak{M} \models \varphi$. Hence, for every structure $\mathfrak{M}$ with $\mathfrak{M} \models \Gamma$, $\mathfrak{M} \models \varphi$, so $\Gamma \models \varphi$. □

Proposition 6.32. *If $\Gamma \subseteq \Gamma'$ and $\Gamma \models \varphi$, then $\Gamma' \models \varphi$.*

Proof. Suppose that $\Gamma \subseteq \Gamma'$ and $\Gamma \models \varphi$. Let $\mathfrak{M}$ be such that $\mathfrak{M} \models \Gamma'$; then $\mathfrak{M} \models \Gamma$, and since $\Gamma \models \varphi$, we get that $\mathfrak{M} \models \varphi$. Hence, whenever $\mathfrak{M} \models \Gamma$, $\mathfrak{M} \models \varphi$, so $\Gamma' \models \varphi$. □

Theorem 6.33 (Semantic Deduction Theorem)**.** *$\Gamma \cup \{\varphi\} \models \psi$ iff $\Gamma \models \varphi \to \psi$.*

Proof. For the forward direction, let $\Gamma \cup \{\varphi\} \models \psi$ and let $\mathfrak{M}$ be a structure so that $\mathfrak{M} \models \Gamma$. If $\mathfrak{M} \models \varphi$, then $\mathfrak{M} \models \Gamma \cup \{\varphi\}$, so since $\Gamma \cup \{\varphi\}$ entails ψ, we get $\mathfrak{M} \models \psi$. Therefore, $\mathfrak{M} \models \varphi \to \psi$, so $\Gamma \models \varphi \to \psi$.

For the reverse direction, let $\Gamma \models \varphi \to \psi$ and $\mathfrak{M}$ be a structure so that $\mathfrak{M} \models \Gamma \cup \{\varphi\}$. Then $\mathfrak{M} \models \Gamma$, so $\mathfrak{M} \models \varphi \to \psi$, and since $\mathfrak{M} \models \varphi$, $\mathfrak{M} \models \psi$. Hence, whenever $\mathfrak{M} \models \Gamma \cup \{\varphi\}$, $\mathfrak{M} \models \psi$, so $\Gamma \cup \{\varphi\} \models \psi$. □

Chapter 7

The Sequent Calculus

7.1 Rules and Proofs

Let $\mathcal{L}$ be a first-order language with the usual constants, variables, and logical symbols and auxiliary symbols (parentheses and the comma).

Definition 7.1 (sequent)**.** A *sequent* is an expression of the form

$$\Gamma \Rightarrow \Delta$$

where Γ and Δ are finite (possibly empty) sets of **Undefined token sentences** of the language $\mathcal{L}$. The formulas in Γ are the *antecedent formulae*, while the formulae in Δ are the *succedent formulae*.

The intuitive idea behind a sequent is: if all of the antecedent formulas hold, then some of the succedent formulas hold. That is, if $\Gamma = \{\Gamma_1, \ldots, \Gamma_m\}$ and $\Delta = \{\Delta_1, \ldots, \Delta_n\}$, then $\Gamma \Rightarrow \Delta$ abbreviates

$$\Gamma_1 \wedge \cdots \wedge \Gamma_m \rightarrow \Delta_1 \vee \cdots \vee \Delta_n$$

When $m = 0$, $\quad \Rightarrow \Delta$ means that $\Delta_1 \vee \cdots \Delta_n$ holds. When $n = 0$, $\Gamma \Rightarrow$ means that $\Gamma_1 \wedge \cdots \wedge \Gamma_m$ gives a contradiction. An empty succedent is sometimes filled with the $\perp$ symbol. The empty sequent $\quad \Rightarrow \quad$ canonically represents a contradiction.

We write Γ, φ (or φ, Γ) for $\Gamma \cup \{\varphi\}$, and Γ, Δ for $\Gamma \cup \Delta$.

Definition 7.2 (Inference)**.** An *inference* is an expression of the form

$$\frac{S_1}{S} \quad \text{or} \quad \frac{S_1 \qquad S_2}{S}$$

where S, S_1, and S_2 are sequents. S_1 and S_2 are called *upper sequents* and S is a *lower sequent*.

Inferences represent the idea that whenever the upper sequent(s) is (are) asserted, from it, we may logically infer the lower sequent.

For the following, let $\Gamma, \Delta, \Pi, \Lambda$ represent finite sets of **Undefined token sentence**.

The rules for **LK** are divided into two main types: *structural* rules and *logical* rules. The logical rules are further divided into *propositional* rules (quantifier-free) and *quantifier* rules.

Structural rules:

Weakening:

$$\frac{\Gamma \Rightarrow \Delta}{\varphi, \Gamma \Rightarrow \Delta} \quad \text{and} \quad \frac{\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \varphi}$$

where φ is called the *weakening formula*.

Cut:

$$\frac{\Gamma \Rightarrow \Delta, \varphi \qquad \varphi, \Pi \Rightarrow \Lambda}{\Gamma, \Pi \Rightarrow \Delta, \Lambda}$$

Logical rules: The rules are named by the logical symbol of the *principal formula* of the inference (the formula containing φ and/or ψ in the lower sequent). The designations "left" and "right" indicate whether the logical symbol has been introduced in an antecedent formula or a succedent formula (to the left or to the right of the sequent symbol).

Propositional Rules:

$$\frac{\Gamma \Rightarrow \Delta, \varphi}{\neg\varphi, \Gamma \Rightarrow \Delta} \neg \text{ left} \quad \frac{\varphi, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \neg\varphi} \neg \text{ right}$$

$$\frac{\varphi, \Gamma \Rightarrow \Delta}{\varphi \wedge \psi, \Gamma \Rightarrow \Delta} \wedge \text{ left} \quad \frac{\psi, \Gamma \Rightarrow \Delta}{\varphi \wedge \psi, \Gamma \Rightarrow \Delta} \wedge \text{ left} \quad \frac{\Gamma \Rightarrow \Delta, \varphi \qquad \Gamma \Rightarrow \Delta, \psi}{\Gamma \Rightarrow \Delta, \varphi \wedge \psi} \wedge \text{ right}$$

$$\frac{\varphi, \Gamma \Rightarrow \Delta \qquad \psi, \Gamma \Rightarrow \Delta}{\varphi \vee \psi, \Gamma \Rightarrow \Delta} \vee \text{ left} \quad \frac{\Gamma \Rightarrow \Delta, \varphi}{\Gamma \Rightarrow \Delta, \varphi \vee \psi} \vee \text{ right} \quad \frac{\Gamma \Rightarrow \Delta, \psi}{\Gamma \Rightarrow \Delta, \varphi \vee \psi} \vee \text{ right}$$

$$\frac{\Gamma \Rightarrow \Delta, \varphi \qquad \psi, \Pi \Rightarrow \Lambda}{\varphi \to \psi, \Gamma, \Pi \Rightarrow \Delta, \Lambda} \to \text{ left} \quad \frac{\varphi, \Gamma \Rightarrow \Delta, \psi}{\Gamma \Rightarrow \Delta, \varphi \to \psi} \to \text{ right}$$

Quantifier Rules:

$$\frac{\varphi(t), \Gamma \Rightarrow \Delta}{\forall x\, \varphi(x), \Gamma \Rightarrow \Delta} \forall \text{ left} \quad \frac{\Gamma \Rightarrow \Delta, \varphi(a)}{\Gamma \Rightarrow \Delta, \forall x\, \varphi(x)} \forall \text{ right}$$

where t is a ground term, and a is a constant which does not occur anywhere in the lower sequent of the $\forall$ right rule. We call a the *eigenvariable* of the $\forall$ right inference.

$$\frac{\varphi(a), \Gamma \Rightarrow \Delta}{\exists x\, \varphi(x), \Gamma \Rightarrow \Delta}\ \exists \text{ left} \qquad \frac{\Gamma \Rightarrow \Delta, \varphi(t)}{\Gamma \Rightarrow \Delta, \exists x\, \varphi(x)}\ \exists \text{ right}$$

where t is a ground term, and a is a constant which does not occur in the lower sequent of the $\exists$ left rule. We call a the *eigenvariable* of the $\exists$ left inference.

The condition that an eigenvariable not occur in the upper sequent of the $\forall$ right or $\exists$ left inference is called the *eigenvariable condition*.

We use the term "eigenvariable" even though a in the above rules is a constant. This has historical reasons.

Definition 7.3 (Initial Sequent)**.** An *initial sequent* is a sequent of the form $\varphi \Rightarrow \varphi$ for any **Undefined token sentence** φ in the language.

Definition 7.4 (LK derivation)**.** An **LK**-derivation of a sequent S is a tree of sequents satisfying the following conditions:

1. The topmost sequents of the tree are initial sequents.
2. Every sequent in the tree (except S) is an upper sequent of an inference whose lower sequent stands directly below that sequent in the tree.

We then say that S is the *end-sequent* of the derivation and that S is *derivable in* **LK** (or **LK**-derivable).

Definition 7.5 (LK theorem)**.** A **Undefined token sentence** φ is a *theorem* of **LK** if the sequent $\Rightarrow \varphi$ is **LK**-derivable.

7.2 Proving Things

Example 7.6. Give an **LK**-derivation for the sequent $\varphi \wedge \psi \Rightarrow \varphi$.

We begin by writing the desired end-sequent at the bottom of derivation.

$$\frac{}{\varphi \wedge \psi \Rightarrow \varphi}$$

Next, we need to figure out what kind of inference could have a lower sequent of this form. This could be a structural rule, but it is a good idea to start by looking for a logical rule. The only logical connective occurring in a formula in the lower sequent is $\wedge$, so we're looking for an $\wedge$ rule, and since the $\wedge$ symbol occurs in the antecedent formulas, we're looking at the $\wedge$ left rule.

$$\frac{}{\varphi \wedge \psi \Rightarrow \varphi}\ \wedge \text{ left}$$

There are two options for what could have been the upper sequent of the $\wedge$ left inference: we could have an upper sequent of $\varphi \Rightarrow \varphi$, or of $\psi \Rightarrow \varphi$. Clearly, $\varphi \Rightarrow \varphi$ is an initial sequent (which is a good thing), while $\psi \Rightarrow \varphi$ is not derivable in general. We fill in the upper sequent:

$$\dfrac{\varphi \Rightarrow \varphi}{\varphi \wedge \psi \Rightarrow \varphi} \wedge \text{left}$$

We now have a correct **LK**-derivation of the sequent $\varphi \wedge \psi \Rightarrow \varphi$.

Example 7.7. Give an **LK**-derivation for the sequent $\neg\varphi \vee \psi \Rightarrow \varphi \to \psi$.

Begin by writing the desired end-sequent at the bottom of the derivation.

$$\dfrac{}{\neg\varphi \vee \psi \Rightarrow \varphi \to \psi}$$

To find a logical rule that could give us this end-sequent, we look at the logical connectives in the end-sequent: $\neg$, $\vee$, and $\to$. We only care at the moment about $\vee$ and $\to$ because they are *main connectives* of **Undefined token sentnce** in the end-sequent, while $\neg$ is inside the scope of another connective, so we will take care of it later. Our options for logical rules for the terminal inference are therefore the $\vee$ left rule and the $\to$ right rule. We could pick either rule, really, but let's pick the $\to$ right rule (if for no reason other than it allows us to put off splitting into two branches). According to the schema of the rule to the lower sequent, this must look like:

$$\dfrac{\dfrac{}{\varphi, \neg\varphi \vee \psi \Rightarrow \psi}}{\neg\varphi \vee \psi \Rightarrow \varphi \to \psi} \to \text{right}$$

Now we can apply the $\vee$ left rule. According to the schema, this must split into two upper sequents as follows:

$$\dfrac{\dfrac{\dfrac{}{\varphi, \neg\varphi \Rightarrow \psi} \quad \dfrac{}{\varphi, \psi \Rightarrow \psi}}{\varphi, \neg\varphi \vee \psi \Rightarrow \psi} \vee \text{left}}{\neg\varphi \vee \psi \Rightarrow \varphi \to \psi} \to \text{right}$$

Remember that we are trying to wind our way up to initial sequents; we seem to be pretty close!The right branch is just one weakening away from an initial sequent and then it is done:

$$\dfrac{\dfrac{\dfrac{}{\varphi, \neg\varphi \Rightarrow \psi} \quad \dfrac{\psi \Rightarrow \psi}{\varphi, \psi \Rightarrow \psi}}{\varphi, \neg\varphi \vee \psi \Rightarrow \psi} \vee \text{left}}{\neg\varphi \vee \psi \Rightarrow \varphi \to \psi} \to \text{right}$$

Now looking at the left branch, the only logical connective in any **Undefined token sentence** is the $\neg$ symbol in the antecedent **Undefined token sentence**, so we're looking at an instance of the $\neg$ left rule.

$$\dfrac{\dfrac{\dfrac{}{\varphi \Rightarrow \psi, \varphi}}{\varphi, \neg\varphi \Rightarrow \psi}\neg\text{ left} \quad \dfrac{\dfrac{}{\psi \Rightarrow \psi}}{\varphi, \psi \Rightarrow \psi}}{\dfrac{\varphi, \neg\varphi \vee \psi \Rightarrow \psi}{\neg\varphi \vee \psi \Rightarrow \varphi \to \psi}\to\text{ right}}\vee\text{ left}$$

Similarly to how we finished off the right branch, we are just one weakening away from finishing off this left branch as well.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\varphi \Rightarrow \varphi}}{\varphi \Rightarrow \psi, \varphi}}{\varphi, \neg\varphi \Rightarrow \psi}\neg\text{ left} \quad \dfrac{\dfrac{}{\psi \Rightarrow \psi}}{\varphi, \psi \Rightarrow \psi}}{\dfrac{\varphi, \neg\varphi \vee \psi \Rightarrow \psi}{\neg\varphi \vee \psi \Rightarrow \varphi \to \psi}\to\text{ right}}\vee\text{ left}$$

Example 7.8. Give an **LK**-derivation of the sequent $\neg\varphi \vee \neg\psi \Rightarrow \neg(\varphi \wedge \psi)$

Using the techniques from above, we start by writing the desired end-sequent at the bottom.

$$\dfrac{}{\neg\varphi \vee \neg\psi \Rightarrow \neg(\varphi \wedge \psi)}$$

The available main connectives of **Undefined token sentence** in the end-sequent are the $\vee$ symbol and the $\neg$ symbol. It would work to apply either the $\vee$ left or the $\neg$ right rule here, but we start with the $\neg$ right rule because it avoids splitting up into two branches for a moment:

$$\dfrac{\dfrac{}{\varphi \wedge \psi, \neg\varphi \vee \neg\psi \Rightarrow}}{\neg\varphi \vee \neg\psi \Rightarrow \neg(\varphi \wedge \psi)}\neg\text{ right}$$

Now we have a choice of whether to look at the $\wedge$ left or the $\vee$ left rule. Let's see what happens when we apply the $\wedge$ left rule: we have a choice to start with either the sequent $\varphi, \neg\varphi \vee \psi \Rightarrow$ or the sequent $\psi, \neg\varphi \vee \psi \Rightarrow$. Since the proof is symmetric with regards to φ and ψ, let's go with the former:

$$\dfrac{\dfrac{\dfrac{}{\varphi, \neg\varphi \vee \neg\psi \Rightarrow}}{\varphi \wedge \psi, \neg\varphi \vee \neg\psi \Rightarrow}\wedge\text{ left}}{\neg\varphi \vee \neg\psi \Rightarrow \neg(\varphi \wedge \psi)}\neg\text{ right}$$

Continuing to fill in the derivation, we see that we run into a problem:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\varphi \Rightarrow \varphi}}{\varphi, \neg\varphi \Rightarrow}\neg\text{ left} \quad \dfrac{\dfrac{}{\varphi \Rightarrow \psi}?}{\varphi, \neg\psi \Rightarrow}\neg\text{ left}}{\varphi, \neg\varphi \vee \neg\psi \Rightarrow}\vee\text{ left}}{\varphi \wedge \psi, \neg\varphi \vee \neg\psi \Rightarrow}\wedge\text{ left}}{\neg\varphi \vee \neg\psi \Rightarrow \neg(\varphi \wedge \psi)}\neg\text{ right}$$

The top of the right branch cannot be reduced any further, and it cannot be brought by way of structural inferences to an initial sequent, so this is not the right path to take. So clearly, it was a mistake to apply the $\wedge$ left rule above. Going back to what we had before and carrying out the $\vee$ left rule instead, we get

$$\dfrac{\dfrac{\dfrac{}{\varphi \wedge \psi, \neg\varphi \Rightarrow} \quad \dfrac{}{\varphi \wedge \psi, \neg\psi \Rightarrow}}{\varphi \wedge \psi, \neg\varphi \vee \neg\psi \Rightarrow} \vee \text{ left}}{\neg\varphi \vee \neg\psi \Rightarrow \neg(\varphi \wedge \psi)} \neg \text{ right}$$

Completing each branch as we've done before, we get

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\varphi \Rightarrow \varphi}}{\varphi \wedge \psi \Rightarrow \varphi} \wedge \text{ left}}{\varphi \wedge \psi, \neg\varphi \Rightarrow} \neg \text{ left} \quad \dfrac{\dfrac{\dfrac{}{\psi \Rightarrow \psi}}{\varphi \wedge \psi \Rightarrow \psi} \wedge \text{ left}}{\varphi \wedge \psi, \neg\psi \Rightarrow} \neg \text{ left}}{\varphi \wedge \psi, \neg\varphi \vee \neg\psi \Rightarrow} \vee \text{ left}}{\neg\varphi \vee \neg\psi \Rightarrow \neg(\varphi \wedge \psi)} \neg \text{ right}$$

(We could have carried out the $\wedge$ rules lower than the $\neg$ rules in these steps and still obtained a correct derivation).

Example 7.9. Give an **LK**-derivation of the sequent $\exists x\, \neg\varphi(x) \Rightarrow \neg\forall x\, \varphi(x)$.

When we are 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 lower down in the finished proof). Also, it is a good idea to try and look ahead and try to guess what the initial sequent might look like. In our case, it will have to be something like $\varphi(a) \Rightarrow \varphi(a)$. That means that when we are "reversing" the quantifier rules, we will have to pick the same term - what we will call a - for both the $\forall$ and the $\exists$ rule. If we picked different terms for each rule, we would end up with something like $\varphi(a) \Rightarrow \varphi(b)$, which, of course, is not derivable.

Starting as usual, we write

$$\dfrac{}{\exists x\, \neg\varphi(x) \Rightarrow \neg\forall x\, \varphi(x)}$$

We could either carry out the $\exists$ left rule or the $\neg$ right rule. Since the $\exists$ left rule is subject to the eigenvariable condition, it's a good idea to take care of it sooner rather than later, so we'll do that one first.

$$\dfrac{\dfrac{}{\neg\varphi(a) \Rightarrow \neg\forall x\, \varphi(x)}}{\exists x\, \neg\beta(x) \Rightarrow \neg\forall x\, \varphi(x)} \exists \text{ left}$$

Applying the $\neg$ left and right rules to eliminate the $\neg$ signs, we get

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\forall x\, \varphi(x) \Rightarrow \varphi(a)}}{\Rightarrow \neg\forall x\, \varphi(x), \varphi(a)}\neg\text{ right}}{\neg\varphi(a) \Rightarrow \neg\forall x \varphi(x)}\neg\text{ left}}{\exists x \neg\varphi(x) \Rightarrow \neg\forall x \varphi(x)}\exists\text{ left}$$

At this point, our only option is to carry out the $\forall$ left rule. Since this rule is not subject to the eigenvariable restriction, we're in the clear. Remember, we want to try and obtain an initial sequent (of the form $\varphi(a) \Rightarrow \varphi(a)$, so we should choose a as our argument for F when we apply the rule.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\varphi(a) \Rightarrow \varphi(a)}{\forall x\, \varphi(x) \Rightarrow \varphi(a)}\forall\text{ left}}{\Rightarrow \neg\forall x\, \varphi(x), \varphi(a)}\neg\text{ right}}{\neg\varphi(a) \Rightarrow \neg\forall x\, \varphi(x)}\neg\text{ left}}{\exists x\, \neg\varphi(x) \Rightarrow \neg\forall x\, \varphi(x)}\exists\text{ left}$$

It is important, especially when dealing with quantifiers, to double check at this point that the eigenvariable condition has not been violated. Since the only rule we applied that is subject to the eigenvariable condition was $\exists$ left, and the eigenvariable a does not occur in its lower sequent (the end-sequent), this is a correct derivation.

7.3 Proof-Theoretic Notions

Just as we've defined a number of important semantic notions (validity, entailment, satisfiabilty), we now define corresponding *proof-theoretic notions*. These are not defined by appeal to satisfaction of sentences in structures, but by appeal to the provability or unprovability of certain sequents. It was an important discovery, due to Gödel, that these notions coincide. That they do is the content of the *completeness theorem*.

Definition 7.10 (Theorems)**.** A sentence φ is a *theorem* if there is a proof in LK of the sequent $\Rightarrow \varphi$. We write $\vdash \varphi$ if φ is a theorem and $\nvdash \varphi$ if it is not.

Definition 7.11 (Provability)**.** A sentence φ is *provable from* a set of sentences Γ, $\Gamma \vdash \varphi$, if there is a finite subset $\Gamma_0 \subseteq \Gamma$ such that LK derives $\Gamma_0 \Rightarrow \varphi$. If φ is not provable from Γ we write $\Gamma \nvdash \varphi$.

Definition 7.12 (Consistency)**.** A set of sentences Γ is *consistent* iff $\Gamma \nvdash \bot$. If Γ is not consistent we say it is *inconsistent*.

Proposition 7.13. *$\Gamma \vdash \varphi$ iff $\Gamma \cup \{\neg A\}$ is inconsistent.*

Proposition 7.14. *Γ is inconsistent iff $\Gamma \vdash \varphi$ for every sentence φ.*

7.4 Properties of Provability

Proposition 7.15. *1. If $\Gamma \vdash_{\mathbf{LK}} \varphi$ and $\Gamma \cup \{\neg\varphi\} \vdash_{\mathbf{LK}} \bot$, then Γ is inconsistent.*

2. If $\Gamma \cup \{\varphi\} \vdash_{\mathbf{LK}} \bot$, then $\Gamma \vdash_{\mathbf{LK}} \neg\varphi$.

3. If $\Gamma \cup \{\varphi\} \vdash_{\mathbf{LK}} \bot$ and $\Gamma \cup \{\neg\varphi\} \vdash_{\mathbf{LK}} \bot$, then $\Gamma \vdash_{\mathbf{LK}} \bot$.

4. If $\Gamma \cup \{\varphi\} \vdash \bot$ and $\Gamma \cup \{\psi\} \vdash \bot$, then $\Gamma \cup \{\varphi \vee \psi\} \vdash \bot$.

5. If $\Gamma \vdash_{\mathbf{LK}} \varphi$ or $\Gamma \vdash_{\mathbf{LK}} \psi$, then $\Gamma \vdash_{\mathbf{LK}} \varphi \vee \psi$.

Proof. 1. Let the **LK**-derivation of $\Gamma_0 \Rightarrow \varphi$ be Π_0 and the **LK**-derivation of $\Gamma_1 \cup \{\varphi\} \Rightarrow \bot$ be Π_1. We can then derive

$$\dfrac{\dfrac{\begin{array}{c}\vdots\,\Pi_0\\ \Gamma_0 \Rightarrow \varphi\end{array}}{\Gamma_0, \Gamma_1 \Rightarrow \varphi} \qquad \dfrac{\begin{array}{c}\vdots\,\Pi_1\\ \Gamma_1, \varphi \Rightarrow \bot\end{array}}{\Gamma_0, \Gamma_1, \varphi \Rightarrow \bot}}{\Gamma_0, \Gamma_1 \Rightarrow \bot}\ \text{cut}$$

Since $\Gamma_0 \subseteq \Gamma$ and $\Gamma_1 \subseteq \Gamma$, $\Gamma_0 \cup \Gamma_1 \subseteq \Gamma$, hence $\Gamma \vdash_{\mathbf{LK}} \bot$.

2. Suppose that $\Gamma \cup \{\varphi\} \vdash_{\mathbf{LK}} \bot$. Then there is a finite set $\Gamma_0 \subseteq \Gamma$ with $\vdash_{\mathbf{LK}} \Gamma_0 \cup \{\varphi\} \Rightarrow \bot$. Let Π_0 be an **LK**-derivation of $\Gamma_0 \cup \{\varphi\} \Rightarrow \bot$, and consider

$$\dfrac{\begin{array}{c}\vdots\,\Pi_0\\ \Gamma_0 \cup \{\varphi\} \Rightarrow \bot\end{array}}{\Gamma_0 \Rightarrow \neg\varphi}\ \neg\,\text{right}$$

3. There are finite sets $\Gamma_0 \subseteq \Gamma$ and $\Gamma_1 \subseteq \Gamma$ and **LK**-derivations Π_0 and Π_1 of $\Gamma_0, \varphi \Rightarrow \bot$ and $\Gamma_1, \neg\varphi \Rightarrow \bot$, respectively. We can then derive

$$\dfrac{\dfrac{\begin{array}{c}\vdots\,\Pi_0\\ \Gamma_0, \varphi \Rightarrow \bot\end{array}}{\Gamma_0 \Rightarrow \neg\varphi}\ \neg\,\text{right} \qquad \dfrac{\begin{array}{c}\vdots\,\Pi_1\\ \Gamma_1, \neg\varphi \Rightarrow \bot\end{array}}{\Gamma_0, \Gamma_1 \Rightarrow \bot}\ \text{cut}$$

Since $\Gamma_0 \subseteq \Gamma$ and $\Gamma_1 \subseteq \Gamma$, $\Gamma_0 \cup \Gamma_1 \subseteq \Gamma$. Hence $\Gamma \vdash_{\mathbf{LK}} \bot$.

4. There are finite sets $\Gamma_0, \Gamma_1 \subseteq \Gamma$ and **LK**-derivations Π_0 and Π_1 such that

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \Pi_0\end{array}}{\Gamma_0 \varphi \Rightarrow \bot}\qquad \dfrac{\begin{array}{c}\vdots\\ \Pi_1\end{array}}{\Gamma_1, \psi \Rightarrow \bot}}{\Gamma_0, \Gamma_1, \varphi \Rightarrow \bot}\qquad \dfrac{}{\Gamma_0, \Gamma_1, \psi \Rightarrow \bot}}{\Gamma_0, \Gamma_1, \varphi \vee \psi \Rightarrow \bot} \vee \text{left}$$

Since $\Gamma_0, \Gamma_1 \subseteq \Gamma$ and $\Gamma \cup \{\varphi \vee \psi\} \vdash \bot$.

5. There is an **LK**-derivation Π_0 and a finite set $\Gamma_0 \subseteq \Gamma$ such that we can derive

$$\dfrac{\begin{array}{c}\vdots\\ \Pi_0\end{array}}{\dfrac{\Gamma_0 \Rightarrow \varphi}{\Gamma_0 \Rightarrow \varphi \vee \psi}} \vee \text{right}$$

Therefore $\Gamma \vdash \varphi \vee \psi$. The proof for when $\Gamma \vdash_{\mathbf{LK}} \psi$ is similar.

□

Problem 7.1. Complete the proof of Proposition 7.15.

Proposition 7.16 (Monotony)**.** *If $\Gamma \subseteq \Delta$ and $\Gamma \vdash \varphi$, then $\Delta \vdash \varphi$.*

Proof. Any finite $\Gamma_0 \subseteq \Gamma$ is also a finite subset of Δ, so derivation of $\Gamma_0 \Rightarrow \varphi$ also shows $\Delta \vdash \varphi$. □

Theorem 7.17. *If c is a constant not occurring in Γ or $\varphi(x)$ and $\Gamma \vdash \varphi(c)$, then $\Gamma \vdash \forall x\, \varphi(c)$.*

Proof. Let Π_0 be an **LK**-derivation of $\Gamma_0 \Rightarrow \varphi(c)$ for some finite $\Gamma_0 \subseteq \Gamma$. By adding a $\forall$ right inference, we obtain a proof of $\Gamma \Rightarrow \forall x\, \varphi(x)$, since c does not occur in Γ or $\varphi(x)$ and thus the eigenvariable condition is satisfied. □

Theorem 7.18. *If $\Gamma \vdash \varphi(t)$ then $\Gamma \vdash \exists x\, \varphi(x)$.*

Proof. Suppose $\Gamma \vdash \varphi(t)$. Then for some finite $\Gamma_0 \subseteq \Gamma$, **LK** derives $\Gamma_0 \Rightarrow \varphi(t)$. Add an $\exists$ right inference to get a derivation of $\Gamma_0 \Rightarrow \exists x\, \varphi(x)$.

□

7.5 Soundness

A proof system, such as the sequent calculus, is *sound* if it cannot prove things that do not actually hold. Soundness is thus a kind of guaranteed safety property for proof systems. Depending on which proof theoretic property is in question, we would like to know for instance, that

1. every provable sentence is valid;
2. if a sentence is provable 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 proof sytem. If any of them do not hold, the proof system is deficient—it would proves too much. Consequently, establishing the soundness of a proof system is of the utmost importance.

Because all these proof-theoretic properties are defined via provability in the sequent calculus of certain sequents, proving (1)–(3) above requires proving something about the semantic properties of provable sequents. We will first define what it means for a sequent to be *valid*, and then show that every provable sequent is valid. (1)–(3) then follow as corollaries from this result.

Definition 7.19. A structure $\mathfrak{M}$ *satisifes* a sequent $\Gamma \Rightarrow \Delta$ is *valid* if either $\mathfrak{M} \not\models \alpha$ for some $\alpha \in \Gamma$ or $\mathfrak{M} \models \alpha$ for some $\alpha \in \Delta$.

A sequent is *valid* if every structure $\mathfrak{M}$ satisfies it.

Theorem 7.20 (Soundness)**.** *If LK derives $\Gamma \Rightarrow \Delta$, then $\Gamma \Rightarrow \Delta$ is valid.*

Proof. Let Π be a derivation of $\Gamma \Rightarrow \Delta$. We proceed by induction on the number of inferences in Π.

If the number of inferences is 0, then Π consists only of an initial sequent. Every initial sequent $\varphi \Rightarrow \varphi$ is obviously valid, since for every $\mathfrak{M}$, either $\mathfrak{M} \not\models \varphi$ or $\mathfrak{M} \models \varphi$.

If the number of inferences is greater than 0, we distinguish cases according to the type of the lowermost inference. By induction hypothesis, we can assume that the premises of that inference are valid.

First, we consider the possible inferences with only one premise $\Gamma' \Rightarrow \Delta'$.

1. The last inference is a weakening. Then $\Gamma' \subseteq \Gamma$ and $\Delta = \Delta'$ if it's a weakening on the left, or $\Gamma = \Gamma'$ and $\Delta' \subseteq \Delta$ if it's a weaking on the right. In either case, $\Delta' \subseteq \Delta$ and $\Gamma' \subseteq \Gamma$. If $\mathfrak{M} \not\models \alpha$ for some $\alpha \in \Gamma'$, then, since $\Gamma' \subseteq \Gamma$, $\alpha \in \Gamma$ as well, and so $\mathfrak{M} \not\models \alpha$ for the same $\alpha \in \Gamma$. Similarly, if $\mathfrak{M} \models \alpha$ for some $\alpha \in \Gamma'$, as $\alpha \in \Gamma$, $\mathfrak{M} \models \alpha$ for some $\alpha \in \Delta$. Since $\Gamma' \Rightarrow \Delta'$ is valid, one of these cases obtains for every $\mathfrak{M}$. Consequently, $\Gamma \Rightarrow \Delta$ is valid.

2. The last inference is $\neg$ left: Then for some $\varphi \in \Gamma'$, $\neg\varphi \in \Delta$. Also, $\Gamma' \subseteq \Gamma$, and $\Delta' \setminus \{\varphi\} \subseteq \Delta$.

   If $\mathfrak{M} \models \varphi$, then $\mathfrak{M} \not\models \neg\varphi$, and since $\neg\varphi \in \Gamma$, $\mathfrak{M}$ satisifes $\Gamma \Rightarrow \Delta$. Since $\Gamma' \Rightarrow \Delta'$ is valid, if $\mathfrak{M} \not\models \varphi$, then either $\mathfrak{M} \not\models \alpha$ for some $\alpha \in \Gamma'$ or $\mathfrak{M} \models \alpha$ for some $\alpha \in \Delta'$ different from φ. Consequently, $\mathfrak{M} \not\models \alpha$ for some $\alpha \in \Gamma$ (since $\Gamma' \subseteq \Gamma$) or $\mathfrak{M} \models \alpha$ for some $\alpha \in \Delta'$ different from φ (since $\Delta' \setminus \{\varphi\} \subseteq \Delta$).

3. The last inference is $\neg$ right: Exercise.

4. The last inference is $\wedge$ left: There are two variants: $\varphi \wedge \psi$ may be inferred on the left from φ or from ψ on the left side of the premise. In the first case, $\varphi \in \Gamma'$. Consider a structure $\mathfrak{M}$. Since $\Gamma' \Rightarrow \Delta'$ is valid, (a) $\mathfrak{M} \not\models \varphi$, (b) $\mathfrak{M} \not\models \alpha$ for some $\alpha \in \Gamma' \setminus \{\varphi\}$, or (c) $\mathfrak{M} \models \alpha$ for some $\alpha \in \Delta'$. In case (a), $\mathfrak{M} \not\models \varphi \wedge \psi$. In case (b), there is an $\alpha \in \Gamma \setminus \{\varphi \wedge \psi\}$ such that $\mathfrak{M} \not\models \alpha$, since $\Gamma' \setminus \{\varphi\} \subseteq \Gamma \setminus \{\varphi \wedge \psi\}$. In case (c), there is a $\alpha \in \Delta$ such that $\mathfrak{M} \models \alpha$, as $\Delta = \Delta'$. So in each case, $\mathfrak{M}$ satisfies $\varphi \wedge \psi, \Gamma \Rightarrow \Delta$. Since $\mathfrak{M}$ was arbitrary, $\Gamma \Rightarrow \Delta$ is valid. The case where $\varphi \wedge \psi$ is inferred from ψ is handled the same, changing φ to ψ.

5. The last inference is $\vee$ right: There are two variants: $\varphi \vee \psi$ may be inferred on the right from φ or from ψ on the right side of the premise. In the first case, $\varphi \in \Delta'$. Consider a structure $\mathfrak{M}$. Since $\Gamma' \Rightarrow \Delta'$ is valid, (a) $\mathfrak{M} \models \varphi$, (b) $\mathfrak{M} \not\models \alpha$ for some $\alpha \in \Gamma'$, or (c) $\mathfrak{M} \models \alpha$ for some $\alpha \in \Delta' \setminus \{\varphi\}$. In case (a), $\mathfrak{M} \models \varphi \vee \psi$. In case (b), there is $\alpha \in \Delta$ such that $\mathfrak{M} \not\models \alpha$, as $\Gamma = \Gamma'$. In case (c), there is an $\alpha \in \Delta$ such that $\mathfrak{M} \models \alpha$, since $\Delta' \setminus \{\varphi\} \subseteq \Delta$. So in each case, $\mathfrak{M}$ satisfies $\varphi \wedge \psi, \Gamma \Rightarrow \Delta$. Since $\mathfrak{M}$ was arbitrary, $\Gamma \Rightarrow \Delta$ is valid. The case where $\varphi \vee \psi$ is inferred from ψ is handled the same, changing φ to ψ.

6. The last inference is $\rightarrow$ right: Then $\varphi \in \Gamma'$, $\psi \in \Delta'$, $\Gamma' \setminus \{\varphi\} \subseteq \Gamma$ and $\Delta' \setminus \{\psi\} \subseteq \Delta$. Since $\Gamma' \Rightarrow \Delta'$ is valid, for any structure $\mathfrak{M}$, (a) $\mathfrak{M} \not\models \varphi$, (b) $\mathfrak{M} \models \psi$, (c) $\mathfrak{M} \not\models \alpha$ for some $\alpha \in \Gamma' \setminus \{\varphi\}$, or $\mathfrak{M} \models \alpha$ for some $\alpha \in \Delta' \setminus \{\psi\}$. In cases (a) and (b), $\mathfrak{M} \models \varphi \rightarrow \psi$. In case (c), for some $\alpha \in \Gamma$, $\mathfrak{M} \not\models \alpha$. In case (d), for some $\alpha \in \Delta$, $\mathfrak{M} \models \alpha$. In each case, $\mathfrak{M}$ satisfies $\Gamma \Rightarrow \Delta$. Since $\mathfrak{M}$ was arbitrary, $\Gamma \Rightarrow \Delta$ is valid.

7. The last inference is $\forall$ left: Then there is a formula $\varphi(x)$ and a ground term t such that $\varphi(t) \in \Gamma'$, $\forall x\, \varphi(x) \in \Gamma$, and $\Gamma' \setminus \{\varphi(t)\} \subseteq \Gamma$. Consider a structure $\mathfrak{M}$. Since $\Gamma' \Rightarrow \Delta'$ is valid, (a) $\mathfrak{M} \not\models \varphi(t)$, (b) $\mathfrak{M} \not\models \alpha$ for some $\alpha \in \Gamma' \setminus \{\varphi(t)\}$, or (c) $\mathfrak{M} \models \alpha$ for some $\alpha \in \Delta'$. In case (a), $\mathfrak{M} \not\models \forall x\, \varphi(x)$. In case (b), there is an $\alpha \in \Gamma \setminus \{\varphi(t)\}$ such that $\mathfrak{M} \not\models \alpha$. In case (c), there is a $\alpha \in \Delta$ such that $\mathfrak{M} \models \alpha$, as $\Delta = \Delta'$. So in each case, $\mathfrak{M}$ satisfies $\Gamma \Rightarrow \Delta$. Since $\mathfrak{M}$ was arbitrary, $\Gamma \Rightarrow \Delta$ is valid.

8. The last inference is $\exists$ right: Exercise.

9. The last inference is $\forall$ right: Then there is a formula $\varphi(x)$ and a constant symbol a such that $\varphi(a) \in \Delta'$, $\forall x\, \varphi(x) \in \Delta$, and $\Delta' \setminus \{\varphi(a)\} \subseteq \Delta$. Furthermore, $a \notin \Gamma \cup \Delta$. Consider a structure $\mathfrak{M}$. Since $\Gamma' \Rightarrow \Delta'$ is valid, (a) $\mathfrak{M} \models \varphi(a)$, (b) $\mathfrak{M} \not\models \alpha$ for some $\alpha \in \Gamma'$, or (c) $\mathfrak{M} \models \alpha$ for some $\alpha \in \Delta' \setminus \{\varphi(a)\}$.

   First, suppose (a) is the case but neither (b) nor (c), i.e., $\mathfrak{M} \models \alpha$ for all $\alpha \in \Gamma'$ and $\mathfrak{M} \not\models \alpha$ for all $\alpha \in \Delta' \setminus \{\varphi(a)\}$. In other words, assume $\mathfrak{M} \models \varphi(a)$ and that $\mathfrak{M}$ does not satisfy $\Gamma' \Rightarrow \Delta' \setminus \{\varphi(a)\}$. Since $a \notin \Gamma \cup \Delta$, also $a \notin \Gamma' \cup (\Delta' \setminus \{\varphi(a)\})$. Thus, if $\mathfrak{M}'$ is like $\mathfrak{M}$ except that $a^{\mathfrak{M}} \neq a^{\mathfrak{M}'}$, $\mathfrak{M}'$ also does not satisfy $\Gamma' \Rightarrow \Delta' \setminus \{\varphi(a)\}$ by extensionality. But since $\Gamma' \Rightarrow \Delta'$ is valid, we must have $\mathfrak{M}' \models \varphi(a)$.

   We now show that $\mathfrak{M} \models \forall x\, \varphi(x)$. To do this, we have to show that for every variable assignment s, $\mathfrak{M}, s \models \forall x\, \varphi(x)$. This in turn means that for every x-variant s' of s, we must have $\mathfrak{M}, s' \models \varphi(x)$. So consider any variable assignment s and let s' be an x-variant of s. Since Γ' and Δ' consist entirely of sentences, $\mathfrak{M}, s \models \alpha$ iff iff $\mathfrak{M}, s' \models \alpha$ iff $\mathfrak{M} \models \alpha$ for all $\alpha \in \Gamma' \cup \Delta'$. Let $\mathfrak{M}'$ be like $\mathfrak{M}$ except that $a^{\mathfrak{M}'} = s'(x)$. Then $\mathfrak{M}, s' \models \varphi(x)$ iff $\mathfrak{M}' \models \varphi(a)$ (as $\varphi(x)$ does not contain a). Since we've already established that $\mathfrak{M}' \models \varphi(a)$ for all $\mathfrak{M}'$ which differ from $\mathfrak{M}$ at most in what they assign to a, this means that $\mathfrak{M}, s' \models \varphi(x)$. Thus weve shown that $\mathfrak{M}, s \models \forall x\, \varphi(x)$. Since s is an arbitrary variable assignment and $\forall x\, \varphi(x)$ is a sentence, then $\mathfrak{M} \models \forall x\, \varphi(x)$.

   If (b) is the case, there is a $\alpha \in \Gamma$ such that $\mathfrak{M} \not\models \alpha$, as $\Gamma = \Gamma'$. If (c) is the case, there is an $\alpha \in \Delta' \setminus \{\varphi(a)\}$ such that $\mathfrak{M} \models \alpha$. So in each case, $\mathfrak{M}$ satisfies $\Gamma \Rightarrow \Delta$. Since $\mathfrak{M}$ was arbitrary, $\Gamma \Rightarrow \Delta$ is valid.

10. The last inference is $\exists$ left: Exercise.

Now let's consider the possible inferences with two premises: cut, $\vee$ left, $\wedge$ right, and $\rightarrow$ left.

1. The last inference is a cut: Suppose the premises are $\Gamma' \Rightarrow \Delta'$ and $\Pi' \Rightarrow \Lambda'$ and the cut formula φ is in both Δ' and Π'. Since each is valid, every structure $\mathfrak{M}$ satisfies both premises. We distinguish two cases: (a) $\mathfrak{M} \not\models \varphi$ and (b) $\mathfrak{M} \models \varphi$. In case (a), in order for $\mathfrak{M}$ to satisfy the left premise, it must satisfy $\Gamma' \Rightarrow \Delta' \setminus \{\varphi\}$. But $\Gamma' \subseteq \Gamma$ and $\Delta' \setminus \{\varphi\} \subseteq \Delta$, so $\mathfrak{M}$ also satisfies $\Gamma \Rightarrow \Delta$. In case (b), in order for $\mathfrak{M}$ to satisfy the right premise, it must satisfy $\Pi' \setminus \{\varphi\} \Rightarrow \Lambda'$. But $\Pi' \setminus \{\varphi\} \subseteq \Gamma$ and $\Lambda' \subseteq \Delta$, so $\mathfrak{M}$ also satisfies $\Gamma \Rightarrow \Delta$.

2. The last inference is $\wedge$ right. The premises are $\Gamma \Rightarrow \Delta'$ and $\Gamma \Rightarrow \Delta''$, where $\varphi \in \Delta'$ an $\psi \in \Delta''$. By induction hypothesis, both are valid. Consider a **Undefined token strcuture** $\mathfrak{M}$. We have two cases: (a) $\mathfrak{M} \not\models \varphi \wedge \psi$ or (b) $\mathfrak{M} \models \varphi \wedge \psi$. In case (a), either $\mathfrak{M} \not\models \varphi$ or $\mathfrak{M} \not\models \psi$. In the

former case, in order for $\mathfrak{M}$ to satisfy $\Gamma \Rightarrow \Delta'$, it must already satisfy $\Gamma \Rightarrow \Delta' \setminus \{\varphi\}$. In the latter case, it must satisfy $\Gamma \Rightarrow \Delta'' \setminus \{\psi\}$. But since both $\Delta' \setminus \{\varphi\} \subseteq \Delta$ and $\Delta'' \setminus \{\psi\} \subseteq \Delta$, that means $\mathfrak{M}$ satisfies $\Gamma \Rightarrow \Delta$. In case (b), $\mathfrak{M}$ satisfies $\Gamma \Rightarrow \Delta$ since $\varphi \wedge \psi \in \Delta$.

3. The last inference is $\vee$ left: Exercise.

4. The last inference is $\rightarrow$ right. The premises are $\Gamma \Rightarrow \Delta'$ and $\Gamma' \Rightarrow \Delta$, where $\varphi \in \Delta'$ an $\psi \in \Gamma'$. By induction hypothesis, both are valid. Consider a **Undefined token strcuture** $\mathfrak{M}$. We have two cases: (a) $\mathfrak{M} \not\models \varphi \rightarrow \psi$ or (b) $\mathfrak{M} \models \varphi \rightarrow \psi$. In case (a), either $\mathfrak{M} \not\models \varphi$ or $\mathfrak{M} \models \psi$. In the former case, in order for $\mathfrak{M}$ to satisfy $\Gamma \Rightarrow \Delta'$, it must already satisfy $\Gamma \Rightarrow \Delta' \setminus \{\varphi\}$. In the latter case, it must satisfy $\Gamma' \setminus \{\psi\} \Rightarrow \Delta$. But since both $\Delta' \setminus \{\varphi\} \subseteq \Delta$ and $\Gamma' \setminus \{\psi\} \subseteq \Gamma$, that means $\mathfrak{M}$ satisfies $\Gamma \Rightarrow \Delta$. In case (b), $\mathfrak{M}$ satisfies $\Gamma \Rightarrow \Delta$ since $\varphi \rightarrow \psi \in \Gamma$.

□

Problem 7.2. Complete the proof of Theorem 7.20.

Corollary 7.21. *If $\vdash \varphi$ then φ is valid.*

Corollary 7.22. *If $\Gamma \vdash \varphi$ then $\Gamma \models \varphi$.*

Proof. If $\Gamma \vdash \varphi$ then for some finite subset $\Gamma_0 \subseteq \Gamma$, there is a derivation of $\Gamma_0 \Rightarrow \varphi$. By Theorem 7.20, every structure $\mathfrak{M}$ either makes some $\psi \in \Gamma_0$ false or makes φ true. Hence, if $\mathfrak{M} \models \Gamma_0$ then also $\mathfrak{M} \models \varphi$. □

Corollary 7.23. *If Γ is satisfiable, then it is consistent.*

Proof. We prove the contrapositive. Suppose that Γ is not consistent. Then $\Gamma \vdash \bot$, i.e., there is a finite $\Gamma_0 \subseteq \Gamma$ and a proof of $\Gamma_0 \Rightarrow \bot$. By Theorem 7.20, $\Gamma_0 \Rightarrow \bot$ is valid. Since $\mathfrak{M} \not\models \bot$ for every structure $\mathfrak{M}$, for $\mathfrak{M}$ to satisfy $\Gamma_0 \Rightarrow \bot$ there must be an $\alpha \in \Gamma_0$ so that $\mathfrak{M} \not\models \alpha$, and since $\Gamma_0 \subseteq \Gamma$, that α is also in Γ. In other words, no $\mathfrak{M}$ satisfies Γ, i.e., Γ is not satisfiable. □

Chapter 8

The Completeness Theorem

8.1 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 somethign fundamental about the relationship between semantic consequence and our proof system: if a sentences φ follows from some sentences Γ, then there is also a proof of φ from Γ. Thus, the proof 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.

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 proof of φ from Γ 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 a consequence of already a finite subset. This is called *compactness*. It also follows from the proof of the completeness theorem that any satisfiable set of sentences has a finite or denumerable model. This result is called the Löwenheim-Skolem theorem.

8.2 Maximally Consistent Sets of Sentences

Definition 8.1. A set Γ of sentences is *maximally consistent* iff

1. Γ is consistent, and
2. if $\Gamma \subsetneq \Gamma'$, then Γ' is inconsistent.

An alternate definition equivalent to the above is: a set Γ of sentences is *maximally consistent* iff

1. Γ is consistent, and

2. If $\Gamma \cup \{\varphi\}$ is consistent, then $\varphi \in \Gamma$.

Maximally consistent sets are important in the completeness proof since we can guarantee that every consistent set of sentences Γ is contained in a maximally consistent set Γ^*, and a maximally consistent set contains, for each sentence φ, either φ or its negation $\neg\varphi$. This is true in particular for atomic sentences, so from a maximally consistent set in a language suitably expanded by constant symbols, we can construct a structure where the interpretation of **Undefined token predicates** 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 in Γ) true. The proof of this latter fact requires that $\neg\varphi \in \Gamma^*$ iff $\varphi \notin \Gamma^*$, $\varphi \vee \psi \in \Gamma^*$ iff $\varphi \in \Gamma^*$ or $\psi \in \Gamma^*$, etc.

Proposition 8.2. *Suppose Γ is maximally consistent. Then:*

1. *If $\Gamma \vdash \varphi$, then $\varphi \in \Gamma$.*
2. *For any φ, either $\varphi \in \Gamma$ or $\neg\varphi \in \Gamma$.*
3. *$\varphi \vee \psi \in \Gamma$ iff either $\varphi \in \Gamma$ or $\psi \in \Gamma$.*

Proof. Let us suppose for all of the following that Γ is maximally consistent.

1. If $\Gamma \vdash \varphi$, then $\varphi \in \Gamma$.

   Suppose that $\Gamma \vdash \varphi$. Suppose to the contrary that $\varphi \notin \Gamma$: then since Γ is maximally consistent, $\Gamma \cup \{\varphi\}$ is inconsistent, hence $\Gamma \cup \{\varphi\} \vdash \bot$. By Proposition 7.15(1) 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. For any φ, either $\varphi \in \Gamma$ or $\neg\varphi \in \Gamma$.

   Suppose to the contrary that there is an φ such that $\varphi \notin \Gamma$ and $\neg\varphi \notin \Gamma$. Since Γ is maximally consistent, $\Gamma \cup \{\varphi\}$ and $\Gamma \cup \{\neg\varphi\}$ are both inconsistent, so $\Gamma \cup \{\varphi\} \vdash \bot$ and $\Gamma \cup \{\neg\varphi\} \vdash \bot$. By Proposition 7.15(3), Γ is inconsistent, a contradiction. Hence there cannot be such a φ and, for every φ, $\varphi \in \Gamma$ or $\neg\varphi \in \Gamma$.

3. $\varphi \vee \psi \in \Gamma$ iff either $\varphi \in \Gamma$ or $\psi \in \Gamma$.

   For the contrapositive of the forward direction, suppose that $\varphi \notin \Gamma$ and $\psi \notin \Gamma$. We want to show that $\varphi \vee \psi \notin \Gamma$. Since Γ is maximally consistent, $\Gamma \cup \{\varphi\} \vdash \bot$ and $\Gamma \cup \{\psi\} \vdash \bot$. By Proposition 7.15(4), $\Gamma \cup \{\varphi \vee \psi\}$ is inconsistent. Hence, $\varphi \vee \psi \notin \Gamma$, as required.

   For the reverse direction, suppose that $\varphi \in \Gamma$ or $\psi \in \Gamma$. Then $\Gamma \vdash \varphi$ or $\Gamma \vdash \psi$. By Proposition 7.15(5), $\Gamma \vdash \varphi \vee \psi$. By (1), $\varphi \vee \psi \in \Gamma$, as required.

□

8.3 Henkin Expansion

Part of the challenge in proving the completeness theorem is that the model we construct from a maximally consistent set Γ must make all the quantified formulas 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 constants and adding, for each formula with one free variable $\varphi(x)$ a formula of the form $\neg\forall x\, \varphi \to \neg\varphi(c)$, where c is one of the new constant symbols. When we construct the structure satisfying Γ, this will guarantee that each false universal sentence has a counterexample among the new constants.

Lemma 8.3. *If Γ is consistent in $\mathcal{L}$ and $\mathcal{L}'$ is obtained from $\mathcal{L}$ by adding countably many new constants $c_1, c_2, \ldots$, then Γ is consistent in $\mathcal{L}'$.*

Definition 8.4. A set Γ of formulas of a language $\mathcal{L}$ is *saturated* if and only if for each formula $\varphi \in \mathrm{Frm}(\mathcal{L})$ and variable x there is a constant c such that $\neg\forall x\, \varphi \to \neg\varphi(c) \in \Gamma$.

The following definition will be used in the proof of the next theorem.

Definition 8.5. Fix an enumeration $\langle\varphi_1, x_1\rangle, \langle\varphi_2, x_2\rangle, \ldots$ of all formula-variable pairs of $\mathcal{L}'$, where x_i is the only free variable in φ_i. We define the sentences θ_n by recursion on n. Assuming that $\theta_1, \ldots, \theta_n$ have been defined, denote by c_{n+1} the first new constant not occurring in $\theta_1, \ldots, \theta_n$, and let θ_{n+1} be the formula $\neg\forall x_{n+1}\, \varphi_{n+1}(x_{n+1}) \to \neg\varphi_{n+1}(c_{n+1})$. This includes the case where $n = 0$ and the list of previous θ's is empty, i.e., θ_1 is $\neg\forall x_1 \varphi_1 \to \neg\varphi_1(c_1)$.

Theorem 8.6. *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 countably many new constants to $\mathcal{L}'$. By the previous Lemma, Γ is still consistent in the richer language. Further, let θ be as in the previous definition: then $\Gamma \cup \theta$ is saturated by construction. Let

$$\begin{aligned}\Gamma_0 &= \Gamma\\ \Gamma_{n+1} &= \Gamma_n \cup \{\theta_{n+1}\}\end{aligned}$$

i.e., $\Gamma_n = \Gamma \cup \{\theta_1, \ldots, \theta_n\}$, and let $\Gamma' = \bigcup_n \Gamma_n$. 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-1} is consistent but $\Gamma_n = \Gamma_{n-1} \cup \{\theta_n\}$ is inconsistent. Recall that θ_n is $\neg\forall x_n\, \varphi_n(x_n) \to \neg\varphi_n(c_n)$. where $\varphi(x)$ is a formula of $\mathcal{L}'$ with only the variable x_n free and no containing any constants c_i where $i \geq n$.

If $\Gamma_{n-1} \cup \{\theta_n\}$ is inconsistent, then $\Gamma_{n-1} \vdash \neg\theta_n$, and hence both of the following hold:

$$\Gamma_{n-1} \vdash \neg\forall x_n\, \varphi_n(x_n) \qquad \Gamma_{n-1} \vdash \varphi_n(c_n)$$

But c_n does not occur in Γ_{n-1} and $\varphi_n(x_n)$ By Theorem 7.17, from $\Gamma \vdash \varphi_n(c_n)$, we obtain $\Gamma \vdash \forall x_n\, \varphi_n(x_n)$. Thus we have that both $\Gamma_{n-1} \vdash \neg\forall x_n\, \varphi_n(x_n)$ and $\Gamma_{n-1} \vdash \forall x_n\, \varphi_n(x_n)$, so Γ itself is inconsistent. Contradiction: Γ_{n-1} was supposed to be consistent. Hence $\Gamma_n \cup \{\theta_n\}$ is consistent. □

8.4 Lindenbaum's Lemma

Lemma 8.7 (Lindenbaum's Lemma)**.** *Every consistent set Γ can be extended to a maximally consistent saturated set Γ^*.*

Proof. Let Γ be consistent, and let Γ' be as in the previous definition: we already proved that $\Gamma \cup \Gamma'$ is a consistent saturated set in the richer language $\mathcal{L}'$ (with the countably many new constants). Let $\varphi_0, \varphi_1, \ldots$ be an enumeration of all the formulas of $\mathcal{L}'$. Define $\Gamma_0 = \Gamma \cup \Gamma'$, and

$$\Gamma_{n+1} = \begin{cases} \Gamma_n \cup \{\varphi_n\} & \text{if } \Gamma_n \cup \{\varphi_n\} \text{ is consistent;} \\ \Gamma_n \cup \{\neg\varphi_n\} & \text{otherwise.} \end{cases}$$

Let $\Gamma^* = \bigcup_{n\geq 0} \Gamma_n$. Since $\Gamma' \subseteq \Gamma^*$, for each formula φ, Γ^* contains a formula of the form $\exists x\, \varphi \to \varphi[c/x]$ and thus is saturated.

Each Γ_n is consistent: Γ_0 is consistent by definition. If $\Gamma_{n+1} = \Gamma_n \cup \{\varphi\}$, this is because the latter is consistent. If it isn't, $\Gamma_{n+1} = \Gamma_n \cup \{\neg\varphi\}$, which must be consistent. If it weren't, i.e., both $\Gamma_n \cup \{\varphi\}$ and $\Gamma_n \cup \{\neg\varphi\}$ are inconsistent, then $\Gamma_n \vdash \neg\varphi$ and $\Gamma_n \vdash \varphi$, so Γ_n would be inconsistent contrary to induction hypothesis.

Any formula 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 that means that Γ^* is maximally consistent. □

8.5 Construction of Model

We will begin by showing how to construct a model which satisfies a maximally consistent, saturated set of sentences in a language $\mathcal{L}$ without the identity symbol.

Definition 8.8. Let Γ^* be a maximally consistent, saturated set of sentences in a language $\mathcal{L}$. The *term model* $\mathfrak{M}(\Gamma^*)$ of Γ^* is the first-order 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 c is c itself: $c^{\mathfrak{M}(\Gamma^*)} = c$.
3. The function f is assigned the function

$$f^{\mathfrak{M}(\Gamma^*)}(t_1, \ldots, t_n) = f(\mathrm{Val}^{\mathfrak{M}(\Gamma^*)}(t_1), \ldots, \mathrm{Val}^{\mathfrak{M}(\Gamma^*)}(t_1))$$

4. If R is an n-place predicate symbol, then $\langle t_1, \ldots, t_n \rangle \in R^{\mathfrak{M}(\Gamma^*)}$ iff $R(t_1, \ldots, t_n) \in \Gamma^*$.

Lemma 8.9 (Truth Lemma). $\mathfrak{M}(\Gamma^*) \models \varphi$ *iff* $\varphi \in \Gamma^*$

Proof. For the forward direction, suppose $\mathfrak{M}(\Gamma^*) \models \varphi$. We want to show that $\varphi \in \Gamma^*$.

1. $\varphi \equiv R(t_1, \ldots, t_n)$: $\langle t_1, \ldots, t_n \rangle \in R^{\mathfrak{M}(\Gamma^*)}$ by the definition of satisfaction, and by the construction of $\mathfrak{M}(\Gamma^*)$, $R(t_1, \ldots, t_n) \in \Gamma^*$.

2. $\varphi \equiv \neg\psi$: since $\mathfrak{M}(\Gamma^*) \models \varphi$, $\mathfrak{M}(\Gamma^*) \not\models \psi$. By induction hypothesis, $\psi \notin \Gamma^*$. By Proposition 8.2(2), $\neg\psi \in \Gamma$.

3. $\varphi \equiv \psi \vee \chi$: since $\mathfrak{M}(\Gamma^*) \models \varphi$, we have at least one of $\mathfrak{M}(\Gamma^*) \models \psi$ or $\mathfrak{M}(\Gamma^*) \models \chi$. If $\mathfrak{M}(\Gamma^*) \models \psi$, then by the induction hypothesis, $\psi \in \Gamma^*$, so by Proposition 8.2(3), $\psi \vee \chi \in \Gamma^*$ (and similar if $\mathfrak{M}(\Gamma^*) \models \chi$).

4. $\varphi \equiv \exists x\, \psi(x)$: since $\mathfrak{M}(\Gamma^*) \models \varphi$, for some variable assignment s, $\mathfrak{M}(\Gamma^*), s \models \psi(x)$. The value $s(x)$ is some constant symbol $c \in |\mathfrak{M}(\Gamma^*)|$. Thus, $\mathfrak{M}(\Gamma^*) \models \psi(c)$, and by our induction hypothesis, $\psi(c) \in \Gamma^*$. By **??** we have $\Gamma^* \vdash \exists x\, \psi(x)$. Then, by Proposition 8.2(1), we can conclude that $\exists x\, \psi(x) \in \Gamma^*$.

For the reverse direction, let $\varphi \in \Gamma^*$. Suppose to the contrary that $\mathfrak{M}(\Gamma^*) \not\models \varphi$. Then $\mathfrak{M}(\Gamma^*) \models \neg\varphi$, so by the forward direction of the proof, $\neg\varphi \in \Gamma^*$. Contradiction: Γ^* is consistent. Hence, we must have that $\mathfrak{M}(\Gamma^*) \models \varphi$. □

8.6 Identity

The construction of the term model given in the preceding section is enough to establish completeness for first-order logic without identity. We could define satisfaction and provability for a language without *leq*, and construct a maximally consistent saturated set Γ^* for a given se Γ. The term model would then be a model of Γ^* (and hence of Γ). It does not work, however, if identity is present. The reason is that Γ might 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 diffrent and so $t = t'$ is false. We can fix this, however, using a construction known as "factoring." This construction works generally for so-called congruence relations.

Definition 8.10. Let Γ^* be a maximally consistent 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 8.11. *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 $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t, \ldots t_n) \approx f(t_1, \ldots, t', \ldots, t_n)$.*

Definition 8.12. Define Trm$/\approx$.

Definition 8.13. Define term model mod $\approx$

Proposition 8.14. *$\mathfrak{M}(\Gamma^*)/\approx \models \varphi$ iff $\varphi \in \Gamma^*$.*

8.7 The Compactness Theorem

Definition 8.15. A set Γ of formulas is *finitely satisfiable* if and only if every finite $\Gamma_0 \subseteq \Gamma$ is satisfiable.

Theorem 8.16 (Compactness Theorem)**.** 1. *If $\Gamma \models \varphi$ then there is a finite $\Gamma_0 \subseteq \Gamma$ such that $\Gamma_0 \models \varphi$;*
2. *Γ is satisfiable if and only if it is finitely satisfiable.*

8.8 The Löwenheim-Skolem Theorems

Theorem 8.17. *If Γ is consistent then it has a countable model, i.e., it is satisfiable in a structure whose domain is either finite or denumerably infinite.*

Proof. If Γ is consistent, the structure $\mathfrak{M}$ delivered by the proof of the completeness theorem has a domain $|\mathfrak{M}|$ whose cardinality is bounded by that of the set of the terms of the language $\mathcal{L}$. So $\mathfrak{M}$ is at most denumerably infinite. □

Theorem 8.18. *If Γ is consistent set of sentences in the language of first-order logic without identity, then it has a denumerable model, i.e., it is satisfiable in a structure whose domain is denumerably infinite.*

Proof. If Γ is consistent and contains no sentences in which identity appears, then the structure $\mathfrak{M}$ delivered by the proof of the completness theorem has a domain $|\mathfrak{M}|$ whose cardinality is identical to that of the set of the terms of the language $\mathcal{L}$. So $\mathfrak{M}$ is denumerably infinite. □

Chapter 9

Undecidability

9.1 Decision Problems

We have a number of important logical notions, such as those of satisfiability, validity, and consequence, which are properties of (sets) formulas or relations between them. The definitions of these logical notions provide *criteria* for when, say, a sentence is valid. (In this case, it is valid iff it is satisfied in every first-order structure.) These criteria, however, do not in general provide a *method for deciding* if a sentence or set of sentences has the property in question. The question of whether there is such a method for a given notion and class of formulas or sentences is called a *decision problem*.

Example 9.1. The *decision problem for validity of sentences in a first-order language* is the question of whether there is a procedure for deciding, given a sentence in a first-order language, if it is valid or not.

Example 9.2. The *decision problem for satisfiability of sentences in a first-order language* is the question of whether there is a procedure for deciding, given a sentence in a first-order language, if it is satisfiable or not.

We say that a decision problem is *solvable* if there is such a procedure, and *unsolvable* otherwise.

To show that a decision problem is solvable, you typically simply write down the procedure that solves it (and prove that it in fact solves it correctly in every case). Proving that a decision problem is unsolvable is a lot harder: you have to show that there can *in principle* be no procedure whatsoever that solves it. This can only be done rigorously if there is a precise definition of what a "procedure" is.

One such precise definition is provided by the notion of a *Turing machine*. A decision problem is solvable by a Turing machine if there is a Turing machine which, when started on the description of an instance of the decision problem as input, eventually halts with simply "1" or "0" on the tape, representing

"yes" or "no", respectively. To show that a decision problem is not solvable it suffices to show that no such Turing machine exists.

There are decision problems that can relatively easily be shown to be unsolvable by Turing machines, such as the Halting problem. To show logical decision problems undecidable, you show that a known unsolvable problem can be "reduced" to it. A decision problem A can be reduced to a decision problem B iff the answer to an instance of A can be obtained by an answer to an instance of B. For instance, the problem of validity of a sentence can be reduced to that of satisfiability: Given a sentence φ, form the sentence $\neg\varphi$: the former is valid iff the latter is not satisfiable. So the decision problem for validity can be solved by taking an instance, transforming that instance into its negation by putting "$\neg$" in front of it, obtaining the answer to "Is $\neg\varphi$ satisfiable?," and switching the answer. This shows that the problem of validity can be reduced to the problem of satisfiability.

A reduction of a decision problem A to a decision problem B shows two things:

1. If B is solvable, then A is solvable.

2. If A is unsolvable, then B is unsolvable.

A decision problem can therefore be shown to be unsolvable by reducing a known unsolvable problem, such as the Halting Problem, to it.

9.2 Representing Turing Machines

In order to represent Turing machines and their behavior by a sentence of first-order logic, we have to define a suitable language. The language consists of two parts: predicates for describing configurations of the machine, and expressions for counting execution steps ("moments") and positions on the tape. The latter require an initial moment, 0, a "successor" function which is traditionally written as a postfix $'$, and an ordering $x<y$ of "before."

Definition 9.3. Given a Turing machine $M = \langle Q, \Sigma, \delta, s\rangle$, the language $\mathcal{L}_M$ consists of:

1. A two-place predicate $Q_q(x, y)$ for every state $q \in Q$.

2. A two-place predicate $S_\sigma(x, y)$ for every symbol $\sigma \in \Sigma$

3. A constant 0

4. A one-place function $'$

5. A two-place predicate $<$

For each number n there is a canonical term $\overline{n}$, the *numeral* for n, which represents it in $\mathcal{L}_M$. $\overline{0}$ is o, $\overline{1}$ is o', $\overline{2}$ is o'', and so on. More formally:

$$\overline{0} = \mathsf{o}$$
$$\overline{n+1} = \overline{n}'$$

The sentences describing the operation of the Turing machine M on input w are the following:

I Axioms describing numbers:

a) A sentence that says that the successor function is injective:

$$\forall x \forall y (x' = y' \to x = y)$$

b) A sentence that says that every number is less than its successor:

$$\forall x (x < x')$$

c) A sentence that ensures that $<$ is transitive:

$$\forall x \forall y \forall z ((x < y \wedge y < z) \to x < z)$$

II. Axioms describing the input configuration:

a) M is in the inital state s at time 0, scanning square 0:

$$Q_s(\mathsf{o}, \mathsf{o})$$

b) The first n squares contain the symbols $\triangleright, \sigma_{i_1}, \ldots, \sigma_{i_n}$:

$$S_{\triangleright}(\mathsf{o}, \mathsf{o}) \wedge S_{i_1}(\overline{1}, \mathsf{o}) \wedge \cdots \wedge S_{i_n}(\overline{n}, \mathsf{o})$$

c) Otherwise, the tape is empty:

$$\forall x (\overline{n} < x \to S_{\textvisiblespace}(x, \mathsf{o})$$

III. Axioms describing the transition from one configuration to the next:

For the following, let $A(x, y)$ be the conjunction of all sentences of the form

$$\forall z ((z < x \vee x < z \wedge S_\sigma(z, y)) \to S_\sigma(z, y'))$$

where $\sigma \in \Sigma$.

1. For every instruction $\langle q_i, \sigma, q_j, \sigma', L \rangle$, the sentence:

$$\forall x \forall y ((Q_{q_i}(x', y) \wedge S_\sigma(x, y)) \to (Q_{q_j}(x, y) \wedge S_{\sigma'}(x, y) \wedge A(x, y)))$$

2. For every instruction $\langle q_i, \sigma, q_j, \sigma', R\rangle$, the sentence:

$$\forall x \forall y((Q_{q_i}(x,y) \wedge S_\sigma(x,y)) \rightarrow (Q_{q_j}(x',y) \wedge S_{\sigma'}(x,y) \wedge A(x,y)))$$

(Probably also need axioms saying every square as exactly one symbol on it at all times, machine always in exactly one state.)

Let $T(M,w)$ be the conjunction of all the above sentences for Turing machine M and input w

The sentence $H(M,w)$:

$$\exists x \exists y\, Q_h(x,y)$$

expresses that the Turing machine M halts on input w.

9.3 Verifying the Representation

In order to verify that our representation works, we first have to make sure that if M halts on input w, then $T(m,w) \rightarrow H(M,w)$ is valid. We can do this simly by proving that $T(m,w)$ implies a descriptin of the configuration of M for each step of the execution of M on input w. If M halts on input w, then for some n, M will be in a halting configuration at step n (and be scanning square m, for some m). Hence, $T(M,w)$ implies $Q_h(\overline{m},\overline{n})$.

Definition 9.4. Let $C(M,w,n)$ be the sentence

$$Q_q(\overline{m},\overline{n}) \wedge S_{\sigma_0}(\overline{0},\overline{n}) \wedge \cdots \wedge S_{\sigma_k}(\overline{k},\overline{n})$$

where q is the state of M at time n, M is scanning sqare m at time n, square i contains symbol σ_i at time n for $0 \leq i \leq k$ and k is the right-most non-blank square of the tape at time m.

Lemma 9.5. *For each n, $T(M,w)$ implies $C(M,w,n)$.*

Proof. By induction on n.

If $n = 0$, then $C(M,w,n)$ is a conjunct of $T(M,w)$, so implied by it.

Suppose $n > 0$ and at time n, M started on w is in state q scanning square m, and the content of the tape is $\sigma_0, \ldots, \sigma_k$.

... to be completed

□

To complete the verification of our clam, we also have to establish the reverse direction: if $T(M,w) \rightarrow H(M,w)$ is valid, then M does in fact halt when started on input m.

Lemma 9.6. *If $T(M,w)$ entails $H(M,w)$, then M halts on input w.*

Proof. Consider the $\mathcal{L}_M$-structure $\mathfrak{M}$ with domain $\mathbb{N}$ which inerprets o as 0, $'$ as the successor function, and $<$ as the lest-than relation, and the predicates Q_q and S_σ as follows:

$$Q_q^{\mathfrak{M}} = \{(m,n) : \text{after } n \text{ steps, } M \text{ started on } w \text{ is in state } q \text{ scanning square } m\}$$
$$S_\sigma^{\mathfrak{M}} = \{(m,n) : \text{after } n \text{ steps, } M \text{ started on } w \text{ has symbol } \sigma \text{ on square } m\}$$

Clearly, $\mathfrak{M} \models T(M,w)$. If $T(M,w)$ implies $H(M,w)$, then $\mathfrak{M} \models H(M,w)$, i.e.,

$$\mathfrak{M} \models \exists x \exists y\, Q_h(x,y).$$

As $|\mathfrak{M}| = \mathbb{N}$, there must be $m, n \in \mathbb{N}$ so that $\mathfrak{M} \models Q_h(\overline{n}, \overline{m})$. By the definition of $\mathfrak{M}$, this means that M started on input w is in state h after m steps, i.e., has halted. □