Logic and Computation

Lecture notes

Jeremy Avigad
Assistant Professor, Philosophy
Carnegie Mellon University

Version: January 2001

Preface

This document comprises a set of lecture notes that I prepared in the fall of 1998, while teaching a course called *Logic and Computation* in the Philosophy Department at Carnegie Mellon University. I distributed these notes to the class and then followed them almost word for word, in the hopes that doing so would enable students to pay more attention to the contents of the lectures, and free them from the drudgery of transcription. The notes were designed to supplement the textbook rather than to replace it, and so they are more wordy and informal than most textbooks in some places, and less detailed in others.

Logic and Computation is cross-listed as an introductory graduate course (80-610) and as an advanced course for undergraduates (80-310). Most students were in the philosophy department, in either the undergraduate major in *Logic and Computation*, the masters' program of the same name, or the Ph.D. program in *Logic, Computation, and Methodology*. There was also a contingency of students from computer science, and a smaller one from mathematics (which maintains its own series of logic courses). With this in mind, I tried to present the course as a rigorous introduction to mathematical logic, which lent some attention to philosophical context and computational relevance, but did not require a good deal of mathematical background. As prerequisites (described in more detail in Chapter 1), I required some familiarity with first-order logic, and some background in reading and writing mathematical proofs.

For a textbook I used Dirk van Dalen's *Logic and Structure*, which I like very much, even though it is written for a more mathematical audience. At various points in the semester I also circulated a few pages taken from Enderton's *Mathematical Logic*, and, for historical perspective, supplementary writings from Tarski, Russell, etc.

I have continued to use the notes in subsequent years, and I have made some minor additions and corrections along the way. But use these notes with caution: they are not of publication quality, and there are bound to be many errors and inconsistencies.

Please let me know if you find these notes useful in any way. I will be happy to make weekly homework assignments and other handouts available, on request.

Jeremy Avigad
January 2001

Contents

Chapter 1

Introduction

1.1 Goals

The word "logic" derives from the Greek word "logos," or reason, and logic can be broadly construed as the study of the principles of reasoning. Understood this way, logic is the subject of this course.

I should qualify this remark, however. In everyday life, we use different modes of reasoning in different contexts. We can reason about our experiences, and try to determine causal relations between different types of events; this forms the basis of scientific inquiry. We can reason probabilistically, and try to determine the "odds" that the Pirates will win the World Series; or we can employ subjunctive reasoning, and wonder what would have happened had Bill Clinton lost the election. We can reason about events occuring in time, or space; we can reason about knowledge, and belief; or we can reason about moral responsibility, and ethical behavior. We can even try to reason about properties that are vague and imprecise, or try to draw "reasonable" conclusions from vague or incomplete data.

It will soon become clear that in this course we will only address a small fragment of such reasoning. This fragment is *amodal* and *atemporal*, which is to say that we wish to consider reasoning about what is universally true, independent of time, and without concern for what might have been the case had the world been somehow different. Also, we will be concerned with a kind of reasoning that aims for absoluteness and certainty, allowing us to conclude that a certain assertion *necessarily* follows from some assumptions; this ignores the probabilistic, inductive, and "fuzzy" reasoning alluded to above. One can think of the kind of reasoning we will address as the "mathematical" kind, and our subject matter sometimes goes by the title "mathematical logic."

To study mathematical logic, we will employ the usual methods of analytic philosophy: we will present a formal model of the kind of reasoning we wish to capture, and then we will use rigorous methods to study this model. That is, we will try to identify the aspects of language that are important in mathematical reasoning, and present formal definitions of truth and logical consequence; and then we will explore the ramifications of these definitions.

Why restrict ourselves to mathematical logic? In part, because it forms an independently interesting "core" of human reasoning, and in part because studying it is much more tractable than studying more general kinds. Even if you are interested in more general notions of truth, language, knowledge, and rationality, it is a good idea to start with mathematics, where things are neat and precise, and then branch out from there. In short, mathematical reasoning forms a paradigmatic subset of reasoning, and one that is amenable to rigorous analysis.

1.2 Overview

When it comes to logical reasoning, certain words seem to play a central role, among them "and", "if ... then," "every," and so on, so our first task will be to describe these linguistic constructions formally. In our approach, statements will be "built up" from basic undefined terms using certain logical connectives.

With this analysis in hand, will can try to give an account of what it means for a statement φ to follow "logically" from a set of hypotheses Γ. One intuitive approach is to say that φ follows from Γ if whenever every statement in Γ is true, then so is φ. More precisely, we will say that φ is a logical consequence of Γ, or Γ logically implies φ (written $\Gamma \models \varphi$) if, no matter how we interpret the undefined symbols in the language, if everything in Γ is true, then φ is true as well. This is a *semantic* notion: it forces us to explain what we mean by "interpretation," as well as "true under an interpretation."

Of course, one might also say that φ follows from Γ if there is a *proof* of φ from Γ. We will use $\Gamma \vdash \varphi$, and say, informally, "Γ proves φ," when we want to express this fact. This is a syntactic notion: it forces us to explain what we mean by a proof.

This leaves us with two notions of logical consequence: a semantic one, and a syntactic one. What is the relationship between them? One of logic's most impressive achievements, beyond the suitable formalization of the two concepts, is the demonstration that in fact, in the case of first-order predicate logic, they *coincide*. In other words, for every first order sentence φ and set of sentences Γ, Γ proves φ if and only if Γ logically implies φ. The forwards direction is known as "soundness": it asserts that our proof system does not lead us astray. The backwards direction is more interesting (and more difficult to prove). Known as "completeness," this property asserts that the proof rules are robust enough to let us derive all the logical consequences.

Before diving into predicate logic, we will start with an even simpler fragment, known as propositional (or sentential) logic. It is far less expressive than first-order logic, but a good starting point. We will define the language formally, define the semantic and syntactic notions, and then prove completeness.

Once we are done analyzing propositional logic, we will repeat the process in the richer first-order setting. Here we will explore an interesting consequences of the completeness theorem known as compactness, which will lead us to some interesting theorems on the limitations of first-order definability, and the existence of "nonstandard" models of arithmetic.

Finally, if time allows, we will consider other kinds of logic, such as intuitionistic, modal, and temporal logic. We may also consider aspects of proof search and automated deduction.

1.3 Prerequisites

There are two prerequisites for this course:

1. You should be able to read and write clear, rigorous proofs.

2. You should be familiar with first-order logic.

In the philosophy department, *Arguments and Inquiry* (80-211) has been designed to meet these two requirements.

Regarding the first, if you have taken any course in abstract mathematics which centers on the notion of proof, you should also have sufficient background. To help you out, I have recommended Solow's "How to Read and Do Proofs" as a supplementary text. But note that writing readable proofs takes some practice, so if you are trying to pick up this skill on the fly, you may have a hard time.

Regarding the second, you may have gathered some familiarity with first-order logic from any number of sources. In this course, we will devote most of our time to studying first-order logic, with the implicit understanding that this kind of logic represents an interesting form of reasoning. If symbolic logic is alien to you, you might not feel convinced of this last fact, which may make the formal analysis seem dull and unmotivated. If you want to get a better sense of first-order logic and how it works, I recommend *Tarski's World* by Barwise and Etchemendy most highly; the package includes both software and exercises that act efficiently to get you comfortable "thinking in" a first-order language.

1.4 Mathematical preliminaries

The following section establishes some of the notation we will use later on, and reviews some of the mathematical notions you should be familiar with.

A staple of modern mathematics is the notion of a *set* of objects. Sets have elements; the relation "x is an element of A" is written $x \in A$. If A and B are sets then A is a subset of B, written $A \subseteq B$, if every element of A is an element of B. A and B are equal, i.e. the same set, if $A \subseteq B$ and $B \subseteq A$. Notice that saying $A = B$ is equivalent to saying that every element of A is an element of B and vice-versa; so two sets are equal if they have exactly the same elements.

$\mathbb{N}$, $\mathbb{Q}$, and $\mathbb{R}$ denote the natural numbers, the rationals, and the reals respectively. Given a set A, one can describe a subset of A by a property; if P is such a property, the notation

$$\{x \in A \mid P(x)\}$$

is read "the set of all elements of A satisyfing P" or "the set of $x \in A$ such that $P(x)$." For example, the set

$$\{x \in \mathbb{N} \mid \text{for some } y \in \mathbb{N},\ x = 2y\}$$

is just a fancy way of describing the set of even numbers.

If A and B are sets, $A \cup B$ denotes their union, i.e. the set of things that are in either one, and $A \cap B$ denotes their intersection, i.e. the set of things that are in both. If $\mathcal{A}$ is a collection of sets, $\bigcup \mathcal{A}$ and $\bigcap \mathcal{A}$ denote the union and intersection, respectively, of all the sets in $\mathcal{A}$; if $A_0, A_1, A_2, \ldots$ is a sequence of sets indexed by natural numbers, then $\bigcup_i A_i$ and $\bigcap_i A_i$ denote their union and intersection. There are other ways of building more sets. For example, if A is any set, $P(A)$, "the power set of A," denotes the set of all subsets of A. If A and B are sets, $A \times B$, "the cross product of A and B," is the set of all ordered pairs $\langle a, b\rangle$ made up from of an element $a \in A$ and an element $b \in B$.

Another staple of modern mathematics is the notion of a *function*. Formally (from a set-theoretic, foundational point of view) a function from A to B is a subset of $A \times B$ such that for every a in A there is exactly one $b \in B$ such that $\langle a, b \rangle$ is in the set. Try to match this up with the intuition that a function is a "map" from A to B; the notation $f(a) = b$ means that the pair $\langle a, b \rangle$ is in the set. I will write $f : A \to B$ to denote that f is a function from A to B; A is called the domain of f, and B is called the range (or codomain).

Having seen the formal definition of a function, it won't hurt you much to forget it. You should sleep better at night knowing that it is there, and knowing that there a precise set-theoretic framework that provides a clear set of rules for talking about sets and functions. However, in practice, it is easier to learn to speak "mathematics" by studying examples (e.g. in these notes and in the textbook).

I will often use $\vec{a}$ to indicate a finite sequence of elements $a_1, a_2, \ldots, a_k$. I will assume that you are familiar with proofs by induction on the natural numbers, though I will review this briefly in the next chapter, mainly as a way to motivate more abstract forms of induction and recursion.

1.5 The use-mention distinction

One thing that may cause some confusion is the fact that in this course, with will be using informal logical arguments to study formal logical arguments. Which is to say, we will be making informal statements about formal statements, proving informal theorems about formal proofs, and so on.

Logicians find it useful to distinguish between "theory" and "metatheory"–that is, the theory that you are studying and the theory you are using to study it. Keep this distinction clear! On the informal level, we will be using the same types of rigorous arguments that you will find in many courses in mathematics and analytic philosophy (this is the metatheory). On the formal level, we will define a symbolic language and proof system to model informal logical discourse (this is the theory). But the formal model should not affect the informal practice. For example, just because we define symbols $\wedge$ and $\to$ to represent the informal notions conjunction and implication, you are not likely to write a letter home to your family and write

> Dear Mom and Dad,
>
> I went to my logic class today $\wedge$ learned a lot of neat things. I make it through the class $\to$ I will learn a lot more.

In much the same way, the informal proofs on your homework should not be suffused with formal symbolism, which makes them harder to read.

A related issue is known to philosophers as the "use-mention" distinction. Consider the following four statements:

- Jeremy is a nice name.
- "Jeremy" is a nice name.
- Jeremy is a nice person.
- "Jeremy" is a nice person.

Clearly, two of these make sense, and two don't. This illustrates the fact that there is a difference between *using* a syntactic object (e.g. to refer to something) and *mentioning* (referring to the syntactic object itself). This issue comes up in this course because we will be stating theorems about syntactic objects, and using variables to refer to them. For example, if φ stands for the propositional formula "$A \to B$", then the following two statements mean the same thing:

- φ has three symbols
- "$A \to B$" has three symbols

In practice, I will sometimes be sloppy about making the distinction between use and mention, but you should be able to supply the necessary corrections on your own.

Chapter 2

Generalized Induction and Recursion

2.1 Induction and recursion on the natural numbers

If P represents some property that a given natural number may or may not have, I will write $P(n)$ to indicate that P holds of n. The principal of induction on the natural numbers is as follows:

Theorem 2.1.1 (induction principle) *Suppose P is some property of natural numbers, such that $P(0)$ is true, and for every natural number n, if $P(n)$ is true then so is $P(n+1)$. Then P holds of every number.*

The upshot of this principle is that we can carry out proofs by induction on the natural numbers. That is, to prove that some statement holds for every natural number, first show that it holds of 0 (this is called the base case) and then show that if it holds of some number n, then it holds of $n+1$. For example:

Theorem 2.1.2 *For every natural number n,*

$$0+1+2+\ldots+n = n(n+1)/2.$$

Proof. Let $P(n)$ be the property "0 + 1 + ... + n = n(n+1) /2."

Base case: $0 = 0$.

Induction step: We need to show that $P(n+1)$ follows from the assumption that $P(n)$ is true. So suppose $P(n)$, i.e. $0+1+\ldots+n = n(n+1)/2$, and let us try to prove $P(n+1)$, namely

$$0 + 1 + \ldots + n + (n+1) = (n+1)(n+2)/2.$$

(Note that $P(n+1)$ is just the result of replacing n by $n+1$ in $P(n)$.) We have that

$$\begin{aligned} 0 + 1 + \ldots + n + (n+1) &= (0 + 1 + \ldots + n) + (n+1) \\ &= n(n+1)/2 + (n+1) \\ &= (n+1)(n+2)/2. \end{aligned}$$

The first line is just a matter of rewriting the equation (technically, the associativity of addition). The second line uses the inductive hypothesis, and the third line follows using ordinary algebra. □

In Solow's book you will find many variations on induction; for example, you can start the induction at any number (i.e. prove that P is true for all numbers greater than k, for some k); or, in the inductive hypothesis, use the fact that P is true for *every* number smaller than $n+1$ (and not just n). In important variation of induction is given by

Theorem 2.1.3 (the least element principle) *Suppose P is true for some natural number. Then there is a smallest natural number for which P is true.*

You should convince yourself that the least element principle is equivalent to induction, which is to say that each one can be proved from the other using only some other basic properties of the natural numbers. (Hint: the least element principle can be rephrased as follows: if there is no smallest natural number for which P is true, then for every natural number, P is false. Given P satisfying the hypothesis, let $P'(n)$ be the property "P does not hold of any number smaller than n" and use induction to show that P' holds of every natural number.)

Recall that a number greater than or equal to 2 is *composite* if it can be written as the product of two smaller numbers. A number is *prime* if it is not composite. We will say that a number n can be factored into primes if we can write

$$n = p_1 p_2 \ldots p_k$$

where each p_i is prime. Here is an example of the way that one can use the least element principle:

Theorem 2.1.4 *Every natural number greater than or equal to 2 can be factored into primes.*

Proof. Suppose there were some number n greater than or equal to 2 that could not be factored into primes. By the least element principle, there would be a smallest such n. Now, there are two cases:

Case 1: n is prime. Then trivially n can be factored into primes. This is a contradiction.

Case 2: n is not prime. Then n is composite, which is to say we can write n as the product of two smaller numbers, p and q. Since we are assuming that n is the least natural number that cannot be factored into primes, p and q can each be factored into primes. But now combining the prime factorization of p with the prime factorization of q results in a prime factorization of n. This is also a contradiction.

The assumption that there is some number n greater than or equal to 2 that cannot be factored into primes resulted in a contradiction. Hence, there is no such number. □

By now you have seen most of the basic proof methods that you will need for this course. For example, to prove "if A then B," we can suppose that A is true, and show that B follows from this assumption. If we want to prove that a statement A is true, we can show that the assumption that it is false leads to a contradiction. Saying "not every number can be factored into primes" is equivalent to saying "there is some number than can't be factored into primes." And so on.

Pay careful attention to the structure of such proofs! I will try to highlight it in lectures; but if this is very alien to you, you have a lot of catching up to do (using Solow). Later in the course, we will see these informal modes of reasoning mirrored in a formal deductive system.

Soon it will be helpful to have the induction principle in a slightly different form. Since every property of the natural numbers determines a subset of the natural numbers (and vice-versa), we can state induction as follows:

Theorem 2.1.5 *Let A be any subset of the natural numbers, with the property that 0 is an element of A, and whenever some number n is an element of A, so is $n+1$. Then $A = \mathbb{N}$.*

The "flip-side" of induction is recursion, which can be described as follows. Suppose want to define a function f from natural numbers to some set. We can do this by specifying the value of f at 0, and then for each natural number n, specify the value of f at $n+1$ in terms of the value at n.

For example, consider the functions given by

$$\begin{aligned} f(0) &= 1 \\ f(n+1) &= 2 \cdot f(n) \end{aligned}$$

and

$$\begin{aligned} g(0) &= 1 \\ g(n+1) &= n \cdot g(n). \end{aligned}$$

Can you give explicit equations for computing f and g?

In fact, even addition and multiplication can be specified recursively. For example, we can define $x + y$ by

$$\begin{aligned} x + 0 &= x \\ x + (y+1) &= (x+y) + 1. \end{aligned}$$

More formally, we are specifying a function $f(x, y)$ by recursion on y, with x carried along as a "parameter."

Here is the general theorem.

Theorem 2.1.6 *Suppose a is an element of a set A, and h is a function from A to A. Then there is a unique function $g : \mathbb{N} \to A$, having the properties*

$$\begin{aligned} g(0) &= a \\ g(n+1) &= h(g(n)) \quad \textit{for every natural number } n. \end{aligned}$$

Actually, this form of the theorem corresponds to the first two examples above; more generally, there are some extra parameters $\vec{z}$ floating around, so we have

$$\begin{aligned} g(0, \vec{z}) &= a(\vec{z}) \\ g(n+1, \vec{z}) &= h(g(n, \vec{z}), \vec{z}) \quad \text{for every natural number } n. \end{aligned}$$

There are, in fact, even fancier forms of recursion. For example, in specifying the value of $g(n+1)$, you might want to use values of g for arbitrary inputs less than $n+1$. For simplicity, however, I will stick with the simple version.

An important thing to notice is that what justifies definition by recursion is really the principle of induction. That is, suppose we are given the specification of g as in the statement of the theorem. I claim that for each n, there is a unique function

$$g_n : \{0, \ldots, n\} \to A$$

satisfying the specifying equations, up to n. This is not hard to show, by induction. We get the final function g by setting

$$g(n) = g_n(n).$$

2.2 Inductively defined sets

We can now summarize what makes the natural numbers so special: we have a special element, 0, and an operation, "+1," which "generate" the entire set. This is what gives us the principal of induction, and allows us to define functions recursively.

Of course, this setup is so simple that we can hope to generalize it. And, in fact, this is exactly what is going on in the Enderton handout. Read that first; you can consider the information here to be supplementary.

Let's try to clarify what we mean when we say that the set of natural numbers is generated from 0 and the successor function (as "+1" is usually known). Suppose we had a large universe U of objects, containing the natural numbers, but possibly containing other things as well (real numbers, functions, Bob, Joe, Susan, the Beatles' greatest hits,...). Suppose we also have a generalization of the successor function defined on U, which maps every natural number to its successor and behaves arbitrarily on other members of U. Can we explain what makes the natural numbers special, by characterizing them in terms of U, 0, and the successor operation?

Intuitively, the set of natural numbers is the *smallest* set containing 0 and closed under this operation. To make this more precise, call a set *inductive* if it contains 0 and is closed under the successor operation. Of course, the natural numbers are such a set, but there are other inductive sets that are too big: for example, take the natural numbers together with $\{3.5, 4.5, 5.5, \ldots\}$. In a sense, we want to say that the natural numbers is the *smallest* inductive subset of U. Formally, we can define

$$\mathbb{N}^* = \bigcap \{I \mid I \text{ is an inductive subset of } U\}.$$

Another way of characterizing the set of natural numbers is to say that the natural numbers are the things you can get to, starting from 0 and applying the successor function at most finitely many times. This is the idea behind the definition of a construction sequence in Enderton, or a formation sequence in van Dalen. Then we define $\mathbb{N}_*$ to be the set of all things we can reach with a formation sequence, using 0 and successor.

Of course, we all know that $\mathbb{N}^* = \mathbb{N}_*$, and they are both equal to the set of natural numbers, that we all know and love. But the approach works more generally. In the more general setup, we have

- An underlying universe, U
- A set of initial elements, $B \subseteq U$
- A set of functions on U, $f_1, f_2, \ldots, f_k$ of various "arities" (i.e. taking various numbers of arguments)

We want to define the set C of objects generated from B by the functions f_i. Following Enderton, I will give two definitions: one, C^*, "from above," and one, C_*, "from below." Then I will show that C^* and C_* are the same.

For the definition from above, say a subset A of U is *inductive* if B is a subset of A, and whenever $a_1, \ldots, a_m$ is in A and f_j is an m-ary function, then $f_j(a_1, \ldots, a_m)$ is in A as well. Notice that U itself is inductive. (Why?) Let C^* be the intersection of all inductive subsets of U. In other words, an element is in C^* if and only if it is in every inductive subset of U.

Lemma 2.2.1 *C^* is inductive.*

Proof. Left to reader: a homework exercise. □

For the definition from below, say a finite sequence $\langle a_0, a_1, \ldots, a_k \rangle$ of elements of U is a *formation sequence* (or, in Enderton's terminology, a *construction sequence*) if, for each i, either a_i is an element of B, or there is a function f_l and numbers $j_1, \ldots, j_m$ less than i, such that $a_i = f_l(a_{j_1}, \ldots, a_{j_m})$. In other words, the inclusion of each a_i is "justified" either by the fact that a_i is in B, or by the fact that a_i is "generated" by prior elements in the sequence. Finally, let C_* be the set of elements a in U for which there is a formation sequence ending in a.

Lemma 2.2.2 *C^* is a subset of C_*.*

Proof. It is not hard to show that C_* is inductive: If a is an element of B, then $\langle b \rangle$ is a one-element formation sequence. And if $a_1, \ldots, a_m$ are in C_*, joining together the formation sequences for these and then appending $f_j(a_1, \ldots, a_m)$ yields a formation sequence for the latter.

But anything in C^* is in *every* inductive set. So everything in C^* is in C_*. $\square$

Lemma 2.2.3 *C_* is a subset of C^*.*

Proof. Let a be any element of C_*. Then there is a formation sequence $\langle a_0, \ldots, a_k \rangle$ with $a_k = a$. Show by ordinary induction on the natural numbers, that for each i, a_i is in C^*. So a is in C^*. $\square$

So $C^* = C_*$. From now on, let's drop the stars and call this set C.

Lemma 2.2.4 *The principle of induction holds for C. In other words, if D is any inductive subset of C, then $D = C$.*

Proof. Let D be an inductive subset of C. By the definition of C^*, every element of C is in D, so C is a subset of D. Hence $D = C$. $\square$

Put slightly differently, we have the following:

Theorem 2.2.5 (induction principle for C) *Suppose P is a property of elements of C, such that P holds of all the elements of B, and for each function f_i, if P holds of each of the elements $a_1, \ldots, a_k$, then P holds of $f_i(a_1, \ldots, a_k)$. Then P holds of every element of C.*

Proof. Let D be the set of elements of C with property P; the hypothesis says that D is an inductive set. $\square$

As far as terminology goes, I will say that C is the "smallest subset of U containing B and closed under the f_i." Or, leaving U implicit, I will write that C is the set inductively defined by the following clauses:

- B is contained in C
- If $a_1, \ldots, a_k$ are all in C, so is $f_1(a_1, \ldots, a_k)$
- $\ldots$

For example, the set of "AB-strings" is the smallest set of strings of symbols such that

- $\emptyset$ (the empty string) is an AB-string

- If s is an AB-string, so is "A"$\hat{}s$
- If s is an AB-string, so is "B"$\hat{}s$

where $s\hat{}t$ denotes the result of concatenating the strings s and t. For another example, we can define the set of "arithmetic expressions" to be the smallest set of strings of symbols such that

- If n is a natural number, then the decimal representation of n is an arithmetic expression
- If s and t are arithmetic expressions, so are $(s + t)$ and $(s \times t)$

Here $(s + t)$ is really an abbreviation for "("$\hat{}s\hat{}$"+"$\hat{}t\hat{}$")", but for the most part, I will stick with the lazier notation. Finally, we can think of the natural numbers themselves as being the smallest set of "mathematical objects" containing 0 and closed under the successor function. By the time this course is done, you will have seen lots more examples of inductive definitions.

Here is a very simple example of how one can use the principle of induction on an inductively defined set.

Proposition 2.2.6 *Every arithmetic proposition has the same number of left and right parentheses.*

Proof. If n is a natural number, the decimal representation of n has no left parentheses and no right parentheses, so the statement holds in the base case.

For the induction step, suppose the claim holds for s and t, and let us show that it holds for $(s + t)$. In this last expression, the number of left parentheses is equal to one plus the number of left parentheses in s plus the number of left parentheses in t. The number of right parentheses is equal to one plus the number of right parentheses in s plus the number of right parentheses in t. By the inductive hypothesis, these two numbers are equal.

A similar argument shows that if the claim holds for s and t, it holds for the expression $(s \times t)$. This covers all the cases, so we are done. $\square$

A historical note: the definition of the natural numbers "from above," which involves taking an intersection over all inductive sets, make strike you either as being really nifty, or really bizarre. This characterization first appeared, essentially, in Frege's "Begriffschrift" of 1879 and his *Grundlagen der Arithmetik* of 1884. It is also beautifully described in an essay by Richard Dedekind called "Was sind und was sollen die Zahlen," (roughly, "What are the numbers, and what should they be?"), written in 1888. Dedekind claimed to have been unaware of Frege's work, but some historians question whether this could really have been the case. Nonetheless, many feel that Dedekind's essay, with its emphasis on the abstract characterization of mathematical *structures*, marked a revolution in mathematical thought.

2.3 Recursion on inductively defined sets

In the last section we saw that we have a principal of induction on any inductively defined set. What about recursion?

A complication arises. Note that we defined the set of AB-strings inductively as the smallest subset of U, satisfying the following:

- $\emptyset$ is an AB-string
- If s is an AB-string, so is $f_1(s)$
- If s is an AB-string, so is $f_2(s)$

where U is the set of all strings of symbols, $f_1(s) =$ "A"$\hat{\ }s$, and $f_2(s) =$ "B"$\hat{\ }s$. We can then define a function which "translates" AB-strings to strings of 0's and 1's, by

- $F(\emptyset) = \emptyset$
- $F(f_1(s)) =$ "0"$\hat{\ }F(s)$
- $F(f_2(s)) =$ "1"$\hat{\ }F(s)$

Then $F($"AAB"$) = F(f_1(f_1(f_2(\emptyset)))) =$ "001", and all is fine and dandy.

But what if instead we had used *different* functions f_1 and f_2 to define a set inductively? For example, let C be the set generated as above, with $f_1(s) =$ "*"$\hat{\ }s$ and $f_2(s) =$ "**"$\hat{\ }s$. With the definition of F just given, what is $F($"***"$)$?

We have a problem, in that "***" can be generated in different ways. For example, it is equal to both $f_1(f_1(f_1(\emptyset)))$ and $f_2(f_1(\emptyset))$, and as a result, we don't know how to compute the value F for this input. The following definition disallows this amibiguity:

Definition 2.3.1 *Let C be the set inductively generated by B and some functions $\vec{f}$. Then we will say that C is* freely generated *if, on C, each f_i is injective, and the ranges of the functions f_i are disjoint from each other and from B. In other words, if $\vec{c}$ and $\vec{d}$ are sequences of elements from C,* *then* $f_i(\vec{c})$ *and* $f_j(\vec{d})$ *are equal if and only if in fact* $i = j$ *and* $\vec{c} = \vec{d}$*; and for each* i *and* $\vec{c}$*,* $f_i(\vec{c})$ *is not in* B*.*

Enderton shows that we *do* have a principal of recursive definition on freely generated structures. The proof is very similar to the proof that one can do recursion on the natural numbers.

Theorem 2.3.2 *Suppose* C *is freely generated from* B *and* $\vec{f}$*. Suppose* V *is another set,* h *is a function from* B *to* V*, and for each* i*,* g_i *is a function from* V *to* V *with the same arity as* f_i*. Then there is exactly one function* F *from* C *to* V*, satisfying the following conditions:*

- *For each* $a \in B$*,* $F(a) = h(a)$
- *If* $a_1, \ldots, a_k$ *are in* C*, then* $F(f_i(a_1, \ldots, a_k)) = g_i(F(a_1), \ldots, F(a_k))$*.*

Think of the g_i's being "translations" of the operations on f_i. I will try to illustrate this with a picture on the board. Here are some examples. First, on define a function $length()$ from AB-strings to $\mathbb{N}$

- $length(\emptyset) = 0$
- $length(\text{"A"}\hat{\ }s) = length(s) + 1$
- $length(\text{"B"}\hat{\ }s) = length(s) + 1$

Also, define $ab2bin()$ from AB-strings to strings of 0's and 1's, by

- $ab2bin(\emptyset) = \emptyset$
- $ab2bin(\text{"A"}\hat{\ }s) = \text{"0"}\hat{\ }ab2bin(s)$
- $ab2bin(\text{"B"}\hat{\ }s) = \text{"1"}\hat{\ }ab2bin(s)$

We can also define a function $val()$ from "arithmetic expressions" to $\mathbb{N}$ by

- $val(s) = n$, if s is the decimal representation of n
- $val((s + t)) = val(s) + val(t)$
- $val((s \times t)) = val(s) \times val(t)$

Steve Awodey tells me that category theorists think of freely generated inductively defined structures as having "no junk, no noise." "No junk" means that there is nothing in the set that doesn't have to be there, which stems from the fact that the set is inductively defined; and "no noise" means that anything in the set got there in just one way, arising from the fact that the elements are freely generated.

2.4 Induction on length

Given an inductively defined structure, C, the principal of induction on C states that to prove that every element of C has a certain property, it is sufficient to show that the "starting" elements have that property, and that this property is maintained by the generating functions. Some of the proofs we will do later on, however, use slightly different forms of induction, and so here I would like to justify some of these variations.

Theorem 2.4.1 (induction on length) *Suppose C is any set, and we are given a function length from C to $\mathbb{N}$. Suppose P is a property of elements of C such that for every n, if P is true of elements of C of length less than n, then it is true of every element of length n as well. Then P holds of every element of C.*

Proof. By induction on n, show that P holds of every element of C having length less than n. $\square$

Though I have called the function "length," it really need not have anything to do with length. All that is necessary is that the function measure the "complexity" of the elements of C, in such a way that we can establish the desired inductive hypothesis.

In much the same way, one can adapt the least element principle to C: to show P is true of every element of C, suppose there is some counterexample. Take a "shortest" counterexample, and derive a contradiction.

Chapter 3

Propositional Logic

3.1 Overview

We are now ready to start applying formal analytic tools to the study of logic. We will start with propositional logic, which is sometimes also referred to as "sentential logic."

Given some basic declarative sentences, like "it is raining" or "the sky is green," we can form more complex sentences like "it is raining and the sky is green" or "if it is raining, then the sky is green." Propositional logic aims to explain how the meaning of these complex sentences are related to the meaning of the basic ones; and to understand what kinds of inferences we can make that depend only on the "logical structure" of a given complex sentence, independent of the meaning of the basic propositions. In other words, we are not trying to study the meaning of "rain," or "green"; but rather, the meaning of "and" and "if ... then" constructions. To that end, we will represent the basic declarative assertions with variables $p_0, p_1, \ldots$, and then model the buildup of more complex sentences with formal symbols.

I have already said that there are many aspects of human language that will not be captured by our formalism. Section 1.1 in van Dalen provides a good overview. I will just add a few remarks and clarifications.

The kinds of connectives we are interested include $\wedge$ (and), $\vee$ (or), $\rightarrow$ (implies), $\neg$ (not), $\leftrightarrow$ (iff). We will see later that we can define others connectives from these, and, in fact, we can define some of these in terms of others. Note that the word "or" can be used in two ways:

- Either Sarah is home or she is on campus.

- I hope my mystery date is either tall or handsome.

The first "or" is exclusive; Sarah can't be both home and on campus. The second "or" is presumably inclusive: the speaker presumably won't be upset if the mystery date is *both* tall and handsome. By default, when I say "or" in this class, I mean to use an inclusive "or," which is denoted by $\vee$. The exclusive "or" is usually denoted $\oplus$.

Material implication $p \rightarrow q$ can be defined as $\neg p \vee q$. Many students find this confusing, since it means that if the antecedent is false, the implication is automatically true. For example, if you walk outside this classroom and say to someone in the hallway,

> If Jeremy is alone in that room, then he is drunk and naked and dancing on the chairs.

this statement is vacuously true, for the simple reason that the antecedent, or hypothesis, is false. Of course, the sentence seems to be saying that even though I may not be alone right now, if I *were* alone, I would be drunk and naked; but this introduces a "modality" that we are trying to avoid. To appreciate the difficulties that arise with this modality, consider the sentences

> If Bill Clinton were George Bush, he would have lost the last election.

or

> If my grandmother had wheels, she would be a motorcycle.

This forces us to consider alternative universes, where Bill Clinton *is* George Bush, or where my grandmother has wheels; and it's hard to reason about these universes. To understand what material implication tries to model, consider the statement

> If x is a prime number that is greater than 2, then x is odd.

Intuitively, we would like to say that this is true, no matter what we substitute for x. But that commits us to accepting the truth of

> If 8 is a prime number that is greater than 2, then 8 is odd.

Another comment I would like to make is that we are adopting what is sometimes called a "classical" view of truth: we assume that the kind of statements that the variables can range over are clear and well-defined, and hence either true or false. That is, we accept $p \vee \neg p$ as being a true statement, for anything that we may wish to substitute for p. This precludes vague statements, like "Bill Clinton has been an effective president"; it even causes problems with the examples above, since the sky may have a sort-of greenish tint and it might be kind of foggy and drizzly outside but not really raining. In short, we are just adopting the convention that the basic statements we are dealing with have a well-defined truth value. (The question as to whether or not mathematical statements fit this description lies at the heart of foundational disputes between classical and intuitionistic views of mathematics.)

Suppose we are given a propositional formula like $(p \wedge q) \to r$. Is it true? Well, the truth depends on what statements the variables are supposed to represent; or, more precisely, it depends on their truth values. (In class I'll compute the truth table of this sentence, though I will assume that most of you have done this before and will therefore be somewhat bored. Incidentally, van Dalen likes to identify "true" with 1 and "false" with 0, so that, for the record, we know what they "are." Feel free to take this position, if you'd like.)

Of course, some formulas will come out true no matter how we assign truth values to the propositional variables. We will say that such statements are *tautologies*, or *tautologically true*, or *valid*. The intuition is that they are *necessarily* true, purely in virtue of their logical form, and independent of the meaning of the basic propositions. This will give us the semantic notion, $\models \varphi$. We will then look at ways that we can establish that a formula is valid, without having to compute the entire truth table. This will give is the syntactic notion, $\vdash \varphi$. Finally, we will show that the deductive procedure is sound and complete, so the syntactic and semantic notions coincide.

That's the basic idea. We will spend most of the rest of this section filling in the details, giving a formal definition of "propositional formula," "true in an interpretation," "valid," "proof," and so on. Why bother, when the details are more or less obvious? There are a couple of answers. Contrary to what you might suppose, the right way to fill in details is *not* always obvious, and doing so can clarify basic issues and yield surprising results. Second of all, a fully rigorous development will point the way to mechanizing propositional reasoning. I will come back to this second point below.

3.2 Syntax

I will define the language of propositional logic and the set *PROP* of propositions as in van Dalen, Definition 1.1.1 and 1.1.2. Note that the latter is simply an inductive definition of the kind we have already discussed. For the inductive definition to make sense, we need to specify an underlying universe U. Since we have to start somewhere, I will take the notions "symbol" and "strings of symbols" to be basic, and assume concatenation of strings to have the obvious properties. When it comes to defining $PROP$, we can then take U to consist of the set of all strings involving the following symbols:

$$(,\ ),\ \wedge,\ \vee,\ \rightarrow,\ \leftrightarrow,\ \neg,\ \bot,\ p_0,\ p_1,\ p_2,\ \ldots$$

Now that we have a definition of $PROP$, we have the formal means to determine whether or not something is a propositional formula. For example, given the closure conditions on $PROP$, we can show sequentially that each of the following is a propositional formula:

$$p_0,\ p_{17},\ (p_0 \wedge p_{17}),\ p_6,\ ((p_0 \wedge p_{17}) \rightarrow p_6).$$

Alternatively, note that the sequence above is, in fact, a formation sequence.

How can we show that, say "$((p_0 \rightarrow$" is *not* in $PROP$? This requires a little bit more thought. But if it were in $PROP$, it would have to be the last element of a formation sequence. But then it would have to be either a propositional variable (which it clearly isn't), or it would have to arise from previous elements of $PROP$ by one of the inductive clauses; but anything arising from one of the inductive clauses ends with a right parenthesis.

We can now do proofs by induction on $PROP$. For example, we can show that every propositional formula has the same number of left and right parentheses.

We can also define functions by recursion. All of these examples are in van Dalen.

- A function *length* from $PROP$ to $\mathbb{N}$
- A function *parsetree* from $PROP$ to the set of trees whose leaves are labelled with propositional variables, and whose internal nodes are labelled with logical connectives. Note that if you want this function to be *injective*, you need to assume that the children of any given node are ordered; otherwise, $(p \wedge q)$ and $(q \wedge p)$ have the same parse tree.
- A function *rank* from $PROP$ to $\mathbb{N}$. This measures, essentially, the depth of the parse tree.
- A function *subformulas*, from $PROP$ to the power set of $PROP$ (that is, the set of all subsets of $PROP$).

As an exercise, you should try to define a function *complexity* that counts the number of nodes in the parse tree. We can now prove more interesting theorems by induction. For example, if we let $|A|$ denote the number of elements in the (finite) set A, we have the following:

Proposition 3.2.1 *For every propositional formula φ, $|subformulas(\varphi)| \leq 2^{rank(\varphi)+1} - 1$.*

Proof. Suppose φ is atomic. Then $|subformulas(\varphi)| = 1$ and $rank(\varphi) = 0$, and $1 \leq 2^{0+1} - 1$.

For the induction step, suppose φ is of the form $\theta \wedge \eta$. Then

$$\begin{aligned} |subformulas(\varphi)| &\leq |subformulas(\theta)| + |subformulas(\eta)| + 1 \\ &\leq (2^{rank(\theta)+1} - 1) + (2^{rank(\eta)+1} - 1) + 1 \quad \text{by the IH} \\ &\leq 2 \cdot 2^{\max(rank(\theta), rank(\eta))+1} - 1 \\ &= 2^{rank(\varphi)+1} - 1 \end{aligned}$$

The other binary connectives are handled in the same way. The case where φ is of the form $\neg\theta$ is left to the reader. □

To save myself some chalk, I will sometimes omit parenthesis, under the convention that they should be mentally reinserted according to the following "order of operations":

1. $\neg$ binds most tightly (read from left to right)
2. $\wedge$ and $\vee$ come next (from left to right)
3. $\rightarrow$ and $\leftrightarrow$ come last

For example, $p_0 \wedge p_1 \rightarrow p_2 \vee p_3$ is really $((p_0 \wedge p_1) \rightarrow (p_2 \vee p_3))$. (Actually, logicians often adopt the convention that multiple $\rightarrow$'s are read from right to left. But if I ever pull that on you, just shoot me.) I will often use variables like $p, q, r, \ldots$ instead of $p_0, p_1, p_2, \ldots$.

3.3 Unique readability

In defining the functions on *PROP* above by recursion, I glossed over one important fact, namely, that *PROP* is freely generated! Since free-generation implies that the map from *PROP* to parse trees is well-defined, it essentially amounts to saying that there is only one way to "parse" a given string of symbols. As a result, in the case of *PROP*, this is known as "unique readability."

I will go over Enderton's proof the the propositional formulas have this property. There is no way of avoiding the nitty gritty details, since the definition of *PROP* depends on the generating mappings from strings to strings.

3.4 Computational issues

The title of this course is *Logic and Computation*. So far I have emphasized the "logic" part. Where does the computation come in?

Actually, aspects of computation will be implicit in almost everything we do. As I mentioned at the end of Section 3.1, our goal is to give an account of logical reasoning that admits mechanization. Basing our work on inductively defined structures does just that. Indeed, many introductory programming courses seem to be founded on the idea that the essence of computation lies in finding the right inductively defined data structures (lists, trees, etc.), and that just about any problem can be solved with a clever recursion. Many among you will have already recognized the fact that our inductive definition of *PROP* leads to the representation of propositional formulas with a natural data structure. For example, Jesse Hughes was kind enough to supply me with the following class definitions (untested) in Java:

```
public abstract class PropForm{

}

public class Connective extends Character{
  //If you prefer "->" and "<->", then this should extend String,
  //not Character.

  public Connective(Character c)
    throws FormulaConstructorException{
      char ch = c.charValue();

      if ((ch != 'v') && //Disjunction
        (ch != '&') && //Conjunction
        (ch != '>') && //Implication
        (ch != '=') && //Biconditional (a bad choice, I know)
        (ch != '~')) //Negation
```

```
        throw(new FormulaConstructorException("Illegal connective"));
    }

    //No toString() method is required, since we us the Character.toString()
    //method. Alternatively, you could change the toString() method here to
    //print "->" and "<->", while keeping Connective a subclass of Character.
}

public class AtomicForm extends PropForm {

    Character symbol;
    Integer index;

    public AtomicForm(Character sym,Integer in)
    throws FormulaConstructorException{
        if (('A' >= sym) && (sym <= 'Z')){
            symbol = sym;
            index = in;
        }
        else
            throw(new FormulaConstructorException("Illegal atomic symbol"));
    }

    public String toString(){
        return symbol.toString() + index.toString();

    }
}

public class BinaryForm{

    PropForm left, right;

    public BinaryForm(Connective c,PropForm l,PropForm r)
        throws FormulaConstructorException{
            conn = c;
            left = l;
            right = r;
    }
```

```
    public String toString(){
        return "(" + left.toString()+") "+
            conn.toString()+
            " ("+right.toString()+")";
    }
}

public class UnaryForm{

    PropForm sub;

    public BinaryForm(Connective c,PropForm s)
        throws FormulaConstructorException{
            conn = c;
            sub = s;
    }

    public String toString(){
        return conn.toString()+
            " ("+sub.toString()+")";
    }

}

public class FormulaConstructorException extends RuntimeException{
    public FormulaConstructorException(){
        super();
    }

    public FormulaConstructorException(String s){
        super(s);
    }
}
```

A propositional formula is either an atomic formula (which consists of a symbol and an integer, like $p17$); or a unary formula (which consists of a connective, and another propositional formula); or a binary formula (which consists of a connective, and two more propositional formulas). Jesse has also implemented the method *toString*(), which converts a propositional formula to a string that can be printed out.

The representation is even more natural in ML, which supports recursively defined inductive types. The type of $PROP$ can be defined as follows (I think):

```
datatype PropForm = Atomic of AtomicForm |
  Unary of UnaryForm |
  Binary of BinaryForm

datatype AtomicForm = char * int

datatype UnaryForm = char * PropForm

datatype BinaryForm = char * PropForm * PropForm
```

ML will then support recursive function definitions for objects of type PropForm.

Now, if the user types in a string that represents a propositional formula, unique readability implies that this string can be parsed as a propositional formula in only one way. Enderton notes that in fact the proof of unique readability gives hints as to the right algorithm for parsing. Assuming that you globally keep track of the string and a pointer to the next character, the recursive algorithm looks something like this:

$Parseformula()$: (returns a formula)
  Read the first symbol, s
  If s is p_i or $\bot$
    Return $AtomicForm(s)$
  Else if s is "("
    Peek at the next symbol, t
    If t is $\neg$
      Read the $\neg$
      $\varphi = ParseFormula()$
      If the next symbol is ")"
        Return $UnaryForm(\neg, \varphi)$
      Else error
    Else
      $\varphi = ParseFormula()$
      Read next symbol, u
      If u is a binary connective
        $\psi = ParseFormula()$

```
            If the next symbol is ")"
               Return BinaryForm(u, φ, ψ)
            Else error
         Else error
      Else error
   Else error
```

Computer scientists can specify "languages" like *PROP* in a number of ways. On of the most common and flexible is to provide what is called a "context free grammar," a notion due to Noam Chomsky. Indeed, the Unix tool "yacc" ("yet another compiler-compiler") takes such a grammer as input, and returns code which parses the language generated by this grammar. Context-free grammars can be "ambiguous," which means that the corresponding inductively defined set is not freely generated. In that case, parsing a string means finding *any* suitable parse tree.

The goal of mechanizing logical reasoning has had a long and colorful history, which long predates the advent of computers. For example, in the 13th century a Franciscan monk named Ramon Lully constructed wheels that would assist people in reasoning about the God's glorious attributes, and in the 17th century Leibniz wrote of a grand, symbolic logical calculus that would reduce reason to calculation. In the 18th century Stanley Jevons constructed a "logical calculator" to implement some of George Boole's ideas, affectionately known as "Jevon's logical piano." Martin Gardner's book, *Logic Machines and Diagrams*, provides a very nice history of such efforts.

3.5 Semantics

Suppose I wanted to define a new connective, $p \star q$, to be read "p unless q," and designed to model such statements as "John is in class unless he slept late." A moment's reflection indicates that there is some abiguity here. For example, it is clear that the statement is true if John is in class, and he did not sleep late; but it is not clear whether we want to interpret the statement as true if John is in class, but also slept late. As a result, you might ask me to clarify what I *mean* by the word "unless." And the clearest answer I could provide is to tell you under what conditions an "unless" statement should be considered true — essentially providing you with a truth-table for the connective.

More generally, we can say that the "meaning" of a complex propositional formula is determined by the conditions under which we accept it as true; and in the caes of propositional logic, the only relevant "conditions" are the truth-values of the atomic statements.

Below I will depart from van Dalen's presentation only slightly, mainly in the organizational definitions. Where van Dalen talks about extending a valuation defined on atomic formulas to another valuation, I prefer to think of starting with a "truth assignment" and extending it to an valuation on all of $PROP$.

Definition 3.5.1 *A truth assignment is simply a function from propositional variables to* $\{0, 1\}$.

For example, we can define a truth assignment v such that $v(p_0) = 1$, $v(p_5) = 1$, and $v(p_i) = 0$ for every other value of i. Here, for convenience (and to follow van Dalen), I am identifying 0 with "false" and 1 with "true."

Theorem 3.5.2 *Let v be a truth assignment. Then there is a unique function $\bar{v}$ from PROP to $\{0, 1\}$ satisfying the following conditions:*

- $\bar{v}(p_i) = v(p_i)$ *for every* i
- $\bar{v}(\bot) = 0$
- $\bar{v}(\varphi \wedge \psi) = \min(\bar{v}(\varphi), \bar{v}(\psi))$
- $\bar{v}(\varphi \vee \psi) = \max(\bar{v}(\varphi), \bar{v}(\psi))$
- $\bar{v}(\varphi \to \psi) = \begin{cases} 1 & \textit{if } \bar{v}(\varphi) = 0 \textit{ or } \bar{v}(\psi) = 1 \\ 0 & \textit{otherwise} \end{cases}$
- $\bar{v}(\varphi \leftrightarrow \psi) = \begin{cases} 1 & \textit{if } \bar{v}(\varphi) = \bar{v}(\psi) \\ 0 & \textit{otherwise} \end{cases}$

Proof. This is simply an application of the recursion theorem. □

Note that, for example, the definition for $\bar{v}(\varphi \wedge \psi)$ means that $\bar{v}(\varphi \wedge \psi) = 1$ iff $\bar{v}(\varphi) = 1$ and $\bar{v}(\psi) = 1$. Since $\bar{v}(\theta)$ is supposed to represent the "truth value" of θ, the preceeding amounts to saying that $\varphi \wedge \psi$ is true (under the truth assignment v) iff φ is true and ψ is true. In other words, the equation explains the meaning of the symbol $\wedge$ in the formal language, by making it correspond to the notion of "and" in the metalanguage. A similar thing can be said for each of the other connectives. The first clear enunciation of this way of "defining" truth is due to Tarski, and is sometimes referred to as "Tarskian semantics."

Like van Dalen, we will use the notation $[\![\varphi]\!]_v$ for $\bar{v}(\varphi)$. I will now follow van Dalen's development quite closely. Starting at the bottom of page 18, I will discuss:

- Lemma 1.2.3. This means that for a given formula, $[\![\varphi]\!]_v$ only depends on the (finitely many) values that v assigns to propositional variables occuring in φ.
- Definition 1.2.4. Notation and terminology varies somewhat in the literature, but in each group below all the entries express the same notion:

  1. "$[\![\varphi]\!]_v = 1$," "$\bar{v}(\varphi) = 1$," "$v \models \varphi$," "φ is true under assignment v," "φ is true under the interpretation v," "v satisfies φ"
  2. "$\models \varphi$, "φ is a tautology," "φ is logically valid," "φ is valid,"
  3. "$\Gamma \models \varphi$," "φ is a semantic consequence of Γ," "Γ logically implies φ."

  Note that the symbol "$\models$" is severely overloaded. Make sure you are clear on the differences between "$\models \varphi$", "$v \models \varphi$", and "$\Gamma \models \varphi$".

We now have the means to *prove* that a given formula is logically valid, or that a formula φ is a logical consequence of a set of formulas Γ. Here some examples; pay close attention to the way the arguments go, and the way the formal logical notions translate to informal notions in the metatheory, using Tarski's conditions.

1. $\models p_1 \wedge p_2 \to p_1$

   Proof. Let v be any truth assignment; we need to show $[\![p_1 \wedge p_2 \to p_1]\!]_v = 1$. If $[\![p_1 \wedge p_2]\!]_v = 0$, we're done (why?). So we only need to show that if $[\![p_1 \wedge p_2]\!]_v = 1$, then $[\![p_1]\!] = 1$.

   So, suppose $[\![p_1 \wedge p_2]\!]_v = 1$. By the definition of $\bar{v}$, this means that $[\![p_1]\!]_v = 1$ and $[\![p_2]\!]_v = 1$. In particular, $[\![p_1]\!]_v = 1$ we're done. $\square$

2. $\{p_1, p_1 \to p_2\} \models p_2$

   Proof. Let v be any truth assignment, and suppose $[\![p_1]\!] = 1$ and $[\![p_1 \to p_2]\!] = 1$. We need to show that $[\![p_2]\!]_v = 1$.

   By the definition of $\bar{v}$, the second assumption implies that if $[\![p_1]\!]_v = 1$, then $[\![p_2]\!]_v = 1$. The first assumption then tells us that $[\![p_2]\!] = 1$, as needed. $\square$

3. $\{p_1 \wedge p_2, p_1 \vee p_4, \neg p_3\} \not\models \neg p_3 \rightarrow p_4$.

   Proof. If we let $v(p_1) = 1$, $v(p_2) = 1$, $v(p_3) = 0$, and $v(p_4) = 0$, then all the formulas on the left come out true, while $p_3 \rightarrow p_4$ comes out false. □

4. For any formulas φ and ψ, $\models \varphi \rightarrow \psi$ if and only if $\{\varphi\} \models \psi$

   Proof. For the forwards direction: suppose $\models \varphi \rightarrow \psi$, and v is a truth assignment such that $[\![\varphi]\!]_v = 1$. Since $\varphi \rightarrow \psi$ is valid, we have $[\![\varphi \rightarrow \psi]\!]_v = 1$, and so $[\![\psi]\!] = 1$ as in example 2.

   For the backwards direction: suppose $\{\varphi\} \models \psi$, and let v be any truth assignent. We need to show that $[\![\varphi \rightarrow \psi]\!]_v = 1$; in other words, we need to show that if $[\![\varphi]\!]_v = 1$, then $[\![\psi]\!]_v = 1$. But this follows directly from the assumption that $\{\varphi\} \models \psi$. □

Coming back to van Dalen, I will discuss

- Definition 1.2.5. (the definition of $\varphi[\psi/p_i]$)
- Theorem 1.2.6, the substitution theorem.

Van Dalen's proof of Theorem 1.2.6 is a little complicated. I would prefer instead to use the following:

Lemma 3.5.3 *If ψ, θ, and φ are any formulas, v is a truth assignment, and $[\![\psi]\!]_v = [\![\theta]\!]_v$, then $[\![\varphi[\psi/p_i]]\!]_v = [\![\varphi[\theta/p_i]]\!]_v$.*

The proof is a routine induction on φ, and the substitution theorem follows easily from this.

3.6 The algebraic point of view

From a mathematical point of view, the reliance on syntactic notions may seem kludgy. While mathematicians study numbers, functions, geometric spaces, and other mathematical objects, some might object to admitting "strings of symbols" as bona-fide citizens of the mathematical universe. In this section I will sketch a more algebraic approach to logic, covered in pages 21–23 of van Dalen.

Definition 3.6.1 *Two propositional formulas φ and ψ are said to be* equivalent*, written $\varphi \equiv \psi$, if $\varphi \leftrightarrow \psi$ is valid.*

Notice that this is the same as saying that $\{\varphi\} \models \psi$ and $\{\psi\} \models \varphi$. Van Dalen uses $\approx$ instead of $\equiv$.

Proposition 3.6.2 *$\equiv$ is an equivalence relation. In other words, we have that for every φ, ψ, and θ*

1. *$\varphi \equiv \varphi$ (reflexivity)*
2. *If $\varphi \equiv \psi$ then $\psi \equiv \varphi$ (symmetry)*
3. *If $\varphi \equiv \psi$ and $\psi \equiv \theta$ then $\varphi \equiv \theta$ (transitivity)*

In general, an equivalence relation is a relation that "looks like" equality. A good example of an equivalence relation comes from "clock arithmetic," or equivalence modulo some number. For example, if you want to know what day of the week it will be 1,000 days from now, it is enough to count 6 days forward from today, since 7 divides $1000 - 6$. Another way of saying this is, up to multiples of 7, 6 and 1000 are the same.

Now, we can bunch together all the Mondays that ever were (a scary thought!) and just call them the same "weekday"; and then we can reasonably say that adding 1000 weekdays to Monday yields Saturday. This act of bunching like things together is known to mathematicians as "modding out by an equivalence relation."

Now we can do the same thing in logic, and talk about formulas "up to logical equivalence." Formally, for each formula φ, we let $[\varphi]$ denote the set of all formulas that are logically equivalent to φ. In other words, we think of $[\varphi]$ as being a new mathematical object, called "the equivalence class of φ." We can then use the logical operations on formulas to define operations on equivalence classes: for example, we can define a new operation $\bar{\wedge}$ on equivalence classes, by

$$[\varphi]\bar{\wedge}[\psi] = [\varphi \wedge \psi].$$

There is something sneaky going on here: I am defining an operation on equivalence classes by referring to representative members, so I need to show that the definition doesn't depend on which representative I choose. In other words, I need to show that if $[\varphi] = [\varphi']$ and $[\psi] = [\psi']$, then $[\varphi \wedge \psi] = [\varphi' \wedge \psi']$. And this really amounts to showing that if $\varphi \equiv \varphi'$ and $\psi \equiv \psi'$, then $\varphi \wedge \psi \equiv \varphi' \wedge \psi'$. I will have you do this on an upcoming homework assignment, which will also contain some problems designed to get you used to thinking about the act of modding out by an equivalence relation. We will need to do this when we prove the completeness theorem for first-order logic. The approach can be used more generally to give the completeness and compactness theorems very "natural" algebraic proofs.

In practice, mathematicians usually use the same symbol, $\wedge$, to denote the derived operation on equivalence classes. The upshot is that we have a new mathematical structure, consisting of the equivalence classes of propositional formulas, and operations $\wedge$, $\vee$, $\rightarrow$, and so on, on this structures. Such a structure is known as a *Boolean algebra.* In fact, one way to define a Boolean algebra is to say that it is a structure with the operations above, such that every propositional equivalence becomes an equality in the algebra. (For example: a and b are any elements of the algebra, then $\neg(a \vee b)$ is equal to $(\neg a \wedge \neg b)$.)

You should be familiar and comfortable with the equivalences given by van Dalen in Theorem 1.3.1 on page 20, which can be viewed providing propositional equivalences, or equations that must hold in every Boolean algebra. Using the substitution theorem we now have a means to "calculate" with formulas. For example, we have

$$\begin{aligned} p \wedge (p \rightarrow q) &\equiv p \wedge (\neg p \vee q) \quad \text{def of } \rightarrow \\ &\equiv (p \wedge \neg p) \vee (p \wedge q) \quad \text{distributivity} \\ &\equiv \bot \vee (p \wedge q) \\ &\equiv p \wedge q \end{aligned}$$

Also,

$$\begin{aligned} (p \rightarrow q) \vee (q \rightarrow r) &\equiv (\neg p \vee q) \vee (\neg q \vee r) \\ &\equiv (q \vee \neg q) \vee (\neg p \vee r) \quad \text{distributivity, commutativity} \\ &\equiv \top \vee (\neg p \vee r) \\ &\equiv \top, \end{aligned}$$

so the first line is the tautology. (This is probably the kind of rational calculus that Leibniz had in mind.) There are more examples on page 23 of van Dalen.

Incidentally, $\top$ and $\bot$ are sometimes called "top" and "bottom," instead of "true" and "false." This makes sense if you imagine the elements of the boolean algebra laid out in front of you, so that an element p is "below" an element q whenever $p \rightarrow q$ is true.

3.7 Complete sets of connectives

Many of you have already noticed that there is some redundancy in our choice of connectives. For example, consider the following equivalences:

$$\begin{aligned}
\varphi \to \psi &\equiv \neg\varphi \vee \psi \\
\varphi \leftrightarrow \psi &\equiv (\varphi \to \psi) \wedge (\psi \to \varphi) \\
\varphi \oplus \psi &\equiv (\varphi \vee \psi) \wedge \neg(\varphi \wedge \psi) \\
&\equiv \neg(\varphi \leftrightarrow \psi) \\
(\varphi \wedge \psi) &\equiv \neg(\neg\varphi \vee \neg\psi)
\end{aligned}$$

Reading these as *definitions*, it becomes clear that we can get away with as little as $\neg$ and $\vee$ as our basic connectives, and take all the other connectives to be defined in terms of these. (We can define $\perp$ as $p_0 \wedge \neg p_0$, though this has the drawback that p_0 now artificially appears in any formula involving $\perp$. If you don't like this artifact, simply keep $\perp$ as a propositional constant.)

But how do we know that *any* connective we might dream up, with any number of arguments, can be represented with $\vee$ and $\neg$? This is the force of Theorem 1.3.6 on page 24 of van Dalen. The following is just a slight variation of van Dalen's presentation.

Say f is a "k-ary truth function" if f is a k-ary function from $\{0, 1\}$ to $\{0, 1\}$; in other words, $f(x_1, \ldots, x_k)$ takes k values of true/false, and returns a value of true/false. You can think of f is being a "truth table" for a k-ary connective. Let φ be a formula with at most the propositional variables $p_1, \ldots, p_k$. Say that φ *represents* f if the following holds: for every truth assignment v,

$$f(v(p_1), \ldots, v(p_k)) = [\![\varphi]\!]_v.$$

So computing $f(x_1, \ldots, x_k)$ is equivalent to evaluating $[\![\varphi]\!]_v$, where v is chosen so that for each i, $v(p_i) = x_i$. In other words, the truth table of φ is just f.

Theorem 3.7.1 *Let $k \leq 1$. Then every k-ary truth function is represented by a formula using only the connectives $\vee$ and $\neg$.*

Proof. By induction on k. When $k = 1$, there are four unary truth functions: the constant 0, the constant 1, the identify function, and the function $f(x) = 1 - x$. These are represented by $\neg(p_1 \vee \neg p_1)$, $p_1 \vee \neg p_1$, p_1, and $\neg p_1$, respectively.

In the induction step, let $f(x_1, \ldots, x_{k+1})$ be a $k+1$-ary truth function. Define

$$g_0(x_1, \ldots, x_k) = f(x_1, \ldots, x_k, 0)$$

and

$$g_1(x_1, \ldots, x_k) = f(x_1, \ldots, x_k, 1).$$

By the induction hypothesis, there are formulas ψ_0 and ψ_1 representing g_0 and g_1, respectively. Let φ be the formula

$$(\neg p_{k+1} \wedge \psi_0) \vee (p_{k+1} \wedge \psi_1).$$

I have cheated a little, by using $\wedge$; so go back and rewrite this formula, replacing $A \wedge B$ by $\neg(\neg A \vee \neg B)$. It is not difficult to show (and I will do this in more detail in class) that φ represents f. □

If one interprets 0-ary functions as constants, then there are two 0-ary functions, 0 and 1. The thoerem above is false for $k = 0$, unless we allow either $\bot$ or $\top$. If we allow, say, $\bot$, then 0 is represented by $\bot$, and 1 is represented by $\neg\bot$.

Say that a set of connectives is *complete* if the conclusion of the theorem above holds, i.e. every truth function (of arity greater than 0) can be represented using those connectives. So the theorem above says, more concisely, that $\{\neg, \vee\}$ is a *complete* set of connectives.

Showing that a set of connectives is complete is now routine: just show that one can define $\neg$ and $\vee$. For example, $\{\neg, \wedge\}$ is a complete set of connectives, because $\varphi \wedge \psi$ is equivalent to $\neg(\neg\varphi \vee \neg\psi)$. For homework I will have you show that the Sheffer stroke, $|$, which represents "nand," is, by itself, complete.

Showing that a set of connectives is *not* complete poses more of a challenge: you have to find some clever way of showing that there is something that *cannot* be represented. For example:

Proposition 3.7.2 *$\{\leftrightarrow\}$ is not a complete set of connectives.*

Proof. Let us show that any formula involving only $\leftrightarrow$ and a single variable p_0 is equivalent to either p_0, or $\top$. This will imply that we can't represent $\neg p_0$.

Use induction on formulas. The base case is immediate. For the induction step, suppose φ is given by $(\theta \leftrightarrow \psi)$. By the induction hypothesis, each of φ and θ are equivalent to either p_0 or $\top$. So φ is equivalent to either

$(p_0 \leftrightarrow p_0)$, $(p_0 \leftrightarrow \top)$, $(\top \leftrightarrow p_0)$, or $(\top \leftrightarrow \top)$. But these are equivalent to $\top$, p_0, p_0, $\top$, respectively. $\square$

In fact, one can prove that $\{\bot, \top, \neg, \leftrightarrow, \oplus\}$ is not complete, even if one allows extra free variables in the representing formula. (I will provide hints to showing this on an upcoming homework assignment.)

3.8 Normal forms

The last section we saw that every formula φ can be expressed in an equivalent way as a formula φ' involving only $\vee$ and $\neg$. We can think of φ' as representing a "normal form" for φ. It is often useful to translate formulas to such "canonical" representatives.

In class I will discuss *conjunctive* and *disjunctive normal form*, as described on pages 25 and 26 of van Dalen. For example, if φ is the formula

$$(p_1 \vee p_2 \vee p_3) \wedge (\neg p_1 \vee \neg p_2 \vee p_4) \wedge (p_4 \vee p_5),$$

then φ is already in conjunctive normal form, and $\neg\varphi$ is equivalent to

$$(\neg p_1 \wedge \neg p_2 \wedge \neg p_3) \vee (p_1 \wedge p_2 \wedge \neg p_4) \vee (\neg p_4 \wedge \neg p_5),$$

which is in disjunctive normal form. Converting the latter to conjunctive normal form requires some work, but note that there is an algorithm implicit in the proof of Theorem 1.3.9. As another example, $p_1 \wedge p_2 \to p_3 \wedge p_4$ is equivalent to

$$(\neg p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee \neg p_2 \vee p_4)$$

in normal form.

I will also state the duality theorem on page 27, but I will leave it to you to look up the proof.

Chapter 4

Deduction for Propositional Logic

4.1 Overview

Suppose we are given a propositional formula φ. We now have a number of ways of demonstrating that this formula is a tautology:

- We can write out the entire truth table of φ
- We can use an informal argument, in the metatheory
- We can use algebraic methods, as described in the last chapter.

Let us consider each of these in turn. The first option can be tedious and inflexible. After all, the truth table of a formula with 17 variables has 2^{17} lines, and if this formula is of the form $\psi \vee \neg\psi$, the complicated check is a waste of time. The second option is much more flexible, but a lot more vague and difficult to mechanize. (Giving a formal definition of the phrase "informal argument" is almost a contradiction in terms.) The third option is more amenable to formalization, provided we are careful to list the starting axioms and algebraic rules of inference. But it does not really correspond closely to our informal arguments, and one might still worry about whether the axioms are rules are "complete," which is to say, sufficient to prove every tautology.

In this chapter I will present a formal deductive system (one of my favorites) due to Gerhard Gentzen, from the early 1930's. Thus far, in class, I have been trying to emphasize the "logical structure" of our informal arguments. For example, I have frequently pointed out that to prove "if A

then B," it suffices to assume A, and show that B necessarily follows. We will see these informal methods modelled in the proof rules, and this fact, to an extent, justifies calling the system "natural deduction." At the same time, our definition of the proof system will be rigorous and mathematical. This will allow us, on the one hand, to show that the system is sound and complete, which is to say that one can derive all and only the tautologies. On the other hand, the formality makes it amenable to mechanization. (In fact, it forms the basis for PVS, a computer-aided verification system for protocols, circuits, and algorithms.)

The advantage of a formal deductive system over the truth-table method will become clearer when we discuss predicate logic — where there isn't any reasonable analogue of the truth-table method.

Natural deduction is far from the only game in town. For example, for many applications, so-called sequent calculi are more convenient. In contrast to both of these, "Hilbert-style" calculi typically use many axioms, and fewer rules — sometimes even just one, modus ponens. Finally, when it comes to proof search, resolution and tableaux systems are also popular.

4.2 Natural deduction

In presenting natural deduction, I will follow the account in Section 1.4 of van Dalen very closely. Start by reading the very nice expository introduction on pages 30–32. I will review this in class, including the overall framework of proving statements from open hypotheses, and the specific rules for $\wedge$, $\rightarrow$, and $\bot$. Definition 1.4.2 in van Dalen introduces the notation that we will use: if Γ is a set of propositional formulas, and φ is a propositional formula, when I write $\Gamma \vdash \varphi$ or say "Γ proves φ," I mean that there is a derivation of the formula φ, such that all the open hypotheses are in Γ. By $\vdash \varphi$ I mean that φ is provable outright, that is, provable from an empty set of hypotheses. I will often use the abbreviations Γ, ψ for $\Gamma \cup \{\psi\}$ and Γ, Δ for $\Gamma \cup \Delta$.

To complement the examples in van Dalen, I will go over the following examples in class:

- $\varphi \wedge \psi \rightarrow \varphi$
- $\{\varphi \rightarrow \psi, \psi \rightarrow \theta\} \vdash \varphi \rightarrow \theta$
- $\{\varphi \rightarrow (\psi \rightarrow \theta)\} \vdash \varphi \wedge \psi \rightarrow \theta$

I have just given you an informal definition of the set of "derivations". You should be aware, however, that one can make the definition more precise. In fact— you guessed it— we can define the set of derivations inductively. First, we need a suitable set for the universe U. Let us suppose that given a finite set of formulas Γ and a formula φ, we have some way of expressing that φ follows from Γ, e.g. as a string of symbols, $\Gamma \Rightarrow \varphi$. We can then take U to be the set of finite trees that are labelled with such assertions, and assume that we have operations $\mathit{tree}_k(\Gamma \Rightarrow \varphi, t_1, \ldots, t_k)$, that take the assertion $\Gamma \Rightarrow \varphi$ and trees t_1 to t_k in U, and return a new tree with the assertion at the bottom subtrees t_1 to t_k. (If $k = 0$, this function just returns a 1-node tree.) Then the set of proofs is defined inductively, as the smallest subset of U satisfying the following:

- For every Γ and φ, $\mathit{tree}_0(\Gamma \cup \{\varphi\} \Rightarrow \varphi)$ is a proof.
- If d is a proof with $\Gamma, \varphi \Rightarrow \psi$ as the bottom-most label, then $\mathit{tree}_1(\Gamma \Rightarrow \varphi \to \psi, d)$ is a proof.
- If d and e are proofs with bottom-most labels $\Gamma \Rightarrow \varphi$ and $\Delta \Rightarrow \varphi \to \psi$ respectively, then $\mathit{tree}_2(\Gamma \cup \Delta \Rightarrow \psi, d, e)$ is a proof.
- If d and e are proofs with bottom-most labels $\Gamma \Rightarrow \varphi$ and $\Delta \Rightarrow \psi$ respectively, then $\mathit{tree}_2(\Gamma \cup \Delta \Rightarrow \varphi \wedge \psi, d, e)$ is a proof.
- If d is a proof with bottom-most label $\Gamma \Rightarrow \varphi \wedge \psi$ then $\mathit{tree}_1(\Gamma \Rightarrow \varphi, d)$ and $\mathit{tree}_1(\Gamma \Rightarrow \psi, d)$ are proofs.
- If d is a proof with $\Gamma \Rightarrow \perp$ as bottom-most element, then $\mathit{tree}_1(\Gamma \Rightarrow \varphi, d)$ is a proof, for every formula φ.
- If d is a proof with $\Gamma, \neg\varphi \Rightarrow \perp$ as bottom-most element, so is $\mathit{tree}_1(\Gamma \Rightarrow \varphi, d)$.

4.3 Proof by contradiction

If you are trying to find a proof of φ from Γ, in general, you can work forwards (generating conclusions from Γ) or backwards (looking for sufficient conditions to conclude φ). In general, however, the overall "logical form" of the hypotheses and desired conclusion usually dictates how you should proceed.

Sometimes, however, you will find yourselves simply stuck. Fortunately, you have one more trick up your sleeve: proof by contradiction, corresponding to the rule RAA.

Proofs using RAA can be a little bit tricker. From an intuitionistic point of view, RAA is not a valid form of inference. In fact, intuitionistic (constructive) first-order logic is exactly what you get by deleting this rule from the natural deduction calculus. The provides us with a nice characterization of the statements φ which can only be proved using RAA: they are exactly those statements which are classically valid but not intuitionistically valid.

I will go over the following two examples in class.

1. Prove $\varphi \to \psi$ from $\neg(\varphi \wedge \neg\psi)$.

$$\dfrac{\dfrac{\dfrac{\neg(\varphi \wedge \psi) \qquad \dfrac{[\varphi]_2 \qquad [\psi]_1}{\varphi \wedge \neg\psi}}{\bot}}{\psi}\,1 \quad (RAA)}{\varphi \to \psi}\,2$$

2. Classically, φ is equivalent to $\neg\neg\varphi$. The forwards direction holds intuitionistically (make sure you can find a derivation!), but the converse direction requires RAA:

$$\dfrac{\dfrac{\dfrac{[\neg\neg\varphi]_2 \qquad [\neg\varphi]_1}{\bot}}{\phi}\,1 \quad (RAA)}{\neg\neg\varphi \to \varphi}\,2$$

4.4 Soundness

From a pedagogical point of view, I have found it convenient to describe the semantics of propositional logic before presenting a system of deduction. After all, the semantics does seem to capture the intuitive notions well, and once it is in place we can use it to evaluate our deductive system objectively. In other words, if our proof system fails to be both sound and complete, we know that we have to either "repair" some of the rules or look for more.

Historically, however, formal deductive systems appeared before semantic issues were clearly articulated. Perhaps the first account of what modern logicians would consider a formal system appears in Frege's "Begriffschrift" ("Concept writing") of 1879, albeit with a quirky two-dimensional diagrammatic notation for formulas. In 1885 Peirce arguably came close to having a formal system for what is, essentially, first-order logic. Peano also came close to having a formal proof system for arithmetic in 1889: he presented a formal symbolic language and axioms, but, oddly enough, did not specify the rules of inference. A clear formal system for something that we would now classify as a kind of higher-order logic appears in Russell and Whitehead's *Principia* in 1910; and first-order logic finally evolved into its modern form through work of Hilbert, Bernays, Ackermann, Skolem, and others around 1920.

All of these systems were designed to model the kind of logical and mathematical reasoning that we are concerned with. But it is important to note that they were developed from "the inside" (designed to capture the relevant forms of reasoning) instead of from "the outside" (designed to conform to a well-defined semantics). The first rigorous account of a semantics for the propositional calculus, together with completeness proofs for specific deductive systems, appeared independently in Bernays' *Habilitationsschrift* of 1918, and independently in a paper by Post, published in 1921. Hilbert and Bernays clearly articulated the problem of proving the completeness of a calculus for the first-order logic in the 1920's, and this problem was solved by Gödel in his dissertation in 1929.

In this section I will show that our deductive system is sound, and in the next I will show that it is complete. Of the two directions, soundness is the easier one to prove. Indeed, it is a straightforward consequence of the fact that the deductive rules we have chosen "agree" with the semantics.

Theorem 4.4.1 *(soundness) Let Γ be any set of propositional formulas, and let φ be any formula. If $\Gamma \vdash \varphi$ then $\Gamma \models \varphi$.*

Proof. Use induction on derivations (i.e. an induction corresponding to the inductive definition of derivations described above). I will go over the case where the bottom inference of the proof as an $\rightarrow$ introduction:

$$\frac{\Gamma, \varphi \Rightarrow \psi}{\Gamma, \Rightarrow \varphi \rightarrow \psi}$$

The full proof, in all its glory, is on pages 40–42 of van Dalen. □

Why is having a semantics useful? For one thing, it gives us a means to show that a certain formula *can't* be derived from a certain set of hypotheses. In general, it is easier to show that something *can* be done (just show how to do it), whereas proving "negative" results usually requires much more ingenuity. In the first-order case, the semantics provides a formal way of showing that one can't prove, say, Euclid's fifth postulate from the other four; namely, one shows that there is a "model" of the first four that fails to satisfy the fifth. In the same way, the example from the previous section shows that

$$\{p_1 \wedge p_2, p_1 \vee p_4, \neg p_3\} \nvdash \neg p_3 \rightarrow p_4.$$

After all, if $p_3 \rightarrow p_4$ were provable from the hypotheses, by soundness it would be a logical consequence of them. But the truth assignment in the previous section shows that this is not the case.

4.5 Completeness

We are now ready to face the harder direction of the equivalence, and show that $\Gamma \models \varphi$ implies $\Gamma \vdash \varphi$. To do so, we will be using some very powerful methods— powerful enough so that, with the right modifications, the same kind of proof will work in the first-order case. One can even generalize this kind of argument to "infinitary" languages and deductive systems. For propositional logic, the are proof that are more simple and direct, and I will discuss some of them below. So using these techniques in this setting is like using a sledgehammer to drive in a thumbtack, or, as the Germans sometimes say (I'm told), using cannons to shoot sparrows. But the power and generality of the abstract methods indicates that they embody, in a sense, the "right" ideas.

The first step is to reformulate the problem in a slightly different way. I will discuss:

- van Dalen's definition 1.5.2, which says that a set of formulas is *consistent* if it doesn't prove $\perp$; and
- Lemma 1.5.3, which gives three equivalent characterizations of consistency.

The notion of consistency is central in logic. When Cantor and Dedekind introduced abstract and transfinite methods to mathematics in the 19th century, they faced criticism that their methods strayed too far from any "concrete" content. Cantor responded forcefully, asserting that mathematicians should be free to use whatever methods are fruitful, so long as these methods are consistent. In the early 20th century, Hilbert turned this into a program for the formal justification of mathematics: to justify classical methods, model these methods with formal systems and use ordinary, combinatorial mathematics to prove the consistency of these formal systems. Though Gödel's incompleteness theorems put severe restrictions on the kind of consistency proofs we can hope for, the program brought into currency two of my favorite German words: *widerspruchsfreiheit* (consistency, or, literally, freedom-from-speaking-against) and *Widerspruchsfreiheitbeweis* (consistency proof).

Let Γ be a set of propositional formulas. I will go over Lemma 1.5.4, which shows that Γ is consistent if there is a truth assignment which makes every formula in Γ true. In other words, Γ is consistent if there is a truth assignment v such that $[\![\psi]\!]_v = 1$ for every formula ψ in Γ. If we are willing to overload the symbol $\models$ even more, we can express this relationship between v and Γ by writing $v \models \Gamma$, and saying "v satisfies Γ."

Consider now Lemma 1.5.5. It is easy to see that the converse directions also hold, so that for every set of formulas Γ and formula φ, we have that Γ, φ is consistent iff Γ doesn't prove $\neg\varphi$, and $\Gamma, \neg\varphi$ is consistent iff Γ doesn't prove φ.

Now consider the statement of the completeness theorem:

> For every set of formulas Γ and formula φ, if $\Gamma \models \varphi$ then $\Gamma \vdash \varphi$.

By taking the contrapositive, this is equivalent to saying

> For every set of formulas Γ and formula φ, if $\Gamma \nvdash \varphi$, then $\Gamma \not\models \varphi$.

By the equivalence above and our semantic definitions, this is amounts to saying

> For every set of formulas Γ and formula φ, if $\Gamma \cup \{\neg\varphi\}$ is consistent, then there is a truth assignment v satisfying $\Gamma \cup \{\neg\varphi\}$.

And so, to prove the completeness theorem, it suffices to prove

> For every set of formulas Γ, if Γ is consistent, then there is a truth assignment v satisfying Γ.

In fact, taking φ to be $\neg\bot$, the last two statements are actually equivalent.

Similarly, soundness is equivalent to saying

> For every set of formulas Γ, if there is a truth assignment v satisfying Γ, then Γ is consistent.

You should review the two ways of stating soundness and completeness until the equivalence seems natural to you (even though showing the equivalence of two "if ... then" statements can be confusing).

The rest of this section (and the corresponding section in van Dalen) is concerned with proving the completeness theorem, in the revised form.

Given a consistent set of propositional formulas Γ, we need to show that there is a truth assignment that satisfies it. And where can we find such a truth assignment? Here is where things get clever: we will extract it from Γ itself. More precisely:

1. First we will extend Γ to a bigger set, Γ', which is "maximally consistent." In other words, Γ' is consistent, but it is so full that you can't add a single formula without making it inconsistent.

2. Then we will show that Γ' has so much information, that it "looks like" a truth valuation.

3. The fact that Γ' looks like a truth valuation will enable us to "read off" a suitable truth assignment.

Now for the details.

Definition 4.5.1 *A set of formulas Γ is said to be* maximally consistent *if and only if*

1. Γ is consistent, and

2. If $\Gamma' \supsetneq \Gamma$, then Γ' is inconsistent.

This is Definition 1.5.6 in van Dalen. Note that the second clause is equivalent to saying that whenever $\psi \notin \Gamma$, $\Gamma \cup \psi$ is inconsistent.

It is not immediately clear that there are *any* maximally consistent sets of formulas. But suppose v is a truth assignment, and let $\Gamma = \{\varphi \mid [\![\varphi]\!]_v = 1\}$. A moment's reflection shows that Γ is maximally consistent: if $\psi \notin \Gamma$, then $[\![\psi]\!]_v = 0$, so $[\![\neg\psi]\!] = 1$, $\neg\psi \in \Gamma$, and $\Gamma \cup \{\psi\}$ is inconsistent. Soon we will see that every maximally consistent set is of this form; in other words, for every maximally consistent set Γ, there is a truth assignment v such that $\Gamma = \{\varphi \mid [\![\varphi]\!]_v = 1\}$.

Our first task is to show that every consistent set Γ of formulas is included in a maximally consistent set Γ^*. This is Lemma 1.5.7 in van Dalen. (Van Dalen mentions that, using Zorn's lemma, this part can be generalized to languages of higher cardinality.)

Next, we will prove that maximally consistent sets have some nice properties.

Lemma 4.5.2 *Suppose Γ is maximally consistent. Then the following hold:*

1. Γ is deductively closed. In other words, if $\Gamma \vdash \varphi$, then φ is already in Γ.

2. *For every formula φ, φ is in Γ if and only if $\neg\varphi$ is not in Γ.*

3. *For every pair of formulas φ and ψ, $\varphi \to \psi$ is in Γ if and only if either φ is not in Γ, or ψ is in Γ.*

4. *For every pair of formulas φ and ψ, $\varphi \wedge \psi$ is in Γ if and only if φ is in Γ and ψ is in Γ.*

Proof. Clause 1: suppose $\Gamma \vdash \varphi$. Since Γ is consistent, $\Gamma \cup \{\varphi\}$ is consistent. Since Γ is maximally consistent, $\varphi \in \Gamma$. (Incidentally, logicians sometimes express the fact that Γ is deductively closed by saying that Γ is a "theory.")

Clause 2: for the forwards direction, suppose $\varphi \in \Gamma$. Since Γ is consistent, $\neg\varphi \notin \Gamma$. Conversely, suppose $\neg\varphi \notin \Gamma$. Since Γ is maximally consistent, $\Gamma \cup \neg\varphi$ is inconsistent. By Lemma 1.5.5, $\Gamma \vdash \varphi$. By clause 1, $\varphi \in \Gamma$.

Clause 3: Suppose $\varphi \to \psi \in \Gamma$. If φ is in Γ, then $\Gamma \vdash \psi$ and $\psi \in \Gamma$. So, either $\varphi \notin \Gamma$ or $\psi \in \Gamma$. Conversely, suppose either $\varphi \notin \Gamma$ or $\psi \in \Gamma$. In the first case, by maximality, $\Gamma \cup \{\varphi\}$, is inconsistent; hence $\Gamma \vdash \neg\varphi$, so $\Gamma \vdash \varphi \to \psi$, and $\varphi \to \psi \in \Gamma$. In the second case, $\Gamma \vdash \psi$ and so $\Gamma \vdash \varphi \to \psi$, so $\varphi \to \psi \in \Gamma$. Either way, $\varphi \to \psi \in \Gamma$.

Clause 4: Suppose $\varphi \wedge \psi \in \Gamma$. Then $\Gamma \vdash \varphi$ and $\Gamma \vdash \psi$, and hence, by clause 1, $\varphi \in \Gamma$ and $\psi \in \Gamma$. Conversely, suppose $\varphi \in \Gamma$ and $\psi \in \Gamma$. Then $\Gamma \vdash \varphi \wedge \psi$, and hence $\varphi \wedge \psi$ is in Γ. □

This amounts to Lemma 1.5.9 and Corollary 1.5.10 in van Dalen, plus a short argument for clause 4.

Finally, we will read off a truth assignment. Given a maximally consistent set Γ, define a truth assignment v by

$$v(p_i) = \begin{cases} 1 & \text{if } p_i \in \Gamma \\ 0 & \text{otherwise} \end{cases}$$

Inductively, using the facts in the last paragraph, we can show that for every formula φ, $[\![\varphi]\!]_v = 1$ if and only if $\varphi \in \Gamma$. In van Dalen, all this is contained in the proof of Lemma 1.5.11.

Putting it all together, we can summarize the proof of the completeness theorem (stated in its original form) as follows.

Theorem 4.5.3 *For any set of formulas Γ and formula φ, if $\Gamma \models \varphi$, then $\Gamma \vdash \varphi$.*

Proof (overview). Let Γ be any set of formulas, and φ any formula. Suppose Γ doesn't prove φ. Then $\Gamma \cup \{\neg\varphi\}$ is consistent. Extend this to a maximally consistent set Γ' containing $\Gamma \cup \{\neg\varphi\}$. Read off a truth asignment v satisfying Γ'. In particular, v satisfies $\Gamma \cup \{\neg\varphi\}$, so v is a truth assignment satisfying Γ but not φ. This means that Γ does not logically imply φ. □

Make sure you are comfortable with this proof. On an exam I may ask you to sketch the entire argument, or to prove any of the particular details that are needed along the way.

4.6 Compactness

The following is an immediate consequence of the completeness theorem.

Theorem 4.6.1 *(compactness) Suppose Γ is a set of propositional formulas, and every finite subset of Γ is satisfiable. Then Γ is satisfiable.*

Proof. Suppose Γ is not satisfiable (which is the same as saying $\Gamma \models \bot$). By the completeness theorem, Γ is inconsistent (which is to say, $\Gamma \vdash \bot$). Consider a proof of $\bot$ from hypotheses in Γ. In this proof only finitely many hypotheses are used; call the set of these hypotheses Γ'. Then Γ' is a finite subset of Γ, and Γ' is inconsistent. By soundness, Γ' is not satisfiable. □

Now, the compactness theorem is much more interesting in the first-order setting. Indeed, it is probably the most widely used tool in model theory. But even in the propositional setting, compactness has some interesting consequences.

Here is one nice example. Say a set of "tiles" is a set of oriented squares, with each edge painted a certain color. Given a finite set of tiles A, an $n \times n$ *tiling* from A is an $n \times n$ square made up of copies of tiles in A, such that adjacent edges have the same color. A *tiling of the plane from A* is an arrangement of such tiles that is infinite in both directions; you can think of this as a function which assigns to every pair of integer coordinates i, j a tile $f(i, j)$, subject to the condition that adjacent edges are compatible.

Theorem 4.6.2 *Suppose A is a set of tiles, and for every n there is an $n \times n$ tiling from A. Then there is an infinite tiling of the plane.*

Proof. Label the tiles in A with the numbers $1, \ldots, l$. Intuitively, we will use variables $p_{i,j,k}$ to represent the assertion "tile k is in position i, j." Let Γ be the following set of formulas:

- $p_{i,j,1} \vee p_{i,j,2} \vee \ldots p_{i,j,l}$, for each i and j. (These assert that every position has a tile in it.)

- $\neg(p_{i,j,k} \wedge p_{i,j,k'})$ for every i, j, k, and k', with $k \neq k'$. (This says that at most on tile is in any position.)

- $\neg(p_{i,j,k} \wedge p_{i+1,j,k'})$ for every i, j, and pairs of tiles k and k' such that the right edge of k does not match the left edge of k'. (This asserts that no two incompatible tiles are next to each other horizontally.)

- $\neg(p_{i,j,k} \wedge p_{i,j+1,k'})$ for every i, j, and pairs of tiles k and k' such that the right edge of k does not match the left edge of k'. (This asserts that no two incompatible tiles are next to each other vertically.)

By hypothesis, every finite subset Γ' of Γ is satisfiable: just read off the truth assignment from an $n \times n$ tiling from A big enough to cover all the positions mentioned in Γ'. By compactness, Γ is satisfiable. But any truth assignment that satisfies Γ determines a tiling of the plane from A.

4.7 The other connectives

I will discuss Section 1.6 of van Dalen, which provides rules for the other connectives, $\vee$, $\neg$, and $\leftrightarrow$. I find the elimination rule for $\vee$ to be particularly nice, since it represents the informal method of reasoning by cases. Here is one example of a proof using the $\vee$ rule:

$$\dfrac{\dfrac{\dfrac{[\varphi \wedge (\psi \vee \sigma)]_2}{\psi \vee \sigma} \qquad \dfrac{\dfrac{\dfrac{[\varphi \wedge (\psi \vee \sigma)]_2}{\varphi} \qquad [\psi]_1}{\varphi \wedge \psi}}{(\varphi \wedge \psi) \vee (\varphi \wedge \sigma)} \qquad \dfrac{\dfrac{\dfrac{[\varphi \wedge (\psi \vee \sigma)]_2}{\varphi} \qquad [\sigma]_1}{\varphi \wedge \sigma}}{(\varphi \wedge \psi) \vee (\varphi \wedge \sigma)}}{(\varphi \wedge \psi) \vee (\varphi \wedge \sigma)}\,1}{(\varphi \wedge (\psi \vee \sigma)) \to ((\varphi \wedge \psi) \vee (\varphi \wedge \sigma))}\,2$$

Here is another one, showing that from $\varphi \vee \psi$ and $\neg\varphi$ one can prove ψ:

$$\dfrac{\varphi \vee \psi \qquad \dfrac{\dfrac{[\varphi]_1 \qquad \neg\varphi}{\perp}}{\psi} \qquad [\psi]_1}{\psi}\,1$$

Finally, the following examples shows that using RAA one can derive the law of the excluded middle:

$$\dfrac{\dfrac{\dfrac{\dfrac{[\neg(\varphi \vee \neg\varphi)]_2 \qquad \dfrac{[\varphi]_1}{\varphi \vee \neg\varphi}}{\bot}}{\neg\varphi}\,1}{\varphi \vee \neg\varphi} \qquad [\neg(\varphi \vee \neg\varphi)]_2}{\dfrac{\bot}{\varphi \vee \neg\varphi}\,2}$$

As van Dalen points out, you can take the new connectives and rules as basic, or you can take the new connectives to be abbreviations for the corresponding formulas with $\wedge$, $\rightarrow$, and $\bot$, and show that under these definitions the new rules given can be *derived* from the old ones.

If you follow the first approach, you have to worry about whether the new system is sound and complete. One option is to go back to the proofs of soundness and completeness, and add new clauses dealing with the new rules. (This is a pain in the neck.) Another option is as follows. Given any formula φ, let φ^* be the equivalent formula using only $\wedge$, $\rightarrow$, and $\bot$. Show that in the new system, you can prove $\varphi \leftrightarrow \varphi^*$; and show that all the new rules correspond, under the translation, to rules in the old system. Then you can show that

1. The new system proves φ from some hypotheses Γ if and only if the old system proves φ^* from their translations Γ^*.

2. Γ logically implies φ if and only if Γ^* logically implies φ^*.

Soundness and completeness for the new system then follow from soundness and completeness for the old system.

4.8 Computational issues

We have already observed the following

Theorem 4.8.1 *The set of propositional tautologies is decidable. In other words, there is a computer program which takes a propositional formula as input, and decides whether or not the formula is a tautology.*

Proof. Given φ, let the computer program loop through all the possible assignments of truth values to the variables which occur in φ, and compute $[\![\varphi]\!]_v$ for each such assignment. If the result is always 1, φ is a tautology. $\square$

The completeness theorem provides another proof, which is useful in other contexts.

Proof 2. Given φ, simultaneously do the following:

- Systematically look for a proof of φ.
- Systematically look for an assignment that makes φ false.

Completeness guarantees that sooner or later you'll find one or the other. In the first case, φ is a tautology. In the second case, it is not. $\square$

Why is this interesting? For one thing, the corresponding theorem for first-order logic is *false*, under some very minimal conditions on the underlying language. When it comes to logic, simple yes/no questions are often computationally unsolvable.

When they are computationally solvable, however, one can ask further how *hard* it is to solve them. We noted above that the simple "truth table" algorithm for testing to see if a formula is a tautology runs in exponential time. Those of you who are familiar with the notion of a nondeterministic Turing machine will recognize right away that the property of *not* being a tautology is in nondeterministic polynomial time: given a propositional formula, a nondeterministic Turing machine can just guess a falsifying assignment, and then check it. Cook's theorem states, moreover, that this problem is NP-complete. So the complement of this problem — that is, the set of formulas which *are* tautologies — is coNP-complete. The question as to whether there is an efficient (polynomial-time computable) algorithm for determining whether or not a formula is a tautology is just the famous open question, "P=NP?," in disguise.

One can ask the following weaker question: is there a proof system which has short proofs of every tautology? Here "short" means "polynomially bounded in the length of the input." If the term "proof system" is broadly construed, this is exactly equivalent to asking whether or not NP is equal to coNP. Most people in the field feel that the answer to this question is also "no," but we are a long way from proving it.

Nonetheless, we can try to evaluate *specific* proof systems and show that they do not have short proofs of every tautology. There *has* been some success in this endeavor — we have good lower bounds for resolution and tableau proof systems, for example. However, the problem of showing that natural deduction (or any of a number of systems that are equivalent from a complexity point of view) is one of the biggest open problems in proof complexity today.

4.9 Constructive completeness proofs

The completeness theorem tells us that if φ is a tautology, there is a natural deduction derivation of φ. One might hope that the proof of the completeness theorem might tell us how to *find* such a derivation. But in that sense, the proof is disappointingly nonconstructive: it just tells us that if there *were* no derivation of φ, there would have to be a truth assignment satisfying $\neg\varphi$ — which we are assuming is not the case.

More constructive proofs are available. Van Dalen sketches one very briefly at on page 47, using conjunctive normal form. (I will give you some problems on the homework designed to elaborate on this.) Another method is to just "simulate" the truth-table method internally. For example, suppose that φ contains three variables, p_1, p_2, p_3, and for every assignment to these variables, φ comes out true. Then one should be able to prove all of the following, mechanically:

$$\begin{array}{rcl} p_1 \wedge p_2 \wedge p_3 & \to & \varphi \\ p_1 \wedge p_2 \wedge \neg p_3 & \to & \varphi \\ p_1 \wedge \neg p_2 \wedge p_3 & \to & \varphi \\ p_1 \wedge \neg p_2 \wedge \neg p_3 & \to & \varphi \\ \neg p_1 \wedge p_2 \wedge p_3 & \to & \varphi \\ \neg p_1 \wedge p_2 \wedge \neg p_3 & \to & \varphi \\ \neg p_1 \wedge \neg p_2 \wedge p_3 & \to & \varphi \\ \neg p_1 \wedge \neg p_2 \wedge \neg p_3 & \to & \varphi \end{array}$$

Then one can put all these proofs together (with a proof that at least one of the eight cases must hold) and obtain a proof of φ.

Neither one of these methods will work for predicate logic. Also, even though we don't really expect to find, in general, derivations that are significantly shorter than the ones given by these procedures, for even simple examples these methods are *blatantly* inefficient.

There has been a lot of research on finding efficient derivations (and finding them efficiently). Typically, given a proof system, one describes some algorithm for searching for proofs of φ in a systematic way; one also shows that if the system *fails* to find a proof, then from this failure one can determine an assignment that makes φ false. As a result, work in the field of automated deduction often yields new proofs of the completeness theorem that go hand in hand with the search method being considered.

Chapter 5

Predicate Logic

5.1 Overview

We are now ready for the big time: predicate logic, also known as first-order logic. Propositional logic provides us with a framework to analyze the basic logical connectives like "and," "or," "not," and so on, but these logical terms will only take us so far. Mathematicians make typically statements about "all" or "some" elements of a given domain, and we would like to analyze the forms of reasoning that are valid for these logical notions.

Start with the discussion in Sections 2.1 and 2.2 of van Dalen. These are the kinds of informal mathematical statements that we would like to analyze:

1. "Every natural number is even or odd."
2. "There are infinitely many prime numbers."
3. "Between any two rational numbers, there is another rational."
4. "If a, b, and c are sets, a is a subset of b, and b is a subset of c, then a is a subset of c."
5. "Every continuous function attains a maximum value on $[0, 1]$."
6. (of groups) "Every element has an inverse," or "the group operation is associative."

Of course, we can use first order logic to model parts of nonmathematical discourse as well, such as the following:

1. "Every politician is corrupt."

2. "Not every politician is corrupt."

3. "Some politicians are not corrupt."

4. "Every politician that is corrupt gets caught."

5. "Some corrupt politicians do not get caught."

6. "Any given politician is either corrupt and is caught, or is not corrupt."

Every use of the word "all" or "some" ranges over some domain, which can be explicit or implicit. If I say "everything is greater than or equal to 0," I may be referring to the set natural numbers implicitly. If I say "every natural number is greater than or equal to 0," I am explicitly using the word "every" to range over natural numbers. To handle cases where the explicit information is absent, we will assume that for any given statement there is some implicit "universe" of objects under consideration.

As in the case of propositional logic, we would like to focus on the meaning of the logical elements of a given sentence, without worrying about the meaning of the basic terms. For example, we would like to say that examples 4 is logically equivalent to the negation of 5 (interpreting "some" to mean "at least one"), independent of how we interpret the terms "corrupt" or "politician," and independent of the world's political status at any given moment.

The basic setup is as follows. We will take a "first-order language" to be given by a collection of function and relation symbols. Intuitively, there is some universe or domain that we would like to with this language: this can be the set of natural numbers, the set of politicians, a set of colored blocks, or whatever. The function symbols are meant to denote functions on the domain of discourse, such as the successor function on the natural numbers, or the "biggest political opponent of" function on the set of politicians. The relation symbols are meant to denote relationships that may or may note hold of members of the universe: for example $Even$ or $<$ on the set of natural numbers, or "is more corrupt than" on the set of politicians.

Since we usually have some particular structure in mind, van Dalen finds it convenient to introduce the notion of a structure before discussing the syntax of first-order logic. It is important to note, however, that the underlying structures only form the motivation behind the choice of a particular set of symbols for the language, and the things that we may do with the them. In other words, the "structure" we wish to address has nothing to do with the syntax of the language, but rather, represents the intended semantics. If you write down "$\forall x \; Corrupt(x)$," you may wish to express the assertion that all politicians are corrupt, but, as far as syntax is concerned, this is just a string of symbols on the page.

In short, first-order logic is designed to represent logical statements about well-defined structures, with well-defined functions and relations on these structures. All the restrictions mentioned on the first day of class apply: there are no built in mechanisms to handle modal or temporal concepts, fuzzy concepts, default assumptions, and so on.

In many introductory logic courses, after presenting the syntax of first-order logic informally, one spends a lot of time looking at the way one can formalize particular statements in mathematics or everyday discourse. The more one does this, the more one realizes how flexible and expressive first-order logic really is. Regrettably, we will not have time to consider very many examples here; I am assuming that you have already been exposed to the basics of first-order logic and are comfortable "reading" and "writing" in this language. That way, we can focus on the formal analysis.

We will proceed much the same way we did in the development of propositional logic. In this section, I will discuss the syntax and semantics of predicate logic, and derive some basic properties. In the next section, we will consider a formal proof system, and prove that it is sound and complete. Although the details are more complex in the first-order case, we will be able to reuse and adapt the tools we have already developed.

5.2 Syntax

Definition 5.2.1 *A* similarity type *(sometimes called a "signature" or "language") consists of*

1. *Some relation symbols $r_1, r_2, \ldots, r_n$ of various arities (recall, "arity" means "# of arguments")*

2. *Some function symbols $f_1, f_2, \ldots, f_m$ of various arities*

3. *Some constant symbols $\bar{c}_i$ (where i is in some index set I)*

Relations are also sometimes called "predicates"; the two words are interhangeable in these notes, and van Dalen favors the latter.

Since only the arities are really important, van Dalen takes the similarity type to be just a list of arities for the relation and function symbols. In this section and the next, we will only encounter similarity types with finitely many relation and function symbols, but we will later need to have infinitely many constants. Note that we can think of propositional constants as relation symbols with arity 0 (they just stand for "true" or "false"), and constants as function symbols with arity 0 (they just stand for some object in the universe).

Now, given a similarity type, we will define a set of formulas that enable us to "say things" about structures having this similarity type. Our formulas will be strings involving the following symbols:

- Relation symbols $r_1, \ldots, r_n, =$
- Function symbols $f_1, \ldots, f_m$
- Constant symbols $\bar{c}_i$
- Variables $x_1, x_2, \ldots$
- Connectives $\wedge$, $\vee$, $\rightarrow$, $\neg$, $\leftrightarrow$, $\perp$, $\forall$, $\exists$
- Parentheses (,)

Most of the languages we discuss have a built in relation symbol, "=", to denote equality. Since equality is a central logical notion, we will handle it in a special way when we come to the semantics. When I want to exclude this symbol from the language, I will specify that we are using "first-order logic without equality."

Now we want to have "terms" that denote objects in the intended universe. For example, assuming we have constants 0 and 1, and functions $+$ and $\times$, we want to have terms like

$$((1 + (1 + (0 + 1))) \times (1 + 1)).$$

In our syntactic discussions we will use regular functional notation, so you should think of $1 + 1$ as an abbreviation for $+(1, 1)$.

Definition 5.2.2 *The set TERM of terms is defined inductively, as follows:*

- *every constant $\bar{c}_i$ is a term*
- *if $t_1, \ldots, t_k$ are all terms, and f is a function symbol of arity k, then $f(t_1, \ldots, t_k)$ is a term.*

So, for example, if f is a binary function symbol and g is a unary function symbol, and c and d are constants, $f(f(c, g(d)), g(c))$ is a term.

Terms are meant to denote objects in the intended domain of discourse. Formulas are meant to make assertions about this domain of discourse, and can involve terms.

Definition 5.2.3 *The set FORM of formulas is defined inductively, as follows:*

- $\perp$ *is a formula*
- *if* t_1 *and* t_2 *are terms, then* $t_1 = t_2$ *is a formula*
- *if* $t_1, \ldots, t_k$ *are terms and* r *is a relation symbol of arity* k *then* $r(t_1, \ldots, t_k)$ *is a formula*
- *if* φ *and* ψ *are formulas and* $\Box$ *is one of the connectives* $\wedge$, $\vee$, $\rightarrow$, *or* $\leftrightarrow$, *then* $(\varphi \Box \psi)$ *is a formula*
- *if* φ *is a formula then so are* $(\neg\varphi)$, $(\forall x\ \varphi)$, *and* $(\exists x\ \varphi)$.

We'll adopt the old conventions regarding dropping parentheses, with the addition that $\forall$ and $\exists$ are parsed first, with $\neg$. So, for example, $\neg\forall x\ \varphi \rightarrow \psi$ is shorthand for $((\neg(\forall x\ \varphi)) \rightarrow \psi)$. Intuitively, $\forall x\ \varphi$ means "for every x, φ is true of x" and $\exists x\ \varphi$ means "for some x, φ is true of x."

By way of motivation, in class I will write down first-order formulas to represent the following notions. Note that, e.g., in the first example I am referring to a first-order language with similarity type $\langle 0, 1, +, \times, < \rangle$, where the first two are constants, the second two are binary function symbols, and the last is a binary relation symbol.

1. On the natural numbers (language: 0, 1, $+$, $\times$, $<$)
   (a) x is even
   (b) no number is less than 0
2. On the real numbers (language: 0, 1, $+$, $\times$, $<$)
   (a) x^2 is greater than or equal to 0 (using $<$ and $=$)
   (b) x is the square root of y
   (c) negative numbers have no square root
   (d) between any two numbers there is another number
3. On people in this room (language: *Likes*, of arity 2)
   (a) Everybody likes somebody
   (b) Nobody likes anyone who doesn't like anyone

Let me add one more example, which is not strictly in the language of first-order logic. Many-sorted logic, which is discussed in Chapter 8 of these notes, is a slight extension of first-order logic that allows multiple "universes" of objects. There are quantifiers and variables ranging over each universe, or "sort," and the function and relation symbols are specified in a way that indicates which sorts the arguments range over.

4. On numbers and sets of numbers (in a two-sorted language, with a binary relation $x \in Y$ between numbers x and sets Y)
   (a) a is a subset of b
   (b) there is an empty set
   (c) for every x and y there is a set whose elements consist of the elements of x together with the elements of y.

Incidentally, these formalizations also work in the language of Zermelo-Fraenkel set theory, an axiomatic basis for mathematics in which every mathematical object is viewed as a set.

Once again, remember that the formal statements are just strings of symbols. We've attached intuitive meanings to them, but we haven't explained where this "meaning" comes from! The semantics gives us a formal way of showing that these formal statements have the intended meaning, when interpreted in the right structure.

Since we have defined terms and formulas inductively, we can (oh, joy!) do proofs by induction and define functions by recursion on these sets. The latter really require us to prove unique readability for terms and formulas; they gory details are left to you.

You may have noticed that some formulas, like $\forall x\ \exists y\ (y > x)$ "say things" outright. Others "say things" that depend on the value assigned to some variable: for example $\exists x\ (y + 1 < x < z)$ "says" something whose truth depends on the values of y and z. In this case, we say that y and z are free variables in this formula; this is the motivation behind the Definitions 2.3.6 and 2.3.7 of van Dalen. I will discuss them in class. A term or formula is *closed* if it has no free variables; a closed formula is also called a *sentence*. A formula without quantifiers is called *open* or *quantifier-free*.

In the first example above, we can say that x and y are *bound variables*; in the second example, only x is bound. But there is a subtle issue lurking here, since in any given formula the same variable may be both free and bound: this is the case for x in the formula $\exists x\ (x > 0) \wedge (x < z)$. So we should really be talking about free and bound *occurences* of variables in a formula, i.e. variables together with an index as to where they occur. As an exercise, you can try to make this notion more precise with an inductive definition; I will use it freely in these lectures.

As in the case of propositional logic, we can talk about substitutions. In this case, we have two kinds of substitutions: substituting a term for a variable, or substituting a formula for a 0-ary relation symbol. And in the first case, we can talk about substituting a term for a variable in another term, and in another formula, for a total of three different kinds of substitutions. The formal definitions are given as 2.3.9, 2.3.10, and 2.3.11; they are fully routine inductive definitions. I won't write them all out in full, but I will give some examples in class.

Finally, let me point out that some substitutions of a term for a variable in a formula are just downright "fishy." For example, take the formula $\exists y\ (y > x)$, with free variable y. This seems to say that there is something bigger than x; if we are thinking of the natural numbers, for example, we expect this formula to come out "true" no matter what we "plug in" for y. But what if we plug in y itself? Then we have $\exists y\ (y > y)$, which is patently false.

What went wrong? The variable y was bound in the example above; it was really a place holder. When we substituted y for x, it became "captured" by the quantifier — an unfortunate occurence. To rule out cases like these, we will say that "t is free for x in φ" if substituting t for x in φ is not fishy. Informally, t is free for x in φ if no variable of t falls under the scope of some quantifier of φ when it is substituted for x; Definition 2.3.12 on page 66 of van Dalen provides a more formal definition.

5.3 Semantics

In the case of propositional logic, it made little sense to ask whether a formula like $p \vee q \to r$ was "true," since the truth value of such a formula depended on the truth values of p, q, and r. Instead, we could ask for the truth value of this formula relative to a given *truth assignment*. Similarly, it makes little sense to ask whether a sentence $\forall x\ \exists y\ R(x, y)$ is "true," without a context; we can only evaluate the truth of such a sentence once we have specified what R means, and what "universe" of objects the quantifiers are supposed to range over. For a given language (i.e. similarity type), a *structure* provides exactly this information.

Definition 5.3.1 *A* structure *(or* "model"*) for the similarity type described in the last section consists of*

1. *A set A (the "universe" of the structure)*

2. *Relations $R_1, R_2, \ldots, R_n$ on A, of the same arity as the relation symbols $r_1, r_2, \ldots, r_n$*

3. *Functions $F_1, F_2, \ldots, F_m$ on A, of the same arity as the function symbols $f_1, f_2, \ldots, f_m$*

4. *Elements c_i of A, corresponding to the constants $\bar{c}_i$*

Such a structure is usually denoted

$$\mathfrak{A} = \langle A, R_1, \ldots, R_n, F_1, \ldots, F_n, \ldots, c_i, \ldots \rangle.$$

(Note that $\mathfrak{A}$ is just a gothic version of the letter A! When I was a student it was a long time before I finally figured that out.) The idea is that each *function symbol* f_i denotes the corresponding *function* F_i, and so on. Make sure you do not confuse the function with the symbol. I should really keep a convention of using lower case letters to denote symbols and upper case letters to denote functions and relations, but even if I were to try to be consistent in doing so, I would probably slip up; so I won't really try. But this means that you will have to work extra hard to keep them separate in your minds. Ask me if you are ever unsure as to which I mean if I write "f_i" or "R_j" on the board.

If $\mathfrak{A}$ is a structure, $|\mathfrak{A}|$ is often used to denote the universe of $\mathfrak{A}$, $f_i^{\mathfrak{A}}$ and $r_j^{\mathfrak{A}}$ denote the functions and relations, etc. In short, the structure $\mathfrak{A}$ gives us "interpretations" of the basic symbols of our language:

$$\begin{array}{ccc} f_i & \rightsquigarrow & f_i^{\mathfrak{A}} \\ r_j & \rightsquigarrow & r_j^{\mathfrak{A}} \\ c_k & \rightsquigarrow & c_k^{\mathfrak{A}} \end{array}$$

We would like to extend the interpretation to terms and formulas, so that

- Closed terms denote elements of $|\mathfrak{A}|$

- Sentences denote either "true" or "false"

More generally,

- A term with free variables $x_1, \ldots, x_k$ denotes a k-ary function on $\mathfrak{A}$

- A formula with free variables $x_1, \ldots, x_k$ denotes a k-ary relation on $\mathfrak{A}$

That is, $t(x_1, \ldots, x_k)$ denotes the function that takes k values to the "interpretation" of t in $\mathfrak{A}$ and $\varphi(x_1, \ldots, x_k)$ denotes the k-ary relation that holds of its arguments if φ is "true" of the arguments in the structure $\mathfrak{A}$.

As you may have guessed, we can do this with a pair of recursive definitions. But first, let us consider some examples of structures.

1. $\langle \mathbb{N}, 0, 1, +, \times, < \rangle$
2. $\langle \mathbb{R}, 0, 1, +, \times, \sqrt{\cdot}, < \rangle$
3. $\langle G, \cdot, \cdot^{-1} \rangle$ (a group)
4. $\langle$People in this room$, Likes(x, y), Male(x), Female(x)\rangle$
5. $\langle$Sets of natural numbers$, \subset, \cup, \cap, \emptyset\rangle$

Note that I am being *very* sloppy here, by using the "+" to denote the addition *symbol* as well as the addition *function* on the natural numbers. In other words, if I use "+" to denote the symbol, I should write something like "$+^{\mathbb{N}}$" for the function. I will not always be so noble; again, I will rely on you to add the superscript. But the point to keep in mind is that as far as syntax is concerned, "3 + 5" is just a string of symbols, which we can interpret in any structure we like. If "+" is interpreted as addition on the natural numbers (and "3" and "5" are interpreted as 3 and 5), then this will be interpreted as the number 8. But if "3" and "5" are interpreted as the real numbers π and e and "+" is interpreted as multiplication on the real numbers, "3 + 5" denotes πe. What fun!

The semantic definitions are given on page 70 of van Dalen, by Definitions 2.4.1 and 2.4.2. A slight twist is given by the fact that to define the semantics for a first-order language L relative to an appropriate L-structure $\mathfrak{A}$, it is convenient to work with an expanded language $L(\mathfrak{A})$, in which you've added a constant name $\bar{a}$ for every element a of the universe of $\mathfrak{A}$. In other words, to give a semantic interpretation of L using the structure $\mathfrak{A}$, we actually give a semanatic interpretation of the larger language $L(\mathfrak{A})$.

That aside, the recursive definitions are perfectly routine. To keep the notation from getting in the way, you may prefer to think of Definition 2.4.1 as a recursive definition of a function $Val_{\mathfrak{A}}$ from the set closed terms to $|\mathfrak{A}|$. Then $t^{\mathfrak{A}}$ is just shorthand for $Val_{\mathfrak{A}}(t)$. Similarly, definition 2.4.2 gives a recursive definition of a function $True_{\mathfrak{A}}$ from the set of sentences to $\{0, 1\}$, and $[\![\varphi]\!]_{\mathfrak{A}}$ is just an abbreviation for $True_{\mathfrak{A}}(\varphi)$.

We're just about ready for the definitions of the semantic notions. There's a minor issue that needs to be settled, though: if φ has variables $x_1, \ldots, x_k$

free, how do we interpret $\mathfrak{A} \models \varphi$? One option is to just rule this out as meaningless; another is to simply adopt the convention that this means that for *very* substitution of names for the variables of φ, the assertion holds. Van Dalen does the latter, via definition 2.4.3. In general, though, I will try to avoid writing $\mathfrak{A} \models \varphi$ when φ is not a sentence.

Definition 2.4.4 is the big one:

- $\mathfrak{A} \models \varphi$, "$\mathfrak{A}$ satisfies φ," "$\mathfrak{A}$ is a model of φ"
- $\models \varphi$, "φ is valid," "φ is true in every model"
- $\mathfrak{A} \models \Gamma$, "$\mathfrak{A}$ satisfies Γ," "$\mathfrak{A}$ is a model of Γ"
- $\Gamma \models \varphi$, "Γ logically implies φ," or "φ is a semantic consequence of Γ"

As was the case for propositional logic, $\models$ is overloaded. I expect you to keep the various uses straight.

Lemma 2.4.5 just says that the definition of satisfaction works the way you expect it to work. (This is not surprising; if Lemma 2.4.5 didn't go through, we would simply go back and repair the definitions.) Whenever I ask you to prove that something holds "using the definition of satisfaction," I am implicitly allowing you to use the clauses of Lemma 2.4.5, since they follow directly from the definition. At this stage, you should pause to compare the notions we have just introduced to the corresponding notions for propositional logic.

In essence, Lemma 2.4.5 just allows you to "translate" symbolic expressions into ordinary mathematical language. For example, if $\mathfrak{A}$ is a structure for a language with a binary relation symbol R, then $\mathfrak{A} \models \forall x \; \exists y \; R(x, y)$ if and only if for every element a in the universe of $\mathfrak{A}$, there is an element b in the universe of $\mathfrak{A}$ such that such the interpretation of R in $\mathfrak{A}$ holds of a and b; more concisely, for every a in $|\mathfrak{A}|$, there is a b in $|\mathfrak{A}|$ such that $R^{\mathfrak{A}}(a, b)$. The latter formulations of $\mathfrak{A} \models \forall x \; \exists y \; R(x, y)$ simply express what the formula $\forall x \; \exists y \; R(x, y)$ "says" about $\mathfrak{A}$. The fact that the translation reflects the intuition behind our choice of symbols indicates that we have defined $\mathfrak{A} \models \varphi$ in the right way.

Now, one way to prove things about the semantic relation is to translate various assertions into ordinary mathematical language, as above, and use ordinary mathematical reasoning. You will see examples of this in the next section. This takes some getting used to, and when translating symbols into words you may not feel like you are really doing anything; but you are.

5.4 Properties of predicate logic

We can now verify some basic properties of the semantic definitions. I will try to move through them quickly, since the proofs are routine; occasionally, I will work out a case or two, but most of the time I will leave the verifications to you. (You should go home and practice proving them until you believe that you can just crank them out on demand.)

These include:

- Theorem 2.5.1
- Theorem 2.5.2
- Theorem 2.5.3
- Lemma 2.5.4 (we will mainly be interested in parts (i) and (ii))
- Theorem 2.5.6, which confirms the intuition that the bound variables are nothing more than "place holders."

Pay close attention to the warning on page 74. Rest assured, I will trip you up on homework and exams if I can. Theorem 2.5.8 is the analogue of the substitution theorem for propositional logic; again, we are mostly interested in parts (i) and (ii).

Definition 2.5.10 says that a formula is prenex if it consists of a string of quantifiers, followed by an open (quantifier-free) formula. In other words, a formula is prenex if all the quantifiers are "in front." Using the theorems and lemmas above, we can translate any given formula to an equivalent one in prenex form: first, get rid of $\to$ and $\leftrightarrow$ in favor of $\vee$, $\wedge$, $\neg$, and then use the theorems and lemmas to bring the quantifiers to the front. In class, I will carry out a "proof by example," by converting the following formula to prenex form:

$$\forall w\ ((\forall x\ \exists y\ R(x,y) \to \exists z\ S(z,w)) \wedge \forall x\ T(x)).$$

I will leave it to you to read the proof of Theorem 2.5.11, which, unsurprisingly, proceeds by induction. In fact, implicit in the proof there is a recursive procedure to carry out the "prenexation," though the implementation requires some work. Kudos to anyone who gets such a routine up and running in ML, C++, or Java.

On page 79 of van Dalen there is a discussion of relativization, which provides a way of restricting the range of a quantifier. For example, if the universe consists of politicians, $\forall x\ Corrupt(x)$ asserts that all politicians are corrupt, and $\exists x\ Corrupt(x)$ asserts that some politicians are corrupt. But how might one assert that every Republican politician is corrupt, or that some Republican politicians are corrupt? A moment's reflection reveal that

$$\forall x\ (Republican(x) \rightarrow Corrupt(x))$$

and

$$\exists x\ (Republican(x) \wedge Corrupt(x))$$

do the trick. Make sure you understand why switching $\rightarrow$ and $\wedge$ in these examples do not produce the desired result. A prime mathematical example of relativization is given in van Dalen; make sure you understand this example, and the notation of Definition 2.5.12.

In Section 2.6 van Dalen points out that equality satisfies some basic axioms, I_1 to I_4. I will discuss the proofs briefly in class; they are routine. Make sure that you understand how I_1 to I_3 express the reflexivity, symmetry, and transitivity of equality. For a given language, we could replace the equality symbol by another symbol $\approx$, satisfying these axioms. But note that these axioms don't "determine" equality uniquely! That is to say, in a given structure $\mathfrak{A}$ the relation $\approx^{\mathfrak{A}}$ might satisfy I_1 to I_4, and yet not coincide with the equality relation on the universe of $\mathfrak{A}$. Make sure you understand how this can happen.

This is a good place to introduce some useful notation for formulas, that emulates the kind of notation that mathematicians use for functions. Specifically, mathematicians will sometimes define a function by writing $f(x) = ax^2$, to indicate that x is the dependent variable and a should be thought of as a "parameter." In a similar way, if I mention a formula $\varphi(x)$ in a particular context, I am really refering to a formula φ and "tagging" a particular variable x as special. If I later write $\varphi(t)$, I really mean $\varphi[t/x]$, where I have renamed the bound variables of φ to prevent collisions, if necessary.

5.5 Using predicate logic

Equipped with a formal description of what it means for a first-order sentence to be true in a structure, as well as a formal notion of logical consequence, we can now put these concepts to work. You should keep in mind that predicate logic is a formal *language*, designed to model certain essential features of informal language. One might therefore devote some energy to showing how one can capture various elements of natural language in a first-order way, as in Russell's analysis of definite descriptions. Of course, many features of natural language cannot be forced into the first-order framework, which might lead one look for formal extensions of the language that capture these features as well.

On the other hand, when it comes to *mathematical* discourse (as characterized in Chapter 1) first-order logic proves to be remarkably flexible and expressive. Restricting our attention to mathematical concerns, we may wish to use first-order logic in any of the following ways:

1. *To describe (portions of) the mathematical universe.* This illustrates logic's foundational role. If we wish to think of a "mathematical proof" as a logical argument from suitable axioms, the foundationalist's task is to indentify those axioms. Such axioms will typically provide descriptions of the objects that we take to inhabit the mathematical universe, such as numbers, sets, functions, relations, and so on. For example, the axioms of Zermelo-Fraenkel set theory, ZFC, describe a ("the") universe of sets; with a little creativity, we can construe familiar mathematical objects as particular kinds of sets.

2. *To describe particular mathematical structures.* We might want to write down axioms that describe interesting structures, like the natural numbers or the real numbers, or the Euclidean plane.

3. *Describing classes of structures.* We can use the language of first-order logic to describe various *kinds* of structures we are interested in, such as graphs, groups, various kinds of orderings, and so on.

4. *Defining a function or relation within a given structure.* Fixing a specific structure, we can use first-order formulas to describe certain elements and relationships in that structure.

In this section I will make these ideas more precise, and discuss some examples. For the most part, however, I will give these applications short-shrift, but you should keep in mind that it is these applications that provide the underlying motivation for much of what is to follow.

You may have already noted that there isn't a sharp distinction between 1 and 2 above. That is, we can think of 1 as being a special case of 2, where the "universe of mathematical objects" is the structure in question. More suprisingly, it is hard to distinguish between 2 and 3: we will see that every first-order description captures more structures than the intended one, so that there are "nonstandard" models of any theory of arithmetic or set theory. In fact, a motivating factor in our formal study of first-order logic is to understand such limitations. Much of Chapter 7 will be devoted to a formal exploration of the strengths and weaknesses of first-order logic, with respect to these goals.

To summarize the discussion up to this point, there are essentially two things we can do with first-order logic:

- define a class of structures, or
- define a relation within a given structure.

The formal definitions that capture these notions are as follows.

Definition 5.5.1 *Let Γ be a set of sentences in a first-order language L. Γ is said to* define *the class of structures*

$$\{\mathfrak{A} \mid \mathfrak{A} \models \Gamma\}.$$

In other words, Γ "defines" the class of models which satisfy it. I have used the word "class" because, technically speaking, the thing I have written down is too big to be a set. If a class A is defined by a singleton set $\{\varphi\}$, I will also say that A is defined by φ. (Exercise: show that if a class of structures is defined by a finite set of sentences, then it can be defined by a single sentence.)

Definition 5.5.2 *Suppose $\varphi(x_1, \ldots, x_k)$ is a formula in a language L with the free variables shown, and suppose $\mathfrak{A}$ is a structure for L. Then φ is said to (explicitly)* define *the k-ary relation R in $\mathfrak{A}$ given by*

$$R(a_1, \ldots, a_k) \quad \textit{iff} \quad \mathfrak{A} \models \varphi(\bar{a}_1, \ldots, \bar{a}_k).$$

In other words, φ "defines" the relation "satisfies φ." (There is also a separate notion of implicit definability that we will not consider here.)

Let us consider some examples of classes of structures that are first-order definable. Many of these are in Section 2.7 of van Dalen, on pages 83–90.

- In the language with only equality, a "structure" is just a set. With a single sentence we can define the class of structures with cardinality at least n, at most n, or exactly n. With a set of sentences, we can define the class of infinite structures.

  (With equality, one can also express the notion "there exists a unique x such that ...," written, in symbols, "$\exists! x \ldots$." I will discuss this in class.)

- In the language with equality and a binary relation $\leq$, we can define the classes of partial orders, linear (or total) orders, dense orders, and so on. I will draw some pictures on the board.

- In the language with a binary relation $\approx$, we can define the class of equivalence relations.

- In the language with symbols $e, \circ, \cdot^{-1}$, we can define the class of groups. (Similarly for rings, fields, algebraically closed fields, and other algebraic structures. Note that the axioms are quantifier-free, or "universal.")

- In the language with a binary relation symbol R, we can describe undirected graphs. Note that these are particularly nice structures to work with; to describe one, you can just draw a picture.

- The axioms of Peano arithmetic describe structures that "look like" the natural numbers.

- The axioms of real closed fields describe structures that "look like" the real numbers.

- One can formalize Hilbert's axiomatization of geometry, which describes structures that "look like" Euclidean space. One can interpret these axioms in the theory of real closed fields.

- The axioms of Zermelo Fraenkel set theory describe structures that "look like" the universe of sets.

The list goes on. For practice, consider the following sentences:

- $\forall x, y\ (S(x) = S(y) \rightarrow x = y)$

- $\exists x\ \forall y\ \neg(S(y) = x)$

What kinds of structures satisfy them?

Here are some additional questions to consider:

- In the language of equality, can you define the class of all finite structures?

- In the language with $\leq$, can you define the class of all well-orders?

- In the *any* language containing the language of groups, can you define the class of all (expansions of) finite groups?

Finally, let us consider some examples of definable relations on the structure $\mathfrak{A} = \langle \mathbb{N}, +^{\mathbb{N}} \rangle$:

1. Define the usual relation $\leq^{\mathbb{N}}$ with the formula

$$\varphi_{\leq}(x, y) \equiv \exists z \ (y = x + z)$$

   Then define $<^{\mathbb{N}}$ as "less than or equal to, but not equal to."

2. Define the element $0^{\mathbb{N}}$ (that is, the relation "is equal to $0^{\mathbb{N}}$"), by

$$\varphi_0(x) \equiv x + x = x$$

3. Define the element $1^{\mathbb{N}}$ by

$$\varphi_1(x) \equiv \forall z \ (z < x \leftrightarrow z = 0)$$

   where $z < x$ and $z = 0$ are really abbreviations for $\varphi_{<}(z, x)$ and $\varphi_0(z)$.

4. The successor function $S^{\mathbb{N}}$, thought of as a relation "$S^{\mathbb{N}}(x) = y$" between x and y, is defined by the formula

$$\varphi_S(x, y) \equiv x < y \wedge \neg \exists z \ (x < z \wedge z < y).$$

I have been good about maintaining a clear distinction between symbols and the formulas they represent, using, for example, $+^{\mathbb{N}}$ for the function and $+$ for the symbol. But having shown you the high moral ground, I will now adopt the logician's practice of abusing notation and using the same symbol for both.

For further examples, think of defining the set of primes in the structure

$$\mathfrak{B} = \langle \mathbb{N}, 0, 1, +, \times \rangle.$$

Also, consider the following questions:

1. Can you define multiplication in $\mathfrak{A}$?

2. Can you define exponentiation in $\mathfrak{B}$?

3. Can you define addition in the structure $\langle \mathbb{N}, < \rangle$?

We will learn the answers to these questions in Chapter 7.

Chapter 6

Deduction for Predicate Logic

6.1 Natural deduction

Now that we have the syntax and semantics for predicate logic, the next step is to find a deductive system that is sound and complete for the semantics. In contrast to the situation with propositional logic, having a deductive system for predicate logic is more crucial. Whereas to show that a propositional formula φ is a tautology it is enough to test all assignments of truth values to the propositional variables of φ, there are first-order sentences that are only satisfied by infinite structures, and so we cannot, in general, determine the validity of φ by "computing" its truth value in every relevant structure. Roughly speaking, there are "too many" such structures, and they are simply "too big."

For our proof system, we will keep all the old rules from propositional logic, and add some new ones to handle $\forall$, $\exists$, and $=$. (Van Dalen prefers to think of the rules for $\exists$ as *derived* from the rules for $\forall$, with $\exists x\ \varphi$ taken to abbreviate $\neg \forall x\ \neg \varphi$. While I may ask you to show that the rules for $\exists$ can be derived in this way on a homework assignment, in class I will take them as basic.)

In general, our proofs will involve formulas with free variables. This presents a conceptual difficulty: how are we supposed to interpret a proof of a formula φ from Γ, if both φ and the formulas in Γ have such free variables? Intuitively, you can think of the proof as establishing that φ follows from Γ, no matter what the free variables are taken to stand for. When we prove the soundness theorem, this intuition will be given a more precise formulation.

In class I will follow the development of van Dalen, Sections 2.8 to 2.10 (pages 92 to 103). First I will treat the universal quantifier, and:

- give the introduction and elimination rules, on page 92.
- show that the restrictions are necessary (pages 92 and 93).
- go over the three examples in van Dalen (page 92).

Since one can think of the existential quantifier as a kind of infinite disjunction, it is not surprising that the rules for $\exists$ are similar to the ones for $\forall$. The introduction rule is designed to model the following sort of argument: suppose you know that there is a politician at Amy's party. You may wish to call this politician "X," and, using the fact that X is a politician at Amy's party, show that nobody there will have any fun. If you can do this, making no other assumptions about X, then all in all, you will have shown that nobody will have fun at Amy's party.

This example may strike you as a little bit frivolous (it should), but I hope it gets the point across. If you know that an object with a certain property exists, you can simply "call it X" and then reason about X. If doing so allows you to reach a conclusion that doesn't refer to X, then you can claim to have proved the conclusion outright, simply forget about the "temporary" use of the name X.

In any event, I will

- present the rules for $\exists$ (see Lemma 2.9.1 on page 97 of van Dalen, and the discussion which follows).
- present some examples, that are not in van Dalen:
  - $\exists x \; (\varphi(x) \wedge \psi(x)) \rightarrow \exists y \; \varphi(y)$
  - $\exists x \; \forall y \; \varphi(x, y) \rightarrow \forall y \; \exists x \; \varphi(x, y)$
  - $\exists x \; \varphi(x) \rightarrow \exists y \; \varphi(y)$
- give examples, to show that the restrictions are necessary. (That is, I will give a "bad" derivation of $\exists x \; \forall y \; (y = x)$ from $\forall y \; (y = y)$, and another "bad" derivation of $\exists x \; \varphi(x) \rightarrow \forall x \; \varphi(x)$.)

I will also discuss van Dalen's more careful wording on page 99.

Finally, I will discuss the rules for identity. In contrast to van Dalen, I will allow instances of RI_1 to RI_4 for arbitrary terms $r, s, t, \ldots$ instead of just variables $x, y, z, \ldots$. Actually, it doesn't matter which version you use, or even if you replace the rules by the corresponding axioms (e.g. $x = y \to y = x$ for RI_2); I will discuss this. I will also indicate (as does van Dalen), that for RI_4 one can restrict the rules to just function and relation symbols.

It is important to note that if you begin a proof with $x = x$, then this is an *axiom*, not an *assumption*; that is, there are no open assumptions in the corresponding one line proof.

Once you have had a chance to experiment with natural deduction, I may come back to this and do a few more examples. These may include:

1. Proving $\exists x\ \varphi \leftrightarrow \neg\forall x\ \neg\varphi$

2. Proving $\forall u, v, w\ (u + w = v + w \to u = v)$ from the following axioms:

   (a) $\forall x\ (x + 0 = x)$

   (b) $\forall x\ \exists y\ (x + y = 0)$

   (c) $\forall x, y, z\ ((x + y) + z = x + (y + z))$

6.2 Soundness

As was the case for propositional logic, we can describe the set of derivations by an inductive definition; and then use that to prove soundness by induction on derivations. Here we need to be more careful, since for a given derivation of a formula φ from a set of formulas Γ, both φ and Γ may have free variables. Then we interpret soundness as follows: for every derivation d, if d is a derivation of φ from Γ, then for any structure $\mathfrak{A}$ and any assignment of elements of $|\mathfrak{A}|$ to the variables of φ and Γ, if $\mathfrak{A}$ satisfies Γ, then $\mathfrak{A}$ satisfies φ.

A more careful formulation appears on page 94 of van Dalen. I will discuss this in class, as well the proof on page 95, focusing on the case of $\forall$ introduction.

The soundness theorem has many applications: to show $\Gamma \nvdash \varphi$, we can prove $\Gamma \not\models \varphi$. For example, from a previous homework assignment you now know that

$$\nvdash \forall x\ \exists y\ R(x, y) \to \exists y\ \forall x\ R(x, y)$$

and

$$\nvdash \exists x\ R(x) \land \exists x\ S(x) \to \exists x\ (R(x) \land S(x)).$$

We also have a formal means for showing that Euclid's fifth postulate is not derivable from the other axioms of Euclidean geometry: once all these axioms have been suitably formalized, we need only find a model of non-Euclidean geometry, which is to say, a structure that falsifies the fifth postulate but satisfies the others.

6.3 Completeness

Let us now prove the completeness theorem for predicate logic. For the moment, we will restrict our attention to sentences; using Lemma 6.3.3 below, we can easily generalize the theorem to allow free variables in Γ and φ.

Theorem 6.3.1 *Let L be any first-order language, let Γ be any set of sentences in L, and let φ be any sentence in L. If $\Gamma \models \varphi$, then $\Gamma \vdash \varphi$.*

From now on, I will omit explicit reference to L; just assume that we've settled on some fixed (but arbitrary) language, from the start. As was the case for propositional logic, the previous theorem follows from (and is, in fact, equivalent to) the following:

Theorem 6.3.2 *Let Γ be any set of sentences. If Γ is consistent, then Γ has a model.*

Recall that "Γ is consistent" means $\Gamma \nvdash \bot$, and "Γ has a model" means that there is a structure $\mathfrak{A}$ such that $\mathfrak{A} \models \Gamma$. The proof that the second form of the completeness theorem implies the first is runs just as it did for propositional logic: if $\Gamma \nvdash \varphi$, then $\Gamma \cup \{\neg\varphi\}$ is consistent, and hence has a model; therefore $\Gamma \not\models \varphi$. Make sure you can fill in the details.

From now on, then, we will focus on the second theorem, which van Dalen calls the "model existence lemma." Given a consistent set of sentences Γ, we need to cook up a model of Γ somehow. This is bound to be more difficult than cooking up a truth assignment: instead of simply assigning 0's and 1's to propositional variables, now we have to find a suitable *universe* for our structure, and then find suitable denotations for the function and relation symbols of L.

And where can we find such a universe? There is a saying, "when all you have is a hammer, everything looks like a nail." All we have to work with is a set of sentences Γ — a set of syntactic objects. So it makes sense to look for the right syntactical objects, extracted from Γ, to comprise the universe of the structure we are building.

One natural idea is to take the universe to be the set of closed terms in the language. For example, in the language of arithmetic, the terms 0,

$S(0)$, $S(S(0))$, and so on provide good candidates for the inhabitants of our universe. Of course, there may not *be* any closed terms in our language — our language may not have any constants — or there may not be "enough." So, starting with a consistent set of sentences, Γ, our strategy is roughly as follows:

1. Extend Γ to a consistent set of sentences Γ', in an expanded language which has "enough names."

2. Extend Γ' to a maximally consistent set of sentences Γ''.

3. Read off a structure $\mathfrak{A}$, whose universe consists of closed terms in the language of Γ''.

Sounds reasonable, doesn't it? Actually, this works just fine if we do not include equality. Otherwise, we need to do a little more work. First, we temporarily replace the equality symbol with a relation symbol $\approx$, and add axioms to Γ asserting that $\approx$ "acts like" equality, which is to say that $\approx$ is an equivalence relation that is compatible with the function and relation symbols of L. Following the prescription above, we can build a model of Γ with an intepretation for $\approx$ that "looks like" equality, though we may have $b \approx^{\mathfrak{A}} c$ for distinct elements b and c of $\mathfrak{A}$. But we can then construct another model $\mathfrak{A}'$ by "bunching" together groups of elements that are $\approx$-equivalent. $\mathfrak{A}'$ will be the structure we are after. In short, we simply need to add one more step to the list above:

4. Construct another structure $\mathfrak{A}'$, whose universe consists of equivalence classes (relative to $\approx^{\mathfrak{A}}$) of the universe of $\mathfrak{A}$.

Although the first completeness proof for first-order logic was due to Gödel, the approach we will follow here is essentially due to Henkin; it is both relatively easy to understand, and relatively easy to generalize to other situations.

With this general overview in mind, we can now better understand van Dalen's proof. I will follow his development fairly closely, changing only some of the organizational details.

I will:

- define the notion of a theory;

- say what it means to be a set of axioms for a theory;

- point out that if Γ is any set of sentences, $\{\sigma \mid \Gamma \vdash \sigma\}$ is a theory

Note that the use of the word "theory" has something to do with the informal use of the term; if you "believe" a set of sentences T to be true, you should also believe their deductive consequences. (One sometimes says that a theory is a set of sentences that is "deductively closed.") I will also:

- define the notion of an extension of a theory, and a conservative extension of a theory;
- define the notion of a Henkin theory.

All these are found on page 106 of van Dalen. Being a Henkin theory means that the language has "enough names," in the sense described above. Note that the sentence $\exists x\ \varphi(x) \to \varphi(c)$ does not mean that $\varphi(c)$ is necessarily true. Instead, think of it as saying that c is a prime candidate for satisfying $\varphi(x)$; if anything satisfies $\varphi(x)$, then c does. (Along the lines of "if ever the was an honest man, Abe Lincoln was," or something like that.) Call this sentence a "special axiom for $\exists x\ \varphi(x)$," and c a "special constant."

The first step in the proof of the completeness theorem is to prove that any theory T has a conservative extension T_ω that is a Henkin theory. This uses a number of lemmas, as follows.

Lemma 6.3.3 *Suppose $\Gamma(x)$ is a set of formulas with a free variable x and $\Gamma(c)$ denotes the result of replacing every occurence of x in Γ with a new constant c. If $\Gamma(c) \vdash \varphi(c)$ then $\Gamma(x) \vdash \varphi(x)$.*

By a "new" constant, I mean that c does not occur in any sentence of Γ, or in φ. To prove this use induction on the length of the derivation of $\varphi(c)$ from $\Gamma(c)$; it is not hard.

Lemma 6.3.4 *Suppose Γ is a set of sentences, ψ is a sentence, and c is a new constant. Then if*

$$\Gamma \cup \{\exists x\ \varphi(x) \to \varphi(c)\} \vdash \psi$$

then $\Gamma \vdash \psi$.

This corresponds to part (a) of the proof of Lemma 3.1.7 in van Dalen. It says, roughly, that we can add a single special axiom without getting a stronger theory. The proof uses exercise 7 of section 2.9 in an important way; the exercise is a difficult one, but I will try to guide you through it in a future homework assignment.

The next Lemma says we can add a whole bunch of special axioms, all at once. This involves simply "iterating" the previous lemma finitely many times, corresponding to part (b) of the proof of Lemma 3.1.7.

Lemma 6.3.5 *Let T be a theory with language L. Let L^* be a language obtained by adding a constant c_φ to L for every sentence of the form $\exists x\ \varphi(x)$, and let T^* be the theory axiomatized by T and a special axiom $\exists x\ \varphi(x) \to \varphi(c_\varphi)$ for each such formula. Then T^* is a conservative extension of T.*

Is T^* a Henkin theory? It looks like it, since we've added special constants for every formula in the language L. But wait! Now there are *new* sentences of the form $\exists x\ \varphi(x)$ in the language L^*, and *they* don't have special constants! So we are led to defining the language L^{**} and the theory T^{**}. We have to keep going, defining a theory T_n for each natural number n, where T_0 is just T and T_{n+1} is T_n^*. Their union, T_ω, is the theory we are looking for.

Lemma 6.3.6 *T_ω is a Henkin theory and a conservative extension of T.*

This is just Lemma 3.1.8 in van Dalen. Taking the last four lemmas, we've completed the first step of our program. Given a consistent set of sentences Γ, let T be the theory axiomatized by Γ, and let T_ω be a Henkin theory that is a conservative extension of T. The last fact implies that T_ω is consistent as well.

Step 2 is easy; just as we did for propositional logic, we can extend T_ω to a maximally consistent set of sentences in the same language. Call the resulting set $\hat{T}$. As in the case of propositional logic, we can show that $\hat{T}$ is a theory (i.e. $\hat{T}$ is deductively closed).

Now, ignoring equality for the moment, the last step is to "read off" the structure $\mathfrak{A}$. Take, for the universe, the set of closed terms:

$$|\mathfrak{A}| = \{t \in L_\omega \mid t \text{ is a closed term}\}.$$

If f is a function symbol of L, how should we interpret it in $\mathfrak{A}$? This amounts to choosing a value of $f^{\mathfrak{A}}(a_1, \ldots, a_k)$, for each sequence of elements $a_1, \ldots, a_k$ in the universe of $\mathfrak{A}$. But these elements $a_1, \ldots, a_k$ are just terms! So there is only one reasonable choice:

$$f^{\mathfrak{A}}(a_1, \ldots, a_k) = f(a_1, \ldots, a_k).$$

Make sure you understand what is going on here: the *function* $f^{\mathfrak{A}}$ works by prepending the *function symbol* f to its arguments (and adding parentheses and commas). Now, to handle the relation symbols, there is only one choice; we have essentially come down to the propositional case. For each relation symbol r and elements $a_1, \ldots, a_k$ in the universe of $\mathfrak{A}$, "read off" the truth value of $r^{\mathfrak{A}}(a_1, \ldots, a_k)$ from the maximally consistent theory, $\hat{T}$:

$$r^{\mathfrak{A}}(a_1, \ldots, a_k) \text{ iff } r(a_1, \ldots, a_k) \in \hat{T}.$$

Once again, make sure you are clear as to what is going on here: the *relation* $r^{\mathfrak{A}}$ is said to hold of $a_1, \ldots, a_k$ in and only if the corresponding *atomic formula* is in $\hat{T}$.

The following two lemmas show that the model we have constructed satisfies $\hat{T}$. The niggling technical details are daunting: for example, we have to deal with the language $L(\mathfrak{A})$, which has names of elements of the universe of $\mathfrak{A}$— which means we have to deal with terms that include names of terms, and so on. *Mon dieu!* But though there are a lot of technical details, the intuitions are straightforward. And the upshot is that the restriction of $\mathfrak{A}$ to the language of L satisfies Γ, which is exactly what we wanted.

Lemma 6.3.7 *If t is any closed term in the language L_ω, then $t^{\mathfrak{A}} = t$. Moreover, if $t(x_1, \ldots, x_k)$ is any term in L_ω and $\bar{s}_1, \ldots, \bar{s}_k$ are names (in $L_\omega(\mathfrak{A})$) for elements of $|\mathfrak{A}|$, then $t(\bar{s}_1, \ldots, \bar{s}_k)^{\mathfrak{A}} = t(s_1, \ldots, s_k)$.*

The proof is simply by induction on terms. There is almost nothing to say, though you have to be very careful while you are saying it.

Lemma 6.3.8 *Let $\varphi(x_1, \ldots, x_k)$ be any formula in L_ω, and let $\bar{a}_1, \ldots, \bar{a}_k$ be any names in $L_\omega(\mathfrak{A})$. Then*

$$\mathfrak{A} \models \varphi(\bar{a}_1, \ldots, \bar{a}_k) \text{ iff } \varphi(a_1, \ldots, a_k) \in \hat{T}.$$

Proof. The proof is by induction on formulas. It is convenient to assume that the only logical constants in the language are $\rightarrow, \wedge, \perp$, and $\forall$ and $\exists$. Of course, van Dalen takes $\exists$ to be defined in terms of $\forall$, eliminating one the case for $\exists$; but that case is illuminating, so I will include it.

First of all, in the base case, we have

$$\mathfrak{A} \models r(t_1(\bar{a}_1, \ldots, \bar{a}_k), \ldots, t_k(\bar{a}_1, \ldots, \bar{a}_k))$$

if and only if

$$r^{\mathfrak{A}}((t_1(\bar{a}_1, \ldots, \bar{a}_k))^{\mathfrak{A}}, \ldots, (t_k(\bar{a}_1, \ldots, \bar{a}_k))^{\mathfrak{A}}),$$

which holds if and only if

$$r^{\mathfrak{A}}(t_1(a_1, \ldots, a_k), \ldots, t_k(a_1, \ldots, a_k)),$$

by the previous lemma; and this, by the definition of $r^{\mathfrak{A}}$, holds if and only if

$$r(t_1(a_1, \ldots, a_k), \ldots, t_k(a_1, \ldots, a_k)) \in \hat{T}.$$

The propositional constants are handled just as in Chapter 4 of these notes. Let us consider the case where φ is of the form $\exists y\ \psi(y, x_1, \ldots, x_k)$. We have

$$\mathfrak{A} \models \exists y\ \psi(y, \bar{a}_1, \ldots, \bar{a}_k)$$

if and only if there is some b in $|\mathfrak{A}|$ such that

$$\mathfrak{A} \models \psi(\bar{b}, \bar{a}_1, \ldots, \bar{a}_k).$$

But elements b of $|\mathfrak{A}|$ are just terms in L_ω, so by the induction hypothesis the last assertion is equivalent to saying that there is some term b in L_ω such that

$$\psi(b, a_1, \ldots, a_k) \in \hat{T}.$$

Since $\hat{T}$ is a theory, if the formula $\psi(b, a_1, \ldots, a_k)$ is in $\hat{T}$ for some term b, then $\exists y\ \psi(y, a_1, \ldots, a_k)$ is in $\hat{T}$ as well. Conversely, if $\exists y\ \psi(y, a_1, \ldots, a_k)$ is in $\hat{T}$, then there is a constant c such that $\psi(c, a_1, \ldots, a_k)$ is also in $\hat{T}$ — since $\hat{T}$ is a Henkin theory, and so has an axiom of the form

$$\exists y\ \psi(y, a_1, \ldots, a_k) \to \psi(c, a_1, \ldots, a_k)$$

for some constant c. In that case, we can just take c to be the term we are looking for.

The case for the universal quantifier is really derivative of the case just described, and is discussed on page 110 of van Dalen. □

You should now go back to the outline at the beginning of this section, and convince yourself that we have successfully completed steps 1–3 of our program. All we have left to do is patch in a procedure to handle the equality symbol.

So suppose we are given a consistent set of sentences Γ in a language L, *with* equality. Let L' be the language L *without* equality, but with an extra binary relation symbol $\approx$. If φ is a formula in L, let φ' be the formula which results from replacing every instance of the equality symbol in φ with $\approx$. Let Γ' be the set of sentences of the form φ', where φ is a sentence in Γ, together with all axioms of the form I_1 to I_4.

It is not hard to see that if Γ proves some formula φ in first-order logic with equality, then Γ' proves φ' in first-order logic without equality, and vice-versa. As a result, Γ' is also a consistent set of sentences. Since L' is a language without equality, we know how to build a model of Γ'; call this model $\mathfrak{A}'$.

$\mathfrak{A}'$ interprets all the symbols of L, as well as $\approx$. Furthermore, the interpretation of $\approx$ "looks" a lot like equality. Axioms I_1 to I_3 guarantee that $\approx^{\mathfrak{A}'}$ is an equivalence relation on the universe of $\mathfrak{A}'$. Furthermore, axioms of the form I_4 guarantee that $\approx^{\mathfrak{A}'}$ is a *congruence* with respect to the function and relation symbols; this simply means that if $a_1 \approx^{\mathfrak{A}'} b_1, \ldots, a_k \approx^{\mathfrak{A}'} b_k$, then $f^{\mathfrak{A}'}(a_1, \ldots, a_k) \approx^{\mathfrak{A}'} f^{\mathfrak{A}'}(b_1, \ldots, b_k)$, and $r^{\mathfrak{A}'}(a_1 \ldots a_k)$ iff $r^{\mathfrak{A}'}(b_1, \ldots b_k)$ (assuming the symbols f and r have arity k). The idea is to use $\mathfrak{A}'$ to cook up a structure $\mathfrak{A}$ for L, with the property that whenever φ is a formula of L and $\mathfrak{A}'$ models φ', then $\mathfrak{A}$ models φ. The strategy is to group together $\approx^{\mathfrak{A}'}$-equivalent elements of $\mathfrak{A}'$ and call each one of those a single element of $\mathfrak{A}$, so that the equivalene relation on $\mathfrak{A}'$ "translates" to real equality on $\mathfrak{A}$.

More precisely, define a new structure $\mathfrak{A}$ for the language L as follows. Let the universe of $\mathfrak{A}$ consist of *equivalence classes* $[a]$ of elements a in the universe of $\mathfrak{A}'$:

$$|\mathfrak{A}| = \{[a] \mid a \in |\mathfrak{A}'|\}.$$

Define

$$f^{\mathfrak{A}}([a_1], \ldots, [a_k]) = [f^{\mathfrak{A}'}(a_1, \ldots, a_k)]$$

and

$$r^{\mathfrak{A}}([a_1], \ldots, [a_l]) \text{ iff } r^{\mathfrak{A}'}(a_1, \ldots, a_l).$$

The fact that $\approx^{\mathfrak{A}'}$ is a congruence means that $f^{\mathfrak{A}}$ and $r^{\mathfrak{A}}$ are well-defined, which is to say that the values on the right do not depend on the choice of representatives of the equivalence classes.

Now we can show inductively that if $t(x_1, \ldots, x_k)$ is any term in the language of L and $a_1, \ldots, a_k$ are elements of the universe of $\mathfrak{A}'$, then

$$(t(\overline{[a_1]}, \ldots, \overline{[a_k]}))^{\mathfrak{A}} = [(t(\bar{a}_1, \ldots, \bar{a}_k))^{\mathfrak{A}'}] \tag{6.1}$$

Furthermore, if $\varphi(x_1, \ldots, x_k)$ is any formula in the language of L (with equality) and φ' is its "translation" to L', we can also show inductively that for any elements $a_1, \ldots, a_k$ in $|\mathfrak{A}'|$, we have

$$\mathfrak{A} \models \varphi(\overline{[a_1]}, \ldots, \overline{[a_k]}).$$

if and only if

$$\mathfrak{A}' \models \varphi(\bar{a}_1, \ldots, \bar{a}_k)$$

Here, all the work is in the base cases, for atomic formulas of the form $r(t_1, \ldots, t_k)$ and $t_1 = t_2$; but these are handled by the definition of $\mathfrak{A}$ and equation (6.1). For example, in the case where φ is the atomic formula $t_1(x_1, \ldots, x_k) = t_2(x_1, \ldots, x_k)$ we have

$$\mathfrak{A} \models t_1(\overline{[a_1]}, \ldots, \overline{[a_k]}) = t_2(\overline{[a_1]}, \ldots, \overline{[a_k]})$$

if and only if

$$(t_1(\overline{[a_1]}, \ldots, \overline{[a_k]}))^{\mathfrak{A}} = (t_2(\overline{[a_1]}, \ldots, \overline{[a_k]}))^{\mathfrak{A}}$$

if and only if

$$[t_1(\bar{a}_1, \ldots, \bar{a}_k))^{\mathfrak{A}'}] = [t_2(\bar{a}_1, \ldots, \bar{a}_k))^{\mathfrak{A}'}]$$

if and only if

$$(t_1(\bar{a}_1, \ldots, \bar{a}_k))^{\mathfrak{A}'} \approx^{\mathfrak{A}'} (t_2(\bar{a}_1, \ldots, \bar{a}_k))^{\mathfrak{A}'}$$

if and only if

$$\mathfrak{A}' \models t_1(\bar{a}_1, \ldots, \bar{a}_k) \approx t_2(\bar{a}_1, \ldots, \bar{a}_k).$$

This shows that the new structure $\mathfrak{A}$ is a model of Γ, thereby completing the proof of the completeness theorem for first-order logic with equality.

6.4 Computational issues

This section and the next round out the chapter with some additional notes on the completeness theorem.

In the proof described above, we did not pay much attention to constructivity. In contrast to the propositional case, here we can ask an additional question: when is the model that we obtain at the end "computable," in the sense that the elements of the universe can be represented in a computer in such a way the the functions are computable and the relations are decidable?

You will be able to pose questions like this more precisely if you take a course like 80-311, *Computability and Incompleteness*. But to relieve some of the suspense, I can offer some informal answers right away. The good news is this: if you start with a *decidable* theory T, you can adapt the proof above to show that you *can* get such a computable structure at the end. The bad news is that there are many natural theories, with decidable sets of axioms, that are *not decidable*. This follows (essentially) from Gödel's incompleteness theorem: if T is consistent and includes a little bit of arithmetic, then T is not decidable (though it is "enumerable"). And, a final bit of news (neither good nor bad, in my estimation, just interesting): using the techniques of computability theory, you can characterize the "complexity" of the model you build from any set of axioms. To use technical terms, there is a model which is computable from the Turing jump of the set of axioms, but is *low* in this respect.

Given a sentence φ, one can try to design an algorithm that searches for a proof of φ, assuming there is one. The problem of finding sensible and efficient algorithms is the essence of automated deduction. Here one typically presents an algorithm, and then argues that if the algorithm fails to find the required proof, then one can extract from this failure a structure satisfying $\neg\varphi$. This provides a kind of proof of the completeness theorem that is more closely linked to the "practical" problem of searching for proofs.

6.5 Cardinality issues

Some of you may be familiar with the notion of the "cardinality" (roughly, "size") of a set, and the fact that there are infinite sets of different cardinalities. If you are, you might wonder, how "large" is the universe of the structure constructed in the proof of the completeness theorem?

If the language we start with is countable — which is to say, there are only countably many constant, function, and relation symbols — then a bit of "cardinal arithmetic" shows that the model obtained is again countable. Indeed, there are only countably many sentences in the original language, and hence only countably many special constants in T_0, T_1, and so on. This means that there are only countably many constants in L_ω, and hence only countably many closed terms, from which the universe of $\mathfrak{A}$ is constructed. Taking equivalence classes in $\mathfrak{A}'$ only cuts down on the size of the universe.

Note, however, that one can have even a single formula in a finite language which only has infinite models.

More generally, we can consider languages L that have κ many constant, function, and relation symbols, where κ is an uncountable cardinal. In that case, the model we construct will have a universe of size at most κ.

Chapter 7

Some Model Theory

7.1 Basic definitions

By now, you may have observed that there are at least two ways of thinking of predicate logic. The first is to think of it as the formalization of a certain kind of logical reasoning, for which our deductive system clarifies the allowable inferences and the semantics clarifies the meaning of the logical terms. However, some of the applications described in Section 5.5 illustrate another use of first-order logic, namely, as a tool for specifying properties of (or in) various structures.

This latter conception that forms the basis of model theory, which can be broadly described as the study of the relationship between formal language and structures, relative to a given semantics. More specifically, model theorists are interested in the kinds of properties one can or can't specify in the language, understanding the kinds of models a given set of sentences can have, or classifying theories by the kinds of models which satisfy them.

In this section I will teach you how to "talk the talk"; in the next few sections I will teach you how to "walk the walk." In the last section of this chapter I will discuss the two different ways of thinking about first-order logic, and try to summarize some of the conclusions that we can draw from the formal results.

What follows is a litany of definitions. Most of these are found either at the beginning of Section 3.2 of van Dalen or at the beginning of Section 3.3.

If Γ is set of sentences (in some language L), $Mod(\Gamma)$ denotes the class of its models. If $\mathfrak{A}$ is a structure, $Th(\mathfrak{A})$ denotes that *theory* of $\mathfrak{A}$, this is, the set of sentences that are true in $\mathfrak{A}$. It is not difficult to verify that $Th(\mathfrak{A})$ is maximally consistent, and (hence) a theory.

If $Th(\mathfrak{A}) = Th(\mathfrak{B})$, then $\mathfrak{A}$ and $\mathfrak{B}$ are said to be *elementarily equivalent.* This is written $\mathfrak{A} \equiv \mathfrak{B}$, and, roughly speaking, means that $\mathfrak{A}$ and $\mathfrak{B}$ are "indistinguishable" as far as the language of first-order logic is concerned: everything you can say about $\mathfrak{A}$, you can say about $\mathfrak{B}$, and vice-versa. (In model theory, "elementary" usually means, roughly, "first-order.")

We will see that different structures $\mathfrak{A}$ and $\mathfrak{B}$ can have the same theories. On the other hand, I may ask you to show for homework that if two sets Γ_1 and Γ_2 have exactly the same class of models, then the theories axiomatized by Γ_1 and Γ_2 are the same (this follows from the completeness theorem).

Let L_1 and L_2 be languages, with $L_2 \supseteq L_1$; by this I mean that L_2 has all the constant, function, and relation symbols of L_1, but possibly some extra ones. If $\mathfrak{A}$ is a structure for L_1 and $\mathfrak{B}$ is a structure for L_2, I will say that $\mathfrak{A}$ is a *reduct* of $\mathfrak{B}$, and that $\mathfrak{B}$ is an *expansion* of $\mathfrak{A}$, if $\mathfrak{A}$ and $\mathfrak{B}$ have the same universes, and the interpretation of every symbol of L_1 in $\mathfrak{A}$ is the same as the interpretation in $\mathfrak{B}$. For example,

$$\langle \mathbb{N}, 0, S, +, \times, < \rangle$$

is an expansion of

$$\langle \mathbb{N}, +, < \rangle.$$

On the other hand, if $\mathfrak{A}$ and $\mathfrak{B}$ are structures in the same language, I will say that $\mathfrak{A}$ is a *substructure* of $\mathfrak{B}$, written $\mathfrak{A} \subseteq \mathfrak{B}$, if the universe of $\mathfrak{A}$ is a subset of the universe of $\mathfrak{B}$, the functions of $\mathfrak{A}$ are just the restrictions of the functions of $\mathfrak{B}$ to the universe of $\mathfrak{A}$ (that is, $f^{\mathfrak{A}}(a_1, \ldots, a_k) = f^{\mathfrak{B}}(a_1, \ldots, a_k)$ for every sequence of elements $a_1, \ldots, a_k$ in $\mathfrak{A}$), and the relations of $\mathfrak{A}$ are just the restrictions of the relations of $\mathfrak{B}$ to the universe of $\mathfrak{A}$. For example,

- $\langle \mathbb{Q}, < \rangle$ is a substructure of $\langle \mathbb{R}, < \rangle$
- $\langle \mathbb{N}, 0, +, \times \rangle$ is a substructure of $\langle \mathbb{R}, 0, +, \times \rangle$
- $\langle \mathbb{Q}, < \rangle$ is *not* a substructure of $\langle \mathbb{R}, > \rangle$

$\mathfrak{A}$ is an *elementary substructure* of $\mathfrak{B}$ if it is a substructure of $\mathfrak{B}$, and the following holds: whenever $a_1, \ldots, a_k$ are elements of the universe of $\mathfrak{A}$ and $\varphi(x_1, \ldots, x_k)$ is a formula, then

$$\mathfrak{A} \models \varphi(\bar{a}_1, \ldots, \bar{a}_k)$$

if and only if

$$\mathfrak{B} \models \varphi(\bar{a}_1, \ldots, \bar{a}_k).$$

This is written $\mathfrak{A} \prec \mathfrak{B}$. This expresses a stronger assertion than "$\mathfrak{A} \subseteq \mathfrak{B}$ and $\mathfrak{A} \equiv \mathfrak{B}$," since $\mathfrak{A}$ and $\mathfrak{B}$ have to agree not just on the truth of sentences in the underlying language, but on sentences *with parameters* from the universe of $\mathfrak{A}$ as well. For example, consider the substructure of $\langle \mathbb{N}, S \rangle$ given by $\langle \mathbb{N}^+, S \rangle$, where $\mathbb{N}^+$ consists of all the *positive* natural numbers (excluding 0). The "isomorphism theorem" below shows that these two structures are elementarily equivalent, but the first thinks that $\exists x \; (x < \bar{1})$ is true, while the second thinks that it is false.

I will discuss the definitions of a *homomorphism* and an *isomorphism* between two structures (cf. definition 3.3.1 in van Dalen). A homomorphism F from $\mathfrak{A}$ to $\mathfrak{B}$ is, roughly, a map from the universe of $\mathfrak{A}$ to the universe of $\mathfrak{B}$ that "preserves the structure." Think of F as "translating" elements in $\mathfrak{A}$ to elements of $\mathfrak{B}$ (I will draw a picture on the board to illustrate this). If F is a bijection, then $\mathfrak{B}$ is really the same structure as $\mathfrak{A}$, after a "renaming" of elements. So saying that $\mathfrak{A}$ and $\mathfrak{B}$ are isomorphic, written $\mathfrak{A} \cong \mathfrak{B}$, is a strong way as saying that they are essentially the same. In particular, the following theorem holds:

Theorem 7.1.1 *Suppose F is an isomorphism between two structures $\mathfrak{A}$ and $\mathfrak{B}$, $\varphi(x_1, \ldots, x_k)$ is a formula with the free variables shown, and $a_1, \ldots, a_k$ are elements of $|\mathfrak{A}|$. Then*

$$\mathfrak{A} \models \varphi(\bar{a}_1, \ldots, \bar{a}_k)$$

if and only if

$$\mathfrak{B} \models \varphi(\overline{F(a_1)}, \ldots, \overline{F(a_k)}).$$

The proof is a routine induction on the formula φ. If φ is a sentence, the theorem implies that $\mathfrak{A} \models \varphi$ if and only if $\mathfrak{B} \models \varphi$, so if $\mathfrak{A}$ is isomorphic to $\mathfrak{B}$, then it is also elementarily equivalent to $\mathfrak{B}$. The converse, however, is *not* the case. For example, it turns out that $\langle \mathbb{Q}, < \rangle$ is elementarily equivalent to $\langle \mathbb{R}, < \rangle$, but they can't possibly by isomorphic since the first structure is countable and the second one isn't. We will see more examples of this kind soon (with justifications). So it is important to keep in mind that isomorphism is a stronger notion than elementary equivalence.

To summarize, the notions we have just introduced include all the following:

- $\mathfrak{A} \subseteq \mathfrak{B}$
- $\mathfrak{A} \prec \mathfrak{B}$

- $\mathfrak{A} \equiv \mathfrak{B}$
- $\mathfrak{A} \cong \mathfrak{B}$

By the end of this semester, I expect you to have all these notions clear, as well as the other ones introduced above.

There are just a few more: an injective homomorphism from $\mathfrak{A}$ to $\mathfrak{B}$ is also called an *embedding* of $\mathfrak{A}$ into $\mathfrak{B}$. (Recall that saying F is injective means that whenever a and b are different, so are $F(a)$ and $F(b)$.) Note that if F is an embedding of $\mathfrak{A}$ into $\mathfrak{B}$, then F is also an isomorphism of $\mathfrak{A}$ with its image in $\mathfrak{B}$, taken as a substructure of $\mathfrak{B}$. An embedding is an *elementary embedding* if the image of $\mathfrak{A}$ is an elementary substructure of $\mathfrak{B}$. Again, I will draw a picture on the board to illustrate this. An *automorphism* of a structure $\mathfrak{A}$ is an isomorphism of $\mathfrak{A}$ with itself.

Here are some examples.

1. $\langle \mathbb{N}, 0, +, \times \rangle$ is a substructure of $\langle \mathbb{R}, 0, +, \times \rangle$. The identity function $F : \mathbb{N} \to \mathbb{R}$ is an embedding.

2. The function $F : \mathbb{N} \to \mathbb{R}$ defined by $F(x) = 0$ is a homomorphism from $\langle \mathbb{N}, 0, + \rangle$ to $\langle \mathbb{R}, 0, + \rangle$.

3. Taken as structures for a language with one binary relation symbol R, there is no isomorphism between $\langle \mathbb{N}, < \rangle$ and $\langle \mathbb{N}, > \rangle$. (Why not?)

4. The function $F : \mathbb{Q} \to \mathbb{Q}$ defined by $F(x) = x/2$ is an automorphism of $\langle \mathbb{Q}, 0, +, < \rangle$.

5. The same function F is *not* an automorphism of $\langle \mathbb{Q}, \times \rangle$. (Why not?)

6. Let $\hat{\mathbb{N}}$ be the structure whose universe consists of strings of symbols "x", "xx", "xxx", ... Let $\hat{0}$ be the string "x", and let $\hat{S}$ be the function which appends an 'x' to any given string. Then $\langle \hat{\mathbb{N}}, \hat{0}, \hat{S} \rangle$ is isomorphic to $\langle \mathbb{N}, 0, S \rangle$.

7. The function $F : \mathbb{R} \to \mathbb{R}$ defined by $F(x) = x^3$ is an automorphism of $\langle \mathbb{R}, 0, < \rangle$ but not $\langle \mathbb{R}, 0, + \rangle$.

8. If $\mathfrak{A}$ is any structure, then the identity function on $|\mathfrak{A}|$ is an isomorphism of $\mathfrak{A}$.

We are now in the position to apply the tools we have developed so far to study these notions more carefully, and, in particular, to demonstrate some of the limitations of first-order logic regarding the two forms of definability discussed in Section 5.5. The next three sections are devoted to that purpose.

7.2 Compactness

The compactness theorem is as follows:

Theorem 7.2.1 *Let Γ be any set of sentences. If every finite subset of Γ has a model, then Γ has a model.*

In slightly different words, if every finite subset of Γ is satisfiable, then Γ is satisfiable. This is an easy consequence of the completeness theorem: if Γ is not satisfiable, then it is inconsistent; but then some finite subset is inconsistent (namely, the finite set of sentences used in a proof of $\perp$ from Γ); and so, by soundness, this finite set is not satisfiable. One can also prove the compactness theorem directly, using a proof that is similar to the proof of the completeness theorem, but replaces "consistent" with "finitely satisfiable."

Despite its simplicity, the compactness theorem is one of the most fundamental and powerful tools in model theory. Let us explore some of its consequences.

First of all, note that it provides a negative answer to one of the questions posed in Chapter 5.

Corollary 7.2.2 *The class of finite structures in the language of equality is not definable.*

Proof. Suppose, for the sake of a contradiction, that the class of finite structures is defined by Γ; that is, $Mod(\Gamma)$ is exactly the class of structures with finite universes. For each n, let φ_n be the sentence that "says" there are at least n elements of the universe, and let Γ' be

$$\Gamma \cup \{\varphi_1, \varphi_2, \varphi_3, \ldots\}$$

Then every finite subset of Γ' is consistent, since every finite subset of Γ' consists of a subset of Γ, together with finitely many sentences φ_i; and any finite structure with a suitably large universe will satisfy these sentences. By the compactness theorem, Γ' is consistent. But every model of Γ' has to be infinite; so Γ has an infinite model, which is a contradiction. □

Actually, the argument, modified only slightly, establishes a much stronger result:

Theorem 7.2.3 *Let Γ be any set of sentences that has arbitrarily large finite models. Then Γ has an infinite model.*

Corollary 7.2.4 *Each of the class of finite groups, finite rings, finite partial orders, etc. is not definable in first-order logic.*

A variation of this trick allows us to show that things like the class of torsion groups and well-orderings are not definable. If you don't know what these are, don't worry about it; we will use a similar argument in a moment to "construct" a nonstandard model of arithmetic.

But first, a few simple observations will allow us to answer another question from Chapter 5. Note that if a class of structures is definable by a finite set of sentences $\{\varphi_1, \ldots, \varphi_k\}$, then it is defined by a single sentence $\varphi_1 \wedge \ldots \wedge \varphi_k$. And if a class of structures is defined by a single sentence σ, the complement of that class is defined by $\neg\sigma$. So we can conclude, for example,

Corollary 7.2.5 *The class of infinite structures in the language of equality cannot be defined by any finite set of sentences.*

If it could, then the class of finite structures would be definable, contrary to the corollary above. (You can find additional information on "finite axiomatizability" on page 116 of van Dalen.)

Finally, let us show that the compactness theorem provides us with "nonstandard" models of arithmetic.

Theorem 7.2.6 *Let $\mathfrak{N}$ be the structure $\langle \mathbb{N}, 0, S, +, \times, < \rangle$. There is a structure $\mathfrak{M}$ such that*

- *$\mathfrak{M}$ is elementarily equivalent to $\mathfrak{N}$*
- *$\mathfrak{M}$ is not isomorphic to $\mathfrak{N}$*

Proof. Let L be the language of $\mathfrak{N}$, and L' be L together with a new constant c. Let Γ be the following set of sentences in L':

$$Th(\mathfrak{N}) \cup \{0 < c, S(0) < 2, S(S(0)) < c, \ldots\}.$$

Every finite subset Γ' of Γ has a model of the form $\langle \mathbb{N}, 0, S, +, \times, <, m \rangle$, where m is a natural number that is large enough to satisfy the finitely many sentences involving c in Γ'. By compactness, Γ has a model,

$$\mathfrak{A} = \langle A, 0^{\mathfrak{A}}, S^{\mathfrak{A}}, +^{\mathfrak{A}}, \times^{\mathfrak{A}}, <^{\mathfrak{A}}, c^{\mathfrak{A}} \rangle.$$

Let $\mathfrak{M}$ be the reduct of $\mathfrak{A}$ to L, obtained by just omitting the interpretation of c. Then we have that

$$\mathfrak{M} \models Th(\mathfrak{N})$$

since $Th(\mathfrak{N})$ is a subset of Γ'. This means that $\mathfrak{M}$ is elementarily equivalent to $\mathfrak{N}$. On the other hand, there is an element a (namely, $c^{\mathfrak{A}}$) in the universe of $\mathfrak{M}$ that is bigger than $0^{\mathfrak{M}}, S(0)^{\mathfrak{M}}, S(S(0))^{\mathfrak{M}}, \ldots$. This implies that $\mathfrak{M}$ is not isomorphic to $\mathfrak{N}$. □

Colloquially, we call this element a a "nonstandard" number in the $\mathfrak{M}$, in contrast to the denotations of $0, S(0), S(S(0)), \ldots$, which are the "standard" numbers in $\mathfrak{M}$. In a certain sense, "nonstandard" means infinitely large. But only in a sense — as far as $\mathfrak{M}$ is concerned, a is a natural number, just like any other!

Incidentally, since every natural number is denoted by a numeral, the argument above shows that $\mathfrak{N}$ is isomorphic to an *elementary substructure* of $\mathfrak{M}$. So a stronger statement of Theorem 7.2.6 is that there is a proper (and non-isomorphic) elementary extension of $\mathfrak{N}$.

And what does such a nonstandard model "look like"? Note that all of the following sentences are true of the natural numbers, and hence true in $\mathfrak{M}$.

1. $<$ is a linear order.
2. Every element has a unique successor.
3. Every element other than 0 has a unique predecessor.
4. Every element is even or odd.
5. For every x and y, if $x > y$ then $x + x > x + y$.
6. If x is even, then there is a number y such that $y + y = x$.
7. There are infinitely many primes.
8. Fermat's last theorem is true.

I will use this information to try to draw a "picture" of $\mathfrak{M}$ on the board. Item 5 can be used to show that there are infinitely many "$\mathbb{Z}$-strips"; and item 6 can be used to show that between any two $\mathbb{Z}$-strips, there is another one.

The same trick can be used to construct "nonstandard models" of any theory of the real numbers, in which there are reals that are smaller than $1, 1/2, 1/3, 1/4, \ldots$. Such numbers can be considered "infinitely small," and, interestingly enough, can be used to make some sense of the informal understanding of "infinitesimals" in the historical development of calculus. Using nonstandard models to justify reasoning about infinitesimals forms the basis of a field known as "nonstandard analysis."

7.3 Definability and automorphisms

We have used compactness to demonstrate some of the limitations of first-order logic, with respect to the definability of classes of structures. Let us now consider the limitations of first-order logic with respect to the definability of relations *within* a given structure.

For this purpose, Theorem 7.1.1 above is a useful tool. Roughly speaking, this theorem implies that the truth of a formula is "preserved" under an automorphism. So to show that a relation is not definable in a structure, it suffices to find an automorphism of the structure for which the given relation is not invariant. For example

Theorem 7.3.1 *There is no first-order formula that defines the positive real numbers in* $\langle \mathbb{R}, < \rangle$.

Proof. It is easy to check that the function $f(x) = x - 10$ is an automorphism of this structure. Now suppose the set of positive numbers is definable by a formula $\varphi(x)$; that is,

$$\{a \in \mathbb{R} \mid a > 0\} = \{a \in \mathbb{R} \mid \langle \mathbb{R}, < \rangle \models \varphi(\bar{a})\}.$$

Then, in particular, we have

$$\langle \mathbb{N}, < \rangle \models \varphi(\bar{5}).$$

By Theorem 7.1.1, we have

$$\langle \mathbb{N}, < \rangle \models \varphi(\overline{f(5)}).$$

But $f(5) = -5$, so this is a contradiction. $\square$

In a similar way we can show the following:

Theorem 7.3.2 *There is no first-order formula that defines addition in* $\langle \mathbb{R}, 0, < \rangle$

Proof. As above, suppose $\varphi(x, y, z)$ defines addition in this structure (that is, the relation "$x + y = z$") and consider the automorphism $f(x) = x^3$. $\square$

As a second example, can you show that $<$ is not definable in $\langle \mathbb{Z}, 0, + \rangle$? I will have you consider additional examples as homework. Before closing this section, I would also like to discuss some more refined methods, that you will *not* be responsible for on the final exam.

The first involves augmenting the "automorphism" trick with compactness. Suppose I want to prove that $<$ is not definable in $\langle \mathbb{N}, 0, S\rangle$. The first thing to do is to look for an automorphism of this structure which misbehaves with respect to $<$; but on this structure there are no automorphisms at all, except for the trivial one.

Using the compactness theorem, though, we can find an elementarily equivalent structure $\mathfrak{A}$ with elements a and b satisfying $\bar{a} \neq \bar{b}$, and

$$\bar{a} \neq 0, \bar{a} \neq S(0), \bar{a} \neq S(S(0)), \ldots, \bar{b} \neq 0, \bar{b} \neq S(0), \bar{b} \neq S(S(0)), \ldots$$

and

$$\bar{a} \neq S(\bar{b}), \bar{a} \neq S(S(\bar{b})), \ldots, \bar{b} \neq S(\bar{a}), \bar{b} \neq S(S(\bar{a})), \ldots$$

A little bit of reflection shows that this structure contains a part that is isomorphic to $\mathbb{N}$, as well as distinct "$\mathbb{Z}$-strips" of which a and b are members. And this new structure *does* have an automorphism, namely, the automorphism which switches the two $\mathbb{Z}$ strips, mapping a to b and vice-versa.

Now suppose $\varphi(x, y)$ defines $<$ on $\langle \mathbb{N}, 0, S\rangle$. Then

$$\langle \mathbb{N}, 0, S\rangle \models \forall x, y\ (x \neq y \rightarrow (\varphi(x, y) \leftrightarrow \neg\varphi(y, x)).$$

Since $\mathfrak{A}$ is elementarily equivalent, the same sentence is true in $\mathfrak{A}$. But if

$$\mathfrak{A} \models \varphi(\bar{a}, \bar{b}),$$

then also

$$\mathfrak{A} \models \varphi(\bar{b}, \bar{a}),$$

since there is an automorphism switching a and b. And this is a contradiction.

Another method of demonstrating undefinability involves using what are called "Ehrenfeucht-Fraissé games." The idea is to show that a given formula cannot define the relation in question by showing that there is a strategy that one can use to "trick" the formula into giving an incorrect answer.

Finally, one can sometimes obtain negative results by bringing computability issues into play. For example, it turns out that one cannot define multiplication in $\langle \mathbb{N}, +\rangle$, because the theory of this structure is decidable (a result due to Presburger, a student of Tarski) whereas the theory of $\langle \mathbb{N}, +, \times\rangle$ is not (this follows from Gödel's incompleteness theorem).

7.4 The Löwenheim-Skolem theorems

The Löwenheim-Skolem theorem is sometimes naively stated as follows:

Theorem 7.4.1 *Let κ be an infinite cardinal, let L be a language of cardinality at most κ, and let Γ be a set of sentences in L. Then if Γ has an infinite model (or even arbitrarily large finite models), it has a model of every cardinality greater than or equal to κ.*

The phrase "greater than or equal to κ" is necessary, since with κ many constants c_i one can simply use the set of sentences $\{c_i \neq c_j\}$ to say that different constants denote different elements of the universe. The proof of the theorem uses the strengthened version of the completeness theorem discussed in Section 6.5.

Proof. Let λ be a cardinal greater than or equal to κ, let L' be a language which adds λ new constants c_i to L , and let Γ' be Γ together with the set of sentences $\{c_i \neq c_j \mid i, j \in \lambda, i \neq j\}$. Since Γ has arbitrarily large models, every finite subset of Γ' is consistent, so Γ' has a model $\mathfrak{A}$ of cardinality at most λ. The sentences $c_i \neq c_j$ guarantee that the cardinality of $\mathfrak{A}$ is at least λ, and so the reduct of $\mathfrak{A}$ to L is the structure we want. □

In fact, the "real" Löwenheim-Skolem theorem comes in two parts, which, taken together, constitute a stronger version of the theorem above. The "upwards" Löwenheim-Skolem theorem is as follows:

Theorem 7.4.2 *Let L be a language, and let $\mathfrak{A}$ be a structure for L. Then if κ is any infinite cardinal that is greater than or equal to both the cardinality of L and the cardinality of $\mathfrak{A}$, then there is a structure $\mathfrak{B}$ of cardinality κ such that $\mathfrak{A} \prec \mathfrak{B}$.*

In other words, the conclusion states that $\mathfrak{A}$ has an elementary extension of size κ. This is essentially theorem 3.3.13 on page 126 in van Dalen. The "downwards" version of the Löwenheim-Skolem theorem is as follows:

Theorem 7.4.3 *Let L be a language, and let $\mathfrak{A}$ be a structure for L. Then if κ is any infinite cardinal that is greater than or equal to the cardinality of L and less than or equal to the cardinality of $\mathfrak{A}$, there is a structure $\mathfrak{B}$ of cardinality κ such that $\mathfrak{B} \prec \mathfrak{A}$.*

In other words, the conclusion states that $\mathfrak{A}$ has an elementary substructure of size κ. This is theorem 3.3.12 in van Dalen.

To prove the upwards version, it is useful to invoke a little more machinery. Recall that if $\mathfrak{A}$ is a structure for a language L, in order to define the semantics for $\mathfrak{A}$ we passed to a larger language $L(\mathfrak{A})$, which has a new constant symbol $\bar{a}$ for every element a in the universe of $\mathfrak{A}$. We can now distinguish between the following sets of sentences, each of which, in a sense, provides a "description" of $\mathfrak{A}$:

- $Th(\mathfrak{A}) = \{\varphi \in L \mid \mathfrak{A} \models \varphi\}$.

  This is the set of first-order statements in L that are true of $\mathfrak{A}$. We have called this the "theory of $\mathfrak{A}$."

- $Diag(\mathfrak{A}) = \{\varphi \in L(\mathfrak{A}) \mid \varphi$ is atomic or negation atomic, and $\mathfrak{A} \models \varphi\}$.

  This set is called the "diagram of $\mathfrak{A}$," and gives a full description of $\mathfrak{A}$. For example, it tells you which relations are true of which elements, which elements map to which other elements under the interpretations of the function symbols of L, and so on.

- $Th(\hat{\mathfrak{A}}) = \{\varphi \in L(\mathfrak{A}) \mid \mathfrak{A} \models \varphi\}$.

  This is sometimes called the "elementary diagram of $\mathfrak{A}$"; it includes not just a record of the atomic truths of $\mathfrak{A}$, but rather an evaluation of *all* first-order sentences in parameters from $|\mathfrak{A}|$. If we let $\hat{\mathfrak{A}}$ be the structure for $L(\mathfrak{A})$ which expands $\mathfrak{A}$ by interpreting each constant $\bar{a}$ by a, this is just the theory of $\hat{\mathfrak{A}}$ in the sense above.

By definition, we have that $\mathfrak{A} \equiv \mathfrak{B}$ if and only if $Th(\mathfrak{A}) = Th(\mathfrak{B})$. In addition, it is not hard to prove the following:

Proposition 7.4.4 *Let L be a language and let $\mathfrak{A}$ and $\mathfrak{B}$ be structures for L. The following are equivalent:*

1. *There is an embedding of $\mathfrak{A}$ into $\mathfrak{B}$.*
2. *$\mathfrak{A}$ is isomorphic to a substructure of $\mathfrak{B}$.*
3. *$\mathfrak{B}$ has an expansion which is a model of the diagram of $\mathfrak{A}$.*

The proposition remains true if one adds the word "elementary" to each of the three clauses:

Proposition 7.4.5 *Let L be a language and let $\mathfrak{A}$ and $\mathfrak{B}$ be structures for L. The following are equivalent:*

1. *There is an elementary embedding of $\mathfrak{A}$ into $\mathfrak{B}$.*

2. *$\mathfrak{A}$ is isomorphic to a elementary substructure of $\mathfrak{B}$.*

3. *$\mathfrak{B}$ has an expansion which is a model of the elementary diagram of $\mathfrak{A}$.*

The upwards Löwenheim-Skolem theorem then follows from the first theorem in this section, provided we take Γ to be the elementary diagram of $\mathfrak{A}$. In contrast, the downwards version of the theorem isn't really a theorem about logic, but, rather, a theorem about obtaining structures that are suitably "closed" under some given operations. The proof involves taking a sufficiently large subset of the original structure $\mathfrak{A}$ and closing it under suitable "Skolem functions." This proof is given on page 144 of van Dalen, but I will not discuss it in class.

The two Löwenheim-Skolem theorems challenge our intuitions regarding cardinality. For example, the first implies that there are uncountable elementary extensions of $\langle \mathbb{N}, 0, S, +, \times \rangle$, a structure that seems to scream "countable." In the other direction, we can say that if Zermelo-Fraenkel set theory is consistent, it has a countable model, despite the fact that in this theory one can prove that there are uncountably many real numbers. This is sometimes known as "Skolem's paradox"; you should think about this, and figure out why it really isn't a paradox at all.

7.5 Discussion

The preceeding sections have illustrated some of the limitations of first-order logic. For example, let $\mathfrak{N}$ denote the structure $\langle \mathbb{N}, 0, S, +, \times \rangle$, and suppose we aim to "describe" $\mathfrak{N}$ with a set of first-order sentences. Of course, if $\mathfrak{N}$ satisfies some set of sentences, then so will any structure that is isomorphic to it. This means that we cannot hope to characterize $\mathfrak{N}$ exactly, but rather, at best, up to isomorphism; but this is a minor complaint. More seriously, the Löwenheim-Skolem theorem tells us that no matter what we do, our set of sentences will have models of every infinite cardinality. And even if we restrict our attention to countable models, the compactness theorem tells us that there will "nonstandard" models that are not isomorphic to the structure we have in mind.

A theory T is said to be *complete* if it has the property that for every formula φ, either φ or $\neg\varphi$ is in T. It is not too hard to see that every maximally consistent set of sentences is a complete, consistent theory, and vice-versa. In particular, if $\mathfrak{A}$ is any structure, $Th(\mathfrak{A})$ is a complete, consistent theory. It is a disappointing fact that in most cases a complete theory does not pick out a unique structure, even up to isomorphism.

One can think of a "logic" more generally as consisting of a set of syntactic rules together with an associated semantics, and consider the extent to which syntactic objects can be used to pick out classes of structures. When a set of sentences describes a structure that is unique up to isomorphism, the description is said to be *categorical.* In these terms, we can say that the Löwenheim-Skolem theorem shows that no first-order set of sentences with an infinite model is categorical.

A weaker requirement is that the set of sentences pick out a unique structure (again, up to isomorphism) from among the class of structures of a given cardinality. Here, the situation is much improved: there *are* first-order theories that are categorical for various cardinalities. The Los-Vaught theorem provides a condition that is necessary for a theory to have this property, though it is not always sufficient.

Theorem 7.5.1 *Let L be a language of cardinality κ, and let T be a theory in L that is categorical in some infinite cardinal λ greater than or equal to κ. Then T is complete.*

Proof. If T is not complete, then there is a sentence φ such that neither φ nor $\neg\varphi$ is in T. By the completeness theorem and the Löwenheim-Skolem theorem, both $T \cup \{\varphi\}$ and $T \cup \{\neg\varphi\}$ have models of cardinality λ.; but these models can't possibly be isomorphic. $\square$

To see that the condition above is not always sufficient, recall that the complete theory of $\mathfrak{N}$ is not categorical for countable structures. But, for example, the theory of algebraically closed fields of characteristic 0 is categorical at every uncountable cardinality, and the theory of real closed fields is categorical at every infinite cardinality. In fact, the two structures just mentioned can be represented with an explicit set of axioms; and the Los-Vaught theorem then tells us that the associated theory is complete (and hence decidable).

Categoricity is a *semantic* issue: we can always make a theory categorical by strengthening the semantics. For example, for the language of arithmetic, we can just make the "semantic" stipulation that the first-order quantifiers range over natural numbers, in effect "fixing" the intended structure as part of the semantics. Perhaps less artificially, we will see in the next chapter that the structure $\mathfrak{N}$ has a natural categorical axiomatization in second-order arithmetic, relative to the "full" second-order semantics. Given this fact, why are logicians so keen on first-order logic?

The answer is that using a stronger semantics invariably involves a trade-off. Suppose we are looking for a semantics for which we gives us a categorical description of the natural numbers. The construction we used to obtain a nonstandard model of arithmetic used only compactness, and the fact that assertions of the form $c > 0, c > S(0), c > S(S(0))$ are representable in the logic; this suggests that any reasonable semantics with a categorical description of $\mathfrak{N}$ will fail to be compact. Since compactness followed from the completeness theorem, we have to give this up as well.

Of course, we can limit the semantics for the language of arithmetic to the single structure $\mathfrak{N}$ as described above, and define a proof system in which every true statement of arithmetic is simply an axiom. This trivially gives us a semantics in which the empty set affords a categorical description of the natural numbers, and for which the proof system is sound and complete. But this proof system and this semantics are not very satisfying. For example, we might want the proof system and the axiomatization of $\mathfrak{N}$ to be "effective," in the sense that we can algorithmically determine whether or not a given sequence of symbols represents an axiom or a proof. We can certainly come up with interesting proof systems for $\mathfrak{N}$ which have these properties. But if we identify effectivity with Turing computablility, Gödel's incompleteness theorem shows us that any such proof system is bound to be either inconsistent, or incomplete, despite any apparent utility.

In short, you can't have everything. If you wish to develop mathematical tools to study structures and their properties, it is natural to consider more general logics and their semantics. But if you want a proof system that is effective, sound, and complete, then it is hard to beat first-order logic, even though it means giving up having a categorical (or even just complete) description of $\mathfrak{N}$.

7.6 Other model-theoretic techniques

In this section I would like to give you a hint as to some of the other kinds of things that model theorists do. We do not have time to pursue these activities in more depth, but you can find more information in van Dalen or in any introductory textbook on model theory, and so I will rest content in having pointed out the way.

First of all, one has what are known as *preservation* theorems. For example, a class of structures for a given language L is said to be *closed under substructures* if whenever $\mathfrak{B}$ is in the class and $\mathfrak{A} \subset \mathfrak{B}$, then $\mathfrak{A}$ is in the class also. A sentence in L is said to be *universal* if it is of the form

$$\forall x_1, \ldots, x_k \ \varphi$$

where φ is quantifier-free. It is not hard to see that if φ is universal, $\mathfrak{B} \models \varphi$, and $\mathfrak{A} \subset \mathfrak{B}$, then $\mathfrak{A} \models \varphi$ as well. As a result, if a theory T has a universal set of axioms Γ, then $Mod(T)$ is closed under substructures. It is a little bit more surprising that the converse also holds:

Theorem 7.6.1 *If T is any theory, then T has a universal set of axioms if and only if $Mod(T)$ is closed under substructures.*

This provides an interesting relationship between a syntactic property that a given theory may or may not have, and a semantic property of the corresponding class of models.

(Here are some hints as to proving the converse direction of the theorem. Suppose $Mod(T)$ is closed under substructures, and let

$$\Gamma = \{\varphi \mid \varphi \text{ is universal sentence and } T \vdash \varphi\}.$$

I claim any model of Γ is also a model of T, so that Γ provides the necessary axiomatization. It suffices to show the contrapositive, which states that if some structure is not a model of T, then it is not a model of Γ either.

To show this last fact, suppose $\mathfrak{A}$ is a structure that does not model T. Since $Mod(T)$ is closed under substructures, $\mathfrak{A}$ is not a substructure of any model of T. Let $diag(\mathfrak{A})$, the "diagram of $\mathfrak{A}$," be the set of closed atomic sentences in $L(\mathfrak{A})$ that are true in $\mathfrak{A}$. The fact the $\mathfrak{A}$ is not a substructure of any model of T means that $diag(\mathfrak{A}) \cup T$ is inconsistent. By compactness, some finite subset of this set is inconsistent. Use this to find a universal sentence φ such that $T \vdash \varphi$, but φ is false in $\mathfrak{A}$. This shows that $\mathfrak{A}$ is not a model of Γ.)

Many model-theoretic methods involve the use of *Skolem functions*, which are used, for example, in the proof of the downward Löwenheim-Skolem theorem. See the discussion in Section 3.4 of van Dalen's book.

In the previous section I alluded to the fact that the complete theory of $\mathfrak{N}$ is undecidable; in fact, it turns out that any subtheory of $Th(\mathfrak{N})$ is also undecidable (including "pure logic," which is to say, the theory axiomatized by $\emptyset$ in the language of arithmetic!). But this negative result can be balanced with some positive ones: one generally wants to know when a given set of axioms yields a decidable theory, or when a given structure has a complete theory that is decidable.

Some theories have the property that every formula is equivalent, over the theory, to a quantifier-free one. This syntactic property has a semantic one, called *model completeness*; a theory is called model complete if whenever $\mathfrak{B} \models T$ and $\mathfrak{A} \subset \mathfrak{B}$ then $\mathfrak{B} \prec \mathfrak{A}$. If you have an effective procedure for eliminating the quantifiers, as well as a decision procedure to determine which quantifier-free formulas are provable, then taken together you have a decision procedure for entire theory.

Generally speaking, model theorists study issues relating to cardinality, definability and undefinability, automorphisms, and preservation theorems, not just in the context of first-order logic, but also with respect to variants and extensions thereof.

This is also a good place to mention the field of finite model theory, which is concerned with similar issues, but in which one restricts attention to finite structures only. Though many model-theoretic techniques carry over to the study of finite models, for the most part this is an entirely different ballgame, and is much more closely related to finite combinatorics. In particular, there are interesting and striking relationships to the theory of computational complexity.

Chapter 8

Beyond First-Order Logic

8.1 Overview

In this chapter I will discuss some extensions, fragments, and variations of first-order logic. In other words, we will consider other "logics," where I am using this last word in the vague but technical sense: a logic typically consists of the formal specification of a language, usually, but not always, a deductive system, and usually, but not always, an intended semantics. But the technical use of the term raises an obvious question: what do the logics below have to do with the word "logic," used in the intuitive or philosophical sense? All of the systems described below are designed to model reasoning of some form or another; can we say what makes them logical?

No easy answers are forthcoming. The word "logic" is used in different ways and in different contexts, and the notion, like that of "truth," has been analyzed from numerous philosophical stances. For example, one might take the goal of logical reasoning to be the determination of which statements are necessarily true, true a priori, true independent of the interpretation of the nonlogical terms, true by virtue of their form, or true by linguistic convention; and each of these conceptions requires a good deal of clarification. Even if one restricts one's attention to the kind of "mathematical" logic I described in Chapter 1, there is little agreement as to its scope. For example, in the *Principia Mathematica* Russell and Whitehead tried to develop mathematics on the basis of logic, in the *logicist* tradition begun by Frege. Their system of logic was a form of higher-type logic similar to the one described below. In the end they were forced to introduce axioms which, by most standards, do not seem purely logical (notably, the axiom of infinity, and the axiom of reducibility), but one might nonetheless hold that some forms of higher-order reasoning should be accepted as logical. In contrast, Quine, whose ontology does not admit "propositions" as legitimate objects of discourse, argues that second-order and higher-order logic are really manifestations of set theory in sheep's clothing; in other words, systems involving quantification over predicates are not purely logical.

For now, it is best to leave such philosophical issues for a rainy day, and simply think of the systems below as formal idealizations of various kinds of reasoning, logical or otherwise.

8.2 Many-sorted logic

In first-order logic, variables and quantifiers range over a single universe. But it is often useful to have multiple (disjoint) universes: for example, you might want to have a universe of numbers, a universe of geometric objects, a universe of functions from numbers to numbers, a universe of abelian groups, and so on.

Many-sorted logic provides this kind of framework. One starts with a list of "sorts" — the "sort" of an object indicates the "universe" it is supposed to inhabit. One then has variables and quantifiers for each sort, and (usually) an equality symbol for each sort. Functions and relations are also "typed" by the sorts of objects they can take as arguments. Otherwise, one keeps the usual rules of first-order logic, with versions of the quantifier-rules repeated for each sort.

For example, to study international relations we might choose a language with two sorts of objects, French citizens and German citizens. We might have a unary relation, "drinks wine," for objects of the first sort; another unary relation, "eats wurst," for objects of the second sort; and a binary relation, "forms a multinational married couple," which takes two arguments, where the first argument is of the first sort and the second argument is of the second sort. If we use variables a, b, c to range over French citizens and x, y, z to range over German citizens, then

$$\forall a, x\ (\mathit{MarriedTo}(a, x) \rightarrow (\mathit{DrinksWine}(a) \vee \neg \mathit{EatsWurst}(x))$$

asserts that if any French person is married to a German, either the French person drinks wine or the German doesn't eat wurst.

Many-sorted logic can be embedded in first-order logic in a natural way, by lumping all the objects of the many-sorted universes together into one first-order universe, using unary relations to keep track of the sorts, and relativizing quantifiers. For example, the first-order language corresponding to the example above would have unary relations "German" and "French," in addition to the other relations described, with the sort requirements erased. A sorted quantifier $\forall x\ \varphi$, where x is a variable of the German sort, translates to

$$\forall x\ (German(x) \rightarrow \varphi).$$

We need to add axioms that insure that the sorts are separate — e.g. $\forall x\ \neg(German(x) \land French(x))$ — as well as axioms that guarantee that "drinks wine" only holds of objects satisfying the predicate $French(x)$, etc. With these conventions and axioms, it is not difficult to show that many-sorted statements translate to first-order statements, and many-sorted proofs translate to first-order proofs. Also, many-sorted models "translate" to corresponding first-order models and vice-versa, so we also have a completeness theorem for many-sorted logic.

8.3 Second-order logic

The language of second-order logic allows one to quantify not just over a universe of individuals, but over relations on that universe as well. Given a first-order language L, for each k one adds variables R which range over k-ary relations, and quantification over those variables. If R is a variable for a k-ary relation, and t_1 to t_k are ordinary (first-order) terms, $R(t_1, \ldots, t_k)$ is an atomic formula. Otherwise, the set of formulas is defined just as in the case of first-order logic, with additional clauses for second-order quantification. Note that one only has equality for first-order terms: if R and S are relation variables of the same arity k, we can define $R = S$ to be an abbreviation for

$$\forall x_1, \ldots, x_k\ (R(x_1, \ldots, x_k) \leftrightarrow S(x_1, \ldots, x_k)).$$

The rules for second-order logic simply extend the quantifier-rules to the new second order variables. Here, however, one has to be a little bit careful to explain how these variables interact with the relation symbols of L, and with formulas of L more generally. At the bare minimum, relation variables count as terms, so one has inferences of the form

$$\frac{\psi(R)}{\exists R\ \psi(R)}$$

But if L is the language of arithmetic with a constant relation symbol $<$, one would also expect the following inference to be valid:

$$\frac{x < y}{\exists R\ R(x, y)}$$

or for a given formula φ,

$$\frac{\varphi(x_1, \ldots, x_k)}{\exists R\, R(x_1, \ldots, x_k)}$$

More generally, we might want to have rules of the form

$$\frac{\psi[\lambda \vec{x}\ \varphi(\vec{x})/R]}{\exists R\ \phi}$$

where $\psi[\lambda \vec{x}\ \varphi(\vec{x})/R]$ denotes the result of replacing every atomic formula of the form $R(t_1, \ldots, t_k)$ by $\varphi(t_1, \ldots, t_k)$. van Dalen points out that this last rule is equivalent to having a schema of *comprehension axioms* of the form

$$\exists R\ \forall x_1, \ldots, x_k\ (\varphi(x_1, \ldots, x_k) \leftrightarrow R(x_1, \ldots, x_k)),$$

one for each formula φ in the second-order language, in which R is not a free variable. (Exercise: show that if R is allowed to occur in φ, this schema is inconsistent!)

When logicians refer to the "axioms of second-order logic" they usually mean the full set of rules listed on page 149 of van Dalen, or, equivalently, the minimal extension of first-order logic together with the comprehension schema. But it is often interesting to study weaker subsystems of these axioms and rules. For example, note that in its full generality the axiom schema of comprehension is *impredicative*: it allows one to assert the existence of a relation $R(x_1, \ldots, x_k)$ that "defined" by a formula with second-order quantifiers; and these quantifiers range over the set of all such relations — a set which includes R itself! If one prohibits the use of second-order quantifiers in the formula φ, one has a *predicative* form of comprehension, which is somewhat weaker.

From the semantic point of view, one can think of a second-order structure as consisting of a first-order structure for the language, coupled with a set of relations on the universe over which the second-order quantifiers range (more precisely, for each k there is a set of relations of arity k). Of course, if comprehension is included in the proof system, then we have the added requirement that there are enough relations in the "second-order part" to satisfy the comprehension axioms — otherwise the proof system is not sound! One easy way to insure that there are enough relations around is to take the second-order part to consist of *all* the relations on the first-order part. Such a model is called a *full* model, and, in a sense, is really the "intended model" for the language. If we restrict our attention to full models we have what is known as the *full* second-order semantics. In that case, specifying a structure boils down to specifying the first-order part, since the contents of the second-order part follow from that implicitly.

To summarize, there is some ambiguity when talking about second-order logic. In terms of the proof system, one might have in mind either

1. A "Minimal" second-order proof system, together with some comprehension axioms.

2. The "standard" second-order proof system, with full comprehension.

In terms of the semantics, one might be interested in either

1. The "weak" semantics, where a model consists of a first-order part, together with a second-order part big enough to satisfy the comprehension axioms.

2. The "standard" second-order semantics, in which one considers full models only.

When logicians do not specify the proof system or the semantics they have in mind, they are usually refering to the second item on each list. The advantage to using this semantics is that, as we will see below, it gives us categorical descriptions of many natural mathematical structures; at the same time, the proof system is quite strong, and sound for this semantics. The drawback is that the proof system is *not* complete for the semantics; in fact, *no* effectively given proof system is complete for the full second-order semantics. On the other hand, we will see that the proof system *is* complete for the weakened semantics; this implies that if a sentence is not provable, then there is *some* model, not necessarily the full one, in which it is false.

The language of second-order logic is quite rich. One can identify unary relations with subsets of the universe, and so in particular you can quantify over these sets; for example, one can express induction for the natural numbers with a single axiom

$$\forall R \; (R(0) \wedge \forall x \; (R(x) \rightarrow R(s(x))) \rightarrow \forall x \; R(x)).$$

If one takes the language of arithmetic to have symbols $0, s, +, \times$ and $<$, one can add the following axioms to describe their behavior:

1. $\neg s(x) = 0$

2. $s(x) = s(y) \rightarrow x = y$

3. $x + 0 = x$

4. $x + s(y) = s(x + y)$

5. $x \times 0 = 0$

6. $x \times s(y) = x \times y + x$

7. $x < y \leftrightarrow \exists z\ (y = x + s(z))$

It is not difficult to show that these axioms, together with the axiom of induction above, provide a categorical description of the structure $\mathfrak{N}$ discussed at the end of the last chapter, provided we are using the full second-order semantics. Given any model $\mathfrak{A}$ of these axioms, define a function f from $\mathbb{N}$ to the universe of $\mathfrak{A}$ using ordinary recursion on $\mathbb{N}$, so that $f(0) = 0^{\mathfrak{A}}$ and $f(x + 1) = s^{\mathfrak{A}}(f(x))$. Using ordinary induction on $\mathbb{N}$ and the fact that axioms 1 and 2 hold in $\mathfrak{A}$, we see that f is injective. To see that f is surjective, let P be the set of elements of $|\mathfrak{A}|$ that are in the image of f. Since $\mathfrak{A}$ is full, P is in the second-order universe. By the construction of f, we know that $0^{\mathfrak{A}}$ is in P, and that P is closed under $s^{\mathfrak{A}}$. The fact that the induction axiom holds in $\mathfrak{A}$ (in particular, for P) guarantees that P is equal to the entire first-order universe of $\mathfrak{A}$. This shows that f is a bijection. Showing that f is a homomorphism is no more difficult, using ordinary induction on $\mathbb{N}$ repeatedly.

In set-theoretic terms, a function is just a special kind of relation; for example, a unary function f can be identified with a binary relation R satisfying $\forall x\ \exists! y\ R(x, y)$. As a result, one can quantify over functions too. Using the full semantics, one can then define the class of infinite structures to be the class of structures $\mathfrak{A}$ for which there is an injective function from the universe of $\mathfrak{A}$ to a proper subset of itself.

In addition, one can define the class of finite structures and the class of well-orderings. I will leave the details to you as a homework. In class, though, I will show how to use the former to define the class of finite structures whose universe if even. More strikingly, one can provide a categorical description of the real numbers as a complete ordered field containing the rationals.

That's the good news; now for the bad. I have already mentioned that there is no effective proof system that is complete for the full second-order semantics. For better or for worse, many of the properties of first-order logic are absent, including compactness and the Löwenheim-Skolem theorems.

On the other hand, if one is willing to give up the full second-order semantics in terms of the weaker one, then the minimal second-order proof system is complete for this semantics. In other words, if we read $\vdash$ as "proves in the minimal system" and $\models$ as "logically implies in the weaker semantics", we can show that whenever $\Gamma \models \varphi$ then $\Gamma \vdash \varphi$. If one wants to include specific comprehension axioms in the proof system, one has to restrict the semantics to second-order structures that satisfy these axioms: for example, if Δ consists of a set of comprehension axioms (possibly all of them), we have that if $\Gamma \cup \Delta \models \varphi$, then $\Gamma \cup \Delta \vdash \varphi$. In particular, if φ is not provable using the comprehension axioms we are considering, then there is a model of $\neg\varphi$ in which these comprehension axioms nonetheless hold.

The easiest way to see that the completeness theorem holds for the weaker semantics is to think of second-order logic as a many-sorted logic, as follows. One sort is interpreted as the ordinary "first-order" universe, and then for each k we have a universe of "relations of arity k." We take the language to have built-in relation symbols "$true_k(R, x_1, \ldots, x_k)$" which is meant to assert that R holds of $x_1, \ldots, x_k$, where R is a variable of the sort "k-ary relation" and x_1 to x_k are objects of the first-order sort.

With this identification, the weak second-order semantics is essentially the usual semantics for many-sorted logic; and we have already observed that many-sorted logic can be embedded in first-order logic. Modulo the translations back and forth, then, the weaker conception of second-order logic is really a form of first-order logic in disguise, where the universe contains both "objects" and "relations" governed by the appropriate axioms.

8.4 Higher-order logic

Passing from first-order logic to second-order logic enabled us to talk about sets of objects in the first-order universe, within the formal language. Why stop there? For example, third-order logic should enable us to deal with sets of sets of objects, or perhaps even sets which contain both objects and sets of objects. And fourth-order logic will let us talk about sets of objects of that kind. As you may have guessed, one can iterate this idea arbitrarily.

In practice, Higher-order logic is often formulated in terms of functions instead of relations. (Modulo the natural identifications, this difference is inessential.) Given some basic "sorts" $A, B, C, \ldots$ (which we will now call "types"), we can create new ones by stipulating

> If σ and τ are finite types then so is $\sigma \to \tau$.

Think of types as syntactic "labels," which classify the objects we want in our universe; $\sigma \to \tau$ describes those objects that are functions which take objects of type σ to objects of type τ. For example, we might want to have a type Ω of truth values, "true" and "false," and a type $\mathbb{N}$ of natural numbers. In that case, you can think of objects of type $\mathbb{N} \to \Omega$ as unary relations, or subsets of $\mathbb{N}$; objects of type $\mathbb{N} \to \mathbb{N}$ are functions from natural numers to natural numbers; and objects of type $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ are "functionals," that is, higher-type functions that take functions to numbers.

As in the case of second-order logic, one can think of higher-order logic as a kind of many-sorted logic, where there is a sort for each type of object we want to consider. But it is usually clearer just to define the syntax of higher-type logic from the ground up. For example, we can define a set of finite types inductively, as follows:

1. $\mathbb{N}$ is a finite type

2. If σ and τ are finite types, then so is $\sigma \to \tau$

3. If σ and τ are finite types, so is $\sigma \times \tau$

Intuitively, $\mathbb{N}$ denotes the type of the natural numbers, $\sigma \to \tau$ denotes the type of functions from σ to τ, and $\sigma \times \tau$ denotes the type of pairs of objects, one from σ and one from τ. We can then define a set of terms inductively, as follows:

1. For each type σ, there is a stock of variables $x, y, z, \ldots$ of type σ

2. 0 is a term of type $\mathbb{N}$

3. S (successor) is a term of type $\mathbb{N} \to \mathbb{N}$

4. If s is a term of type σ, and t is a term of type $\mathbb{N} \to (\sigma \to \sigma)$, then R_{st} is a term of type $\mathbb{N} \to \sigma$

5. if s is a term of type $\tau \to \sigma$ and t is a term of type τ, then $s(t)$ is a term of type σ

6. if s is a term of type σ and x is a variable of type τ, then $\lambda x\ s$ is a term of type $\tau \to \sigma$

7. if s is a term of type σ and t is a term of type τ, then $\langle s, t \rangle$ is a term of type $\sigma \times \tau$

8. if s is a term of type $\sigma \times \tau$ then $p_1(s)$ is a term of type σ and $p_2(s)$ is a term of type τ

Intuitively, R_{st} denotes the function defined recursively by

$$\begin{array}{rcl} R_{st}(0) & = & s \\ R_{st}(S(x)) & = & t(x, R_{st}(x)), \end{array}$$

$\langle s, t\rangle$ denotes the pair whose first component is s and whose second component is t, and $p_1(s)$ and $p_2(s)$ denote the first and second elements ("projections") of s. Finally, $\lambda x\ s$ denotes the function f defined by

$$f(x) = s$$

for any x of type σ; so item 6 gives us a form of comprehension, enabling us to define functions using terms. Formulas are built up from equality assertions $s = t$ between terms of the same type, the usual propositional connectives, and higher-type quantification. One can then take the axioms of the system to be the basic equations governing the terms defined above, together with the usual rules of logic with quantifiers and equality.

If one augments the finite type system with a type Ω of truth values, one has to include axioms which govern its use as well. In fact, if one is clever, one can get rid of complex formulas entirely, replacing them with terms of type Ω! The proof system can then be modified accordingly.

As in the case of second-order logic, there are different versions of higher-type semantics that one might want to use. In the full version, variables of type $\sigma \to \tau$ range over the set of *all* functions from the objects of type σ to objects of type τ. As you might expect, this semantics is too strong to admit a complete, effective proof system. But one can consider a weaker semantics, in which a model consists of sets of elements T_τ for each type τ, together with appropriate operations for application, projection, etc. If the details are carried out correctly, one can obtain completeness theorems for the kinds of proof systems I have just described.

Higher-type logic is attractive because it provides a framework in which we can embed a good deal of mathematics in a natural way: starting with $\mathbb{N}$, one can define real numbers, continuous functions, and so on. It is also particularly attractive in the context of intuitionistic logic, since the types have clear "constructive" intepretations. In fact, one can develop constructive versions of higher-type semantics (based on intuitionistic, rather than classical logic) that clarify these constructive interpretations quite nicely, and are, in many ways, more interesting than the classical counterparts.

8.5 Intuitionistic logic

In constrast to second-order and higher-order logic, intuitionistic first-order logic represents a restriction of the classical version, intended to model a more "constructive" kind of reasoning. The following examples may serve to illustrate some of the underlying motivation.

Suppose I came up to you one day and announced that I had determined a natural number x, with the property that if x is prime, the Riemann hypothesis is true, and if x is composite, the Riemann hypothesis is false. Great news! Whether the Riemann hypothesis is true or not is one of the big open questions of mathematics, and here I seem to have reduced the problem to one of calculation, that is, to the determination of whether a specific number is prime or not.

What is the magic value of x? Let me describe it as follows: x is the natural number that is equal to 7 if the Riemann hypothesis is true, and 9 otherwise.

Angrily, you demand your money back. From a classical point of view, the description above does in fact determine a unique value of x; but what you really want is a value of x that is given *explicitly.*

To take another, perhaps less contrived example, consider the following question. We know that it is possible to raise an irrational number to a rational power, and get a rational result. For example, $\sqrt{2}^2 = 2$. What is less clear is whether or not it is possible to raise an irrational number to an *irrational* power, and get a rational result. The following theorem answers this in the affirmative:

Theorem 8.5.1 *There are irrational numbers a and b such that a^b is rational.*

Proof. Consider $\sqrt{2}^{\sqrt{2}}$. If this is rational, we are done: we can let $a = b = \sqrt{2}$. Otherwise, it is irrational. Then we have

$$(\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = \sqrt{2}^{\sqrt{2}\times\sqrt{2}} = \sqrt{2}^2 = 2,$$

which is certainly rational. So, in this case, let a be $\sqrt{2}^{\sqrt{2}}$, and let b be $\sqrt{2}$. □

Does this constitute a valid proof? Most mathematicians feel that it does. But again, there is something a little bit unsatisfying here: we have proved the existence of a pair of real numbers with a certain property, without being able to say *which* pair of numbers it is.

Intuitionistic logic is designed to model a kind of reasoning where moves like this are disallowed. Proving the existence of an x satisfying $\varphi(x)$ means that you have a specific x, and a proof that it satisfies P; proving that φ or ψ holds means that you can prove one or the other.

Formally speaking, intuitionistic first-order logic is what you get if you omit the rule RAA from our natural deduction system. Similarly, there are intuitionistic versions of second-order or higher-order logic. From the mathematical point of view, these are just formal deductive systems, but, as already noted, they are intended to model a kind of mathematical reasoning. One can take this to be the kind of reasoning that is justified on a certain philosophical view of mathematics (such as Brouwer's intuitionism); one can take it to be a kind of mathematical reasoning which is more "concrete" and satisfying (along the lines of Bishop's constructivism); and one can argue about whether or not the formal description captures the informal motivation. But whatever philosophical positions we may hold, we can study intuitionistic logic as a formally presented logic; and for whatever reasons, many mathematical logicians find it interesting to do so.

There is an informal constructive interpretation of the intuitionist connectives, usually known as the Brouwer-Heyting-Kolmogorov interpretation. It runs as follows: a proof of $\varphi \wedge \psi$ consists of a proof of φ paired with a proof of ψ; a proof of $\varphi \vee \psi$ consists of either a proof of φ, or a proof of ψ, where we have explicit information as to which is the case; a proof of $\varphi \rightarrow \psi$ consists of a procedure, which transforms a proof of φ to a proof of ψ; a proof of $\forall x\, \varphi(x)$ consists of a procedure, which returns a proof of $\varphi(x)$ for any value of x; and a proof of $\exists x\, \varphi(x)$ consists of a value of x, together with a proof that this value satisfied φ. One can describe the interpretation in computational terms known as the "Curry-Howard isomorphism" or the "formulas-as-types paradigm": think of a formula as specifying a certain kind of data type, and proofs as computational objects of these data types that enable us to see that the corresponding formula is true.

Intuitionistic logic is often thought of as being classical logic "minus" the law of the excluded middle. This following theorem makes this more precise.

Theorem 8.5.2 *Intuitionistically, the following axiom schemata are equivalent:*

1. *RAA*

2. $\varphi \vee \neg\varphi$

3. $\neg\neg\varphi \rightarrow \varphi$

Obtaining instances of one schema from either of the others is a good exercise in intuitionistic logic.

The first deductive systems for intuitionistic propositional logic, put forth as formalizations of Brouwer's intuitionism, are due, independently, to Kolmogorov, Glivenko, and Heyting. The first formalization of intuitionistic first-order logic (and parts of intuitionist mathematics) is due to Heyting. Though a number of classically valid schemata do not intuitionistically valid, many are; on page 160 of van Dalen you will find a long list of examples.

The *double-negation translation* describes an important relationship between classical and intuitionist logic. It is defined inductively follows (think of φ^N as the "intuitionist" translation of the classical formula φ):

$$\begin{aligned}
\varphi^N &\equiv \neg\neg\varphi \quad \text{for atomic formulas } \varphi \\
(\varphi \wedge \psi)^N &\equiv \varphi^N \wedge \psi^N \\
(\varphi \vee \psi)^N &\equiv \neg\neg(\varphi^N \vee \psi^N) \\
(\varphi \to \psi)^N &\equiv \varphi^N \to \psi^N \\
(\forall x\ \varphi)^N &\equiv \forall x\ \varphi^N \\
(\exists x\ \varphi)^N &\equiv \neg\neg\exists x\ \varphi^N
\end{aligned}$$

Kolmogorov and Glivenko had versions of this translation for propositional logic; for predicate logic, it is due to Gödel and Genzten, independently. We have

Theorem 8.5.3 *1. $\varphi \leftrightarrow \varphi^N$ is provable classically*

2. If φ is provable classically, then φ^N is provable intuitionistically.

We can now envision the following dialogue. Classical mathematician: "I've proved φ!" Intuitionist mathematician: "Your proof isn't valid. What you've really proved is φ^N." Classical mathematician: "Fine by me!" As far as the classical mathematician is concerned, the intuitionist is just splitting hairs, since the two are equivalent. But the intuitionist insists there is a difference.

Note that the above translation concerns pure logic only; it does not address the question as to what the appropritiate *nonlogical* axioms are for classical and intuitionistic mathematics, or what the relationship is between them. But the following slight extension of the theorem above provides some useful information:

Theorem 8.5.4 *If Γ proves φ classically, Γ^N proves φ^N intuitionistically.*

In other words, if φ is provable from some hypotheses classically, then φ^N is provable from their double-negation translations. This is Theorem 5.2.8 on page 164 of van Dalen.

To show that a sentence or propositional formula is intuitionistically valid, all you have to do is provide a proof. But how can you show that it is not valid? For that purpose, we need a semantics that is sound, and preferrably complete. For that purpose, a semantics due to Kripke nicely fits the bill.

We can play the same game we did for classical logic: define the semantics, and prove soundness and completeness. It is worthwhile, however, to note the following distinction. In the case of classical logic, the semantics was the "obvious" one, in a sense implicit in the meaning of the connectives. Though one can provide some intuitive motivation for Kripke semantics, the latter does not offer the same feeling of inevitability. In addition, the notion of a classical structure is a natural mathematical one, so we can either take the notion of a structure to be a tool for studying classical first-order logic, or take classical first-order logic to be a tool for studying mathematical structures. In contrast, Kripke structures can only be viewed as a logical construct; they don't seem to have independent mathematical interest.

The definition of a Kripke structure for first-order logic is given in Section 5.3 of van Dalen. If time allows, I will discuss the restriction to propositional logic. In that case, a Kripke structure consists of a partial order $\mathfrak{P}$ with a least element, and an "monotone" assignment of propositional variables to the elements of $\mathfrak{P}$. The intuition is that the elements of $\mathfrak{P}$ represent "worlds," or "states of knowledge"; an element $p \geq q$ represents a "possible future state" of q; and the propositional variables assigned to p are the propositions that are known to be true in state p. The forcing relation $p \Vdash \varphi$ then extends this relationship to arbitrary formulas in the language; read $p \Vdash \varphi$ as "φ is true in state p." The relationship is defined inductively, as follows:

1. $p \Vdash p_i$ if p_i is one of the propositional variables assigned to p.

2. $p \Vdash \varphi \wedge \psi$ if $p \Vdash \varphi$ and $p \Vdash \psi$.

3. $p \Vdash \varphi \vee \psi$ if $p \Vdash \varphi$ or $p \Vdash \psi$.

4. $p \Vdash \varphi \to \psi$ if whenever $q \geq p$ and $q \Vdash \varphi$, then $q \Vdash \psi$.

5. No node forces $\perp$.

I will discuss this in class, and give some examples. It is a good exercise to try to show that $\neg(p \wedge q) \to \neg p \vee \neg q$ is not intuitionistically valid, by cooking up a Kripke-structure that provides a counterexample.

8.6 Modal logics

When it came to illustrating material implication in Chapter 3, I gave the following example:

> If Jeremy is alone in that room, then he is drunk and naked and dancing on the chairs.

This was an example of an implication assertion that may be materially true but nonetheless misleading, since it seems to suggest that there is a stronger link between the hypothesis and conclusion. That is, the wording suggests that the claim is not only true in this particular world (where it may be trivially true, because Jeremy is not alone in the room), but that, moreover, the conclusion *would have* been true *had* the antecedent been true. In other words, one can take the assertion to mean that the claim is true not just in this world, but in any "possible" world; or that it is *necessarily* true, as opposed to just true in this particular world.

Modal logic was designed to make sense of this kind of necessity. One obtains modal propositional logic from ordinary propositional logic by adding a box operator; which is to say, if φ is a formula, so is $\Box\varphi$. Intuitively, $\Box\varphi$ asserts that φ is *necessarily* true, or true in any possible world. $\diamond\varphi$ is usually taken to be an abbreviation for $\neg\Box\neg\varphi$, and can be read as asserting that φ is *possibly* true. Of course, modality can be added to predicate logic as well.

Kripke structures can be used to provide a semantics for modal logic; in fact, Kripke first designed this semantics with modal logic in mind. Rather than restricting to partial orders, more generally one has a set of "possible worlds," P, and a binary "accessibility" relation $R(x, y)$ between worlds. Intuitively, $R(p, q)$ asserts that the world q is compatible with p; i.e. if we are "in" world p, we have to entertain the the possibility that the world could have been like q.

Modal logic is sometimes called an "intensional" logic, as opposed to an "extensional" one. The intended semantics for an extensional logic, like classical logic, will only refer to a single world, the "actual" one; while the semantics for an "intensional" logic relies on a more elaborate ontology. In addition to modeling necessity, one can use modality to model other linguistic constructions, reinterpreting $\Box$ and $\diamond$ according to the application. For example:

1. In provability logic, $\Box\varphi$ is read "φ is provable" and $\diamond\varphi$ is read "φ is consistent."
2. In epistemic logic, one might read $\Box\varphi$ as "I know φ" or "I believe φ."
3. In temporal logic, one can read $\Box\varphi$ as "φ is always true" and $\diamond\varphi$ as "φ is sometimes true."

One would like to augment logic with rules and axioms dealing with modality. For example, the proof system S4 consists of the ordinary axioms and rules of propositional logic, together with the following axioms:

1. $\Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi)$
2. $\Box\varphi \to \varphi$
3. $\Box\varphi \to \Box\Box\varphi$

as well as a rule, "from φ conclude $\Box\varphi$." S5 adds the following axiom:

1. $\diamond\varphi \to \Box \diamond \varphi$

Variations of these axioms may be suitable for different applications; for example, S5 is usually taken to characterize the notion of logical necessity. And the nice thing is that one can usually find a semantics for which the proof system is sound and complete by restricting the accessibility relation in the Kripke models in natural ways. For example, S4 corresponds to the class of Kripke models in which the accessibility relation is reflexive and transitive. S5 corresponds to the class of Kripke models in which the accessibility relation is *universal*, which is to say that every world is accessible from every other; so $\Box\varphi$ holds if and only if φ holds in every world.

8.7 Other logics

As you may have gathered by now, it is not hard to design a new logic. You too can create your own a syntax, make up a deductive system, and fashion a semantics to go with it. You might have to be a bit clever if you want the proof system to be complete for the semantics, and it might take some effort to convince the world at large that your logic is truly interesting. But, in return, you can enjoy hours of good, clean fun, exploring your logic's mathematical and computational properties.

Recent decades have witnessed a veritable explosion of formal logics. Fuzzy logic is designed to model reasoning about vague properties. Probabilistic logic is designed to model reasoning about uncertainty. Default logics and nonmonotonic logics are designed to model defeasible forms of reasoning, which is to say, "reasonable" inferences that can later be overturned in the face of new information. There are epistemic logics, designed to model reasoning about knowledge; causal logics, designed to model reasoning about causal relationships; and even "deontic" logics, which are designed to model reasoning about moral and ethical obligations. Depending on whether the primary motivation for introducing these systems is philosophical, mathematical, or computational, you may find such creatures studies under the rubric of mathematical logic, philosophical logic, artificial intelligence, cognitive science, or elsewhere.

The list goes on and on, and the possibilities seem endless. We may never attain Leibniz' dream of reducing all of human reason to calculation — but that can't stop us from trying.