Mathematical Logic
An Introduction to Model Theory

A.H. Lightstone

Mathematical Logic
An Introduction to Model Theory

A.H. Lightstone
Queen's University
Kingston, Ontario, Canada

Edited by
H.B. Enderton
University of California
Los Angeles, California

PLENUM PRESS · NEW YORK AND LONDON

Library of Congress Cataloging in Publication Data

Lightstone, A H
Mathematical logic.

(Mathematical concepts and methods in science and engineering; 9)
Bibliography: p.
Includes index.
1. Logic, Symbolic and mathematical. 2. Model theory. I. Title.
QA9.L54 511'.3 77-17838
ISBN 0-306-30894-0

A Division of Plenum Publishing Corporation
227 West 17th Street, New York, N.Y. 10011

Printed in the United States of America

Foreword

Before his death in March, 1976, A. H. Lightstone delivered the manuscript for this book to Plenum Press. Because he died before the editorial work on the manuscript was completed, I agreed (in the fall of 1976) to serve as a surrogate author and to see the project through to completion.

I have changed the manuscript as little as possible, altering certain passages to correct oversights. But the alterations are minor; this is Lightstone's book.

H. B. Enderton

Preface

This is a treatment of the predicate calculus in a form that serves as a foundation for nonstandard analysis. Classically, the predicates and variables of the predicate calculus are kept distinct, inasmuch as no variable is also a predicate; moreover, each predicate is assigned an *order*, a unique natural number that indicates the length of each tuple to which the predicate can be prefixed. These restrictions are dropped here, in order to develop a flexible, expressive language capable of exploiting the potential of nonstandard analysis.

To assist the reader in grasping the basic ideas of logic, we begin in Part I by presenting the *propositional calculus* and *statement systems*. This provides a relatively simple setting in which to grapple with the sometimes foreign ideas of mathematical logic. These ideas are repeated in Part II, where the *predicate calculus* and *semantical systems* are studied.

Finally, in Part III, we present some applications. There is a substantial discussion of *nonstandard analysis*, a treatment of the *Löwenheim–Skolem Theorem*, a discussion of *axiomatic set theory* that utilizes semantical systems, and an account of *complete theories*. The presentation of complete theories includes Vaught's test, but is mainly devoted to an exposition of Robinson's notion of *model completeness* and its connection with *completeness*. Chapter 16 is taken from the author's *The Axiomatic Method: An Introduction to Mathematical Logic*,* with only a few minor changes.

This book contains many ideas due to Abraham Robinson, the father of nonstandard analysis. The author is also indebted to Prof. Ernest Heighton for several stimulating conversations and many valuable suggestions.

A. H. Lightstone

* Prentice-Hall, Englewood Cliffs (1964).

Contents

Introduction

A theory of deduction utilizes various ideas of logic that may appear strange, even foreign, to mathematics students with little background in logic. The main concern of this book is to develop the important theory of deduction known as the *predicate calculus*. In an effort to overcome the strangeness of the logical ideas and methods involved, we shall first present the theory of deduction based on the connectives $\neg$ (not) and $\vee$ (or). This theory, known as the *propositional calculus*, characterizes the conclusions, or consequences, of a given set of assumptions, and so provides us with the formal side of arguments. The question of the validity of a given argument of this sort is easy to solve by the truth-table method, and so is really trivial. Therefore, in studying the accompanying theory of deduction we are able to concentrate on the formal apparatus and methods of a theory of deduction, without the complications owing to the subject matter under investigation. In short, the propositional calculus is a convenient device for making clear the nature of a theory of deduction.

In rough outline, the steps in setting up a theory of deduction are as follows. First, the propositions (statements) of a language are characterized. This is achieved by actually creating a specific formal language possessing its own alphabet and rules of grammar; in fact, the sentences of the formal language are effectively spelled out by suitably chosen rules of grammar. Finally, the notion of *truth* within this specialized and highly artificial language is characterized in terms of the concept of a "proof."

To explain in a little more detail, a theory of deduction is based on an alphabet, which consists of symbols of several sorts. There are the *connectives* and *parentheses*, which are used to construct compound propositions from given propositions of the language; and there are symbols that yield the basic, atomic propositions of the language. As a first step toward characterizing the propositions of the language, it is convenient to introduce the notion of an *expression* of the language. An expression is any finite string of symbols of the language, with repetitions allowed. Certain sequences of propositions are recognized as "proofs," and the last proposition of each proof is said to be *provable.* Using the notion of a provable proposition, we can characterize in a purely formal manner the consequences of a set of propositions.

As we have suggested, we shall present two theories of deduction in this book. The first of these, the propositional calculus, involves a language whose connectives are "not" and the "inclusive or." We shall motivate this theory of deduction, which is highly abstract, by considering *statement systems*, which are simple and concrete. In this setting, we can utilize the truth-table approach of symbolic logic.

The second theory of deduction that we develop in this book is a version of the classical predicate calculus. This involves a language whose connectives are "not," the "inclusive or," and "for each" (the universal quantifier). We shall motivate this theory of deduction by considering *semantical systems*, a much simpler notion.

PART I
Statement Systems and Propositional Calculus

I

Statement Systems

1.1. Statement Systems

A statement system consists of a given set of statements, each of which is assigned a truth-value *true* or *false*. Each statement system has its own language, which is built up from its initial statements by connecting them with the logical connectives *not* (denoted by $\neg$) and the *inclusive or* (denoted by $\vee$). Applying the truth-table definitions of these connectives, we easily compute a unique truth-value for each of the compound statements in the language of the statement system. We shall go into this in Section 1.2.

Since a statement system involves a set of objects, called statements, and since each of these objects has a unique truth-value, we shall identify a statement system with a map whose range is included in {true, false}. The domain of this map is the set of initial statements of the statement system. By a statement system, then, we mean any map $\sum$ with a nonempty domain and with range included in {true, false}. We regard the members of dom $\sum$ as *statements* (so we use this term in a generalized sense); each of these objects is assigned a truth-value by the map $\sum$. Thus, a statement system $\sum$ consists of objects, called statements, each of which is assigned a truth-value by the map $\sum$.

Of course, a statement system may involve actual statements, in the usual sense of the term.

Example 1. Let $\sum$ be the map with range {true, false} and domain

{grass is green, oil is cheap, logic is easy, Washington is the capital of the United States} such that $\sum$ associates ***true*** with "grass is green" and "Washington is the capital of the United States," and $\sum$ associates *false* with "oil is cheap" and "logic is easy." Then $\sum$ is a statement system.

Note that we obtain different statement systems from a given set of statements by assigning truth-values in different ways to the statements of the system. To illustrate, let us change the truth-values assigned to the statements of the statement system of Example 1.

Example 2. Let $\sum$ be the map with domain

{grass is green, oil is cheap, logic is easy, Washington is the capital of the United States}

such that $\sum$ associates *false* with each statement in its domain. Then $\sum$ is a statement system.

We point out that the statement system of Example 2 is different from the statement system of Example 1. By a statement system we mean a map whose range is included in {true, false} and whose domain is non-empty; the maps of the two examples are certainly different.

The members of the domain of a statement system (i.e., its statements) need not be actual statements. Here is an example.

Example 3. Let $\sum$ be the map of $\{S, T, U\}$ into {true, false} that associates ***true*** with T and associates *false* with S and U. Then $\sum$ is a statement system.

1.2. Language of a Statement System

Each statement system $\sum$ has its own language, which is built up from the statements in dom $\sum$ by means of the logical connectives ***not*** ($\neg$) and the ***inclusive or*** ($\vee$). Each grammatical expression of this language is said to be a ***statement well-formed formula***, or ***swff*** for short. Thus, each swff of $\sum$ consists of a finite number of objects in dom $\sum$ linked by the connectives $\neg$ and $\vee$ (and parentheses).

More formally, we say that the expression (S) is an ***atomic*** swff of $\sum$ for each $S \in \text{dom} \sum$. The remaining swffs of $\sum$ are defined as follows. Let A and B be any swffs of $\sum$; then we say that $(\neg A)$ and $(A \vee B)$ are swffs of $\sum$. This is subject to the requirement that each swff of $\sum$ involves only a finite number of instances of connectives.

We obtain the truth-value of each atomic swff of $\sum$ directly from the map $\sum$ itself. The truth-value of each compound swff of $\sum$ is obtained by considering the significance of the connectives *not* and the *inclusive or*. Bearing this in mind, we formulate the following definition.

Definition. (i) An atomic swff (S) is ***true*** for $\sum$ if $\sum$ associates "true" with S; (S) is *false* for $\sum$ if $\sum$ associates "false" with S. Here, $S \in \text{dom} \sum$.

(ii) $(\neg A)$ is ***true*** for $\sum$ if A is false for $\sum$; $(\neg A)$ is *false* for $\sum$ if A is true for $\sum$. Here, A is any swff of $\sum$.

(iii) $(A \vee B)$ is ***true*** for $\sum$ if A is true for $\sum$, B is true for $\sum$, or both A and B are true for $\sum$; $(A \vee B)$ is *false* for $\sum$ if A is false for $\sum$ and B is false for $\sum$. Here, A and B are any swffs of $\sum$.

We point out that this definition assigns a unique truth-value to each swff of $\sum$.

To illustrate, let $\sum$ be the statement system of Example 1, Section 1.1. Let

g = grass is green
o = oil is cheap
l = logic is easy
W = Washington is the capital of the United States

Then the following swffs of $\sum$ are each true for $\sum$:

$$(g), \quad (\neg(o)), \quad ((g) \vee (o)), \quad (\neg((\neg(W)) \vee (o)))$$

The following swffs of $\sum$ are each false for $\sum$:

$$(o), \quad (\neg(g)), \quad ((o) \vee (\neg(g)))$$

1.3. Names for Swffs

The purpose of the parentheses that appear in each swff is to avoid ambiguous statements. For example, "$\neg S \vee T$" could represent either $(\neg S) \vee T$ or $\neg(S \vee T)$. Moreover, parentheses impose a certain structure on swffs which we shall find very useful (see Section 2.2).

On the other hand, it is difficult to read a given swff if it involves many parentheses. We can obtain the best of both worlds by introducing conventions for omitting parentheses. This is achieved by introducing *names* for swffs; of course, we must avoid ambiguity, i.e., each of our names must name a unique swff.

We shall usually omit parentheses whenever this does not produce an ambiguous expression. For example, the outermost pair of parentheses of any swff and the pair of parentheses involved in each atomic swff can usually be suppressed without harm. For example, let $S, T \in \operatorname{dom} \sum$ and consider the swff

$$A = ((\neg(S)) \vee ((T) \vee (S)))$$

Under our agreement, "$(\neg S) \vee (T \vee S)$" is a name for A.

Short names for certain swffs are obtained by introducing the logical connectives $\wedge$ (*and*), $\rightarrow$ (*if . . . then*), and $\leftrightarrow$ (*if and only if*). Our agreement is that for any swffs A and B,

$$\begin{array}{lll} (A \wedge B) & \text{is a name for} & (\neg((\neg A) \vee (\neg B))) \\ (A \rightarrow B) & \text{is a name for} & ((\neg A) \vee B) \\ (A \leftrightarrow B) & \text{is a name for} & ((A \rightarrow B) \wedge (B \rightarrow A)) \end{array}$$

We emphasize that $\neg$ and $\vee$ are the basic connectives of our language, whereas the connectives $\wedge$, $\rightarrow$, and $\leftrightarrow$ are defined in terms of $\neg$ and $\vee$. We shall freely drop the outermost pair of parentheses of a *name* for a swff. For example, "$(S \wedge T) \rightarrow T$" is a name for the swff named by

$$(\neg(\neg((\neg S) \vee (\neg T)))) \vee T$$

i.e., the swff $((\neg(\neg((\neg(S)) \vee (\neg(T))))) \vee (T))$.

Here is a very useful convention for omitting parentheses. We shall attribute a built-in bracketing power, or reach, to the connectives in the following order: $\neg$, $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$, where the connectives weakest in reach are written first in this list. Under this convention, $\rightarrow$ is the principal connective of the swff $S \vee T \rightarrow S$, since the bracketing power of $\rightarrow$ is stronger than the bracketing power of $\vee$. In other words, the reach of $\vee$ is blocked by the stronger connective $\rightarrow$ in the name "$S \vee T \rightarrow S$." So, this is an abbreviation for "$(S \vee T) \rightarrow S$," which itself is a name for a swff.

It is sometimes possible to make a dot do the work of several pairs of parentheses. The idea is to strengthen the bracketing power of a connective by placing a dot above it. This means that the bracketing power of the connectives is as follows: $\neg$, $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$, $\dot{\neg}$, $\dot{\vee}$, $\dot{\wedge}$, $\dot{\rightarrow}$, $\dot{\leftrightarrow}$, $\ddot{\neg}$, $\ddot{\vee}$, $\ddot{\wedge}$, $\ddot{\rightarrow}$, $\ddot{\leftrightarrow}$, and so on.

Under this "dot" convention, the swff

$$(S \rightarrow T) \rightarrow ((U \vee S) \rightarrow (T \vee U))$$

can be written as

$$S \rightarrow T \dot{\rightarrow} U \vee S \rightarrow T \vee U$$

without ambiguity.

The intention of the above conventions is not to eliminate all parentheses, but merely to reduce the number of parentheses that appear in a name for a given swff so that the eye will not be lost in a maze of parentheses. One or two pairs of parentheses may not be objectionable in a name for a swff and may even be desirable. Our goal is that names for swffs should be readable. For this reason, we prefer to write

$$S \vee \neg T \rightarrow \neg(S \vee T)$$

rather than $S \vee \neg T \dot{\rightarrow} \neg S \vee T$.

In this section we have followed the usual custom of mentioning a swff A by writing down a name for A. For example, we introduced "$A \wedge B$" as a name for a certain swff C by writing down a name for C rather than C itself.

Exercises

1. Let $\sum$ be the statement system such that dom $\sum = \{S_1, \ldots, S_9\}$, where S_i = "i is prime," for $i = 1, \ldots, 9$, and let $\sum$ assign "true" to S_i when i is prime; i.e., S_2, S_3, S_5, and S_7 are true for $\sum$, whereas S_1, S_4, S_6, S_8, and S_9 are false for $\sum$. Compute the truth-value of each of the following swffs of $\sum$:

   (a) $S_1 \vee S_3$.
   (b) $S_1 \vee \neg S_2$.
   (c) $S_2 \wedge S_3 \rightarrow S_4$.
   (d) $S_3 \wedge S_7 \leftrightarrow \neg S_9$.
   (e) $S_5 \vee \neg S_6 \rightarrow \neg S_5 \vee S_6$.

2. Prove that for every statement system $\sum$, each swff of $\sum$ has a unique truth-value. *Hint:* If there is a swff of $\sum$ that does not have a unique truth-value, then there is a shortest swff of $\sum$ that does not have a unique truth-value.

3. Let A, B, and C be swffs of $\sum$, a statement system. Show that each of the following swffs is true for $\sum$:

   (a) $A \vee A \rightarrow A$.
   (b) $A \rightarrow A \vee B$.
   (c) $A \rightarrow B \dot{\rightarrow} C \vee A \rightarrow B \vee C$.

4. Let A and $A \rightarrow B$ be true swffs of a statement system $\sum$. Prove that B is true for $\sum$.

5. Let A, B, and C be swffs of a statement system $\sum$. Prove that:

   (a) $\neg(A \vee B)$ is true for $\sum$ iff $\neg A \wedge \neg B$ is true for $\sum$.
   (b) $\neg(A \wedge B)$ is true for $\sum$ iff $\neg A \vee \neg B$ is true for $\sum$.
   (c) $A \wedge (B \vee C)$ is true for $\sum$ iff $(A \wedge B) \vee (A \wedge C)$ is true for $\sum$.
   (d) $A \vee (B \wedge C)$ is true for $\sum$ iff $(A \vee B) \wedge (A \vee C)$ is true for $\sum$.
   (e) $A \dot{\rightarrow} B \rightarrow C$ is true for $\sum$ iff $A \wedge B \rightarrow C$ is true for $\sum$.