Preface

This book is an introductory text on mathematical logic and type theory. It is aimed primarily at providing an introduction to logic for students of mathematics, computer science, or philosophy who are at the college junior, senior, or introductory graduate level. It can also be used as an introduction to type theory by researchers at more advanced levels.

The first part of the book (Chapters 1 and 2, supplemented by parts of Chapters 3 and 4) is suitable for use as a text in a one-semester introduction to mathematical logic, and the second part (Chapters 5 – 7) for a second semester course which introduces type theory (more specifically, typed λ-calculus) as a language for formalizing mathematics and proves the classical incompleteness and undecidability theorems in this context. Persons who wish to learn about type theory and have had a prior introduction to logic will have no difficulty starting with Chapter 5.

The book is oriented toward persons who wish to study logic as a vehicle for formal reasoning. It may be particularly useful to those who are interested in issues related to the problem of constructing formal proofs as models of informal reasoning or for use in computerized systems which involve automated deduction. Proofs, which are often the chief end products and principal manifestations of mathematical reasoning, constitute highly significant pathways to truth and are a central concern of this book. Our choice of the title *To Truth Through Proof* is motivated by the consideration that while in most realms one needs more than logic to achieve an understanding of what is true, in mathematics the primary and ultimate tool for establishing truth is logic.

Of course, the study of logic involves reasoning about reasoning, and it is not surprising that complex questions arise. To achieve deep understanding and proper perspective, one must study a variety of logical systems as mathematical objects and look at them from a variety of points of view. We are thus led to study the interplay between syntax and semantics, questions of consistency and independence, formal rules of reasoning, various formats for proofs and refutations, ways of representing basic mathematical concepts in a formal system, the notion of computability, and the completeness, incompleteness, and undecidability theorems which illuminate both the power and the limitations of logic.

One of the basic tasks of mathematical logic is the formalization of mathematical reasoning. Both type theory (otherwise known as higher-order logic) and axiomatic set theory can be used as formal languages for this

purpose, and it is really an accident of intellectual history that at present most logicians and mathematicians are more familiar with axiomatic set theory than with type theory. It is hoped that this book will help to remedy this situation.

In logic as in other realms, there is a natural tendency for people to prefer that with which they are most familiar. However, those familiar with both type theory and axiomatic set theory recognize that in some ways the former provides a more natural vehicle than the latter for formalizing what mathematicians actually do. Both logical systems are necessarily more restrictive than naive axiomatic set theory with the unrestricted Comprehension Axiom, since the latter theory is inconsistent. Axiomatic set theory achieves consistency by restricting the Comprehension Axiom and introducing the distinction between sets and classes. Mathematicians often find this distinction unnatural, ignore the technicalities about existence axioms, and leave it to the specialists to show that their reasoning can be justified. Type theory achieves consistency by distinguishing between different types of objects (such as numbers, sets of numbers, collections of sets of numbers, functions from numbers to numbers, sets of such functions, etc.). Mathematicians make such distinctions too, and even use different letters or alphabets to help them distinguish between different types of objects, so the restrictions which type theory introduces are already implicit in mathematical practice. While some formulations of type theory may seem cumbersome, the formulation $\mathcal{Q}_0$ introduced in Chapter 5 is a very rich and expressive language, with functions (which need not be regarded as sets of ordered pairs) of all types as primitive objects, so most of what mathematicians write can be translated into $\mathcal{Q}_0$ very directly. $\mathcal{Q}_0$ is a version of typed λ-calculus, and the availability of λ-notation in $\mathcal{Q}_0$ enables definitions to be handled very conveniently and eliminates the need for axioms asserting the existence of sets and functions.

One's choice of a formal language will generally depend on what one wishes to do with it. If one is choosing a language for expressing mathematics in computerized systems which deal with mathematics on a theoretical level (such as mathematically sophisticated information retrieval systems, proof checkers, or deductive aids), there are several reasons why type theory may be preferable to set theory. First, the primitive notation of set theory is so economical that this language is only practical to actually use if one expands it by introducing various abbreviations. This is usually done in an informal way (i.e., in the meta-language), but in an automated system it is important that the formal language be the one actually used to express mathematical statements, since this is the language which must be studied rigorously and

manipulated systematically. Thus, the formal language should not only be adequate in principle for expressing mathematical ideas, but it should also be convenient in practice. Indeed, the translation from familiar mathematical notations to the formal language should be as simple and direct as possible. Second, in set theory one can write all sorts of expressions which are not legal in type theory, and which a mathematician would reject as nonsense almost instinctively. A computer program for manipulating mathematical expressions would certainly be very awkward and inefficient if it did not have some way of distinguishing between different types of mathematical objects and rejecting nonsensical expressions. Third, one of the basic operations in automated theorem proving is that of finding appropriate substitutions for variables, and an understanding of the types of various entities allows one to avoid much useless search. Finally, it has been found (see [Andrews *et al.*, 1984] [Andrews *et al.*, 1996]) that unification algorithms for type theory (such as [Huet, 1975] and [Jensen and Pietrzykowski, 1976]) are powerful tools in automating the proofs of simple mathematical theorems.

As noted in [Hanna and Daeche, 1985], [Hanna and Daeche, 1986], and [Gordon, 1986], typed λ-calculus is particularly well suited for specifying and verifying hardware and software. Familiarity with typed λ-calculus also provides fundamental background for the study of denotational semantics for functional programming languages.

While Gödel's Completeness Theorem guarantees that all valid wffs of first-order logic have proofs in first-order logic, extraordinary reductions in the lengths of such proofs can sometimes be achieved by using higher-order logic. (See §54.)

It is easy to discuss the semantics of type theory, and to explain the crucial distinction between standard and nonstandard models which sheds so much light on the mysteries associated with the incompleteness theorems, Skolem's paradox, etc. As will be seen in Chapter 7, in the context of the language $\mathcal{Q}_0$ it is easy to present the incompleteness theorems very elegantly and to show that their significance extends far beyond the realm of formalizing arithmetic.

Obviously, serious students of mathematical logic ought to know about both type theory and axiomatic set theory. Pedagogically, it makes good sense to introduce them to elementary logic (propositional calculus and first-order logic), then type theory, then set theory. In spite of the superficial appearance of simplicity possessed by set theory, its models are more complicated, and from a logical point of view it is a more powerful and complex language. In [Marshall and Chuaqui, 1991] it is argued that the set-theoretical sentences which are preserved under isomorphisms are of fundamental im-

portance, and it is shown that these are the sentences which are equivalent to sentences of type theory.

A brief explanation of the labeling system used in this book may be helpful. The sections in Chapter 1 (for example) are labeled §10, §11, ..., §16. Of course, an alternative labeling would be §1.0, §1.1, ..., §1.6, but the periods are quite redundant once one understands the labeling system. These tags should be regarded as labels, not numbers, although they do have a natural ordering. Similarly, the theorems in §11 (for example) are labeled 1100, 1101, 1102, ..., rather than 1.1.00, 1.1.01, 1.1.02, ... Exercises are labeled in a similar way, but their labels start with an X. Thus, the exercises for section 11 are labeled X1100, X1101, X1102, ... Labeling starts with 0 rather than 1 because 0 is the initial ordinal. (Chapter 0 is the Introduction.)

The reader should examine the table of contents in parallel with the discussion of the individual chapters below. Much that is in this book is simply what one would expect in an introductory text on mathematical logic, so the discussion will concentrate on features one might not expect.

Chapter 1 introduces the student to an axiomatic system $\mathcal{P}$ of propositional calculus and later to more general considerations about propositional calculus. In §14 a convenient two-dimensional notation is introduced to represent wffs in negation normal form. This enables one to readily comprehend the structure of such wffs, find disjunctive and conjunctive normal forms by inspection, and check whether the wffs are contradictory by examining the "vertical paths" through them.

Chapter 2 contains what everyone ought to know about first-order logic. Students can learn a great deal about how to construct proofs by doing the exercises at the end of §21. The discussion of prenex normal form in §22 shows students how to pull out the quantifiers of a wff all at once as well as how to bring them out past each connective one step at a time. This discussion is facilitated by the use of the notation $\exists\!\!\!/$ for the ambiguous quantifier.

In §25 Gödel's Completeness Theorem is proved by an elegant generalization of Henkin's method due to Smullyan. The notion of consistency in Henkin's proof is replaced by abstract consistency, and with very little extra effort a proof is obtained for Smullyan's Unifying Principle, from which readily follow Gödel's Completeness Theorem in §25, Gentzen's Cut-Elimination Theorem in §31, the completeness of the method of semantic tableaux in §32, the completeness of a method for refuting universal sentences in §34, and the Craig–Lyndon Interpolation Theorem in §41. In their first encounter with the proof of Gödel's Completeness Theorem, students may need a less abstract approach than that of Smullyan's Unifying Prin-

ciple, so as an alternative in §25A an elegant and direct simplified proof of the Completeness Theorem is given. It actually parallels the abstract approach very closely and provides a good introduction to the abstract approach. The Löwenheim–Skolem Theorem and the Compactness Theorem are derived in the usual way from the Completeness Theorem. It is shown how the Compactness Theorem can be applied to extend the Four Color Theorem of graph theory from finite to infinite graphs and to prove the existence of nonstandard models for number theory.

Chapter 3 is concerned with the logical principles underlying various techniques for proving or refuting sentences of first-order logic, and provides an introduction to the fundamental ideas used in various approaches to automated theorem proving.

The chapter starts in §30 with a system of natural deduction which is essentially a summary of the most useful derived rules of inference that were obtained in §21. Methods of proving existential formulas are discussed. A cut-free system of logic and semantic tableaux are then introduced in §31 and §32 as systems in which one can search more systematically for a proof (or refutation). When one constructs a semantic tableau, one often notices that it would be nice to postpone deciding how to instantiate universal quantifiers by just instantiating them with variables for which substitutions could be made later. This is facilitated when one eliminates existential quantifiers to obtain a universal sentence by Skolemization, which is the topic of §33. It is noted that several methods of Skolemization can be used.

§34 introduces a method for refuting universal sentences which avoids the branching on disjunctions of semantic tableaux and uses the cut rule to reduce formulas to the contradictory empty disjunction. This method has many elements in common with the resolution method [Robinson, 1965] of automated theorem-proving. Students are often successful at establishing theorems by this method even though they were unable to find proofs for them in natural deduction format.

The proof methods just discussed all yield highly redundant proof structures in which various subformulas may occur many times. §35 introduces a version of Herbrand's Theorem which enables one to establish that a wff is contradictory simply by displaying a suitable compound instance of it.

One of the basic processes of theorem-proving in first- or higher-order logic is that of substituting terms for variables (or instantiating quantified variables) in such a way that certain expressions become the same. §36 provides a description of Robinson's Unification Algorithm for first-order logic and a proof of its correctness. Since substitutions have been defined (in §10) to be functions of a certain sort which map formulas to formulas,

certain facts about substitutions (such as the associativity of composition) can be used without special justification.

Chapter 4 discusses a few additional topics in first-order logic. In §40 the topic of Duality is introduced with a parable about two scholars who argue about an ancient text on logic because of a fundamental ambiguity in the notations for truth and falsehood. Of course, much of this material could appropriately be discussed immediately after §14.

Chapter 5 introduces a system $\mathcal{Q}_0$ of typed λ-calculus which is essentially the system introduced by Alonzo Church in [Church, 1940], except that (following [Henkin, 1963]) equality relations rather than quantifiers and propositional connectives are primitive. Type theory is first introduced in §50 with a discussion of a rather traditional formulation $\mathcal{F}^\omega$ of type theory in which propositional connectives and quantifiers are primitive. It is shown that equality can be defined in such a system in two quite natural ways. It is then shown that this system can be simplified to obtain naive axiomatic set theory, which is inconsistent because Russell's paradox can be derived in it. Thus, one needs some device such as type distinctions or restrictions on the Comprehension Axiom to obtain a consistent formalization of naive set theory. The discussion turns to finding as simple, natural, and expressive a formulation of type theory as possible. (The choice of equality as the primitive logical notion is influenced not only by the desire for simplicity but also by the desire for a natural semantics, as discussed at the end of [Andrews, 1972].) This leads to an exposition of the basic ideas underlying $\mathcal{Q}_0$. While these ideas usually come to be regarded as very natural, they often seem novel at this stage, and need to be absorbed thoroughly before one proceeds to the next section.

By the end of §52 the student should be ready to prove theorems in $\mathcal{Q}_0$, so notations for some simple but basic mathematical ideas are introduced, and in the exercises the student is asked to give formal proofs of some simple mathematical theorems about sets and functions. An exercise in §53 asks for a proof of a concise type-theoretic formulation of Cantor's Theorem.

Henkin's Completeness Theorem is proved for $\mathcal{Q}_0$ in §55. Skolem's paradox (which was first discussed in §25) is discussed again in the context of type theory and resolved with the aid of the important distinction between standard and nonstandard models. It is shown that theories which have infinite models must have nonstandard models.

Chapter 6 is intended to make it clear that mathematics really can be formalized within the system $\mathcal{Q}_0^\infty$ which is the result of adding an axiom of infinity to $\mathcal{Q}_0$. It is shown how cardinal numbers can be defined, Peano's Postulates can be derived, and recursive functions can be represented very

elegantly in $\mathcal{Q}_0^\infty$. Proofs of theorems of $\mathcal{Q}_0^\infty$ are presented in sufficient detail so that students should have no difficulty providing formal justifications for each step. Naturally, each teacher will decide how (or whether) to treat these proofs in class. It is good experience for students to present some proofs, or parts of proofs, in class. By the end of this chapter students should have a firm grasp of the connection between abbreviated formal proofs and the informal proofs seen in other mathematics courses, and should therefore be much more confident in dealing with such proofs.

Chapter 7 presents the classical incompleteness, undecidability and undefinability results for $\mathcal{Q}_0^\infty$. In §70 it is shown that the numerical functions representable in any consistent recursively axiomatized extension of $\mathcal{Q}_0^\infty$ are precisely the recursive functions, and this leads to an argument for Church's Thesis. In §71 Gödel's First and Second Incompleteness Theorems are presented for $\mathcal{Q}_0^\infty$, along with Löb's Theorem about the sentence which says "I am provable". A number of consequences of Gödel's Second Incompleteness Theorem are discussed. In §72 the Gödel–Rosser Incompleteness Theorem is presented for $\mathcal{Q}_0^\infty$, and it is shown how this implies that there is no recursively axiomatized extension of $\mathcal{Q}_0$ whose theorems are precisely the wffs valid in all standard models. §73 is concerned with the unsolvability of the decision problems for $\mathcal{Q}_0$ and $\mathcal{Q}_0^\infty$, and the undefinability of truth. §74 is a brief epilogue reflecting on the elusiveness of truth.

The exercises at the end of each section generally provide opportunities for using material from that section. However, students need to learn how to decide for themselves what techniques to use to solve problems. Therefore, at the end of the book there is a collection of Supplementary Exercises which are not explicitly tied to any particular section.

Some exercises (see §21 and §52, for example) involve applying rules of inference to prove theorems of the formal system being discussed. A computer program called ETPS (which is reviewed in [Goldson and Reeves, 1993] and [Goldson *et al.*, 1993]) has been developed to facilitate work on such exercises. The student using ETPS issues commands to apply rules of inference in specified ways, and the computer handles the details of writing the appropriate lines of the proof and checking that the rules can be used in this way. Proofs, and the active lines of the proof, are displayed using the special symbols of logic in proof windows which are automatically updated as the proof is constructed. The program thus allows students to concentrate on the essential logical problems underlying the proofs, and it gives them immediate feedback for both correct and incorrect actions. Experience shows that ETPS enables students to construct rigorous proofs of more difficult theorems than they would otherwise find tractable.

ETPS permits students to work forwards, backwards, or in a combination of these modes, and provides facilities for rearranging proofs, deleting parts of proofs, displaying only those parts of proofs under active consideration, saving incomplete proofs, and printing proofs on paper. A convenient formula editor permits the student to extract needed formulas which occur anywhere in the proof, and build new formulas from them. Teachers who set up ETPS for use by a class can take advantage of its facilities for keeping records of completed exercises and processing information for grades.

Information about obtaining ETPS without cost can be obtained from http://gtps.math.cmu.edu/tps.html; alternatively, connect to the web page http://www.cmu.edu/ for Carnegie Mellon University or http://www.cs.cmu.edu/afs/cs/project/pal/www/pal.html for CMU's Pure and Applied Logic Program and from there find the author's web page.

I have used versions of this book for a course in mathematical logic at Carnegie Mellon University for about thirty years. Experience has shown that students are best prepared for the course if they have had at least one rigorous mathematics course or a course in philosophy or computer science that has prepared them to appreciate the importance of understanding the nature of deductive reasoning and the art of proving theorems. I have found that if one covers the material rather thoroughly in class, it is difficult to cover all of Chapters 1 – 4 in the first semester, so I normally cover Chapter 1, most of Chapter 2 (using §25A and omitting §24 and most of §26), and also §30, §33, and §34. Naturally, in a course primarily composed of graduate students, or students with more previous exposure to logic, one could move faster, ask students to read some material outside class, and cover all of Chapters 1 – 4 in one semester. In the second semester I normally cover Chapters 5 – 7 fairly completely.

A few comments about notation may be helpful. "U is a subset of V" is written as "$U \subseteq V$", and "U is a proper subset of V" as "$U \subset V$". The composition of functions f and g is written "$f \circ g$". "iff" is an abbreviation for "if and only if". Ends of proofs are marked with the sign ■.

I wish to thank the many people who contributed directly or indirectly to the development of this book. Alonzo Church, John Kemeny, Raymond Smullyan, and Abraham Robinson taught me logic. It will be obvious to all who have read them that this book has been deeply influenced by [Church, 1940] and [Church, 1956]. Numerous students at Carnegie Mellon helped shape the book with their questions, comments, and interests. Special thanks go to Ferna Hartman for her heroic work typing the difficult manuscript.

Bibliography

[Andrews *et al.*, 1984] Peter B. Andrews, Dale A. Miller, Eve Longini Cohen, and Frank Pfenning. Automating Higher-Order Logic. In W. W. Bledsoe and D. W. Loveland, editors, *Automated Theorem Proving: After 25 Years*, Contemporary Mathematics series, vol. 29, pages 169–192. American Mathematical Society, 1984.

[Andrews *et al.*, 1996] Peter B. Andrews, Matthew Bishop, Sunil Issar, Dan Nesmith, Frank Pfenning, and Hongwei Xi. TPS: A Theorem Proving System for Classical Type Theory. *Journal of Automated Reasoning*, 16:321–353, 1996.

[Andrews, 1972] Peter B. Andrews. General Models and Extensionality. *Journal of Symbolic Logic*, 37:395–397, 1972.

[Church, 1940] Alonzo Church. A Formulation of the Simple Theory of Types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[Church, 1956] Alonzo Church. *Introduction to Mathematical Logic*. Princeton University Press, Princeton, N.J., 1956.

[Goldson and Reeves, 1993] Doug Goldson and Steve Reeves. Using Programs to Teach Logic to Computer Scientists. *Notices of the American Mathematical Society*, 40:143–148, 1993.

[Goldson *et al.*, 1993] Douglas Goldson, Steve Reeves, and Richard Bornat. A Review of Several Programs for the Teaching of Logic. *The Computer Journal*, 36:373–386, 1993.

[Gordon, 1986] Mike Gordon. Why higher-order logic is a good formalism for specifying and verifying hardware. In G. J. Milne and P. A. Subrahmanyam, editors, *Formal Aspects of VLSI Design*, pages 153–177. North-Holland, 1986.

[Hanna and Daeche, 1985] F. K. Hanna and N. Daeche. Specification and Verification using Higher-Order Logic. In Koomen and Moto-oka, editors, *Computer Hardware Description Languages and their Applications*, pages 418–433. North Holland, 1985.

[Hanna and Daeche, 1986] F. K. Hanna and N. Daeche. Specification and Verification Using Higher-Order Logic: A Case Study. In G. J. Milne and P. A. Subrahmanyam, editors, *Formal Aspects of VLSI Design*, pages 179–213. North-Holland, 1986.

[Henkin, 1963] Leon Henkin. A Theory of Propositional Types. *Fundamenta Mathematicae*, 52:323–344, 1963.

[Huet, 1975] Gérard P. Huet. A Unification Algorithm for Typed λ-Calculus. *Theoretical Computer Science*, 1:27–57, 1975.

[Jensen and Pietrzykowski, 1976] D.C. Jensen and T. Pietrzykowski. Mechanizing ω-Order Type Theory Through Unification. *Theoretical Computer Science*, 3:123–171, 1976.

[Marshall and Chuaqui, 1991] M. Victoria Marshall and Rolando Chuaqui. Sentences of type theory: the only sentences preserved under isomorphisms. *Journal of Symbolic Logic*, 56:932–948, 1991.

[Robinson, 1965] J. A. Robinson. A Machine-Oriented Logic Based on the Resolution Principle. *Journal of the ACM*, 12:23–41, 1965.

Chapter 1

Propositional Calculus

§10. The Language of $\mathcal{P}$

As discussed in the Introduction, we can study the nature of reasoning by studying logistic systems in which reasoning is represented in a precise way. We shall commence our study with a rather simple logistic system called $\mathcal{P}$, which is one formulation of propositional calculus. Many of the definitions and concepts which we shall introduce while studying $\mathcal{P}$ will also be applicable to richer logical systems. In addition, $\mathcal{P}$ will occur as a subsystem of some of these richer systems. Once we have discussed $\mathcal{P}$, which is an example of a logistic system, we will be in a good position to give a general explanation of what a logistic system is.

In order to make clear what aspects of reasoning will be represented in $\mathcal{P}$, we start by considering three examples of logical inferences.

EXAMPLE 1: Jack is asked whether he will go to the picnic on Saturday. He doesn't like to go to picnics in the rain, and he remembers that if it does not rain, he will be playing in a tennis match which conflicts with the picnic. Hence he replies,

(A_1) "If it rains, I will not go to the picnic."

(B_1) "If it does not rain, I will not go to the picnic."

(C_1) "Therefore, I will not go to the picnic."

EXAMPLE 2: We are given that w, z, and y are sets such that $z - y = \emptyset$ and $w \cap \widetilde{z} \subseteq y$ (where $\widetilde{z}$ is the complement of the set z), and we wish to show

that $w \subseteq y$. So we assume that $z - y = \emptyset$ and $w \cap \widetilde{z} \subseteq y$ and $x \in w$, and try to prove $x \in y$. Using the information that is given, it is not hard to show that

(A_2) If $x \in z$, then $x \in y$.

(B_2) If $x \notin z$, then $x \in y$.

(C_2) Therefore, $x \in y$.

EXAMPLE 3: Let us use $\star\star$ as a name for the sentence "The first sentence in this chapter which contains an occurrence of the symbol @ is not true." One may argue that:

(A_3) If $\star\star$ is true, then $\star\star$ is not true.

(B_3) If $\star\star$ is not true, then $\star\star$ is not true.

(C_3) Therefore, $\star\star$ is not true.

The reader need not be concerned with all the details of the arguments above, or even with whether the statements (A_i) and (B_i) were correctly inferred in each argument. The important point is that (C_i) is indeed a logical consequence of (A_i) and (B_i) in each argument.

Note that the main inference in each argument has the following form:

(A) If p, then q.

(B) If not p, then q.

(C) Therefore, q.

Obviously, the key words involved in this inference are "if", "then", and "not". These are connectives, words which can be used to construct more complex statements from simpler statements. Statements are sometimes called propositions (though some prefer to say that statements express propositions), and certain connectives which are important for logical purposes are represented in logistic systems as *propositional connectives.* A propositional calculus is a logistic system which formalizes the logical use of propositional connectives.

It is customary to use special symbols instead of English words for propositional connectives. This is very convenient, since the connectives are used very often and the symbols are quicker to write than the words (once one

becomes accustomed to them). Also, the symbols allow one to exhibit very clearly the logical structure of a proposition which might be expressed in a variety of ways in English. Logistic systems which use symbols in this way are also referred to as systems of symbolic logic.

We shall express "If p, then q" symbolically as "$p \supset q$", and "not p" as "$\sim p$". In this notation, the inference above can be expressed as follows:

(A') $p \supset q$

(B') $\sim p \supset q$

(C') q

(The word "therefore" indicates that a conclusion is being reached, but has no logical content itself, so we have omitted it from (C').)

Before discussing the details of the logistic system $\mathcal{P}$, let us make a brief survey of the most important propositional connectives.

Negation ($\sim$)

We write the negation of the statement p as $\sim p$. In English this is generally expressed by inserting the word "not" into the sentence at an appropriate point. If w is the statement "everyone gets wet", $\sim w$ can be expressed by the statement "not everyone gets wet", but if r is the statement "it is raining", $\sim r$ is expressed by "it is not raining".

If p is true, then the statement $\sim p$ is false, while if p is a false statement, then $\sim p$ is a true statement. This is summarized in Figure 1.1. which we call the truth table for $\sim$. We write T for True, and F for False.

p	$\sim p$
T	F
F	T

Figure 1.1: Truth table for $\sim$

Conjunction ($\wedge$)

The conjunction of statements p and q is written as $[p \wedge q]$, and is generally expressed in English as "p and q" or "p, but q". Note that the statements "It is raining, and Jack is going to the picnic" and "It is raining, but Jack is going to the picnic" are *logically* equivalent, though they differ in emphasis. They both mean that the statement "It is raining" is true and

p	q	$p \wedge q$	$p \vee q$	$p \supset q$	$p \equiv q$	$p \not\equiv q$
T	T	T	T	T	T	F
T	F	F	T	F	F	T
F	T	F	T	T	F	T
F	F	F	F	T	T	F

Figure 1.2: Truth tables for $\wedge$, $\vee$, $\supset$, $\equiv$, $\not\equiv$

the statement "Jack is going to the picnic" is true. The statement $p \wedge q$ is true precisely when p and q are both true.

In Figure 1.2 we display the truth table for $\wedge$ and the other connectives which we will discuss below. Each horizontal line of the truth table corresponds to one of the logical possibilities with regard to the truth or falsity of the statements p and q. For example, in line 1 (below the bar) of the truth table, p is true and q is true and $[p \wedge q]$ is true, while in line 2, p is true and q is false and $[p \wedge q]$ is false.

Disjunction ($\vee$)

The disjunction of p and q is written as $[p \vee q]$, and generally expressed in English as "p or q". As can be seen from Figure 1.2, we regard $[p \vee q]$ as true if either or both of the statements p and q are true. Thus, we use $\vee$ for *inclusive* disjunction; we include the case where p and q are both true as one of the cases in which $[p \vee q]$ is true. For mathematical purposes, this is the generally accepted usage of the word "or". For example, 24 is regarded as a member of the set of integers which are multiples of 2 or multiples of 3. However, in ordinary usage the word "or" is sometimes used in the *exclusive sense*, in which "p or q" means p is true or q is true, but not both. For example, if Jack says, "I will go to the picnic or I will go to the tennis match", he may mean that he will go to one or the other, but not both. We will discuss exclusive disjunction below.

Implication ($\supset$)

$[p \supset q]$ means that the statement p implies the statement q, i.e., q is true whenever p is true. This can be translated into English in a variety of ways, including "if p, then q"; "q, if p"; "p only if q"; "p is sufficient for q"; "q is necessary for p" (i.e., "it is necessary for q to be true in order for p to be true").

We intend $\supset$ to represent *material implication*, so that the truth of $[p \supset q]$ depends only on the truth or falsity of the statements p and q.

Other notions of implication, in which the truth of "p implies q" may depend on such factors as the relevance of p to q, may be useful for certain purposes, but are much more complex than material implication, and seem not to be necessary for the formalization of mathematics, so we shall not consider them further here.

Note from Figure 1.2 that $[p \supset q]$ is regarded as true when p is false, without regard for q. The appropriateness of this way of defining material implication may be illuminated by supposing that p means "it rains", and q means "the street gets wet", so $[p \supset q]$ means "if it rains, then the street gets wet". If the street is protected from the rain in some way, so that on a certain day it rains, but the street stays dry, then on that day the statement "if it rains, then the street gets wet" is clearly false. However, if the street is not so protected, we would expect the statement "if it rains, then the street gets wet" to be true every day, whether or not it rains that day. In particular, if on a certain day it does not rain but the street gets wet because a pail of water is spilled on it, the statement "if it rains, then the street gets wet" is still true. Similarly, the statement is true on a day when it does not rain and the street remains dry. Thus the only case in which $[p \supset q]$ is false is that in which p is true and q is false.

Equivalence ($\equiv$)

$[p \equiv q]$ means that the statements p and q are equivalent with respect to truth; that is, they are both true or both false.

Figure 1.3 displays truth tables for $[[p \supset q] \wedge [q \supset p]]$ and $[p \equiv q]$. In each line of the truth table, the truth of each component statement is displayed under the main connective of that statement. It is easy to see that in every possible situation, $[[p \supset q] \wedge [q \supset p]]$ and $[p \equiv q]$ are both true or both false; that is, they are equivalent. Thus, p is equivalent to q means that p implies q and that q implies p. This leads to several alternative ways of expressing the assertion that p is equivalent to q which are encountered very frequently in mathematics.

p	q	$[[p \supset q]$	$\wedge$	$[q \supset p]]$	$[p \equiv q]$
T	T	T	T	T	T
T	F	F	F	T	F
F	T	T	F	F	F
F	F	T	T	T	T

Figure 1.3: Truth tables for $[[p \supset q] \wedge [q \supset p]]$ and $[p \equiv q]$

Chapter 2

First-Order Logic

We next consider first-order logic, which is also called quantification theory and predicate calculus.[1] Initially we discuss first-order logic without equality, and then we discuss first-order logic with equality in §26.

§20. The Language of $\mathcal{F}$

Having discussed in Chapter 1 how to combine statements using propositional connectives, we next turn to the problem of formally representing the statements themselves in a way that shows more of their structure. We shall focus on those aspects of the structure of statements which are most important for representing reasoning involving these statements.

Statements make assertions about entities (such as people, animals, plants, colors, qualities, numbers, abstractions, ideas, feelings, etc.), which are designated by nouns and pronouns. For convenience we shall refer to these entities as *objects* or *individuals*, slightly extending the usual meanings of these words in English. In our formal language we shall introduce *terms* (which will be defined below) to denote (serve as names for) these individuals.

Statements can be regarded as asserting that individuals have certain properties, or that individuals are related to each other in certain ways. For example, "Jack will go to the picnic" expresses the assertion that the individual Jack has the property of being an individual who will go to the

[1] Frege [Frege, 1879] is generally credited with first developing those aspects of first-order logic which go beyond propositional calculus, though others independently developed similar ideas somewhat later, and Frege's notation never found general favor. See [Church, 1956, pp. 288–294] for more information about the history of first-order logic.

picnic. "112 is prime" expresses the assertion (which happens to be false) that the individual 112 has the property of being a prime number. "Jill votes for Alice" expresses the assertion that the individuals "Jill" and "Alice" are related by the relation "votes for". "$5 < 112$" expresses the assertion that the individuals 5 and 112 are related by the relation " $<$ ". Another way of saying this is to assert that the relation " $<$ " *holds* between 5 and 112.

Since it is sometimes complicated to analyze a sentence in a natural language such as English or Chinese, it is convenient to have a quite simple and uniform syntax to represent the assertion that the entity t has the property P, or that the relation R holds between the entities s and t. We shall use Pt to represent the former, and Rst to represent the latter. (Sometimes it will be convenient to introduce commas or parentheses and write these as $P(t)$ and $R(s,t)$, but the commas and parentheses are in principle unnecessary.) Thus, if we let "Prime" be the property of being a prime number, we can express "112 is prime" as "Prime(112)". Similarly, we can express "Jill votes for Alice" as "VotesFor(Jill, Alice)". We call names for properties and relations *predicates*, and they play such a fundamental role in first-order logic that it is sometimes called predicate calculus.

In order to be able to deal adequately with reasoning involving individuals, one must be able to represent statements containing words like "all", "every", and "some". To illustrate the relevant ideas, let us suppose that a class of girls holds an election for class president. Each girl is given a ballot containing the names of all the girls in the class, and is allowed to vote for as many members of the class as she pleases. Suppose we want to express in a formal language the assertion that everyone votes for Alice. One way one might try to formalize this is to introduce a term "everyone" into the formal language, and express the assertion as "VotesFor(everyone, Alice)". In a similar way, to express the assertion that Jill votes for someone, one might introduce a term "someone" into the formal language, and write "VotesFor(Jill, someone)". If such statements were permitted in the formal language, it would be very natural to also permit "VotesFor(everyone, someone)". What should this mean? One obvious possibility is "Everyone votes for someone". However, there is another possibility: "There is someone for whom everyone votes". Note that these express quite distinct assertions; the latter statement asserts that everyone votes for the same person, while the former statement does not.

Contemplation of this and other examples (particularly those dealing with more complicated statements) leads to the conclusion that a better way of formally representing assertions involving words like "all", "every", and "some" is needed. The solution that has been developed is to use *indi-*

VotesFor(Jill, Alice)	Jill votes for Alice.
$\forall x$ VotesFor(x, Alice)	Everyone votes for Alice.
$\forall y$ VotesFor(Jill, y)	Jill votes for everyone.
$\forall x$ VotesFor(x, x)	Everyone votes for herself.
$\forall x \forall y$ VotesFor(x, y)	Everyone votes for everyone.
$\forall x \exists y$ VotesFor(x, y)	Everyone votes for someone.
$\exists y \forall x$ VotesFor(x, y)	There is someone for whom everyone votes.
$\exists x \forall y$ VotesFor(x, y)	There is someone who votes for everyone.
$\forall y \exists x$ VotesFor(x, y)	Everyone gets someone's vote.
$\exists x \exists y$ VotesFor(x, y)	Someone votes for someone.
$\exists x$ VotesFor(x, x)	Someone votes for herself.
$\exists x$ VotesFor(x, Alice)	Someone votes for Alice.
$\exists y$ VotesFor(Jill, y)	Jill votes for someone.

Figure 2.1: Statements about voting in symbolic form and in English.

vidual variables to refer to arbitrary (unspecified) individuals, and express the assertion that "Everyone votes for Alice" as "For every x, x votes for Alice", and express the assertion that "Jill votes for someone" as "There is a y such that Jill votes for y." Just as we introduced symbols for propositional connectives, we introduce the symbol $\forall$ to abbreviate the phrase "for every" (or "for all") and the symbol $\exists$ and write "$\exists y$" to abbreviate the phrase "there is a y such that" (where y is any individual variable). Now we can express "Everyone votes for someone" as "$\forall x \exists y$ VotesFor(x, y)" and "There is someone for whom everyone votes" as "$\exists y \forall x$ VotesFor(x, y)". With this symbolic representation, the distinction between these assertions is made very clear. In Figure 2.1 we list a variety of statements about voting in symbolic form and in English. Note how the symbolic representations of these statements make it easy to focus on the essential logical differences between them.

Expressions of the form $\forall \mathbf{x}$ and $\exists \mathbf{x}$ are called *quantifiers*. They play such a fundamental role in first-order logic that it is sometimes called quantification theory. The reason we actually use the name "first-order logic" will become evident when we discuss higher-order logic at the beginning of section §50.

Just as $\vee$ can be defined in terms of $\sim$ and $\wedge$, $\exists$ can be defined in terms of $\sim$ and $\forall$. We can see this from the following example. Suppose our class of girls decides to vote on some issue, but each girl has the option of voting or abstaining (not voting). If we write "x votes" as "Votes(x)", then "$\sim$Votes(x)" means "x does not vote" or "x abstains". Therefore

"$\forall$x$\sim$Votes(x)" means "everyone abstains", and "$\sim \forall x \sim$Votes(x)" means "not everyone abstains", which is of course equivalent to the assertion that "someone votes". From this and other examples it can be seen that $\exists x Px$ always has the same logical meaning as $\sim \forall x \sim Px$, provided that there is at least one individual.[2] This provides a contextual definition of $\exists$ in terms of $\sim$ and $\forall$. Therefore, for the sake of economy, $\exists$ will not be a primitive symbol of the formal system $\mathcal{F}$ of first-order logic which we shall soon introduce.

Sometimes we wish to refer to individuals by more complex expressions than we have used so far, such as "Jill's father", "$x + 3$", or "the value of the diamond". We can do this in a uniform way by having in our language notations for functions which map individuals to individuals. If we let f(x) mean "the father of x", g(x, y) mean "$x + y$", and V(x) mean "the value of x", then we can represent "Jill's father" as f(Jill), "$x + 3$" as g(x, 3), and "the value of the diamond" as V(the diamond).

The symbols we will use to refer to individuals, functions, and predicates will be *variables* or *constants*. Variables are used to refer to arbitrary elements of the appropriate type, whereas constants denote particular elements. Thus, variables may occur in quantifiers, but constants may not.

Now we are ready to start describing the system $\mathcal{F}$ of first-order logic. The *primitive symbols* of $\mathcal{F}$ are the following:

(a) Improper symbols: [ ] $\sim$ $\vee$ $\forall$
(b) Individual variables: $u\ v\ w\ x\ y\ z\ u_1\ v_1\ w_1\ \ldots\ u_2\ \ldots$
(c) n-ary function variables: $f^n\ g^n\ h^n\ f_1^n\ g_1^n\ h_1^n\ \ldots$ for each natural number $n \geq 1$.
(d) Propositional variables: $p\ q\ r\ s\ p_1\ q_1\ r_1\ s_1\ \ldots$
(e) n-ary predicate variables: $P^n\ Q^n\ R^n\ S^n\ P_1^n\ Q_1^n\ R_1^n\ S_1^n \ldots$ for each natural number $n \geq 1$.

An n-ary symbol is called *unary* or *monadic* if $n = 1$, and it is called *binary* if $n = 2$. There are denumerably many variables of each type. Individual variables may be regarded as 0-ary function variables, and propositional variables may be regarded as 0-ary predicate variables. In addition, there may be finitely or infinitely many individual, function, propositional, or predicate constants. We also admit systems in which there are no variables of certain types; however, individual variables must always be present, and there must be predicate variables or at least one predicate constant. Each choice of constants and variables determines a different formulation of the

[2]There is little need for a logical system to discuss nothing, so, as is customary, our logical system will be designed with the tacit assumption that there is at least one individual under discussion.

system $\mathcal{F}$, of course; thus we use $\mathcal{F}$ ambiguously as the name of any one of a variety of quite similar systems.

The *terms* and *wffs* of $\mathcal{F}$ are defined inductively by the following *formation rules*:

(a) Each individual variable or constant is a term.
(b) If $\mathbf{t}_1, \ldots, \mathbf{t}_n$ are terms and $\mathbf{f}^n$ is an n-ary function variable or constant, then $\mathbf{f}^n\mathbf{t}_1 \ldots \mathbf{t}_n$ is a term.
(c) If $\mathbf{t}_1, \ldots, \mathbf{t}_n$ are terms and $\mathbf{P}^n$ is an n-ary predicate variable or constant, then $\mathbf{P}^n\mathbf{t}_1 \ldots \mathbf{t}_n$ is a wff. Also, each propositional variable standing alone is a wff. (Wffs of these forms are called *atomic* or *elementary* wffs.)
(d) If $\mathbf{A}$ is a wff, so is $\sim\mathbf{A}$.
(e) If $\mathbf{A}$ and $\mathbf{B}$ are wffs, so is $[\mathbf{A} \vee \mathbf{B}]$.
(f) If $\mathbf{A}$ is a wff and $\mathbf{x}$ is an individual variable, then $\forall\mathbf{x}\mathbf{A}$ is a wff.

Sometimes $\forall\mathbf{x}\mathbf{A}$ is written as $(\forall\mathbf{x})\mathbf{A}$ or simply as $(\mathbf{x})\mathbf{A}$. We shall often omit the numerical superscripts from function and predicate symbols in terms and wffs, since the context will show what they must be.

We leave it to the reader (exercises X2006 and X2007) to formulate definitions of the sets of terms and wffs in the style of the definition of the set of wffs in §10 and to formulate and prove analogues of Theorem 1000 (the Principle of Induction on the Construction of a Wff) for terms and wffs of $\mathcal{F}$.

As in the case of propositional calculus, we shall use **A, B, C, D, E**, etc., as syntactical variables ranging over wffs. Similarly, we use **u, v, w, x, y, z**, etc., as syntactical variables for individual variables.

While in first-order logic only individual variables may be quantified, in second-order logic the other types of variables may be quantified too. Since variables designate arbitrary elements of the appropriate type, it is normally appropriate to substitute arbitrary expressions of the appropriate type for variables, under suitable conditions. Such substitution for constants is generally not appropriate since it would have the effect of implying that all elements have the special properties of the element denoted by the constant in question.

For the sake of economy, starting with §25 we shall restrict our attention to formulations of $\mathcal{F}$ in which there are no variables other than individual variables.

We adopt from §10 the conventions for regarding $[\mathbf{A} \wedge \mathbf{B}]$, $[\mathbf{A} \supset \mathbf{B}]$, and $[\mathbf{A} \equiv \mathbf{B}]$ as abbreviations for wffs (now wffs of $\mathcal{F}$). In addition,

Chapter 3

Provability and Refutability

We now turn our attention to the practical problems of proving theorems of first-order logic. If one has unlimited resources and patience, one can simply enumerate proofs until one finds a proof of the wff under consideration (provided, of course, that it actually is a theorem). As a practical matter, however, one usually wishes to find a proof as quickly and easily as possible, so various methods have been devised to facilitate efficient searches for proofs. We shall discuss some of these methods in this chapter, along with various results related to them.

A *refutation* of a wff **B** is a proof that **B** is contradictory; this amounts to a proof of $\sim$ **B**. Thus, one way to prove a wff **A** is to refute $\sim$ **A**. This method is proving a wff **A** is often called the indirect method, and it is surprisingly useful, so a number of the procedures we discuss below are refutation procedures rather than proof procedures. Since a refutation of $\sim$ **A** provides a proof of **A**, we shall sometimes speak of refutation procedures as proof procedures.

We say that we refute a set $\mathcal{S}$ of sentences when we drive a contradiction from $\mathcal{S}$. Naturally, such a refutation shows that some finite conjunction of members of $\mathcal{S}$ is contradictory.

§30. Natural Deduction

Proofs of theorems of $\mathcal{F}$ (which is known as a *Hilbert-style* system) are rarely written out in full, since they tend to be long, awkward, and unpleasant to read. In practice, one uses proofs from hypotheses and derived rules of

inference of $\mathcal{F}$. In this section we introduce a system $\mathcal{N}$ of *natural deduction*,[1] in which it is possible to give fairly natural and well structured proofs, and express the forms of arguments which arise in mathematical practice. Of course, the rules of inference of $\mathcal{N}$ are all derived rules of inference of $\mathcal{F}$, and this section may be regarded simply as a summary of those derived rules of inference of $\mathcal{F}$ which are most useful for writing our proofs.

Later in this chapter we discuss a variety of methods and logical tools for establishing the validity of wffs of first-order logic. Generally, proofs can be translated from one format into another (though real insight may be required to design algorithms to do this), and once one has found the essential ingredients of a proof, one may wish to express them in a proof which is reasonably comprehensible. The system $\mathcal{N}$ provides one standard for what such a proof should look like.

The exact choice of primitive connectives and quantifiers for this section does not matter very much, but it is natural to take at least $\sim$, $\vee$, $\wedge$, $\supset$, and f (falsehood) as primitive connectives, and both $\forall$ and $\exists$ as primitive quantifiers. f plays the role of a contradiction in indirect proofs.

Various definitions, such as *free for*, must be adjusted in trivial ways to take account of the fact that $\exists$ is now a primitive quantifier, and we leave this task to the reader. To avoid needless redundancy we shall use $\mathbf{M}$ and $\mathbf{N}$ as syntactical variables for wffs or for the "null formula" (empty disjunction). When $\mathbf{M}$ is null, $\mathbf{M} \vee \mathbf{A}$ and $\mathbf{A} \vee \mathbf{M}$ both stand for $\mathbf{A}$. A null formula standing alone may be regarded as an abbreviation for f. In this section we use $\mathcal{H}$ as a syntactical variable for a finite (possibly empty) *sequence* of wffs, and write $\mathcal{H}, \mathbf{A}$ for the sequence obtained by appending $\mathbf{A}$ to the sequence $\mathcal{H}$. As in §21, we may say that $\mathbf{x}$ is not free in $\mathcal{H}$ when we mean that $\mathbf{x}$ is not free in any wff of $\mathcal{H}$.

A *natural deduction proof* consists of a finite sequence of *lines* of the form $\mathcal{H} \vdash \mathbf{A}$. The members of the sequence $\mathcal{H}$ are called *hypotheses*, and the wff $\mathbf{A}$ is called the *assertion* of the line. Each line must be inferred from zero or more preceding lines by one of the following rules of inference.

Rules of Inference

Hypothesis Rule (Hyp). Infer $\mathcal{H} \vdash \mathbf{A}$ whenever $\mathbf{A}$ is a member of $\mathcal{H}$.

Rule for Expanding or Rearranging Hypotheses. From $\mathcal{H}_1 \vdash \mathbf{A}$ infer $\mathcal{H}_2 \vdash \mathbf{A}$, provided that every wff in $\mathcal{H}_1$ is also in $\mathcal{H}_2$.

[1] Natural deduction was introduced in [Gentzen, 1935]. Also see [Quine, 1950], [Prawitz, 1965], and references cited therein.

Deduction Rule (Ded). From $\mathcal{H}$, $\mathbf{A} \vdash \mathbf{B}$ infer $\mathcal{H} \vdash \mathbf{A} \supset \mathbf{B}$.

Rule P. From $\mathcal{H} \vdash \mathbf{A}_1, \ldots,$ and $\mathcal{H} \vdash \mathbf{A}_n$ infer $\mathcal{H} \vdash \mathbf{B}$, provided that $[[\mathbf{A}_1 \wedge \ldots \wedge \mathbf{A}_n] \supset \mathbf{B}]$ is tautologous.

Negation Rule (Neg.). From $\mathcal{H} \vdash \mathbf{A}$, infer $\mathcal{H} \vdash \mathbf{B}$, where $\mathbf{A}$ is $\sim \forall \mathbf{x}\mathbf{C}$, $\sim \exists \mathbf{x}\mathbf{C}$, $\forall \mathbf{x} \sim \mathbf{C}$, or $\exists \mathbf{x} \sim \mathbf{C}$, and $\mathbf{B}$ is $\exists \mathbf{x} \sim \mathbf{C}$, $\forall \mathbf{x} \sim \mathbf{C}$, $\sim \exists \mathbf{x}\mathbf{C}$, or $\sim \forall \mathbf{x}\mathbf{C}$, respectively.

Rule of Indirect Proof (IP). From $\mathcal{H}$, $\sim \mathbf{A} \vdash \mathsf{f}$ infer $\mathcal{H} \vdash \mathbf{A}$.

Rule of Cases (Cases). From $\mathcal{H} \vdash \mathbf{A} \vee \mathbf{B}$ and $\mathcal{H}$, $\mathbf{A} \vdash \mathbf{C}$ and $\mathcal{H}$, $\mathbf{B} \vdash \mathbf{C}$ infer $\mathcal{H} \vdash \mathbf{C}$.

Rule of Alphabetic Change of Bound Variables (α). From $\mathcal{H} \vdash \mathbf{A}$ infer $\mathcal{H} \vdash \mathbf{A}'$, where $\mathbf{A}'$ is obtained from $\mathbf{A}$ upon replacing an occurrence of $\forall \mathrm{x}\mathrm{C}$ $[\exists \mathbf{x}\mathbf{C}]$ in $\mathbf{A}$ by an occurrence of $\forall \mathbf{y}\mathsf{S}^{\mathbf{x}}_{\mathbf{y}}\mathbf{C}$ $[\exists \mathbf{y}\mathsf{S}^{\mathbf{x}}_{\mathbf{y}}\mathbf{C}]$, where $\mathbf{y}$ is not free in $\mathbf{C}$ and $\mathbf{y}$ is free for $\mathbf{x}$ in $\mathbf{C}$.

Universal Generalization ($\forall$G). From $\mathcal{H} \vdash \mathbf{M} \vee \mathbf{A} \vee \mathbf{N}$ infer $\mathcal{H} \vdash \mathbf{M} \vee \forall \mathbf{x}\mathbf{A} \vee \mathbf{N}$, provided that $\mathbf{x}$ is not free in $\mathcal{H}$, $\mathbf{M}$, or $\mathbf{N}$.

Existential Generalization ($\exists$G). Let $\mathbf{A}(\mathbf{x})$ be a wff and let $\mathbf{t}$ be a term which is free for $\mathbf{x}$ in $\mathbf{A}(\mathbf{x})$. ($\mathbf{t}$ may occur in $\mathbf{A}(\mathbf{x})$.) From $\mathcal{H} \vdash \mathbf{M} \vee \mathbf{A}(\mathbf{t}) \vee \mathbf{N}$ infer $\mathcal{H} \vdash \mathbf{M} \vee \exists \mathbf{x}\mathbf{A}(\mathbf{x}) \vee \mathbf{N}$.

Universal Instantiation ($\forall$I). From $\mathcal{H} \vdash \forall \mathbf{x}\mathbf{A}(\mathbf{x})$ infer $\mathcal{H} \vdash \mathbf{A}(\mathbf{t})$, provided that $\mathbf{t}$ is a term free for $\mathbf{x}$ in $\mathbf{A}(\mathbf{x})$.

Rule C. From $\mathcal{H} \vdash \exists \mathbf{x}\mathbf{B}(\mathbf{x})$ and $\mathcal{H}$, $\mathbf{B}(\mathbf{y}) \vdash \mathbf{A}$ infer $\mathcal{H} \vdash \mathbf{A}$, where $\mathbf{y}$ is an individual variable which is free for $\mathbf{x}$ in $\mathbf{B}(\mathbf{x})$ and which is not free in $\mathcal{H}$, $\exists \mathbf{x}\mathbf{B}(\mathbf{x})$ or in $\mathbf{A}$.

Note that the system $\mathcal{N}$ has no axioms. We remark that the Rule for Expanding or Rearranging Hypotheses is often used tacitly and without explicit mention in combination with other rules. Rule $\forall$G really has four forms:

from $\mathcal{H} \vdash \mathbf{A}$	infer $\mathcal{H} \vdash \forall \mathbf{x}\mathbf{A}$;
from $\mathcal{H} \vdash \mathbf{A} \vee \mathbf{C}$	infer $\mathcal{H} \vdash \forall \mathbf{x}\mathbf{A} \vee \mathbf{C}$;
from $\mathcal{H} \vdash \mathbf{B} \vee \mathbf{A}$	infer $\mathcal{H} \vdash \mathbf{B} \vee \forall \mathbf{x}\mathbf{A}$;
from $\mathcal{H} \vdash \mathbf{B} \vee \mathbf{A} \vee \mathbf{C}$	infer $\mathcal{H} \vdash \mathbf{B} \vee \forall \mathbf{x}\mathbf{A} \vee \mathbf{C}$.

(In each case, $\mathbf{x}$ must not be free in $\mathcal{H}$, $\mathbf{B}$, or $\mathbf{C}$.) Our use of the null formula simply enables us to compress these four statements into one. Rule P can actually be restricted to certain special cases of Rule P (such as certain rules of the system $\mathcal{G}$ of §31), but we shall not discuss that here. Naturally, if one wishes to prove theorems involving $=$, one should add to the rules above certain derived rules of inference of the system discussed in §26. Also, in some contexts it would be natural to include a rule permitting one to infer any previously proved theorem.

The soundness and completeness of $\mathcal{N}$ follow from the corresponding results for $\mathcal{F}$, though a careful proof must deal with the fact that $\exists$ is a primitive symbol of $\mathcal{N}$, but not of $\mathcal{F}$. We leave further consideration of these matters to the reader (Exercises X3003 and X3004).

If $\mathbf{A}(\mathbf{x})$ is a wff in which $\mathbf{x}$ occurs free, and one wishes to prove $\exists\mathbf{x}\mathbf{A}(\mathbf{x})$, a natural approach is to try to find a term $\mathbf{t}$ such that one can prove $\mathbf{A}(\mathbf{t})$, and then derive $\exists\mathbf{x}\mathbf{A}(\mathbf{x})$ by $\exists$G. For example, if $\mathbf{A}(\mathbf{x})$ is $[Qy \supset Q\mathbf{x}]$, one can prove $\mathbf{A}(y)$ (i.e., $[Qy \supset Qy]$), and from this infer $\exists\mathbf{x}\mathbf{A}(\mathbf{x})$ (i.e., $\exists\mathbf{x}[Qy \supset Q\mathbf{x}]$).

Sometimes this approach will not work, as in the case where $\mathbf{A}(\mathbf{x})$ is $[Q\mathbf{x} \supset \cdot Qa \wedge Qb]$. Nevertheless, in this case one can prove $\mathbf{A}(a) \vee \mathbf{A}(b)$ (i.e., $[Qa \supset \cdot Qa \wedge Qb] \vee [Qb \supset \cdot Qa \wedge Qb]$, which is tautologous), from which one can infer $\exists\mathbf{x}\mathbf{A}(\mathbf{x}) \vee \exists\mathbf{x}\mathbf{A}(\mathbf{x})$ and hence infer $\exists\mathbf{x}\mathbf{A}(\mathbf{x})$ by Rule P. Thus, a natural generalization of the approach to proving $\exists\mathbf{x}\mathbf{A}(\mathbf{x})$ mentioned above is to find terms $\mathbf{t}_1, \ldots, \mathbf{t}_n$ such that one can prove $\mathbf{A}(\mathbf{t}_1) \vee \ldots \vee \mathbf{A}(\mathbf{t}_n)$ and from this infer $\exists\mathbf{x}\mathbf{A}(\mathbf{x}) \vee \ldots \vee \exists\mathbf{x}\mathbf{A}(\mathbf{x})$, and hence $\exists\mathbf{x}\mathbf{A}(\mathbf{x})$.

However, there are cases when even this generalized approach does not quite work. Consider the problem of giving a direct natural deduction proof of $\exists x \forall y \cdot Px \supset Py$. One proof is:

$\vdash [Px \supset Py] \vee \cdot Py \supset Pz$	Rule P
$\vdash [Px \supset Py] \vee \forall z \cdot Py \supset Pz$	$\forall$G
$\vdash [Px \supset Py] \vee \exists x \forall z \cdot Px \supset Pz$	$\exists$G
$\vdash \forall y[Px \supset Py] \vee \exists x \forall z \cdot Px \supset Pz$	$\forall$G
$\vdash \exists x \forall y[Px \supset Py] \vee \exists x \forall z \cdot Px \supset Pz$	$\exists$G
$\vdash \exists x \forall y[Px \supset Py] \vee \exists x \forall y \cdot Px \supset Py$	α
$\vdash \exists x \forall y \cdot Px \supset Py$	Rule P

Note that in this proof we do not have terms $\mathbf{t}_1$ and $\mathbf{t}_2$ in which y is not free such that we prove $\forall y[P\mathbf{t}_1 \supset Py] \vee \forall y[P\mathbf{t}_2 \supset Py]$. Could there be such terms? Consider an interpretation $\mathcal{M} = \langle \mathcal{D}, \mathcal{J} \rangle$, where

$\mathcal{D} = \{a, b\}$;

$\mathcal{J}\mathbf{c} = a$ for all constants $\mathbf{c}$;

for each function symbol $\mathbf{f}^n$

$(\mathcal{J}\mathbf{f}^n)d_1 \ldots d_n = a$ for all $d_1, \ldots, d_n$ in $\mathcal{D}$;

$(\mathcal{J}\ P)a = \mathsf{T}$, and $(\mathcal{J}\ P)b = \mathsf{F}$.

Let $\varphi\mathbf{x} = a$ for all individual variables $\mathbf{x}$; then $\mathcal{V}_\varphi^{\mathcal{M}}\mathbf{t} = a$ for each term $\mathbf{t}$. Thus if y does not occur in a term $\mathbf{t}$, and ψ is the assignment which agrees with φ off y, while $\psi y = b$, then

$$\mathcal{V}_\psi[P\mathbf{t} \supset Py] = \mathsf{F}, \quad \text{so} \quad \mathcal{V}_\varphi^{\mathcal{M}}\forall y[P\mathbf{t} \supset Py] = \mathsf{F}.$$

Thus one cannot prove any disjunction of the form $\bigvee_{i=1}^n \forall y[P\mathbf{t}_i \supset Py]$, where the terms $\mathbf{t}_i$ do not contain y.

EXERCISES

Prove the following theorems in $\mathcal{N}$:

X3000. $\sim \forall \mathbf{x}\mathbf{A} \equiv \exists \mathbf{x} \sim \mathbf{A}$.

X3001. $\sim \exists \mathbf{x}\mathbf{A} \equiv \forall \mathbf{x} \sim \mathbf{A}$.

X3002. $\forall u \forall v \forall w[Puv \vee Pvw] \supset \exists x \forall y Pxy$. (*Hint:* recall the advice given near the end of §21 about proving theorems of the form $\mathbf{A} \supset \mathbf{B}$.)

X3003. Prove that the system $\mathcal{N}$ is sound in the sense of 2303.

X3004. Prove that the system $\mathcal{N}$ is complete in the sense of 2509.

X3005. Add rules of inference for $=$ to $\mathcal{N}$, and prove that the resulting system is sound and complete in the sense of 2609 and 2612.

X3006. Are any of the rules of inference of $\mathcal{N}$ dependent (non-independent) in the sense of §13?

X3007. Find a system $\mathcal{N}'$ which can be obtained from $\mathcal{N}$ by deleting certain rules of inference, and whose rules of inference are all independent. Prove the independence of these rules.

Chapter 4

Further Topics in First-Order Logic

§40. Duality

We introduce this subject with a parable about two scholars, named Oren and Nero, who were visiting an archaeologist and were shown a recently discovered tablet, the contents of which are reproduced in Figure 4.1. They soon realized that the figures on the tablet were truth tables, and they set about translating them into more familiar notations. (Before proceeding further, the reader is advised to do this for himself for at least a few of the tables.) Oren produced the translation in Figure 4.2, and Nero produced that in Figure 4.3. As they started to show their translations to the archaeologist, Nero modestly remarked "That was really quite easy, as soon as I realized that $\oplus$ denoted conjunction". "But you're wrong!" exclaimed Oren. "$\oplus$ denoted disjunction!" They argued for some time, and neither was able to persuade the other that he was wrong. All they could agree on was that $\ominus$ denoted negation.

Then the archaeologist showed them an inscription which had been found on another tablet. "We've figured out how to translate most of the language they apparently used in everyday affairs", he said, "but this inscription seems to be in a mixture of two languages. Here's how the inscription looks under the translation we've achieved so far:"

> 'The value of $(\boxplus\ \mathbf{x})\mathbf{A}(\mathbf{x})$ is $\triangledown$ iff there is an n such that the value of $\mathbf{A}(n)$ is $\triangledown$.'

Oren and Nero recognized that this was another fragment of a text on logic,

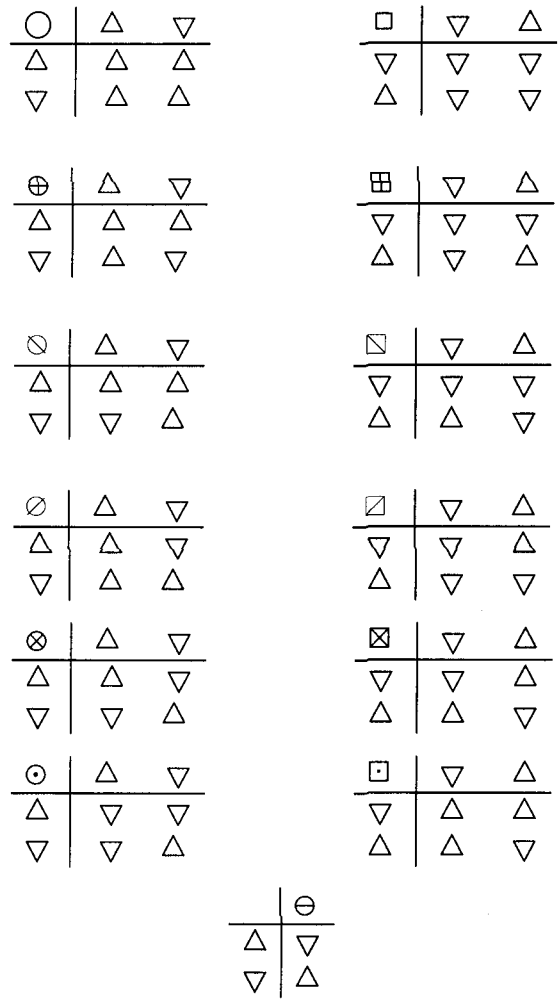

Figure 4.1: The tablet

and each soon produced a translation of it. Oren's translation was:

> 'The value of $\forall \mathbf{x}\mathbf{A}(\mathbf{x})$ is falsehood iff there is an n such that the value of $\mathbf{A}(n)$ is falsehood.'

Nero's translation was:

> 'The value of $\exists \mathbf{x}\mathbf{A}(\mathbf{x})$ is truth iff there is an n such that the value of $\mathbf{A}(n)$ is truth.'

Again, they could find no way to resolve their disagreement.

t	T	F
T	T	T
F	T	T

f	F	T
F	F	F
T	F	F

$\vee$	T	F
T	T	T
F	T	F

$\wedge$	F	T
F	F	F
T	F	T

$\subset$	T	F
T	T	T
F	F	T

$\not\supset$	F	T
F	F	F
T	T	F

$\supset$	T	F
T	T	F
F	T	T

$\not\subset$	F	T
F	F	T
T	F	F

$\equiv$	T	F
T	T	F
F	F	T

$\not\equiv$	F	T
F	F	T
T	T	F

$\downarrow$	T	F
T	F	F
F	F	T

$\mid$	F	T
F	T	T
T	T	F

	$\sim$
T	F
F	T

Figure 4.2: Oren's translation of the tablet

f	F	T
F	F	F
T	F	F

t	T	F
T	T	T
F	T	T

$\wedge$	F	T
F	F	F
T	F	T

$\vee$	T	F
T	T	T
F	T	F

$\not\supset$	F	T
F	F	F
T	T	F

$\subset$	T	F
T	T	T
F	F	T

$\not\subset$	F	T
F	F	T
T	F	F

$\supset$	T	F
T	T	F
F	T	T

$\not\equiv$	F	T
F	F	T
T	T	F

$\equiv$	T	F
T	T	F
F	F	T

$\mid$	F	T
F	T	T
T	T	F

$\downarrow$	T	F
T	F	F
F	F	T

	$\sim$
F	T
T	F

Figure 4.3: Nero's translation of the tablet

"There's one more fragment of a tablet you might be interested in", said the archeologist. "We managed to translate the first word on it as '**AXIOMS**', and there's a line under that which looks like this:"

$$(\boxplus p)[p \oplus \ominus p].$$

"That settles it!" exclaimed Oren. "That translates to '$\forall p[p \vee \sim p]$', which is a fine axiom. Clearly my method of translation is correct." "Not at all", replied Nero. "It translates to '$\exists p[p \wedge \sim p]$'. The people who wrote these

tablets were known throughout the ancient world as deceitful, treacherous, inveterate liars. Apparently they even axiomatized their lies!"

We leave it to the reader to decide what the moral of this story is.

Clearly there is a symmetry, or *duality*, in logic induced by systematically interchanging truth and falsehood. We shall show how this duality can be used. The ideas in this section apply to any sound and complete formulation of first-order logic. We shall present them in terms of $\mathcal{F}$, but they apply equally well to a system in which both $\forall$ and $\exists$, and all binary propositional connectives, are primitive symbols.

When $\subset$, $\not\supset$, $\not\subset$, $\not\equiv$, $\downarrow$, and $\mid$ occur in abbreviations of wffs of $\mathcal{F}$, they are to be regarded as having been introduced by the following definitions, which we hereby add to the definitions of $\wedge$, $\supset$, and $\equiv$ in §10, and of $\exists$ in §20:

$$\begin{array}{lll}
[\mathbf{A} \subset \mathbf{B}] & \text{stands for} & [\mathbf{A} \vee \sim \mathbf{B}]. \\
[\mathbf{A} \not\supset \mathbf{B}] & \text{stands for} & \sim [\sim \mathbf{A} \vee \mathbf{B}]. \\
[\mathbf{A} \not\subset \mathbf{B}] & \text{stands for} & \sim [\mathbf{A} \vee \sim \mathbf{B}]. \\
[\mathbf{A} \not\equiv \mathbf{B}] & \text{stands for} & \sim [\mathbf{A} \equiv \mathbf{B}]. \\
[\mathbf{A} \downarrow \mathbf{B}] & \text{stands for} & \sim [\mathbf{A} \vee \mathbf{B}]. \\
[\mathbf{A} \mid \mathbf{B}] & \text{stands for} & \sim [\mathbf{A} \wedge \mathbf{B}].
\end{array}$$

Definition. Let $\mathbf{A}$ be a wff of $\mathcal{F}$. A wff $\mathbf{B}$ is a *dual* of $\mathbf{A}$ iff there is an abbreviation $\mathbf{C}$ for $\mathbf{A}$ such that $\mathbf{B}$ is the result of interchanging $\forall$ with $\exists$, $\vee$ with $\wedge$, $\supset$ with $\not\subset$, $\subset$ with $\not\supset$, $\equiv$ with $\not\equiv$, and $\downarrow$ with $\mid$ everywhere in $\mathbf{C}$ (See Figure 4.4.) The *principal dual* $\mathbf{A}^d$ of $\mathbf{A}$ is the result of making these interchanges in the unabbreviated form of $\mathbf{A}$.

$\forall$	$\exists$
$\wedge$	$\vee$
$\supset$	$\not\subset$
$\subset$	$\not\supset$
$\equiv$	$\not\equiv$
$\downarrow$	$\mid$

Figure 4.4: Dual Pairs

4000 Proposition. If $\mathbf{B}$ and $\mathbf{D}$ are duals of $\mathbf{A}$, then $\vdash \mathbf{B} \equiv \mathbf{D}$.

Proof: We show that if $\mathbf{B}$ is any dual of $\mathbf{A}$, then $\vdash \mathbf{B} \equiv \mathbf{A}^d$. Clearly this will suffice to prove the proposition. We show, by induction on the construction

of $\mathbf{D}$, that if $\mathbf{D}$ is any abbreviation for a wff $\mathbf{D}_0$ of $\mathcal{F}$, and $\mathbf{D}'$ is the result of interchanging $\forall$ with $\exists$, $\vee$ with $\wedge$, $\supset$ with $\not\subset$, etc., in $\mathbf{D}$, then $\vdash \mathbf{D}' \equiv \mathbf{D}_0^d$.

Of course, this is trivially true if $\mathbf{D}$ is atomic. Now consider the following cases:

(a) $\mathbf{D}$ is $\forall \mathbf{x}\mathbf{M}$. Then $\mathbf{D}' = \exists \mathbf{x}\mathbf{M}'$, but $\vdash \mathbf{M}' \equiv \mathbf{M}_0^d$ by inductive hypothesis, so $\vdash \mathbf{D}' \equiv \exists \mathbf{x}\mathbf{M}_0^d$. Also $\mathbf{D}_0^d = (\forall \mathbf{x}\mathbf{M}_0)^d = \exists \mathbf{x}\mathbf{M}_0^d$, so $\vdash \mathbf{D}' \equiv \mathbf{D}_0^d$.

(b) $\mathbf{D}$ is $\exists \mathbf{x}\mathbf{M}$. Then $\mathbf{D}' = \forall \mathbf{x}\mathbf{M}'$, but $\vdash \mathbf{M}' \equiv \mathbf{M}_0^d$ by inductive hypothesis, so $\vdash \mathbf{D}' \equiv \forall \mathbf{x}\mathbf{M}_0^d$. Also $\mathbf{D}_0^d = (\sim \forall \mathbf{x} \sim \mathbf{M}_0)^d = \sim \exists \mathbf{x} \sim \mathbf{M}_0^d$, so $\vdash \mathbf{D}' \equiv \mathbf{D}_0^d$.

(c) $\mathbf{D}$ is $[\mathbf{M} \vee \mathbf{N}]$. Then $\mathbf{D}' = [\mathbf{M}' \wedge \mathbf{N}']$, but $\vdash \mathbf{M}' \equiv \mathbf{M}_0^d$ and $\vdash \mathbf{N}' \equiv \mathbf{N}_0^d$ by inductive hypothesis, so $\vdash \mathbf{D}' \equiv [\mathbf{M}_0^d \wedge \mathbf{N}_0^d]$. Also $\mathbf{D}_0^d = [\mathbf{M}_0 \vee \mathbf{N}_0]^d = [\mathbf{M}_0^d \wedge \mathbf{N}_0^d]$, so $\vdash \mathbf{D}' \equiv \mathbf{D}_0^d$.

(d) $\mathbf{D}$ is $[\mathbf{M} \wedge \mathbf{N}]$. Then $\mathbf{D}' = [\mathbf{M}' \vee \mathbf{N}']$, but $\vdash \mathbf{M}' \equiv \mathbf{M}_0^d$ and $\vdash \mathbf{N}' \equiv \mathbf{N}_0^d$ by inductive hypothesis, so $\vdash \mathbf{D}' \equiv [\mathbf{M}_0^d \vee \mathbf{N}_0^d]$. Also $\mathbf{D}_0^d = (\sim [\sim \mathbf{M}_0 \vee \sim \mathbf{N}_0])^d = \sim [\sim \mathbf{M}_0^d \wedge \sim \mathbf{N}_0^d] = \sim\sim [\sim\sim \mathbf{M}_0^d \vee \sim\sim \mathbf{N}_0^d]$, so $\vdash \mathbf{D}' \equiv \mathbf{D}_0^d$.

Since the pattern of proof is now clear, we leave it to the reader to check the remaining cases (Exercise X4000). Note that there is some purpose to this exercise, since it should be verified that the duals of the remaining connectives are defined correctly. ■

4001 Corollary. For any wff $\mathbf{A}$ of $\mathcal{F}$, $\vdash \mathbf{A}^{dd} \equiv \mathbf{A}$.

Proof: It is easy to see that $\mathbf{A}$ is a dual of $\mathbf{A}^d$, so $\vdash \mathbf{A}^{dd} \equiv \mathbf{A}$ by 4000. ■

REMARK. Before looking at the next theorem, the reader should test his understanding of the basic idea of duality by trying to figure out what the theorem should say. The theorem has the form $\mathcal{V}_\varphi^{\mathcal{M}} \mathbf{A}^d = ?$, and essentially says that one can compute $\mathcal{V}_\varphi^{\mathcal{M}} \mathbf{A}^d$ if one knows $\mathcal{V}_\varphi^{\mathcal{N}} \mathbf{A}$ for an appropriate interpretation $\mathcal{N}$.

4002 Theorem. Let $\mathcal{M} = \langle \mathcal{D}, \mathcal{J} \rangle$ be any interpretation. Define $\mathcal{J}'$ to agree with $\mathcal{J}$ on all individual and function constants, and for all n-ary predicate constants $\mathbf{P}$, let $(\mathcal{J}'\mathbf{P})d_1 \ldots d_n = \sim (\mathcal{J}\mathbf{P})d_1 \ldots d_n$ for all $d_1, \ldots, d_n \in \mathcal{D}$. Let $\mathcal{M}' = \langle \mathcal{D}, \mathcal{J}' \rangle$. Then for any wff $\mathbf{A}$ and assignment φ, $\mathcal{V}_\varphi^{\mathcal{M}} \mathbf{A} = \sim \mathcal{V}_\varphi^{\mathcal{M}'} \mathbf{A}$.

NOTES:

(1) In accord with the convention introduced in §25, for simplicity we are

Chapter 5

Type Theory

§50. Introduction

So far we have been concerned with first-order logic, and its subsystem propositional calculus, which we might regard as zeroth-order logic. We now wish to discuss higher-order logics.

It seems very natural to extend the system $\mathcal{F}$ of first-order logic by permitting quantification on predicate, propositional, and function variables as well as individual variables. One thus obtains the wffs of a system of *second-order logic*. One might then introduce predicate and function variables of higher type to denote relations and functions whose arguments may be relations and functions of individuals as well as individuals. Thus one would be led to a system of *third-order logic*, and if one permitted quantification with respect to these new variables, one would obtain a system of *fourth-order logic*. This process can be continued indefinitely to obtain logics of arbitrarily high order. Of course, after a while one runs out of different types of letters to use for different types of variables, but this problem can be solved by introducing *type symbols* to indicate the types of variables, and using a letter with type symbol α as subscript for a variable of type α.

To make these ideas precise, we shall briefly describe a system $\mathcal{F}^\omega$ of ω-order logic which has all finite order logics as subsystems. We shall not discuss $\mathcal{F}^\omega$ very extensively, since we prefer to devote most of our attention to an improved formulation of higher-order logic called $\mathcal{Q}_0$ which we shall soon introduce. Nevertheless, $\mathcal{F}^\omega$ provides a convenient starting point for a discussion of higher-order logics. For the sake of simplicity, $\mathcal{F}^\omega$ will have no function symbols. It is well known that assertions about functions can be expressed in terms of relations, since to every n-ary function f there

corresponds an $(n+1)$-ary relation R such that for all $x_1, \ldots, x_n$, and y, $Rx_1 \ldots x_n y$ iff $fx_1 \ldots x_n = y$.

A system of logic which includes logics of all finite orders (but no infinite orders) is known as *ω-order logic*, or *finite type theory*. (There are also systems of transfinite type theory, such as that in [Andrews, 1965], but we shall not discuss them here.) Type theory was invented by Bertrand Russell [Russell, 1908]. In [Whitehead and Russell, 1913], which had extraordinary influence on the development of logic in the early twentieth century, he and Alfred North Whitehead demonstrated that various fundamental parts of mathematics could indeed be formalized in this system. Russell was concerned with providing secure foundations for mathematics, and wished to enforce the *vicious-circle principle* that no totality can contain members defined in terms of itself. He therefore originally advocated using what is now called *ramified* type theory,[1] which is still important in various proof-theoretic studies (such as [Feferman, 1964]) of the consistency of restricted portions of mathematics. However, ramified type theory did not prove adequate for formalizing mathematics in general, and we shall devote our attention to a simplified version of it which is known as *simple* type theory.

Additional information about type theory may be obtained from sources such as [Andrews, 2001], [van Benthem and Doets, 1983], [Hatcher, 1968], [Hindley, 1997], [Jacobs, 1999], [Leivant, 1994], [Mendelson, 1987], [Quine, 1963], [Shapiro, 1991], [Wolfram, 1993], and references cited therein.

The *type symbols* of $\mathcal{F}^\omega$ and their *orders* are defined inductively as follows:

(a) ι is a type symbol (denoting the type of individuals) of order 0.
(b) o is a type symbol (denoting the type of truth values) of order 1.
(c) If $\tau_1, \ldots, \tau_n$ are type symbols (with $n \geq 1$), then $(\tau_1 \ldots \tau_n)$ is a type symbol (denoting the type of n-ary relations with arguments of types $\tau_1, \ldots, \tau_n$, respectively), and its order is 1+ the maximum of the orders of $\tau_1, \ldots, \tau_n$. (One may think of "o" as "( )".)

The primitive symbols of $\mathcal{F}^\omega$ are the improper symbols [,], $\sim$, $\vee$, and $\forall$, a denumerable list of variables of each type, and (optionally) constants of certain types. Variables of type o are called *propositional variables*, and variables of type ι are called *individual variables*.

The formation rules of $\mathcal{F}^\omega$ are the following:

[1] See [Hazen, 1983] or [Church, 1956, p. 347] for more details.

(a) Every propositional variable or constant is a wff.
(b) If $\mathbf{u}_{\tau_1 \ldots \tau_n}, \mathbf{v}^1_{\tau_1} \ldots, \mathbf{v}^n_{\tau_n}$ are variables or constants of the types indicated by their subscripts, then $\mathbf{u}_{\tau_1 \ldots \tau_n} \mathbf{v}^1_{\tau_1} \ldots \mathbf{v}^n_{\tau_n}$ is a wff.
(c) If $\mathbf{A}$ and $\mathbf{B}$ are wffs and $\mathbf{u}$ is a variable of any type, then $\sim \mathbf{A}$, $[\mathbf{A} \vee \mathbf{B}]$, and $\forall \mathbf{u} \mathbf{A}$ are wffs.

The *order* of a variable or constant is the order of its type symbol. For any positive integer m, the wffs of $2m$th-order logic are the wffs in which no variable or constant of order greater than m occurs, and the wffs of $(2m-1)$th-order logic are the wffs of $2m$th-order logic in which no variable of order m is quantified. (Note that first-order logic, as here defined, is just the result of dropping function symbols from the system $\mathcal{F}$ of §20.)

The definitions of connectives and quantifiers in §10 and §20 can be extended to apply to $\mathcal{F}^\omega$. If $\mathbf{x}_\tau$ and $\mathbf{y}_\tau$ are variables or constants of the same type τ, $\mathbf{x}_\tau = \mathbf{y}_\tau$ is defined to be an abbreviation for

$$\forall z_{(\tau)} [z_{(\tau)} \mathbf{x}_\tau \supset z_{(\tau)} \mathbf{y}_\tau] \tag{*1}$$

where $z_{(\tau)}$ is a variable of type (τ). An alternative possibility is to define $\mathbf{x}_\tau = \mathbf{y}_\tau$ as

$$\forall p_{(\tau\tau)} [\forall z_\tau p_{(\tau\tau)} z_\tau z_\tau \supset p_{(\tau\tau)} \mathbf{x}_\tau \mathbf{y}_\tau]. \tag{*2}$$

(See Exercise X5007.) Note that the two basic properties of equality which we took as axioms in §26 are reflexivity and substitutivity. *1 defines equality in terms of substitutivity, and *2 defines equality in terms of reflexivity (indeed, as the intersection of all reflexive relations). Thus, all the properties of equality follow from either of the basic properties of substitutivity or reflexivity when these are expressed in a sufficiently powerful way using the expressiveness of higher-order logic.

The rules of inference of $\mathcal{F}^\omega$ are:

(1) **Modus Ponens**. From $\mathbf{A}$ and $\mathbf{A} \supset \mathbf{B}$ to infer $\mathbf{B}$.
(2) **Generalization**. From $\mathbf{A}$ to infer $\forall \mathbf{x} \mathbf{A}$, where $\mathbf{x}$ is a variable of any type.

The axiom schemata of $\mathcal{F}^\omega$ are:

(1) $\mathbf{A} \vee \mathbf{A} \supset \mathbf{A}$
(2) $\mathbf{A} \supset . \mathbf{B} \vee \mathbf{A}$
(3) $\mathbf{A} \supset \mathbf{B} \supset . \mathbf{C} \vee \mathbf{A} \supset \mathbf{B} \vee \mathbf{C}$

(4) $\forall \mathbf{x}_\tau \mathbf{A} \supset S^{\mathbf{x}_\tau}_{\mathbf{y}_\tau}\mathbf{A}$, where $\mathbf{y}_\tau$ is a variable or constant of the same type as the variable $\mathbf{x}_\tau$, and $\mathbf{y}_\tau$ is free for $\mathbf{x}_\tau$ in $\mathbf{A}$.

(5) $\forall \mathbf{x}[\mathbf{A} \vee \mathbf{B}] \supset \ \centerdot\, \mathbf{A} \vee \forall \mathbf{x}\mathbf{B}$, where $\mathbf{x}$ is any variable not free in $\mathbf{A}$.

(6) **(Comprehension Axioms)**

$\exists \mathbf{u}_\circ[\mathbf{u}_\circ \equiv \mathbf{A}]$, where $\mathbf{u}_\circ$ does not occur free in $\mathbf{A}$.

$\exists \mathbf{u}_{(\tau_1 \ldots \tau_n)} \forall \mathbf{v}^1_{\tau_1} \ldots \forall \mathbf{v}^n_{\tau_n} [\mathbf{u}_{(\tau_1 \ldots \tau_n)} \mathbf{v}^1_{\tau_1} \ldots \mathbf{v}^n_{\tau_n} \equiv \mathbf{A}]$, where $\mathbf{u}_{(\tau_1 \ldots \tau_n)}$ does not occur free in $\mathbf{A}$, and $\mathbf{v}^1_{\tau_1}, \ldots, \mathbf{v}^n_{\tau_n}$ are distinct variables.

(7) **(Axioms of Extensionality)**

$[x_\circ \equiv y_\circ] \supset \ \centerdot\, x_\circ = y_\circ$

$\forall w^1_{\tau_1} \ldots \forall w^n_{\tau_n} [x_{(\tau_1 \ldots \tau_n)} w^1_{\tau_1} \ldots w^n_{\tau_n} \equiv y_{(\tau_1 \ldots \tau_n)} w^1_{\tau_1} \ldots w^n_{\tau_n}] \supset \ \centerdot\, x_{(\tau_1 \ldots \tau_n)} = y_{(\tau_1 \ldots \tau_n)}$

The axioms of kth-order logic are those axioms of $\mathcal{F}^\omega$ which are wffs of kth-order logic. Note that Comprehension Axioms first appear in second-order logic, and Axioms of Extensionality first appear in fourth-order logic.

Note the principal features which are added to first-order logic in order to obtain this formulation of higher-order logic:

1. Variables of arbitrarily high orders.
2. Quantification on variables of all types.
3. Comprehension Axioms.
4. Axioms of Extensionality.

One may also wish to assume additional axioms, such as the Axiom Schema of Choice, an Axiom of Infinity, and perhaps even the Continuum Hypothesis, though there is room for argument whether these are axioms of pure logic or of mathematics.

It should be remarked that one can restrict the formation rules of $\mathcal{F}^\omega$ by requiring that $n = 1$ in the definitions above, and obtain a still simpler system of ω-order logic which is nevertheless adequate to express most mathematical ideas, since one can define ordered pairs (and, more generally, n-tuples) in various ways, and then represent an n-ary relation B by the monadic relation P such that $Bx^1 \ldots x^n$ iff $P\langle x_1, \ldots, x_n \rangle$. Thus, $P\langle x^1, \ldots, x^n \rangle$ would be an abbreviation for $\exists u[Pu \wedge u = \langle x^1, \ldots, x^n \rangle]$, where $u = \langle x^1, \ldots, x^n \rangle$ is defined in some appropriate way, and suitable types are assigned to u and P.

If one wishes to thus restrict the language so that all predicates are monadic, one may find it congenial to write $\mathbf{v}_\tau \in \mathbf{u}_{(\tau)}$ rather than $\mathbf{u}_{(\tau)}\mathbf{v}_\tau$,

and to say "$\mathbf{v}_\tau$ is in the set $\mathbf{u}_{(\tau)}$" rather than "$\mathbf{v}_\tau$ has the property $\mathbf{u}_{(\tau)}$". Of course, this is a mere notational change. A more radical change is obtained by also dropping the type symbols, so that all variables are of the same type. (Propositional variables are customarily dispensed with.) One thus obtains Naive Axiomatic Set Theory, which can be formalized in first-order logic. $\in$ is regarded as a binary predicate constant, and all atomic wffs are of the form $\mathbf{x} \in \mathbf{y}$, where $\mathbf{x}$ and $\mathbf{y}$ are (individual) variables. $\mathbf{x} \in \mathbf{y}$ is interpreted to mean "$\mathbf{x}$ is a member of the set $\mathbf{y}$", which is of course false if $\mathbf{y}$ is not a set at all. The Comprehension Axiom Schema now takes the simple form $\exists \mathbf{u} \forall \mathbf{v}[\mathbf{v} \in \mathbf{u} \equiv \mathbf{A}]$, where $\mathbf{u}$ is not free in the wff $\mathbf{A}$. This axiom schema asserts that for any wff $\mathbf{A}$ in which $\mathbf{v}$ occurs free, there is a set $\mathbf{u}$ consisting of all those elements $\mathbf{v}$ of which $\mathbf{A}$ is true. Unfortunately, the axiom schema guarantees the existence of too many sets. In particular, $\exists u \forall v[v \in u \equiv \sim v \in v]$, i.e., there is a set whose members are all those elements which are not members of themselves. Choosing such a set u, we conclude that $u \in u \equiv \sim u \in u$, which is a contradiction, so Naive Axiomatic Set Theory is inconsistent! The paradox above was discovered by Bertrand Russell, and is known as Russell's Paradox.

One can restrict the Comprehension Axiom Schema, and take only certain carefully chosen instances of it as axioms, to obtain apparently consistent formulations of Axiomatic Set Theory. Alternatively, one can keep type symbols, so that Russell's paradox (and various other paradoxes) cannot be expressed at all within the formal system. Each approach leads to systems which are adequate for formalizing mathematics, and each approach has advantages for certain purposes. At present, Axiomatic Set Theory is more popular with mathematicians simply because it is much better known. However, most mathematicians do make mental distinctions between different types of mathematical objects, and use different types of letters to denote them, so it might be claimed that type theory provides a more natural formalization of mathematics as it is actually practiced. In addition, explicit syntactic distinctions between expressions denoting intuitively different types of mathematical entities are very useful in computerized systems for exploring and applying mathematics (on a theoretical rather than purely computational level). Of course, type symbols need not be explicitly written as long as one knows the types of the variables and constants.

Therefore we shall turn our attention to finding a formulation of type theory which is as expressive as possible, allowing mathematical ideas to be expressed precisely with a minimum of circumlocutions, and which is as simple and economical as is possible without sacrificing expressiveness. The reader will observe that the formal language we find arises very naturally