Symbolic Logic and Mechanical Theorem Proving

CHIN-LIANG CHANG

RICHARD CHAR-TUNG LEE

National Institutes of Health
Bethesda, Maryland

ACADEMIC PRESS
New York San Francisco London
A Subsidiary of Harcourt Brace Jovanovich, Publishers

ACADEMIC PRESS, INC.
111 Fifth Avenue, New York, New York 10003

United Kingdom Edition published by
ACADEMIC PRESS, INC. (LONDON) LTD.
24/28 Oval Road, London NW1

LIBRARY OF CONGRESS CATALOG CARD NUMBER: 72-88358

AMS (MOS) 1970 Subject Classifications: 02B10, 02B05; 02-01, 02-04; 02G99; 68A40, 68A45

PRINTED IN THE UNITED STATES OF AMERICA

80 81 82 9 8 7 6 5 4

TO OUR WIVES

Preface

Artificial intelligence is a relatively new field of recent attraction to many researchers concerned with programming computers to perform tasks that require "intelligence." Mechanical theorem proving is an important subject in artificial intelligence.

It has long been man's ambition to find a general decision procedure to prove theorems. This desire clearly dates back to Leibniz (1646–1716); it was revived by Peano around the turn of the century and by Hilbert's school in the 1920s. A very important theorem was proved by Herbrand in 1930; he proposed a mechanical method to prove theorems. Unfortunately, his method was very difficult to apply since it was extremely time consuming to carry out by hand. With the invention of digital computers, logicians regained interest in mechanical theorem proving. In 1960, Herbrand's procedure was implemented by Gilmore on a digital computer, followed shortly by a more efficient procedure proposed by Davis and Putnam.

A major breakthrough in mechanical theorem proving was made by J. A. Robinson in 1965; he developed a single inference rule, the resolution principle, which was shown to be highly efficient and very easily implemented on computers. Since then, many improvements of the resolution principle have been made. Mechanical theorem proving has been applied to many areas, such as program analysis, program synthesis, deductive question-answering systems, problem-solving systems, and robot technology. The number of applications keeps growing.

There has also been an increasing awareness of the importance of symbolic logic itself for researchers in computer science. Some background in the first-order logic is now a necessary tool for reading papers in many journals related to computer science. However, most of the available books in

symbolic logic are not computer-science oriented; they are almost all for mathematicians and philosophers.

Our goal was to write a book containing an introduction to symbolic logic and a thorough discussion of mechanical theorem proving and its applications. The book consists of three major parts. Chapters 2 and 3 constitute an introduction to symbolic logic, Chapters 4–9 introduce several techniques in mechanical theorem proving, and Chapters 10 and 11 show how theorem proving can be applied to various areas such as question answering, problem solving, program analysis, and program synthesis.

The book can serve several purposes. It can be used for a senior or graduate course on theorem proving. If used for a senior course, the advanced chapters, 6–11, may be neglected. For such a course, the reader needs no background in symbolic logic, only a basic knowledge of elementary set theory. On the other hand, if used as a graduate course in which a background in logic may be assumed, then Chapters 2 and 3 can be skipped. Since we adopted a purely model-theoretic approach to the first-order logic, this book is quite different from other logic books, which have a long tradition of presenting the first-order logic syntactically (axiomatically). Furthermore, this book emphasizes efficient computer implementations of proof techniques which, usually, have been of no concern to logicians. In some universities, mechanical theorem proving is taught as a part of artificial intelligence, not as a separate course. In such a case, this book can be used as a supplementary textbook in artificial intelligence to provide the reader with background in both mechanical theorem proving and application areas.

Since mechanical theorem proving is such a fast-growing field, it is inevitable that some significant results in this field had to be omitted from the book. However, we have tried to include in the Bibliography every published paper related to this field. The Bibliography is divided into three parts. The first part is a collection of material concerning artificial intelligence in general. The second part covers symbolic logic and mechanical theorem proving. The third part consists of material related to the applications of theorem-proving techniques.

Acknowledgments

It is impossible to name all the people who have helped the authors organize the material and clarify our views. Miss D. Koniver and Dr. L. Hodes of the National Institutes of Health, Professor J. Minker of the University of Maryland, Professor Z. Manna of Stanford University, Dr. S. K. Chang of IBM, Yorktown Heights, Dr. R. Waldinger of the Stanford Research Institute, Professor R. Anderson of the University of Houston, Mr. J. Yochelson, Mr. B. Crissey, and Mr. J. Hu of the Johns Hopkins University have all read a part or all of our manuscript. Although not responsible for any error of our own, they have contributed tremendously to the improvement of this book. We also would like to thank Dr. J. Slagle, who taught both of us artificial intelligence at the University of California, Berkeley. It was he who provided us with an intellectual and pleasant working atmosphere which made this work possible. Finally, our deepest thanks go to our loving wives for their patient and understanding support during the writing of this book.

Chapter 1

Introduction

1.1 ARTIFICIAL INTELLIGENCE, SYMBOLIC LOGIC, AND THEOREM PROVING

Ever since the birth of the first modern computer, computer technology has developed at a fantastic speed. Today we see computers being used not only to solve computationally difficult problems, such as carrying out a fast Fourier transform or inverting a matrix of high dimensionality, but also being asked to perform tasks that would be called intelligent if done by human beings. Some such tasks are writing programs, answering questions, and proving theorems. Artificial intelligence is a field in computer science that is concerned with performing such tasks [Ernst and Newell, 1969; Feigenbaum and Feldman, 1963; Nilsson, 1971; Slagle, 1971].

The second half of the 1960s has been phenomenal in the field of artificial intelligence for the increase of interest in mechanical theorem proving. The widespread intensive interest in mechanical theorem proving is caused not only by the growing awareness that the ability to make logical deductions is an integral part of human intelligence, but is perhaps more a result of the status of mechanical theorem-proving techniques in the late 1960s. The foundation of mechanical theorem proving was developed by Herbrand in 1930. His method was impractical to apply until the invention of the digital computer. It was not until the landmark paper by J. A. Robinson in 1965,

together with the development of the resolution principle, that major steps were taken to achieve realistic computer-implemented theorem provers. Since 1965, many refinements of the resolution principle have been made.

Parallel to the progress in improving mechanical theorem-proving techniques was the progress in applying theorem-proving techniques to various problems in artificial intelligence. These were first applied to deductive question answering and later to problem solving, program synthesis, program analysis, and many others.

There are many points of view from which we can study symbolic logic. Traditionally, it has been studied from philosophical and mathematical orientations. In this book, we are interested in the *applications* of symbolic logic to solving intellectually difficult problems. That is, we want to use symbolic logic to represent problems and to obtain their solutions.

We shall now present some very simple examples to demonstrate how symbolic logic can be used to represent problems. Since we have not yet formally discussed symbolic logic, the reader should rely on his intuition to follow the text at this moment.

Let us consider a simple example. Assume that we have the following facts:

F_1: If it is hot and humid, then it will rain.
F_2: If it is humid, then it is hot.
F_3: It is humid now.

The question is: Will it rain?

The above facts are given in English. We shall use symbols to represent them. Let P, Q, and R represent "It is hot," "It is humid," and "It will rain," respectively. We also need some logical symbols. In this case, we shall use $\wedge$ to represent "and" and $\rightarrow$ to represent "imply." Then the above three facts are represented as:

F_1: $P \wedge Q \rightarrow R$

F_2: $Q \rightarrow P$

F_3: Q.

Thus we have translated English sentences into logical formulas. The reader will see later that whenever F_1, F_2, and F_3 are true, the formula

F_4: R

is true. Therefore, we say that F_4 *logically follows from* F_1, F_2, and F_3. That is, it will rain.

Let us consider another example. We have the following facts:

F_1: Confucius is a man.
F_2: Every man is mortal.

To represent F_1 and F_2, we need a new concept, called a predicate. We may let $P(x)$ and $Q(x)$ represent "x is a man" and "x is mortal," respectively. We also use $(\forall x)$ to represent "for all x." Then the above facts are represented by

F_1: $P(\text{Confucius})$
F_2: $(\forall x)(P(x) \rightarrow Q(x))$.

Again, the reader will see later that from F_1 and F_2, we can *logically deduce*

F_3: $Q(\text{Confucius})$

which means that Confucius is mortal.

In the above two examples, we essentially had to prove that a formula *logically follows from* other formulas. We shall call a statement that a formula logically follows from formulas a *theorem*. A demonstration that a theorem is true, that is that a formula logically follows from other formulas, will be called a *proof* of the theorem. The *problem of mechanical theorem proving* is to consider mechanical methods for finding proofs of theorems.

There are many problems that can be conveniently transformed into theorem-proving problems; we shall list some of them.

1. In a *question-answering system*, facts can be represented by logical formulas. Then, to answer a question from the facts, we prove that a formula corresponding to the answer is derivable from the formulas representing the facts.

2. In the *program-analysis problem*, we can describe the execution of a program by a formula A, and the condition that the program will terminate by another formula B. Then, verifying that the program will terminate is equivalent to proving that formula B follows from formula A.

3. In the *graph-isomorphism problem*, we want to know whether a graph is isomorphic to a subgraph of another graph. This problem is not merely an interesting problem in mathematics; it is also a practical problem. For example, the structure of an organic compound can be described by a graph. Therefore, testing whether a substructure of the structure of an organic compound is the structure of another organic compound is a graph-isomorphism problem. For this problem, we can describe graphs by formulas. Thus, the problem may be stated as proving that the formula representing a graph follows from the formula representing another graph.

4. In the *state-transformation problem*, there are a collection of states and a collection of (state) operators. When one operator is applied to a state, a new state will be obtained. Starting with an initial state, one tries to find a sequence of operators that will transform the initial state into a desired state. In this case, we can describe states and the state transition rules by logical formulas. Thus, transforming the initial state into the desired state may be

regarded as verifying that the formula representing the desired state follows from the formula representing both the states and the state transition rules.

Since many problems can be formulated as theorem-proving problems, theorem proving is a very important field in artificial intelligence science. Thanks to the relentless efforts of many researchers, great progress has been made in using computers to prove theorems. In this book, both the theory and applications of mechanical theorem proving will be considered.

1.2 MATHEMATICAL BACKGROUND

In this section, we provide some basic mathematical concepts that will be used in this book. All of these concepts can be easily found in elementary mathematical textbooks [Lightstone, 1964; Pfeiffer, 1964; Stoll, 1961]. The following basic definitions of set theory are used.

A *set* is a collection of elements (members). A set that contains no elements is called the *empty set* $\varnothing$. Let A and B be two sets. $x \in A$ is used to denote that x is a member of A, or that x belongs to A.

The set A is *identical* to the set B, denoted $A = B$, if and only if both A and B have the same elements.

The set A is a *subset* of the set B, denoted $A \subseteq B$, if and only if each element of A is an element of B.

The set A is a *proper subset* of the set B, denoted $A \subset B$, if and only if $A \subseteq B$ and $A \neq B$.

The *union* of two sets A and B, denoted by $A \cup B$, is the set consisting of all elements that belong to A or to B.

The *intersection* of two sets A and B, denoted by $A \cap B$, is the set consisting of all elements that belong both to A and to B.

The *difference* of two sets A and B, denoted by $A - B$, is the set consisting of all elements that belong to A but do not belong to B.

We now define relations and functions.

An *ordered pair* of elements is denoted by (x, y), where x is called the first *coordinate* and y is called the second *coordinate*.

A *relation* is a set of ordered pairs. For example, the equality relation is a set of ordered pairs, each of which has the first coordinate equal to its second coordinate. The *domain* of a relation R is the set of all first coordinates of elements of R, and its *range* is the set of all second coordinates.

A *function* is a relation such that no two distinct elements have the same first coordinate. If f is a function and x is an element of its domain, then $f(x)$ denotes the second coordinate of the unique element of f whose first coordinate is x. $f(x)$ is called the *value* of f at x, and we say that f assigns the value $f(x)$ to x.

Throughout this book, we use many conventional symbols. For example,

$>$ means "greater than"; $\geqslant$, "greater than or equal to"; $<$, "less than"; $\leqslant$, "less than or equal to"; $=$, "equal to"; $\neq$, "not equal to"; $\triangleq$, "defined as"; and so on. The equality sign "$=$" will be used for many purposes. It may be used to mean "is defined by," "is identical to," "is equivalent to," or "is equal to." This will not cause the reader any confusion since he can always determine its exact meaning from the text.

REFERENCES

Ernst, G. W., and A. Newell (1969): "GPS: a Case Study in Generality and Problem Solving," Academic Press, New York.

Feigenbaum, E., and J. Feldman, eds. (1963): "Computers and Thought," McGraw-Hill, New York.

Lightstone, A. H. (1964): "The Axiomatic Method, an Introduction to Mathematical Logic," Prentice Hall, Englewood Cliffs, New Jersey.

Nilsson, N. J. (1971): "Problem Solving Methods in Artificial Intelligence," McGraw-Hill, New York.

Pfeiffer, P. E. (1964): "Sets, Events and Switching," McGraw-Hill, New York.

Slagle, J. R. (1971): "Artificial Intelligence, the Heuristic Programming Approach," McGraw-Hill, New York.

Stoll, R. (1961): "Sets, Logic and Axiomatic Theories," Freeman, San Francisco.

Stoll, R. (1963): "Set Theory and Logic," Freeman, San Francisco.

Chapter 2

The Propositional Logic

2.1 INTRODUCTION

Symbolic logic considers languages whose essential purpose is to symbolize reasoning encountered not only in mathematics but also in daily life. In this chapter, we shall first study the simplest symbolic logic—the propositional logic (or the propositional calculus). In the next chapter, we shall consider a more general one—the first-order logic (or the first-order predicate calculus).

In the propositional logic, we are interested in declarative sentences that can be either *true* or *false*, but not both. Any such declarative sentence is called a proposition. More formally, a *proposition* is a declarative sentence that is either true or false, but not both. Examples of propositions are: "Snow is white," "Sugar is a hydrocarbon," "Smith has a Ph.D. degree." The "true" or "false" assigned to a proposition is called the *truth value* of the proposition. Customarily, we represent "true" by ***T*** and "false" by ***F***. Furthermore, for convenience, we shall use an uppercase symbol or a string of uppercase symbols to denote a proposition. For instance, we may denote the propositions above as follows:

$P \triangleq$ Snow is white,
$Q \triangleq$ Sugar is a hydrocarbon,
$R \triangleq$ Smith has a Ph.D. degree.

The symbols, such as P, Q, and R, that are used to denote propositions are called *atomic formulus*, or *atoms*.

From propositions, we can build *compound* propositions by using *logical connectives*. Examples of compound propositions are: "Snow is white *and* the sky is clear" and "*If* John is not at home, *then* Mary is at home." The logical connectives in the above two compund propositions are "and" and "if $\cdots$ then." In the propositional logic, we shall use five logical connectives: $\sim$ (*not*), $\wedge$ (*and*), $\vee$ (*or*), $\rightarrow$ (*if* $\cdots$ *then*), and $\leftrightarrow$ (*if and only if*). These five logical connectives can be used to build compound propositions from propositions. More generally, they can be used to construct more complicated compound propositions from compound propositions by applying them repeatedly. For example, let us represent "Humidity is high" by P, "Temperature is high" by Q, and "One feels comfortable" by C. Then the sentence "If the humidity is high and the temperature is high, then one does not feel comfortable" may be represented by $((P \wedge Q) \rightarrow (\sim C))$. Therefore, we see that a compound proposition can express a rather complicated idea. In the propositional logic, an expression that represents a proposition, such as P, or a compound proposition, such as $((P \wedge Q) \rightarrow (\sim C))$, is called a well-formed formula.

Definition *Well-formed formulas*, or *formulas* for short, in the propositional logic are defined recursively as follows:

1. An atom is a formula.
2. If G is a formula, then $(\sim G)$ is a formula.
3. If G and H are formulas, then $(G \wedge H)$, $(G \vee H)$, $(G \rightarrow H)$, and $(G \leftrightarrow H)$ are formulas.
4. All formulas are generated by applying the above rules.

It is not difficult to see that expressions such as $(P \rightarrow)$ and $(P \vee)$ are not formulas. Throughout the book, when no confusion is possible, some pairs of parentheses may be dropped. For example, $P \vee Q$ and $P \rightarrow Q$ are the formulas $(P \vee Q)$ and $(P \rightarrow Q)$, respectively. We can further omit the use of parentheses by assigning decreasing ranks to the propositional connectives as follows:

$$\leftrightarrow, \quad \rightarrow, \quad \wedge, \quad \vee, \quad \sim.$$

and requiring that the connective with greater rank always reaches further. Thus $P \rightarrow Q \wedge R$ will mean $(P \rightarrow (Q \wedge R))$, and $P \rightarrow Q \wedge \sim R \vee S$ will mean $(P \rightarrow (Q \wedge ((\sim R) \vee S)))$.

Let G and H be two formulas. Then the truth values of the formulas $(\sim G)$, $(G \wedge H)$, $(G \vee H)$, $(G \rightarrow H)$, and $(G \leftrightarrow H)$ are related to the truth values of G and H in the following way:

1. $\sim G$ is true when G is false, and is false when G is true. $\sim G$ is called the *negation* of G.

2. $(G \wedge H)$ is true if G and H are both true; otherwise, $(G \wedge H)$ is false. $(G \wedge H)$ is called the *conjunction* of G and H.

3. $(G \vee H)$ is true if at least one of G and H is true; otherwise, $(G \vee H)$ is false. $(G \vee H)$ is called the *disjunction* of G and H.

4. $(G \rightarrow H)$ is false if G is true and H is false; otherwise, $(G \rightarrow H)$ is true. $(G \rightarrow H)$ is read as "*If G, then H*," or "*G implies H*."

5. $(G \leftrightarrow H)$ is true whenever G and H have the same truth values; otherwise $(G \leftrightarrow H)$ is false. $(G \leftrightarrow H)$ is read as "G if and only if H."

The above relations can be represented conveniently as in Table 2.1. Based on this table, we shall describe ways to evaluate the truth values of a formula in terms of the truth values of atoms occurring in the formula.

TABLE 2.1

G	H	$\sim G$	$(G \wedge H)$	$(G \vee H)$	$(G \rightarrow H)$	$(G \leftrightarrow H)$
T	*T*	*F*	*T*	*T*	*T*	*T*
T	*F*	*F*	*F*	*T*	*F*	*F*
F	*T*	*T*	*F*	*T*	*T*	*F*
F	*F*	*T*	*F*	*F*	*T*	*T*

2.2 INTERPRETATIONS OF FORMULAS IN THE PROPOSITIONAL LOGIC

Suppose that P and Q are two atoms and that the truth values of P and Q are T and F, respectively. Then, according to the second row of Table 2.1 with P and Q substituted for G and H, respectively, we find that the truth values of $(\sim P)$, $(P \wedge Q)$, $(P \vee Q)$, $(P \rightarrow Q)$, and $(P \leftrightarrow Q)$ are F, F, T, F, and F, respectively. Similarly, the truth value of any formula can be evaluated in terms of truth values of atoms.

Example 2.1

Consider the formula

$$G \triangleq (P \wedge Q) \rightarrow (R \leftrightarrow (\sim S)).$$

The atoms in this formula are P, Q, R, and S. Suppose the truth values of P, Q, R, and S are T, F, T, and T, respectively. Then $(P \wedge Q)$ is F since Q is false; $(\sim S)$ is F since S is T; $(R \leftrightarrow (\sim S))$ is F since R is T and $(\sim S)$ is F; and $(P \wedge Q) \rightarrow (R \leftrightarrow (\sim S))$ is T since $(P \wedge Q)$ is F and $(R \leftrightarrow (\sim S))$ is F. Therefore, the formula G is T if P, Q, R, and S are assigned T, F, T, and T,

respectively. The assignment of the truth values $\{T, F, T, T\}$ to $\{P, Q, R, S\}$, respectively, will be called an *interpretation* of the formula G. Since each one of P, Q, R, and S can be assigned either T or F, there are $2^4 = 16$ interpretations of the formula G. In Table 2.2, we give the truth values of the formula G under all these 16 interpretations.

TABLE 2.2
TRUTH TABLE OF $(P \wedge Q) \rightarrow (R \leftrightarrow (\sim S))$

P	Q	R	S	$\sim S$	$(P \wedge Q)$	$(R \leftrightarrow (\sim S))$	$(P \wedge Q) \rightarrow (R \leftrightarrow (\sim S))$
T	T	T	T	F	T	F	F
T	T	T	F	T	T	T	T
T	T	F	T	F	T	T	T
T	T	F	F	T	T	F	F
T	F	T	T	F	F	F	T
T	F	T	F	T	F	T	T
T	F	F	T	F	F	T	T
T	F	F	F	T	F	F	T
F	T	T	T	F	F	F	T
F	T	T	F	T	F	T	T
F	T	F	T	F	F	T	T
F	T	F	F	T	F	F	T
F	F	T	T	F	F	F	T
F	F	T	F	T	F	T	T
F	F	F	T	F	F	T	T
F	F	F	F	T	F	F	T

A table, such as Table 2.2, that displays the truth values of a formula G for all possible assignments of truth values to atoms occurring in G is called a *truth table* of G.

We now give a formal definition of an interpretation of a propositional formula.

Definition Given a propositional formula G, let $A_1, A_2, \ldots, A_n$ be the atoms occurring in the formula G. Then an *interpretation* of G is an assignment of truth values to $A_1, \ldots, A_n$ in which every A_i is assigned either T or F, but not both.

Definition A formula G is said to be *true under (or in) an interpretation* if and only if G is evaluated to T in the interpretation; otherwise, G is said to be *false under the interpretation.*

If there are n distinct atoms in a formula, then there will be 2^n distinct interpretations for the formula. Sometimes, if $A_1, \ldots, A_n$ are all atoms

occurring in a formula, it may be more convenient to represent an interpretation by a set $\{m_1, \ldots, m_n\}$, where m_i is either A_i or $\sim A_i$. For example, the set $\{P, \sim Q, \sim R, S\}$ represents an interpretation in which P, Q, R, and S are, respectively, assigned T, F, F, and T. That is, if an atom A is in a set that represents an interpretation, then A is assigned T; while if the negation of the atom A is in the set, then A is assigned F. This convention will be kept throughout the book.

2.3 VALIDITY AND INCONSISTENCY IN THE PROPOSITIONAL LOGIC

In this section, we shall consider formulas that are true under all their possible interpretations and formulas that are false under all their possible interpretations.

Example 2.2

Let us consider the formula

$$G \triangleq ((P \rightarrow Q) \wedge P) \rightarrow Q.$$

The atoms in this formula are P and Q. Hence, the formula G has $2^2 = 4$ interpretations. The truth values of G under all its four interpretations are given in Table 2.3. We see that the formula G is true under all its interpretations. This formula will be called a *valid formula* (or a *tautology*).

TABLE 2.3

TRUTH TABLE OF $((P \rightarrow Q) \wedge P) \rightarrow Q$

P	Q	$(P \rightarrow Q)$	$(P \rightarrow Q) \wedge P$	$((P \rightarrow Q) \wedge P) \rightarrow Q$
T	T	T	T	T
T	F	F	F	T
F	T	T	F	T
F	F	T	F	T

Example 2.3

Consider the formula

$$G \triangleq (P \rightarrow Q) \wedge (P \wedge \sim Q).$$

The truth table of G is given in Table 2.4. We see that G is false under all its interpretations. This formula will be called an *inconsistent formula* (or a *contradiction*).

TABLE 2.4

TRUTH TABLE OF $(P \rightarrow Q) \wedge (P \wedge \sim Q)$

P	Q	$\sim Q$	$(P \rightarrow Q)$	$(P \wedge \sim Q)$	$(P \rightarrow Q) \wedge (P \wedge \sim Q)$
T	T	F	T	F	F
T	F	T	F	T	F
F	T	F	T	F	F
F	F	T	T	F	F

We now give formal definitions for validity and inconsistency.

Definition A formula is said to be *valid* if and only if it is true under all its interpretations. A formula is said to be *invalid* if and only if it is not valid.

Definition A formula is said to be *inconsistent* (or *unsatisfiable*) if and only if it is false under all its interpretations. A formula is said to be *consistent* (or *satisfiable*) if and only if it is not inconsistent.

By the above definitions, the following observations are obvious:

1. A formula is valid if and only if its negation is inconsistent.
2. A formula is inconsistent if and only if its negation is valid.
3. A formula is invalid if and only if there is at least one interpretation under which the formula is false.
4. A formula is consistent if and only if there is at least one interpretation under which the formula is true.
5. If a formula is valid, then it is consistent, but not vice versa.
6. If a formula is inconsistent, then it is invalid, but not vice versa.

Example 2.4

By using the truth table techniques, the reader should be able to establish the following:

a. $(P \wedge \sim P)$ is inconsistent; therefore also invalid.
b. $(P \vee \sim P)$ is valid; therefore also consistent.
c. $(P \rightarrow \sim P)$ is invalid, yet it is consistent.

If a formula F is true under an interpretation I, then we say that I *satisfies* F, or F *is satisfied* by I. On the other hand, if a formula F is false under an interpretation I, then we say that I *falsifies* F or F *is falsified* by I. For example, the formula $(P \wedge (\sim Q))$ is satisfied by the interpretation $\{P, \sim Q\}$, but is falsified by the interpretation $\{P, Q\}$. When an interpretation I satisfies a formula F, I is also called a *model* of F.

It will be shown later that the proof of the validity or inconsistency of a

formula is a very important problem. In the propositional logic, since the number of interpretations of a formula is finite, one can always decide whether or not a formula in the propositional logic is valid (inconsistent) by exhaustively examining all of its possible interpretations.

2.4 NORMAL FORMS IN THE PROPOSITIONAL LOGIC

As will be clear later, it is often necessary to transform a formula from one form to another, especially to a "normal form." This is accomplished by replacing a formula in the given formula by a formula "equivalent" to it and repeating this process until the desired form is obtained. By "equivalent," we mean the following:

Definition Two formulas F and G are said to be *equivalent* (or F is *equivalent* to G), denoted $F = G$, if and only if the truth values of F and G are the same under every interpretation of F and G.

Example 2.5

We can verify that $(P \rightarrow Q)$ is equivalent to $(\sim P \vee Q)$ by examining the truth table, Table 2.5.

TABLE 2.5
TRUTH TABLE OF $(P \rightarrow Q)$ AND $(\sim P \vee Q)$

P	Q	$(P \rightarrow Q)$	$(\sim P \vee Q)$
T	T	T	T
T	F	F	F
F	T	T	T
F	F	T	T

We need an adequate supply of equivalent formulas in order to carry out the transformation of formulas. In the propositional logic, let ■ denote the formula that is always true and □ the formula that is always false. Then we have some useful pairs of equivalent formulas given in Table 2.6, where F, G, and H are all formulas. For simplicity, we shall call each of them a "law."

The laws in Table 2.6 can be verified by using truth tables. Laws (2.3a), (2.3b) are often called *commutative* laws; (2.4a), (2.4b) *associative* laws; (2.5a), (2.5b), *distributive* laws; and (2.10a), (2.10b), *De Morgan's* laws.

Because of the associative laws, the parentheses in $(F \vee G) \vee H$ or $F \vee (G \vee H)$ can be dropped. That is, we can write $F \vee G \vee H$ for $(F \vee G) \vee H$ and $F \vee (G \vee H)$. More generally, we can write $F_1 \vee F_2 \vee \cdots \vee F_n$ without

TABLE 2.6

(2.1)	$F \leftrightarrow G = (F \rightarrow G) \wedge (G \rightarrow F)$	
(2.2)	$F \rightarrow G = \sim F \vee G$	
(2.3)	(a) $F \vee G = G \vee F$;	(b) $F \wedge G = G \wedge F$
(2.4)	(a) $(F \vee G) \vee H = F \vee (G \vee H)$;	(b) $(F \wedge G) \wedge H = F \wedge (G \wedge H)$
(2.5)	(a) $F \vee (G \wedge H) = (F \vee G) \wedge (F \vee H)$;	(b) $F \wedge (G \vee H) = (F \wedge G) \vee (F \wedge H)$
(2.6)	(a) $F \vee \square = F$;	(b) $F \wedge \blacksquare = F$
(2.7)	(a) $F \vee \blacksquare = \blacksquare$;	(b) $F \wedge \square = \square$
(2.8)	(a) $F \vee \sim F = \blacksquare$;	(b) $F \wedge \sim F = \square$
(2.9)	$\sim(\sim F) = F$	
(2.10)	(a) $\sim(F \vee G) = \sim F \wedge \sim G$;	(b) $\sim(F \wedge G) = \sim F \vee \sim G$

any ambiguity, where $F_1, F_2, \ldots, F_n$ are formulas. $F_1 \vee F_2 \vee \cdots \vee F_n$ is true whenever at least one of the F_i, $1 \leqslant i \leqslant n$, is true; otherwise, it is false. $F_1 \vee F_2 \vee \cdots \vee F_n$ is called the *disjunction* of $F_1, \ldots, F_n$. Similarly, we can write $F_1 \wedge F_2 \wedge \cdots \wedge F_n$, which is true if all $F_1, F_2, \ldots, F_n$ are true, and which is false otherwise. $F_1 \wedge F_2 \wedge \cdots \wedge F_n$ is called the *conjunction* of $F_1, \ldots, F_n$.

Note that the order in which the F_i appear in a disjunction or a conjunction is immaterial.

For example,

$$F_1 \vee F_2 \vee F_3 = F_1 \vee F_3 \vee F_2 = F_2 \vee F_1 \vee F_3 = F_3 \vee F_2 \vee F_1$$
$$= F_2 \vee F_3 \vee F_1 = F_3 \vee F_1 \vee F_2.$$

We now define normal forms as follows.

Definition A *literal* is an atom or the negation of an atom.

Definition A formula F is said to be in a *conjunctive normal form* if and only if F has the form $F \triangleq F_1 \wedge \cdots \wedge F_n, n \geqslant 1$, where each of $F_1, \ldots, F_n$ is a disjunction of literals.

Example 2.6

Let P, Q, and R be atoms. Then $F \triangleq (P \vee \sim Q \vee R) \wedge (\sim P \vee Q)$ is a formula in a conjunctive normal form. For this formula, $F_1 = (P \vee \sim Q \vee R)$ and $F_2 = (\sim P \vee Q)$. Clearly, F_1 is a disjunction of the literals P, $\sim Q$, and R, and F_2 is a disjunction of the literals $\sim P$ and Q.

Definition A formula F is said to be in a *disjunctive normal form* if and only if F has the form of $F \triangleq F_1 \vee \cdots \vee F_n$, $n \geqslant 1$, where each of $F_1, \ldots, F_n$ is a conjunction of literals.

Example 2.7

Let P, Q, and R be atoms. Then $F \triangleq (\sim P \wedge Q) \vee (P \wedge \sim Q \wedge \sim R)$ is a formula in a disjunctive normal form. For this formula, $F_1 = (\sim P \wedge Q)$

and $F_2 = (P \wedge \sim Q \wedge \sim R)$. Clearly, F_1 is a conjunction of the literals $\sim P$ and Q, and F_2 is a conjunction of the literals $P, \sim Q$, and $\sim R$.

Any formula can be transformed into a normal form. This is accomplished easily by using the laws given in Table 2.6. The following is an *outline of a transformation procedure:*

Step 1 Use the laws

(2.1) $F \leftrightarrow G = (F \rightarrow G) \wedge (G \rightarrow F)$

(2.2) $F \rightarrow G = \sim F \vee G$

to eliminate the logical connectives $\leftrightarrow$ and $\rightarrow$.

Step 2 Repeatedly use the law

(2.9) $\sim(\sim F) = F$

and De Morgan's laws

(2.10a) $\sim(F \vee G) = \sim F \wedge \sim G$

(2.10b) $\sim(F \wedge G) = \sim F \vee \sim G$

to bring the negation signs immediately before atoms.

Step 3 Repeatedly use the distributive laws

(2.5a) $F \vee (G \wedge H) = (F \vee G) \wedge (F \vee H)$

(2.5b) $F \wedge (G \vee H) = (F \wedge G) \vee (F \wedge H)$

and the other laws to obtain a normal form.

Example 2.8

Obtain a disjunctive normal form for the formula $(P \vee \sim Q) \rightarrow R$.
Since

$$\begin{aligned} (P \vee \sim Q) \rightarrow R &= \sim(P \vee \sim Q) \vee R && \text{by (2.2)} \\ &= (\sim P \wedge \sim(\sim Q)) \vee R && \text{by (2.10a)} \\ &= (\sim P \wedge Q) \vee R && \text{by (2.9),} \end{aligned}$$

a disjunctive normal form of $(P \vee \sim Q) \rightarrow R$ is $(\sim P \wedge Q) \vee R$.

Example 2.9

Obtain a conjunctive normal form for the formula $(P \wedge (Q \rightarrow R)) \rightarrow S$.

$$
\begin{aligned}
&(P \wedge (Q \to R)) \to S \\
&\quad = (P \wedge (\sim Q \vee R)) \to S && \text{by (2.2)} \\
&\quad = \sim(P \wedge (\sim Q \vee R)) \vee S && \text{by (2.2)} \\
&\quad = (\sim P \vee \sim(\sim Q \vee R)) \vee S && \text{by (2.10b)} \\
&\quad = (\sim P \vee (\sim(\sim Q) \wedge \sim R)) \vee S && \text{by (2.10a)} \\
&\quad = (\sim P \vee (Q \wedge \sim R)) \vee S && \text{by (2.9)} \\
&\quad = ((\sim P \vee Q) \wedge (\sim P \vee \sim R)) \vee S && \text{by (2.5a)} \\
&\quad = S \vee ((\sim P \vee Q) \wedge (\sim P \vee \sim R)) && \text{by (2.3a)} \\
&\quad = (S \vee (\sim P \vee Q)) \wedge (S \vee (\sim P \vee \sim R)) && \text{by (2.5a)} \\
&\quad = (S \vee \sim P \vee Q) \wedge (S \vee \sim P \vee \sim R) && \text{by (2.4a).}
\end{aligned}
$$

A conjunctive normal form of $(P \wedge (Q \to R)) \to S$ is $(S \vee \sim P \vee Q) \wedge (S \vee \sim P \vee \sim R)$.

2.5 LOGICAL CONSEQUENCES

In mathematics as well as in daily life, we often have to decide whether one statement follows from some other statements. This leads to the concept of "logical consequence." Before giving a formal definition of logical consequence, let us first consider the following example.

Example 2.10

Suppose the stock prices go down if the prime interest rate goes up. Suppose also that most people are unhappy when stock prices go down. Assume that the prime interest rate does go up. Show that you can conclude that most people are unhappy.

To show the above conclusion, let us denote the statements as follows:

$P \triangleq$ Prime interest rate goes up,

$S \triangleq$ Stock prices go down,

$U \triangleq$ Most people are unhappy.

There are four statements in this example. They are

(1) If the prime interest rate goes up, stock prices go down.
(2) If stock prices go down, most people are unhappy.

(3) The prime interest rate goes up.
(4) Most people are unhappy.

These statements are first symbolized as

(1′) $P \to S$

(2′) $S \to U$

(3′) P

(4′) U.

We now show that (4′) is true whenever (1′) $\wedge$ (2′) $\wedge$ (3′) is true.

Let us first transform $((P \to S) \wedge (S \to U) \wedge P)$ (representing (1′) $\wedge$ (2′) $\wedge$ (3′)) into a normal form:

$$
\begin{aligned}
((P \to S) \wedge (S \to U) \wedge P) &= ((\sim P \vee S) \wedge (\sim S \vee U) \wedge P) && \text{by (2.2)} \\
&= (P \wedge (\sim P \vee S) \wedge (\sim S \vee U)) && \text{by (2.3b)} \\
&= (((P \wedge \sim P) \vee (P \wedge S)) \wedge (\sim S \vee U)) && \text{by (2.5b)} \\
&= ((\square \vee (P \wedge S)) \wedge (\sim S \vee U)) && \text{by (2.8b)} \\
&= (P \wedge S) \wedge (\sim S \vee U) && \text{by (2.6a)} \\
&= (P \wedge S \wedge \sim S) \vee (P \wedge S \wedge U) && \text{by (2.5b)} \\
&= (P \wedge \square) \vee (P \wedge S \wedge U) && \text{by (2.8b)} \\
&= \square \vee (P \wedge S \wedge U) && \text{by (2.7b)} \\
&= P \wedge S \wedge U && \text{by (2.6a).}
\end{aligned}
$$

Therefore, if $((P \to S) \wedge (S \to U) \wedge P)$ is true, then $(P \wedge S \wedge U)$ is true. Since $(P \wedge S \wedge U)$ is true only if P, S, and U are all true, we conclude that U is true.

Because U is true whenever $(P \to S)$, $(S \to U)$, and P are true, in logic, U is called a *logical consequence* of $(P \to S)$, $(S \to U)$, and P. More formally, we define a logical consequence as follows.

Definition Given formulas $F_1, F_2, \ldots, F_n$ and a formula G, G is said to be a *logical consequence* of $F_1, \ldots, F_n$ (or G *logically follows from* $F_1, \ldots, F_n$) if and only if for any interpretation I in which $F_1 \wedge F_2 \wedge \cdots \wedge F_n$ is true, G is also true. $F_1, F_2, \ldots, F_n$ are called *axioms* (or *postulates*, *premises*) of G.

Theorem 2.1 Given formulas $F_1, \ldots, F_n$ and a formula G, G is a logical consequence of $F_1, \ldots, F_n$ if and only if the formula $((F_1 \wedge \cdots \wedge F_n) \to G)$ is valid.

Proof ($\Rightarrow$) Suppose G is a logical consequence of $F_1, \ldots, F_n$. Let I be an arbitary interpretation. If $F_1, \ldots, F_n$ are true in I, then, by the definition of

logical consequence, G is true in I. Hence, $((F_1 \wedge \cdots \wedge F_n) \rightarrow G)$ is true in I. On the other hand, if $F_1, \ldots, F_n$ are false in I, then $((F_1 \wedge \cdots \wedge F_n) \rightarrow G)$ is true in I. Thus, we have shown that $((F_1 \wedge \cdots \wedge F_n) \rightarrow G)$ is true under any interpretation. That is, $((F_1 \wedge \cdots \wedge F_n) \rightarrow G)$ is a valid formula.

($\Leftarrow$) Suppose $((F_1 \wedge \cdots \wedge F_n) \rightarrow G)$ is a valid formula. For any interpretation I, if $(F_1 \wedge \cdots \wedge F_n)$ is true in I, G must be true in I. Therefore, G is a logical consequence of $F_1, \ldots, F_n$. Q.E.D.

Theorem 2.2 Given formulas $F_1, \ldots, F_n$ and a formula G, G is a logical consequence of $F_1, \ldots, F_n$ if and only if the formula $(F_1 \wedge \cdots \wedge F_n \wedge \sim G)$ is inconsistent.

Proof By Theorem 2.1, G is a logical consequence of $F_1, \ldots, F_n$ if and only if the formula $((F_1 \wedge \cdots \wedge F_n) \rightarrow G)$ is valid. Hence, G is a logical consequence of $F_1, \ldots, F_n$ if and only if the negation of $((F_1 \wedge \cdots \wedge F_n) \rightarrow G)$ is inconsistent. Since

$$\begin{aligned} \sim((F_1 \wedge \cdots \wedge F_n) \rightarrow G) &= \sim(\sim(F_1 \wedge \cdots \wedge F_n) \vee G) \\ &= (\sim(\sim(F_1 \wedge \cdots \wedge F_n)) \wedge \sim G) \\ &= (F_1 \wedge \cdots \wedge F_n) \wedge \sim G \\ &= F_1 \wedge \cdots \wedge F_n \wedge \sim G, \end{aligned}$$

we conclude that Theorem 2.2 is true.

Theorems 2.1 and 2.2 are very important. They show that proving that a particular formula is a logical consequence of a finite set of formulas is equivalent to proving that a certain related formula is valid or inconsistent.

If G is a logical consequence of $F_1, \ldots, F_n$, the formula $((F_1 \wedge \cdots \wedge F_n) \rightarrow G)$ is called a *theorem*, and G is also called the *conclusion of the theorem*. In mathematics as well as in other fields, many problems can be formulated as problems of proving theorems. This will be elaborated in the next section. Before going to the next section, let us now consider one simple example to show how we can utilize Theorems 2.1 and 2.2.

Example 2.11

Consider the formulas

$$F_1 \triangleq (P \rightarrow Q), \qquad F_2 \triangleq \sim Q, \qquad G \triangleq \sim P.$$

Show that G is a logical consequence of F_1 and F_2.

Method 1 We can use the truth-table technique to show that G is true in every model of $(P \rightarrow Q) \wedge \sim Q$. From Table 2.7, one notes that there is only one model for $(P \rightarrow Q) \wedge \sim Q$, namely, $\{\sim P, \sim Q\}$. $\sim P$ is certainly true in

TABLE 2.7

TRUTH TABLE OF $(P \to Q) \wedge \sim Q$ AND $\sim P$

P	Q	$P \to Q$	$\sim Q$	$(P \to Q) \wedge \sim Q$	$\sim P$
T	*T*	*T*	*F*	*F*	*F*
T	*F*	*F*	*T*	*F*	*F*
F	*T*	*T*	*F*	*F*	*T*
F	*F*	*T*	*T*	*T*	*T*

this model. Thus, by the definintion of logical consequence, we conclude that $\sim P$ is a logical consequence of $(P \to Q)$ and $\sim Q$.

Method 2 We can use Theorem 2.1. This can be done simply by extending the truth table in Table 2.7 or by evaluating the formula $((P \to Q) \wedge \sim Q) \to \sim P$.

Table 2.8 shows that $((P \to Q) \wedge \sim Q) \to \sim P$ is true in every interpretation. Therefore $((P \to Q) \wedge \sim Q) \to \sim P$ is valid and, according to Theorem 2.1, $\sim P$ is a logical consequence of $(P \to Q)$ and $\sim Q$.

TABLE 2.8

TRUTH TABLE OF $((P \to Q) \wedge \sim Q) \to \sim P$

P	Q	$P \to Q$	$\sim Q$	$(P \to Q) \wedge \sim Q$	$\sim P$	$((P \to Q) \wedge \sim Q) \to \sim P$
T	*T*	*T*	*F*	*F*	*F*	*T*
T	*F*	*F*	*T*	*F*	*F*	*T*
F	*T*	*T*	*F*	*F*	*T*	*T*
F	*F*	*T*	*T*	*T*	*T*	*T*

We can also prove the validity of $((P \to Q \wedge \sim Q) \to \sim P$ by transforming it into a conjunctive normal form.

$$
\begin{aligned}
((P \to Q) \wedge \sim Q) \to \sim P &= \sim((P \to Q) \wedge \sim Q) \vee \sim P && \text{by (2.2)} \\
&= \sim((\sim P \vee Q) \wedge \sim Q) \vee \sim P && \text{by (2.2)} \\
&= \sim((\sim P \wedge \sim Q) \vee (Q \wedge \sim Q)) \vee \sim P && \text{by (2.5b)} \\
&= \sim((\sim P \wedge \sim Q) \vee \square) \vee \sim P && \text{by (2.8b)} \\
&= \sim((\sim P \wedge \sim Q)) \vee \sim P && \text{by (2.6a)} \\
&= (P \vee Q) \vee \sim P && \text{by (2.10b)} \\
&= (Q \vee P) \vee \sim P && \text{by (2.3a)}
\end{aligned}
$$

$$= Q \vee (P \vee \sim P) \qquad \text{by (2.4a)}$$

$$= Q \vee \blacksquare \qquad \text{by (2.8a)}$$

$$= \blacksquare \qquad \text{by (2.7a).}$$

Thus, $((P \rightarrow Q) \wedge \sim Q) \rightarrow \sim P$ is valid.

Method 3 We can use Theorem 2.2. In this case, we prove that

$$((P \rightarrow Q) \wedge \sim Q) \wedge (\sim(\sim P)) = (P \rightarrow Q) \wedge \sim Q \wedge P$$

is inconsistent.

Again, as in Method 2, we can use the truth table technique to show that $(P \rightarrow Q) \wedge \sim Q \wedge P$ is false in every interpretation. From Table 2.9, we conclude that $(P \rightarrow Q) \wedge \sim Q \wedge P$ is inconsistent and, according to Theorem 2.2, $\sim P$ is a logical consequence of $(P \rightarrow Q)$ and $\sim Q$.

TABLE 2.9
TRUTH TABLE OF $(P \rightarrow Q) \wedge \sim Q \wedge P$

P	Q	$P \rightarrow Q$	$\sim Q$	$(P \rightarrow Q) \wedge \sim Q \wedge P$
T	*T*	*T*	*F*	*F*
T	*F*	*F*	*T*	*F*
F	*T*	*T*	*F*	*F*
F	*F*	*T*	*T*	*F*

We can also prove the inconsistency of $(P \rightarrow Q) \wedge \sim Q \wedge P$ by transforming it into a disjunctive normal form:

$$(P \rightarrow Q) \wedge \sim Q \wedge P = (\sim P \vee Q) \wedge \sim Q \wedge P \qquad \text{by (2.2)}$$

$$= (\sim P \wedge \sim Q \wedge P) \vee (Q \wedge \sim Q \wedge P) \qquad \text{by (2.5b)}$$

$$= \square \vee \square \qquad \text{by (2.8b)}$$

$$= \square \qquad \text{by (2.6a).}$$

Thus, $(P \rightarrow Q) \wedge \sim Q \wedge P$ is inconsistent.

2.6 APPLICATIONS OF THE PROPOSITIONAL LOGIC

Having discussed the various concepts in the previous sections, we now consider applications of the propositional logic. They are best illustrated by examples.

Example 2.12

Given that if the congress refuses to enact new laws, then the strike will not be over unless it lasts more than one year and the president of the firm resigns, will the strike not be over if the congress refuses to act and the strike just starts?

We first transform statements into symbols:

P: The congress refuses to act,
Q: The strike is over,
R: The president of the firm resigns,
S: The strike lasts more than one year.

Then the facts given in the example can be represented by formulas:

F_1: $(P \to (\sim Q \vee (R \wedge S)))$ $\triangleq$ If the congress refuses to enact new laws, then the strike will not be over unless it lasts more than one year and the president of the firm resigns,
F_2: P $\triangleq$ The congress refuses to act,
F_3: $\sim S$ $\triangleq$ The strike just starts.

From the facts F_1, F_2, and F_3, can we conclude that the strike will not be over? That is, can we show that $\sim Q$ is a logical consequence of F_1, F_2, and F_3? By Theorem 2.1, this is equivalent to showing that

TABLE 2.10

TRUTH TABLE OF $(F_1 \wedge F_2 \wedge F_3) \to \sim Q$, WHERE $F_1 \triangleq (P \to (\sim Q \vee (R \wedge S)))$, $F_2 \triangleq P$, $F_3 \triangleq \sim S$.

P	Q	R	S	F_1	F_2	F_3	$\sim Q$	$(F_1 \wedge F_2 \wedge F_3) \to \sim Q$
T	*T*	*T*	*T*	*T*	*T*	*F*	*F*	*T*
T	*T*	*T*	*F*	*F*	*T*	*T*	*F*	*T*
T	*T*	*F*	*T*	*F*	*T*	*F*	*F*	*T*
T	*T*	*F*	*F*	*F*	*T*	*T*	*F*	*T*
T	*F*	*T*	*T*	*T*	*T*	*F*	*T*	*T*
T	*F*	*T*	*F*	*T*	*T*	*T*	*T*	*T*
T	*F*	*F*	*T*	*T*	*T*	*F*	*T*	*T*
T	*F*	*F*	*F*	*T*	*T*	*T*	*T*	*T*
F	*T*	*T*	*T*	*T*	*F*	*F*	*F*	*T*
F	*T*	*T*	*F*	*T*	*F*	*T*	*F*	*T*
F	*T*	*F*	*T*	*T*	*F*	*F*	*F*	*T*
F	*T*	*F*	*F*	*T*	*F*	*T*	*F*	*T*
F	*F*	*T*	*T*	*T*	*F*	*F*	*T*	*T*
F	*F*	*T*	*F*	*T*	*F*	*T*	*T*	*T*
F	*F*	*F*	*T*	*T*	*F*	*F*	*T*	*T*
F	*F*	*F*	*F*	*T*	*F*	*T*	*T*	*T*

$((P \to (\sim Q \vee (R \wedge S))) \wedge P \wedge \sim S) \to \sim Q$ is a valid formula. The truth values of the above formula under all the interpretations are shown in Table 2.10.

From Table 2.10, there is no interpretation under which the formula is false. Hence the formula $((P \to (\sim Q \vee (R \wedge S))) \wedge P \wedge \sim S) \to \sim Q$ is a valid formula. Therefore, $\sim Q$ is a logical consequence of F_1, F_2, and F_3. That is, we can conclude $\sim Q$ from F_1, F_2, and F_3. Hence, the answer is "The strike will not be over."

Example 2.13 (Chemical Synthesis Problem)

Suppose we can perform the following chemical reactions:

$$\mathrm{MgO} + \mathrm{H_2} \to \mathrm{Mg} + \mathrm{H_2O}$$

$$\mathrm{C} + \mathrm{O_2} \to \mathrm{CO_2}$$

$$\mathrm{CO_2} + \mathrm{H_2O} \to \mathrm{H_2CO_3}.$$

Suppose we have some quantities of MgO, H_2, O_2, and C. Show that we can make H_2CO_3.

For this problem, we may consider MgO, H_2, O_2, and C as atomic formulas. Then the above chemical reactions can be represented by the following formulas:

A_1: $(\mathrm{MgO} \wedge \mathrm{H_2}) \to (\mathrm{Mg} \wedge \mathrm{H_2O})$

A_2: $(\mathrm{C} \wedge \mathrm{O_2}) \to \mathrm{CO_2}$

A_3: $(\mathrm{CO_2} \wedge \mathrm{H_2O}) \to \mathrm{H_2CO_3}$.

Since we have MgO, H_2, O_2, and C, these facts can be represented by the following formulas:

A_4: MgO

A_5: $\mathrm{H_2}$

A_6: $\mathrm{O_2}$

A_7: C.

Now, the problem can be regarded as proving that H_2CO_3 is a logical consequence of $A_1, \ldots, A_7$, which, by Theorem 2.2, is true if $(A_1 \wedge \cdots \wedge A_7) \wedge \sim \mathrm{H_2CO_3}$ is inconsistent. We prove this by transforming the formula

$$(A_1 \wedge \cdots \wedge A_7 \wedge \sim \mathrm{H_2CO_3})$$

into a disjunctive normal form.

$$
\begin{aligned}
(A_1 \wedge \cdots \wedge A_7 \wedge \sim H_2CO_3) &= ((MgO \wedge H_2) \rightarrow (Mg \wedge H_2O)) \wedge ((C \wedge O_2) \\
&\quad \rightarrow CO_2) \wedge ((CO_2 \wedge H_2O) \rightarrow H_2CO_3) \\
&\quad \wedge MgO \wedge H_2 \wedge O_2 \wedge C \wedge \sim H_2CO_3 \\
&= (\sim MgO \vee \sim H_2 \vee Mg) \wedge (\sim MgO \vee \sim H_2 \\
&\quad \vee H_2O) \wedge (\sim C \vee \sim O_2 \vee CO_2) \\
&\quad \wedge (\sim CO_2 \vee \sim H_2O \vee H_2CO_3) \\
&\quad \wedge MgO \wedge H_2 \wedge O_2 \wedge C \wedge \sim H_2CO_3 \\
&= (\sim MgO \vee \sim H_2 \vee Mg) \wedge (\sim MgO \vee \sim H_2 \\
&\quad \vee H_2O) \wedge MgO \wedge H_2 \wedge (\sim C \vee \sim O_2 \\
&\quad \vee CO_2) \wedge C \wedge O_2 \\
&\quad \wedge (\sim CO_2 \vee \sim H_2O \vee H_2CO_3) \wedge \sim H_2CO_3 \\
&= Mg \wedge H_2O \wedge MgO \wedge H_2 \wedge CO_2 \wedge C \wedge O_2 \\
&\quad \wedge (\sim CO_2 \vee \sim H_2O) \wedge \sim H_2CO_3 \\
&= (\sim CO_2 \vee \sim H_2O) \wedge H_2O \wedge CO_2 \wedge Mg \\
&\quad \wedge MgO \wedge H_2 \wedge C \wedge O_2 \wedge \sim H_2CO_3 \\
&= \square \wedge Mg \wedge MgO \wedge H_2 \wedge C \wedge O_2 \\
&\quad \wedge \sim H_2CO_3 \\
&= \square.
\end{aligned}
$$

Since $\square$ is always false, the formula $(A_1 \wedge \cdots \wedge A_7 \wedge \sim H_2CO_3)$ is inconsistent. Therefore H_2CO_3 is a logical consequence of $A_1, \ldots, A_7$. That is, we can make H_2CO_3 from MgO, H_2, O_2, and C. The above procedure for proving an inconsistent formula by transforming it into $\square$ is sometimes called the *multiplication method*, because the transformation process is very similar to multiplying out an arithmetic expression. Example 2.13 is only a simple chemical synthesis problem. In reality, there are hundreds of chemical reactions. In order to use computers efficiently, we need an efficient computer program for proving theorems. This will be discussed in detail in Chapters 4–9.

In the above examples, we have shown that the propositional logic can be applied to many problems. The approach is first to symbolize problems by formulas and then to prove that the formulas are valid or inconsistent. We have used the truth-table method and the multiplication method to prove

whether a formula is valid (inconsistent). In Chapters 4–9, more efficient methods will be given for proving valid and inconsistent formulas.

REFERENCES

Hilbert, D., and W. Ackermann (1950): "Principles of Mathematical Logic," Chelsea, New York.
Kleene, S. C. (1967): "Mathematical Logic," Wiley, New York.
Mendelson, E. (1964): "Introduction to Mathematical Logic," Van Nostrand-Reinhold, Princeton, New Jersey.
Stoll, R. R. (1961): "Sets, Logic and Axiomatic Theories," Freeman, San Francisco.
Whitehead, A. N., and B. Russell, (1927): "Principia Mathematica," Cambridge Univ. Press, London and New York.

EXERCISES

Section 2.1

1. Symbolize the following statements by formulas.
   (a) A relation is an equivalence relation if and only if it is reflexive, symmetric, and transitive.
   (b) If the humidity is so high, it will rain either this afternoon or this evening.
   (c) Cancer will not be cured unless its cause is determined and a new drug for cancer is found.
   (d) It requires courage and skills to climb that mountain.
   (e) If he is a man who campaigns so hard, he probably will be elected.
2. Let

   $P \triangleq$ He needs a doctor, $Q \triangleq$ He needs a lawyer,

   $R \triangleq$ He has an accident, $S \triangleq$ He is sick,

   $U \triangleq$ He is injured.

   State the following formulas in English.

   (a) $(S \to P) \wedge (R \to Q)$ (b) $P \to (S \vee U)$

   (c) $(P \wedge Q) \to R$ (d) $(P \wedge Q) \leftrightarrow (S \wedge U)$

   (e) $\sim(S \vee U) \to \sim P$.

Section 2.2

3. Complete the following truth table (Table 2.11) of the formula

$$(\sim P \vee Q) \wedge (\sim(P \wedge \sim Q)).$$

TABLE 2.11

P	Q	$\sim P$	$\sim Q$	$\sim P \vee Q$	$P \wedge \sim Q$	$\sim(P \wedge \sim Q)$	$(\sim P \vee Q) \wedge (\sim(P \wedge \sim Q))$
T	T						
T	F						
F	T						
F	F						

Section 2.3

4. For each of the following formulas, determine whether it is valid, invalid, inconsistent, consistent, or some combination of these.

   (a) $\sim(\sim P) \rightarrow P$ (b) $P \rightarrow (P \wedge Q)$

   (c) $\sim(P \vee Q) \vee \sim Q$ (d) $(P \vee Q) \rightarrow P$

   (e) $(P \rightarrow Q) \rightarrow (\sim Q \rightarrow \sim P)$ (f) $(P \rightarrow Q) \rightarrow (Q \rightarrow P)$

   (g) $P \vee (P \rightarrow Q)$ (h) $(P \wedge (Q \rightarrow P)) \rightarrow P$

   (i) $P \vee (Q \rightarrow \sim P)$ (j) $(P \vee \sim Q) \wedge (\sim P \vee Q)$

   (k) $\sim P \wedge (\sim(P \rightarrow Q))$ (l) $P \rightarrow \sim P$

   (m) $\sim P \rightarrow P$.

5. Consider the following statement:

   If the congress refuses to enact new laws, then the strike will not be over unless it lasts more than one year and the president of the firm resigns, and if either the congress enacts new laws or the strike is not over then the strike lasts more than one year.

   Is the above statement contradictory? Explain.

Section 2.4

6. Transform the following into disjunctive normal forms

   (a) $(\sim P \wedge Q) \rightarrow R$ (b) $P \rightarrow ((Q \wedge R) \rightarrow S)$

   (c) $\sim(P \vee \sim Q) \wedge (S \rightarrow T)$ (d) $(P \rightarrow Q) \rightarrow R$

   (e) $\sim(P \wedge Q) \wedge (P \vee Q)$.

7. Transform the following into conjunctive normal forms

   (a) $P \vee (\sim P \wedge Q \wedge R)$ (b) $\sim(P \rightarrow Q) \vee (P \vee Q)$

   (c) $\sim(P \rightarrow Q)$ (d) $(P \rightarrow Q) \rightarrow R$

   (e) $(\sim P \wedge Q) \vee (P \wedge \sim Q)$.

8. Is it possible to have a formula that is in conjunctive normal form as well as disjunctive normal form? If yes, give an example.
9. Verify each of the following pairs of equivalent formulas by transforming formulas on both sides of the sign $=$ into the same normal form.
   (a) $P \wedge P = P$, and $P \vee P = P$
   (b) $(P \rightarrow Q) \wedge (P \rightarrow R) = (P \rightarrow (Q \wedge R))$
   (c) $(P \rightarrow Q) \rightarrow (P \wedge Q) = (\sim P \rightarrow Q) \wedge (Q \rightarrow P)$
   (d) $P \wedge Q \wedge (\sim P \vee \sim Q) = \sim P \wedge \sim Q \wedge (P \vee Q)$
   (e) $P \vee (P \rightarrow (P \wedge Q)) = \sim P \vee \sim Q \vee (P \wedge Q)$.

Sections 2.5 and 2.6

10. Prove that $(\sim Q \rightarrow \sim P)$ is a logical consequence of $(P \rightarrow Q)$.
11. If the congress refuses to enact new laws, then the strike will not be over unless it lasts more than one year and the president of the firm resigns. Suppose the congress refuses to act, the strike is over, and the president of the firm does not resign. Has the strike lasted more than one year?
12. Consider the following statements:
    $F_1 \triangleq$ Tom cannot be a good student unless he is smart and his father supports him.
    $F_2 \triangleq$ Tom is a good student only if his father supports him.
    Show that F_2 is a logical consequence of F_1.
13. Show that for the following statements, F_2 is a logical consequence of F_1.
    $F_1 \triangleq$ If the president does not have the appropriate authority or if he does not want to take the responsibility, then neither order will be restored nor will the riots stop spreading unless the rioters become tired of rioting and the local authorities begin to take conciliatory actions.
    $F_2 \triangleq$ If the president does not want to take the responsibility and the rioters are not tired of rioting, then riots will spread.
14. Show that Q is a logical consequence of $(P \rightarrow Q)$ and P. This is related to the so-called *modus ponens* rule.

Chapter 3

The First-Order Logic

3.1 INTRODUCTION

In the propositional logic, the most basic elements are atoms. Through atoms we build up formulas. We then use formulas to express various complex ideas. As discussed in Chapter 2, in this simple logic an atom represents a declarative sentence that can be either true or false, but not both. An atom is treated as a single unit. Its structure and composition are suppressed. However, there are many ideas that cannot be treated in this simple way. For example, consider the following deduction of statements:

Every man is mortal.
Since Confucius is a man, he is mortal.

The above reasoning is intuitively correct. However, if we denote

P: Every man is mortal,
Q: Confucius is a man,
R: Confucius is mortal,

then R is not a logical consequence of P and Q within the framework of the propositional logic. This is because the structures of P, Q, and R are not used in the propositional logic. In this chapter, we shall introduce the first-order logic, which has three more logical notions (called *terms*, *predicates*,

and *quantifiers*) than does the propositional logic. It will be made clear later that much of everyday and mathematical language can be symbolized by the first-order logic.

Just as in the propositional logic, we first have to define atoms in the first-order logic. Before giving the formal definition of an atom, we first consider some examples.

Suppose we want to represent "x is greater than 3." We first define a predicate $GREATER(x,y)$ to mean "x is greater than y." (Note that a predicate is a relation.) Then the sentence "x is greater than 3" is represented by $GREATER(x,3)$.

Similarly, we can represent "x loves y" by the predicate $LOVE(x,y)$. Then "John loves Mary" can be represented by $LOVE(\text{John},\text{Mary})$.

We can also use function symbols in the first-order logic. For example, we can use $plus(x,y)$ to denote "$x + y$" and $father(x)$ to mean the father of x. The sentences "$x + 1$ is greater than x" and "John's father loves John" can be symbolized as $GREATER(plus(x,1),x)$ and $LOVE(father(\text{John}),\text{John})$, respectively.

In the above examples, $GREATER(x,3)$, $LOVE(\text{John},\text{Mary})$, $GREATER(plus(x,1),x)$, and $LOVE(father(\text{John}),\text{John})$ are all atoms in the first-order logic, where $GREATER$ and $LOVE$ are *predicate symbols*, x is a *variable*, 3, John, and Mary are *individual symbols or constants*, and *father* and *plus* are *function symbols*.

In general, we are allowed to use the following four types of symbols to construct an atom:

i. Individual symbols or constants: These are usually names of objects, such as John, Mary, and 3.

ii. Variable symbols: These are customarily *lowercase* unsubscripted or subscripted letters, $x, y, z, \ldots$.

iii. Function symbols: These are customarily *lowercase* letters f, g, h, ... or expressive strings of *lowercase* letters such as *father* and *plus*.

iv. Predicate symbols: These are customarily *uppercase* letters P, Q, R, ... or expressive strings of *uppercase* letters such as $GREATER$ and $LOVE$.

Any function or predicate symbol takes a specified number of arguments. If a function symbol f takes n arguments, f is called an *n-place function symbol*. It is noted that an individual symbol or a constant may be considered a function symbol that takes no argument. Similarly, if a predicate symbol P takes n arguments, P is called an *n-place predicate symbol*. For example, *father* is a one-place function symbol, and $GREATER$ and $LOVE$ are two-place predicate symbols.

A function is a mapping that maps a list of constants to a constant. For example, *father* is a function that maps a person named John to a person

who is John's father. Therefore, *father*(John) represents a person, even though his name is unknown. We call *father*(John) a *term* in the first-order logic. More formally, we have the following definition.

Definition *Terms* are defined recursively as follows:

i. A constant is a term.
ii. A variable is a term.
iii. If f is an n-place function symbol, and $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term.
iv. All terms are generated by applying the above rules.

Example 3.1

Since x and 1 are both terms and *plus* is a two-place function symbol, $plus(x, 1)$ is a term according to the above definition.

Furthermore, the reader can see that $plus(plus(x, 1), x)$ and *father*(*father* (John)) are also terms; the former denotes $(x+1)+x$ and the latter denotes the grandfather of John.

A predicate is a mapping that maps a list of constants to ***T*** or ***F***. For example, $GREATER$ is a predicate. $GREATER(5,3)$ is ***T***, but $GREATER(1,3)$ is ***F***. Having defined terms, we can now formally define an atom in the first-order logic.

Definition If P is an n-place predicate symbol, and $t_1, \ldots, t_n$ are terms, then $P(t_1, \ldots, t_n)$ is an *atom*. No other expressions can be atoms.

Once atoms are defined, we can use the same five logical connectives given in Chapter 2 to build up formulas. Furthermore, since we have introduced variables, we use two special symbols $\forall$ and $\exists$ to characterize variables. The symbols $\forall$ and $\exists$ are called, respectively, the *universal* and *existential quantifiers*. If x is a variable, then $(\forall x)$ is read as "*for all* x," "*for each* x," or "*for every* x," while $(\exists x)$ is read as "*there exists an* x," "*for some* x," or "*for at least one* x." Let us consider a few examples to see how the quantifiers can be used.

Example 3.2

Symbolize the following statements:

(a) Every rational number is a real number.
(b) There exists a number that is a prime.
(c) For every number x, there exists a number y such that $x < y$.

Denote "x is a prime number" by $P(x)$, "x is a rational number" by $Q(x)$, "x is a real number" by $R(x)$, and "x is less than y" by $LESS(x, y)$. Then

the above statements can be denoted, respectively, as

(a′) $(\forall x)(Q(x) \rightarrow R(x))$

(b′) $(\exists x)P(x)$

(c′) $(\forall x)(\exists y)\,LESS(x, y)$.

Each of the expressions (a′), (b′), and (c′) is called a formula. Before we give a formal definition of a formula, we have to distinguish between *bound* variables and *free* variables. To do this, we first define the *scope* of a quantifier occurring in a formula as the formula to which the quantifier applies. For example, the scope of both the universal and existential quantifiers in the formula $(\forall x)(\exists y)\,LESS(x, y)$ is $LESS(x, y)$. The scope of the universal quantifier in the formula $(\forall x)(Q(x) \rightarrow R(x))$ is $(Q(x) \rightarrow R(x))$.

Definition An occurrence of a variable in a formula is *bound* if and only if the occurrence is within the scope of a quantifier employing the variable, or is the occurrence in that quantifier. An occurrence of a variable in a formula is *free* if and only if this occurrence of the variable is not bound.

Definition A variable is *free* in a formula if at least one occurrence of it is free in the formula. A variable is *bound* in a formula if at least one occurrence of it is bound.

In the formula $(\forall x)P(x, y)$, since both the occurrences of x are bound, the variable x is bound. However, the variable y is free since the only occurrence of y is free. We note that a variable can be both free and bound in a formula. For example, y is both free and bound in the formula $(\forall x)P(x, y) \wedge (\forall y)Q(y)$.

We now can formally define a formula by using atoms, logical connectives, and quantifiers.

Definition *Well-formed formulas*, or *formulas* for short, in the first-order logic are defined recursively as follows:

i. An atom is a formula. (Note that "atom" is an abbreviation for an atomic formula.)

ii. If F and G are formulas, then $\sim(F)$, $(F \vee G)$, $(F \wedge G)$, $(F \rightarrow G)$, and $(F \leftrightarrow G)$ are formulas.

iii. If F is a formula and x is a free variable in F, then $(\forall x)F$ and $(\exists x)F$ are formulas.

iv. Formulas are generated only by a finite number of applications of (i), (ii), and (iii).

Throughout this book, parentheses may be omitted by the same conventions established in Chapter 2. We extend these conventions by agreeing that quantifiers have the least ranks. For example, $(\exists x)A \vee B$ stands for $(((\exists x)A) \vee (B))$.

Example 3.3

Translate the statement "Every man is mortal. Confucius is a man. Therefore, Confucius is mortal." into a formula.

Denote "x is a man" by $MAN(x)$, and "x is mortal" by $MORTAL(x)$. Then "every man is mortal" can be represented by

$$(\forall x)(MAN(x) \rightarrow MORTAL(x)),$$

"Confucius is a man" by

$$MAN(\text{Confucius}),$$

and "Confucius is mortal" by

$$MORTAL(\text{Confucius}).$$

The whole statement can now be represented by

$$(\forall x)(MAN(x) \rightarrow MORTAL(x)) \wedge MAN(\text{Confucius}) \rightarrow MORTAL(\text{Confucius}).$$

Example 3.4

The basic axioms of natural numbers are the following:

A_1: For every number, there is one and only one immediate successor.
A_2: There is no number for which 0 is the immediate successor.
A_3: For every number other than 0, there is one and only one immediate predecessor.

Let $f(x)$ and $g(x)$ represent the immediate successor and predecessor of x, respectively. Let "x is equal to y" be denoted by $E(x,y)$. Then the axioms can be represented by the following formulas:

A_1': $(\forall x)(\exists y)(E(y,f(x)) \wedge (\forall z)(E(z,f(x)) \rightarrow E(y,z)))$

A_2': $\sim((\exists x)E(0,f(x)))$

A_3': $(\forall x)(\sim E(x,0) \rightarrow ((\exists y)(E(y,g(x)) \wedge (\forall z)(E(z,g(x)) \rightarrow E(y,z)))))$.

3.2 INTERPRETATIONS OF FORMULAS IN THE FIRST-ORDER LOGIC

In the propositional logic, an interpretation is an assignment of truth values to atoms. In the first-order logic, since there are variables involved, we have to do more than that. To define an interpretation for a formula in the

first-order logic, we have to specify two things, namely, the domain and an assignment to constants, function symbols, and predicate symbols occurring in the formula. The following is the formal definition of an interpretation of a formula in the first-order logic.

Definition An *interpretation* of a formula F in the first-order logic consists of a nonempty domain D, and an assignment of "values" to each constant, function symbol, and predicate symbol occurring in F as follows:

1. To each constant, we assign an element in D.
2. To each n-place function symbol, we assign a mapping from D^n to D. (Note that $D^n = \{(x_1, \ldots, x_n) \mid x_1 \in D, \ldots, x_n \in D\}$).
3. To each n-place predicate symbol, we assign a mapping from D^n to $\{\boldsymbol{T}, \boldsymbol{F}\}$.

Sometimes, to emphasize the domain D, we speak of an interpretation of the formula *over* D. When we evaluate the truth value of a formula in an interpretation over the domain D, $(\forall x)$ will be interpreted as "for all elements x in D," and $(\exists x)$ as "there is an element in D."

For every interpretation of a formula over a domain D, the formula can be evaluated to $\boldsymbol{T}$ or $\boldsymbol{F}$ according to the following rules:

1. If the truth values of formulas G and H are evaluated, then the truth values of the formulas $\sim G$, $(G \wedge H)$, $(G \vee H)$, $(G \rightarrow H)$, and $(G \leftrightarrow H)$ are evaluated by using Table 2.1 in Chapter 2.
2. $(\forall x)G$ is evaluated to $\boldsymbol{T}$ if the truth value of G is evaluated to $\boldsymbol{T}$ for every d in D; otherwise, it is evaluated to $\boldsymbol{F}$.
3. $(\exists x)G$ is evaluated to $\boldsymbol{T}$ if the truth value of G is $\boldsymbol{T}$ for at least one d in D; otherwise, it is evaluated to $\boldsymbol{F}$.

We note that any formula containing free variables cannot be evaluated. In the rest of the book, we shall assume either that formulas do not contain free variables, or that free variables are treated as constants.

Example 3.5

Let us consider the formulas

$$(\forall x)\, P(x) \qquad \text{and} \qquad (\exists x) \sim P(x).$$

Let an interpretation be as follows:

Domain: $D = \{1, 2\}$.

Assignment for P:

$P(1)$	$P(2)$
$\boldsymbol{T}$	$\boldsymbol{F}$

It should be easy for the reader to confirm that $(\forall x)P(x)$ is F in this interpretation because $P(x)$ is not T for both $x = 1$ and $x = 2$. On the other hand, since $\sim P(2)$ is T in this interpretation, $(\exists x) \sim P(x)$ is T in this interpretation.

Example 3.6

Consider the formula

$$(\forall x)(\exists y)P(x, y).$$

Let us define an interpretation as follows:

$D = \{1,2\}$

$P(1,1)$	$P(1,2)$	$P(2,1)$	$P(2,2)$
T	F	F	T

If $x = 1$, we can see that there is a y, namely 1, such that $P(1, y)$ is T.

If $x = 2$, there is also a y, namely 2, such that $P(2, y)$ is T.

Therefore, in the above interpretation, for every x in D, there is a y such that $P(x, y)$ is T; that is $(\forall x)(\exists y)P(x, y)$ is T in this interpretation.

Example 3.7

Consider the formula

G: $(\forall x)(P(x) \rightarrow Q(f(x), a))$.

There are one constant a, one one-place function symbol f, one one-place predicate symbol P, and one two-place predicate symbol Q in G. The following is an interpretation I of G.

Domain: $D = \{1,2\}$.

Assignment for a:

a
1

Assignment for f:

$f(1)$	$f(2)$
2	1

Assignment for P and Q:

$P(1)$	$P(2)$	$Q(1,1)$	$Q(1,2)$	$Q(2,1)$	$Q(2,2)$
F	T	T	T	F	T

If $x = 1$, then

$$\begin{aligned} P(x) \to Q(f(x), a) &= P(1) \to Q(f(1), a) \\ &= P(1) \to Q(2, 1) \\ &= \boldsymbol{F} \to \boldsymbol{F} = \boldsymbol{T}. \end{aligned}$$

If $x = 2$, then

$$\begin{aligned} P(x) \to Q(f(x), a) &= P(2) \to Q(f(2), a) \\ &= P(2) \to Q(1, 1) \\ &= \boldsymbol{T} \to \boldsymbol{T} = \boldsymbol{T}. \end{aligned}$$

Since $P(x) \to Q(f(x), a)$ is true for all elements x in the domain D, the formula $(\forall x)(P(x) \to Q(f(x), a))$ is true under the interpretation I.

Example 3.8

Evaluate the truth values of the following formulas under the interpretation given in Example 3.7.

(a) $(\exists x)(P(f(x)) \wedge Q(x, f(a)))$,

(b) $(\exists x)(P(x) \wedge Q(x, a))$,

(c) $(\forall x)(\exists y)(P(x) \wedge Q(x, y))$.

For (a): If $x = 1$,

$$\begin{aligned} P(f(x)) \wedge Q(x, f(a)) &= P(f(1)) \wedge Q(1, f(a)) \\ &= P(2) \wedge Q(1, f(1)) \\ &= P(2) \wedge Q(1, 2) \\ &= \boldsymbol{T} \wedge \boldsymbol{T} = \boldsymbol{T}. \end{aligned}$$

If $x = 2$,

$$\begin{aligned} P(f(x)) \wedge Q(x, f(a)) &= P(f(2)) \wedge Q(2, f(1)) \\ &= P(1) \wedge Q(2, 1) \\ &= \boldsymbol{F} \wedge \boldsymbol{F} = \boldsymbol{F}. \end{aligned}$$

Since there is an element in the domain D, that is, $x = 1$, such that $P(f(x)) \wedge Q(x, f(a))$ is true, the truth value of the formula $(\exists x)(P(f(x)) \wedge Q(x, f(a)))$ is true under the interpretation I.

For (b): If $x = 1$,

$$P(x) \wedge Q(x,a) = P(1) \wedge Q(1,1)$$
$$= \boldsymbol{F} \wedge \boldsymbol{T} = \boldsymbol{F}.$$

If $x = 2$,

$$P(x) \wedge Q(x,a) = P(2) \wedge Q(2,1)$$
$$= \boldsymbol{T} \wedge \boldsymbol{F} = \boldsymbol{F}.$$

Since there is no element in the domain D such that $P(x) \wedge Q(x,a)$ is true, the formula $(\exists x)(P(x) \wedge Q(x,a))$ is evaluated to be false under the interpretation I.

For (c): If $x = 1$, then $P(x) = P(1) = \boldsymbol{F}$. Therefore $P(x) \wedge Q(x,y) = \boldsymbol{F}$ for $y = 1$ and $y = 2$. Since there exists an x, that is, $x = 1$, such that $(\exists y)(P(x) \wedge Q(x,y))$ is false, the formula $(\forall x)(\exists y)(P(x) \wedge Q(x,y))$ is false under the interpretation I, that is, the formula is falsified by I.

Once interpretations are defined, all the concepts, such as validity, inconsistency, and logical consequence, defined in Chapter 2 can be defined analogously for formulas of the first-order logic.

Definition A formula G is *consistent* (*satisfiable*) if and only if there exists an interpretation I such that G is evaluated to $\boldsymbol{T}$ in I. If a formula G is $\boldsymbol{T}$ in an interpretation I, we say that I is a *model* of G and I *satisfies* G.

Definition A formula G is *inconsistent* (*unsatisfiable*) if and only if there exists no interpretation that satisfies G.

Definition A formula G is *valid* if and only if every interpretation of G satisfies G.

Definition A formula G is a *logical consequence* of formulas $F_1, F_2, \ldots, F_n$ if and only if for every interpretation I, if $F_1 \wedge \cdots \wedge F_n$ is true in I, G is also true in I.

The relations between validity (inconsistency) and logical consequence as stated in Theorems 2.1 and 2.2 are also true for the first-order logic. In fact, the first-order logic can be considered as an extension of the propositional logic. When a formula in the first-order logic contains no variables and quantifiers, it can be treated just as a formula in the propositional logic.

Example 3.9

It is left as an exercise for the reader to prove the following:

(1) $(\forall x)P(x) \wedge (\exists y) \sim P(y)$ is inconsistent.

(2) $(\forall x)P(x) \rightarrow (\exists y)P(y)$ is valid.

(3) $P(a) \rightarrow \sim ((\exists x)P(x))$ is consistent.

(4) $(\forall x)P(x) \vee ((\exists y) \sim P(y))$ is valid.

Example 3.10

Consider formulas

F_1: $(\forall x)(P(x) \rightarrow Q(x))$

F_2: $P(a)$.

We will now prove that formula $Q(a)$ is a logical consequence of F_1 and F_2.

Consider any interpretation I that satisfies $(\forall x)(P(x) \rightarrow Q(x)) \wedge P(a)$. Certainly in this interpretation, $P(a)$ is $\boldsymbol{T}$. Assume $Q(a)$ is not $\boldsymbol{T}$ in this interpretation; then $\sim P(a) \vee Q(a)$, that is, $P(a) \rightarrow Q(a)$ is $\boldsymbol{F}$ in I. This means that $(\forall x)(P(x) \rightarrow Q(x))$ is $\boldsymbol{F}$ in I, which is impossible. Therefore, $Q(a)$ must be $\boldsymbol{T}$ in every interpretation that satisfies

$$(\forall x)(P(x) \rightarrow Q(x)) \wedge P(a).$$

This means that $Q(a)$ is a logical consequence of F_1 and F_2.

In the first-order logic, since there are an infinite number of domains, in general, there are an infinite number of interpretations of a formula. Therefore, unlike in the propositional logic, it is not possible to verify a valid or an inconsistent formula by evaluating the formula under all the possible interpretations. In the subsequent chapters, we shall give procedures for verifying inconsistent formulas in the first-order logic.

3.3 PRENEX NORMAL FORMS IN THE FIRST-ORDER LOGIC

In the propositional logic, we have introduced two normal forms—the conjunctive normal form and the disjunctive normal form. In the first-order logic, there is also a normal form called the "prenex normal form." The reason for considering a prenex normal form of a formula is to simplify proof procedures, which will be discussed in the sequel.

Definition A formula F in the first-order logic is said to be in a *prenex normal form* if and only if the formula F is in the form of

$$(Q_1 x_1) \cdots (Q_n x_n)(M)$$

where every $(Q_i x_i)$, $i = 1, \ldots, n$, is either $(\forall x_i)$ or $(\exists x_i)$, and M is a formula containing no quantifiers. $(Q_1 x_1) \cdots (Q_n x_n)$ is called the *prefix* and M is called the *matrix* of the formula F.

Here are some formulas in prenex normal form:

$$(\forall x)(\forall y)(P(x,y) \wedge Q(y)), \qquad (\forall x)(\forall y)(\sim P(x,y) \rightarrow Q(y)),$$
$$(\forall x)(\forall y)(\exists z)(Q(x,y) \rightarrow R(z)).$$

Given a formula, we now consider a method of transforming it into a prenex normal form. This is to be accomplished by first considering some basic pairs of equivalent formulas in the first-order logic. We remember that two formulas F and G are *equivalent*, denoted by $F = G$, if and only if the truth values of F and G are the same under every interpretation. The basic pairs of equivalent formulas given in Table 2.6 in Chapter 2 are still true for the first-order logic. In addition, there are other pairs of equivalent formulas containing quantifiers. We now consider these additional pairs of equivalent formulas. Let F be a formula containing a free variable x. To emphasize that the free variable x is in F, we represent F by $F[x]$. Let G be a formula that does not contain variable x. Then we have the following pairs of equivalent formulas, where Q is either $\forall$ or $\exists$. For simplicity, we shall call each of the following pairs of equivalent formulas a "law."

(3.1a) $(Qx)F[x] \vee G = (Qx)(F[x] \vee G)$.

(3.1b) $(Qx)F[x] \wedge G = (Qx)(F[x] \wedge G)$.

(3.2a) $\sim((\forall x)F[x]) = (\exists x)(\sim F[x])$.

(3.2b) $\sim((\exists x)F[x]) = (\forall x)(\sim F[x])$.

Laws (3.1a) and (3.1b) are obviously true, since G does not contain x and therefore can be brought into the scope of the quantifier Q. Laws (3.2a) and (3.2b) are not difficult to prove. Let I be any arbitrary interpretation over a domain D. If $\sim((\forall x)F[x])$ is true in I, then $(\forall x)F[x]$ is false in I. This means there is an element e in D such that $F[e]$ is false, that is, $\sim F[e]$ is true in I. Therefore, $(\exists x)(\sim F[x])$ is true in I. On the other hand, if $\sim((\forall x)F[x])$ is false in I, then $(\forall x)F[x]$ is true in I. This means that $F[x]$ is true for every element x in D, that is, $\sim F[x]$ is false for every element x in D. Therefore, $(\exists x)(\sim F[x])$ is false in I. Since $\sim((\forall x)F[x])$ and $(\exists x)(\sim F[x])$ always assume the same truth value for every arbitrary interpretation, by definition, $\sim((\forall x)F[x]) = (\exists x)(\sim F[x])$. Hence, law (3.2a) is proved. Similarly, we can prove law (3.2b).

Suppose $F[x]$ and $H[x]$ are two formulas containing x. There are two other laws:

(3.3a) $(\forall x)F[x] \wedge (\forall x)H[x] = (\forall x)(F[x] \wedge H[x])$.

(3.3b) $(\exists x)F[x] \vee (\exists x)H[x] = (\exists x)(F[x] \vee H[x])$.

That is, the universal quantifier $\forall$ and the existential quantifier $\exists$ can distribute over $\wedge$ and $\vee$, respectively.

The proofs of (3.3a) and (3.3b) are not difficult. We leave the proofs to the reader. However, the universal quantifier $\forall$ and the existential quantifier $\exists$ *cannot* distribute over $\vee$ and $\wedge$, respectively. That is,

$$(\forall x)F[x] \vee (\forall x)H[x] \neq (\forall x)(F[x] \vee H[x])$$

and

$$(\exists x)F[x] \wedge (\exists x)H[x] \neq (\exists x)(F[x] \wedge H[x]).$$

For cases like these, we have to do something special. Since every bound variable in a formula can be considered as a dummy variable, every bound variable x can be renamed z, and the formula $(\forall x)H[x]$ becomes $(\forall z)H[z]$; that is, $(\forall x)H[x] = (\forall z)H[z]$. Suppose we choose the variable z that does not appear in $F[x]$. Then,

$$\begin{aligned}(\forall x)F[x] \vee (\forall x)H[x] &= (\forall x)F[x] \vee (\forall z)H[z] \\ &\quad \text{(by renaming all } x\text{'s occurring in } (\forall x)H[x] \text{ as } z) \\ &= (\forall x)(\forall z)(F[x] \vee H[z]) \\ &\quad \text{(by 3.1a).}\end{aligned}$$

Similarly, we can have

$$\begin{aligned}(\exists x)F[x] \wedge (\exists x)H[x] &= (\exists x)F[x] \wedge (\exists z)H[z] \\ &\quad \text{(by renaming all } x\text{'s occurring in } (\exists x)H[x] \text{ as } z) \\ &= (\exists x)(\exists z)(F[x] \wedge H[z]) \\ &\quad \text{(by 3.1b).}\end{aligned}$$

Therefore, for these two cases, we still can bring all the quantifiers to the left of the formula. In general, we have

(3.4a) $(Q_1 x)F[x] \vee (Q_2 x)H[x] = (Q_1 x)(Q_2 z)(F[x] \vee H[z])$

(3.4b) $(Q_3 x)F[x] \wedge (Q_4 x)H[x] = (Q_3 x)(Q_4 z)(F[x] \wedge H[z])$

where Q_1, Q_2, Q_3, and Q_4 are either $\forall$ or $\exists$, and z does not appear in $F[x]$. Of course, if $Q_1 = Q_2 = \exists$ and $Q_3 = Q_4 = \forall$, then we do not have to rename x's in $(Q_2 x)H[x]$ or $(Q_4 x)H[x]$. We can use (3.3) directly.

Using laws (2.1)–(2.10) and laws (3.1)–(3.4), we can always transform a given formula into prenex normal form. The following is an outline of the transforming procedure.

Transforming Formulas into Prenex Normal Form

Step 1 Use the laws

$$F \leftrightarrow G = (F \rightarrow G) \wedge (G \rightarrow F) \tag{2.1}$$

$$F \rightarrow G = \sim F \vee G \tag{2.2}$$

to eliminate the logical connectives $\leftrightarrow$ and $\rightarrow$.

Step 2 Repeatedly use the law

$$\sim(\sim F) = F \tag{2.9}$$

De Morgan's laws

$$\sim(F \vee G) = \sim F \wedge \sim G \tag{2.10a}$$

$$\sim(F \wedge G) = \sim F \vee \sim G \tag{2.10b}$$

and the laws

$$\sim((\forall x)F[x]) = (\exists x)(\sim F[x]) \tag{3.2a}$$

$$\sim((\exists x)F[x]) = (\forall x)(\sim F[x]) \tag{3.2b}$$

to bring the negation signs immediately before atoms.

Step 3 Rename bound variables if necessary.

Step 4 Use the laws

$$(Qx)F[x] \vee G = (Qx)(F[x] \vee G) \tag{3.1a}$$

$$(Qx)F[x] \wedge G = (Qx)(F[x] \wedge G) \tag{3.1b}$$

$$(\forall x)F[x] \wedge (\forall x)H[x] = (\forall x)(F[x] \wedge H[x]) \tag{3.3a}$$

$$(\exists x)F[x] \vee (\exists x)H[x] = (\exists x)(F[x] \vee H[x]) \tag{3.3b}$$

$$(Q_1 x)F[x] \vee (Q_2 x)H[x] = (Q_1 x)(Q_2 z)(F[x] \vee H[z]) \tag{3.4a}$$

$$(Q_3 x)F[x] \wedge (Q_4 x)H[x] = (Q_3 x)(Q_4 z)(F[x] \wedge H[z]) \tag{3.4b}$$

to move the quantifiers to the left of the entire formula to obtain a prenex normal form.

Example 3.11

Transform the formula $(\forall x)P(x) \rightarrow (\exists x)Q(x)$ into prenex normal form.

$$\begin{aligned}(\forall x)P(x) \rightarrow (\exists x)Q(x) &= \sim((\forall x)P(x)) \vee (\exists x)Q(x) && \text{by (2.2)}\\ &= (\exists x)(\sim P(x)) \vee (\exists x)Q(x) && \text{by (3.2a)}\\ &= (\exists x)(\sim P(x) \vee Q(x)) && \text{by (3.3b).}\end{aligned}$$

Therefore, a prenex normal form of $(\forall x)P(x) \rightarrow (\exists x)Q(x)$ is $(\exists x)(\sim P(x) \vee Q(x))$.

Example 3.12

Obtain a prenex normal form for the formula

$$(\forall x)(\forall y)((\exists z)(P(x,z) \wedge P(y,z)) \rightarrow (\exists u)Q(x,y,u)).$$

$$
\begin{aligned}
&(\forall x)(\forall y)[(\exists z)(P(x,z) \wedge P(y,z)) \rightarrow (\exists u)Q(x,y,u)] \\
&\quad = (\forall x)(\forall y)[\sim((\exists z)(P(x,z) \wedge P(y,z))) && \text{by (2.2)} \\
&\qquad \vee (\exists u)Q(x,y,u)] \\
&\quad = (\forall x)(\forall y)[(\forall z)(\sim P(x,z) \vee \sim P(y,z)) && \text{by (3.2b) and (2.10b)} \\
&\qquad \vee (\exists u)Q(x,y,u)] \\
&\quad = (\forall x)(\forall y)(\forall z)(\exists u)(\sim P(x,z) && \text{using (3.1a), move the} \\
&\qquad \vee \sim P(y,z) \vee Q(x,y,u)) && \text{quantifiers to the left.}
\end{aligned}
$$

Therefore, we obtain the last formula as a prenex normal form of the first formula.

3.4 APPLICATIONS OF THE FIRST-ORDER LOGIC

In this section we shall give a few examples to illustrate some applications of the first-order logic to problem solving. As in the propositional logic, the usual approach is first to symbolize problems by formulas and then to prove that the formulas are valid or inconsistent.

Example 3.13

Consider Example 3.3. There are two axioms:

A_1: $(\forall x)(MAN(x) \rightarrow MORTAL(x))$.
A_2: MAN(Confucius).

From A_1 and A_2, show that Confucius is mortal. That is, show that $MORTAL$(Confucius) is a logical consequence of A_1 and A_2.

We have

$A_1 \wedge A_2$: $(\forall x)(MAN(x) \rightarrow MORTAL(x)) \wedge MAN$(Confucius).

If $A_1 \wedge A_2$ is true in an interpretation I, then both A_1 and A_2 are true in I. Since $(MAN(x) \rightarrow MORTAL(x))$ is true for all x, when x is replaced by "Confucius," $(MAN$(Confucius) $\rightarrow MORTAL$(Confucius)) is true in I. That is, $\sim MAN$(Confucius) $\vee$ $MORTAL$(Confucius) is true in I. However, $\sim MAN$(Confucius) is false in I since MAN(Confucius) is true in I. Hence, $MORTAL$(Confucius) must be true in I. We have therefore shown that $MORTAL$(Confucius) is true in I whenever $(A_1 \wedge A_2)$ is true in I. By definition, $MORTAL$(Confucius) is a logical consequence of A_1 and A_2.

Example 3.14

No used-car dealer buys a used car for his family. Some people who buy used cars for their families are absolutely dishonest. Conclude that some absolutely dishonest people are not used-car dealers.

We let $U(x)$, $B(x)$, and $D(x)$ denote "x is a used-car dealer," "x buys

a used car for his family," and "x is absolutely dishonest," respectively. Then we have

A_1: $(\forall x)(U(x) \rightarrow \sim B(x))$

A_2: $(\exists x)(B(x) \wedge D(x))$.

We have to show that

A_3: $(\exists x)(D(x) \wedge \sim U(x))$

is a logical consequence of A_1 and A_2.

Assume that A_1 and A_2 are true in an interpretation I over a domain D. Since A_2 is true in I, there is an x in D, say a, such that $B(a) \wedge D(a)$ is true in I. Therefore $B(a)$ is true in I, that is, $\sim B(a)$ is false in I. A_1 can be written in the following form:

A_1: $(\forall x)(\sim U(x) \vee \sim B(x))$.

Since A_1 is true in I and $\sim B(a)$ is false in I, $\sim U(a)$ must be true in I. However, since $B(a) \wedge D(a)$ is true in I, $D(a)$ is true in I. Therefore, $D(a) \wedge \sim U(a)$ is true in I. Thus, A_3, that is, $(\exists x)(D(x) \wedge \sim U(x))$, is true in I. Hence, A_3 is a logical consequence of A_1 and A_2.

Example 3.15

Some patients like all doctors. No patient likes any quack. Therefore, no doctor is a quack. Denote

$P(x)$: x is a patient,

$D(x)$: x is a doctor,

$Q(x)$: x is a quack,

$L(x, y)$: x likes y.

Then the facts and the conclusion may be symbolized as follows:

F_1: $(\exists x)(P(x) \wedge (\forall y)(D(y) \rightarrow L(x, y)))$

F_2: $(\forall x)(P(x) \rightarrow (\forall y)(Q(y) \rightarrow \sim L(x, y)))$

G: $(\forall x)(D(x) \rightarrow \sim Q(x))$.

We now show that G is a logical consequence of F_1 and F_2. Let I be an arbitrary interpretation over a domain D. Suppose F_1 and F_2 are true in I. Since F_1, that is, $(\exists x)(P(x) \wedge (\forall y)(D(y) \rightarrow L(x, y)))$ is true in I, there is some element, say e, in D such that $(P(e) \wedge (\forall y)(D(y) \rightarrow L(e, y)))$ is true in I. That is, both $P(e)$ and $(\forall y)(D(y) \rightarrow L(e, y))$ are true in I. On the other hand, since $(P(x) \rightarrow (\forall y)(Q(y) \rightarrow \sim L(x, y)))$ is true in I for all elements of x in D,

certainly $(P(e) \rightarrow (\forall y)(Q(y) \rightarrow \sim L(e,y)))$ is true in I. Since $P(e)$ is true in I, $(\forall y)(Q(y) \rightarrow \sim L(e,y))$ must be true in I. Hence we know that for every element y in D, both $(D(y) \rightarrow L(e,y))$ and $(Q(y) \rightarrow \sim L(e,y))$ are true in I. If $D(y)$ is false in I, then $(D(y) \rightarrow \sim Q(y))$ is true in I. If $D(y)$ is true in I, then $L(e,y)$ must be true in I, since $(D(y) \rightarrow L(e,y))$ is true in I. Therefore, $Q(y)$ must be false in I, since $(Q(y) \rightarrow \sim L(e,y))$ is true in I. Consequently, $(D(y) \rightarrow \sim Q(y))$ is true in I. Therefore, $(D(y) \rightarrow \sim Q(y))$ is true for every y in D; i.e., $(\forall y)(D(y) \rightarrow \sim Q(y))$ is true in I. Hence, we have shown that if F_1 and F_2 are true in I, $(\forall y)(D(y) \rightarrow \sim Q(y))$ is true in I. This shows that G is a logical consequence of F_1 and F_2.

In the above examples, we have shown that the conclusions follow from the given facts. A demonstration that a conclusion follows from axioms is called a *proof.* A procedure for finding proofs is called a *proof procedure.* In Chapters 4–9, we shall give proof procedures that *mechanically* use inference rules to find a proof. These proof procedures are very efficient. The reader will find out that the above three examples are extremely easy to prove if these proof procedures are used.

REFERENCES

Hilbert, D., and W. Ackermann (1950): "Principles of Mathematical Logic," Chelsea, New York.

Kleene, S. C. (1967): "Mathematical Logic," Wiley, New York.

Korfhage, R. R. (1966): "Logic and Algorithms," Wiley, New York.

Mendelson, E., (1964): "Introduction to Mathematical Logic," Van Nostrand-Reinhold, Princeton, New Jersey.

Stoll, R. R. (1961): "Sets, Logic and Axiomatic Theories," Freeman, San Francisco.

Whitehead, A., and B. Russell (1927): "Principia Mathematica," Cambridge Univ. Press, London and New York.

EXERCISES

Section 3.1

1. Let $P(x)$ and $Q(x)$ represent "x is a rational number" and "x is a real number," respectively. Symbolize the following sentences:

   1.1 Every rational number is a real number.
   1.2 Some real numbers are rational numbers.
   1.3 Not every real number is a rational number.

2. Let $C(x)$ mean "x is a used-car dealer," and $H(x)$ mean "x is honest." Translate each of the following into English:

   2.1 $(\exists x)C(x)$

2.2 $(\exists x)H(x)$

2.3 $(\forall x)(C(x) \rightarrow \sim H(x))$

2.4 $(\exists x)(C(x) \wedge H(x))$

2.5 $(\exists x)(H(x) \rightarrow C(x))$.

3. Let $P(x)$, $L(x)$, $R(x,y,z)$, and $E(x,y)$ represent "x is a point," "x is a line," "z passes through x and y," and "$x = y$," respectively. Translate the following:

   For every two points, there is one and only one line passing through both points.

4. An Abelian group is a set A with a binary operator $+$ that has certain properties. Let $P(x,y,z)$ and $E(x,y)$ represent $x+y=z$ and $x=y$, respectively. Express the following axioms for Abelian groups symbolically.

   (a) For every x and y in A, there exists a z in A such that $x+y=z$ (closure).
   (b) If $x+y=z$ and $x+y=w$, then $z=w$ (uniqueness).
   (c) $(x+y)+z=x+(y+z)$ (associativity).
   (d) $x+y=y+x$ (symmetry).
   (e) For every x and y in A, there exists a z such that $x+z=y$ (right solution).

Section 3.2

5. For the following interpretation ($D = \{a,b\}$),

$P(a,a)$	$P(a,b)$	$P(b,a)$	$P(b,b)$
T	***F***	***F***	***T***

   determine the truth value of the following formulas:

   (a) $(\forall x)(\exists y)\,P(x,y)$ (b) $(\forall x)(\forall y)\,P(x,y)$

   (c) $(\exists x)(\forall y)\,P(x,y)$ (d) $(\exists y) \sim P(a,y)$

   (e) $(\forall x)(\forall y)(P(x,y) \rightarrow P(y,x))$ (f) $(\forall x)\,P(x,x)$.

6. Consider the following formula:

   A: $(\exists x)\,P(x) \rightarrow (\forall x)\,P(x)$.

   a. Prove that this formula is always true if the domain D contains only one element.
   b. Let $D = \{a,b\}$. Find an interpretation over D in which A is evaluated to ***F***.

7. Consider the following interpretation:

   Domain: $D = \{1,2\}$.

   Assignment of constants a and b:

a	b
1	2

   Assignment for function f:

$f(1)$	$f(2)$
2	1

   Assignment for predicate P:

$P(1,1)$	$P(1,2)$	$P(2,1)$	$P(2,2)$
T	***T***	***F***	***F***

   Evaluate the truth value of the following formulas in the above interpretation:

   (1) $P(a, f(a)) \wedge P(b, f(b))$

   (2) $(\forall x)(\exists y)\, P(y, x)$

   (3) $(\forall x)(\forall y)(P(x, y) \rightarrow P(f(x), f(y)))$.

8. Let F_1 and F_2 be as follows:

   F_1: $(\forall x)(P(x) \rightarrow Q(x))$

   F_2: $\sim Q(a)$.

   Prove that $\sim P(a)$ is a logical consequence of F_1 and F_2.

Section 3.3

9. Transform the following formulas into prenex normal forms:

   (1) $(\forall x)(P(x) \rightarrow (\exists y) Q(x, y))$

   (2) $(\exists x)(\sim ((\exists y) P(x, y)) \rightarrow ((\exists z) Q(z) \rightarrow R(x)))$

   (3) $(\forall x)(\forall y)((\exists z) P(x, y, z) \wedge ((\exists u) Q(x, u) \rightarrow (\exists v) Q(y, v)))$.

Section 3.4

10. Consider the following statements:

F_1: Every student is honest.
F_2: John is not honest.

From the above statements, prove that John is not a student.

11. Consider the following premises:

(1) Every athelete is strong.
(2) Everyone who is both strong and intelligent will succeed in his career.
(3) Peter is an athelete.
(4) Peter is intelligent.

Try to conclude that Peter will succeed in his career.

12. Assume that St. Francis is loved by everyone who loves someone. Also assume that no one loves nobody. Deduce that St. Francis is loved by everyone.

Chapter 4

Herbrand's Theorem

4.1 INTRODUCTION

In the previous chapters, we have discussed how to solve problems by proving theorems. In this and the following chapters, we shall consider proof procedures. Actually, finding a general decision procedure to verify the validity (inconsistency) of a formula was considered long ago. It was first tried by Leibniz (1646–1716) and further revived by Peano around the turn of the century and by Hilbert's school in the 1920s. It was not until Church [1936] and Turing [1936] that this was proved impossible. Church and Turing independently showed that there is no general decision procedure to check the validity of formulas of the first-order logic. However, there are proof procedures which can verify that a formula is valid if indeed it is valid. For invalid formulas, these procedures in general will never terminate. In view of the result of Church and Turing, this is the best we can expect to get from a proof procedure.

A very important approach to mechanical theorem proving was given by Herbrand in 1930. By definition, a valid formula is a formula that is true under all interpretations. Herbrand developed an algorithm to find an interpretation that can falsify a given formula. However, if the given formula is indeed valid, no such interpretation can exist and his algorithm will halt after a finite number of trials. Herbrand's method is the basis for most modern automatic proof procedures.

Gilmore [1960] was one of the first persons to implement Herbrand's procedure on a computer. Since a formula is valid if and only if its negation is inconsistent, his program was designed to detect the inconsistency of the negation of the given formula. During the execution of his program, propositional formulas are generated that are periodically tested for inconsistency. If the negation of the given formula is inconsistent, his program will eventually detect this fact. Gilmore's program managed to prove a few simple formulas, but encountered decisive difficulties with most other formulas of the first order logic. Careful studies of his program revealed that his method of testing the inconsistency of a propositional formula is inefficient. Gilmore's method was improved by Davis and Putnam [1960] a few months after his result was published. However, their improvement was still not enough. Many valid formulas of the first-order logic still could not be proved by computers in a reasonable amount of time.

A major breakthrough was made by Robinson [1965a], who introduced the so-called resolution principle. Resolution proof procedure is much more efficient than any earlier procedure. Since the introduction of the resolution principle, several refinements have been suggested in attempts to further increase its efficiency. Some of these refinements are semantic resolution [Slagle, 1967; Meltzer, 1966; Robinson, 1965b; Kowalski and Hayes, 1969], lock resolution [Boyer, 1971], linear resolution [Loveland, 1970a, b; Luckham, 1970; Anderson and Bledsoe, 1970; Yates *et al.*, 1970; Reiter, 1971; Kowalski and Kuehner, 1970], unit resolution [Wos *et al.*, 1964; Chang, 1970a], and set-of-support strategy [Wos *et al.*, 1965]. In this chapter, we shall first prove Herbrand's theorem. The resolution principle and some refinements of the resolution principle will be discussed in subsequent chapters.

4.2 SKOLEM STANDARD FORMS

Herbrand's and the resolution proof procedures that are to be discussed later in this book are actually refutation procedures. That is, instead of proving a formula valid, they prove that the negation of the formula is inconsistent. This is just a matter of convenience. There is no loss of generality in using refutation procedures. Furthermore, these refutation procedures are applied to a "standard form" of a formula. This standard form was introduced by Davis and Putnam [1960], and will be used throughout this book.

Essentially what Davis and Putnam did was to exploit the following ideas:

1. A formula of the first-order logic can be transformed into prenex normal form where the matrix contains no quantifiers and the prefix is a sequence of quantifiers.

2. The matrix, since it does not contain quantifiers, can be transformed into a conjunctive normal form.

3. Without affecting the inconsistency property, the existential quantifiers in the prefix can be eliminated by using Skolem functions.

In Chapter 3, we have already discussed how to transform a formula into a prenex normal form. By the techniques given in Chapter 2, we also know how to transform the matrix into a conjunctive normal form. We now discuss how to eliminate the existential quantifiers.

Let a formula F be already in a prenex normal form $(Q_1x_1)\cdots(Q_nx_n)M$, where M is in a *conjunctive* normal form. Suppose Q_r is an existential quantifier in the prefix $(Q_1x_1)\cdots(Q_nx_n)$, $1 \leqslant r \leqslant n$. If no universal quantifier appears before Q_r, we choose a new constant c different from other constants occurring in M, replace all x_r appearing in M by c, and delete (Q_rx_r) from the prefix. If $Q_{s_1}, \ldots, Q_{s_m}$ are all the universal quantifiers appearing before Q_r, $1 \leqslant s_1 < s_2 \cdots < s_m < r$, we choose a new m-place function symbol f different from other function symbols, replace all x_r in M by $f(x_{s_1}, x_{s_2}, \ldots, x_{s_m})$, and delete (Q_rx_r) from the prefix. After the above process is applied to all the existential quantifiers in the prefix, the last formula we obtain is a *Skolem standard form* (*standard form* for short) of the formula F. The constants and functions used to replace the existential variables are called *Skolem* functions.

Example 4.1

Obtain a standard form of the formula

$$(\exists x)(\forall y)(\forall z)(\exists u)(\forall v)(\exists w)P(x, y, z, u, v, w).$$

In the above formula, $(\exists x)$ is preceded by no universal quantifiers, $(\exists u)$ is preceded by $(\forall y)$ and $(\forall z)$, and $(\exists w)$ by $(\forall y)$, $(\forall z)$ and $(\forall v)$. Therefore, we replace the existential variable x by a constant a, u by a two-place function $f(y, z)$, and w by a three-place function $g(y, z, v)$. Thus, we obtain the following standard form of the formula:

$$(\forall y)(\forall z)(\forall v)P(a, y, z, f(y, z), v, g(y, z, v)).$$

Example 4.2

Obtain a standard form of the formula

$$(\forall x)(\exists y)(\exists z)((\sim P(x, y) \wedge Q(x, z)) \vee R(x, y, z)).$$

First, the matrix is transformed into a conjunctive normal form:

$$(\forall x)(\exists y)(\exists z)((\sim P(x, y) \vee R(x, y, z)) \wedge (Q(x, z) \vee R(x, y, z))).$$

Then, since $(\exists y)$ and $(\exists z)$ are both preceded by $(\forall x)$, the existential variables y and z are replaced, respectively, by one-place functions $f(x)$ and $g(x)$.

Thus, we obtain the following standard form of the formula:

$$(\forall x)\big((\sim P(x, f(x)) \vee R(x, f(x), g(x))) \wedge (Q(x, g(x)) \vee R(x, f(x), g(x)))\big).$$

Definition A *clause* is a finite disjunction of zero or more literals.

When it is convenient, we shall regard a set of literals as synonymous with a clause. For example, $P \vee Q \vee \sim R = \{P, Q, \sim R\}$. A clause consisting of r literals is called an *r-literal* clause. A one-literal clause is called a *unit* clause. When a clause contains no literal, we call it the *empty* clause. Since the empty clause has no literal that can be satisfied by an interpretation, the empty clause is always false. We customarily denote the empty clause by $\square$.

The disjunctions $\sim P(x,f(x)) \vee R(x,f(x),g(x))$ and $Q(x,g(x)) \vee R(x,f(x), g(x))$ in the standard form in Example 4.2 are clauses. *A set S of clauses is regarded as a conjunction of all clauses in S, where every variable in S is considered governed by a universal quantifier.* By this convention, a standard form can be simply represented by a set of clauses. For example, the standard form of Example 4.2 can be represented by the set

$$\{\sim P(x,f(x)) \vee R(x,f(x),g(x)), Q(x,g(x)) \vee R(x,f(x),g(x))\}.$$

As we said in the beginning of this section, we can eliminate existential quantifiers without affecting the inconsistency property. This is shown in the following theorem.

Theorem 4.1 Let S be a set of clauses that represents a standard form of a formula F. Then F is inconsistent if and only if S is inconsistent.

Proof Without loss of generality, we may assume that F is in a prenex normal form, that is, $F = (Q_1x_1)\cdots(Q_nx_n)M[x_1, \ldots, x_n]$. (We use $M[x_1, \ldots, x_n]$ to indicate that the matrix M contains variables $x_1, \ldots, x_n$.) Let Q_r be the first existential quantifier. Let $F_1 = (\forall x_1)\cdots(\forall x_{r-1})(Q_{r+1}x_{r+1})\cdots(Q_nx_n)M[x_1, \ldots, x_{r-1}, f(x_1, \ldots, x_{r-1}), x_{r+1}, \ldots, x_n]$, where f is a Skolem function corresponding to x_r, $1 \leqslant r \leqslant n$. We want to show that F is inconsistent if and only if F_1 is inconsistent. Suppose F is inconsistent. If F_1 is consistent, then there is an interpretation I such that F_1 is true in I. That is, for all $x_1, \ldots, x_{r-1}$, there exists at least one element, which is $f(x_1, \ldots, x_{r-1})$, such that $(Q_{r+1}x_{r+1})\cdots(Q_nx_n)M[x_1, \ldots, x_{r-1}, f(x_1, \ldots, x_{r-1}), x_{r+1}, \ldots, x_n]$ is true in I. Thus, F is true in I, which contradicts the assumption that F is inconsistent. Therefore F_1 must be inconsistent. On the other hand, suppose that F_1 is inconsistent. If F is consistent, then there is an interpretation I over a domain D such that F is true in I. That is, for all $x_1, \ldots, x_{r-1}$, there exists an element x_r such that $(Q_{r+1}x_{r+1})\cdots(Q_nx_n)M[x_1, \ldots, x_{r-1}, x_r, x_{r+1}, \ldots, x_n]$ is true in I. Extend the interpretation I to include a function f that maps $(x_1, \ldots, x_{r-1})$ to x_r for all $x_1, \ldots, x_{r-1}$ in D, that is, $f(x_1, \ldots, x_{r-1}) = x_r$. Let

this extension of I be denoted by I'. Then, clearly, for all $x_1, \ldots, x_{r-1}$, $(Q_{r+1}x_{r+1})\cdots(Q_nx_n)M[x_1, \ldots, x_{r-1}, f(x_1, \ldots, x_{r-1}), x_{r+1}, \ldots, x_n]$ is true in I'. That is, F_1 is true in I', which contradicts the assumption that F_1 is inconsistent. Therefore F must be inconsistent. Assume there are m existential quantifiers in F. Let $F_0 = F$. Let F_k be obtained from F_{k-1} by replacing the first existential quantifier in F_{k-1} by a Skolem function, $k = 1, \ldots, m$. Clearly, $S = F_m$. Using the same arguments given above, we can show that F_{k-1} is inconsistent if and only if F_k is inconsistent for $k = 1, \ldots, m$. Therefore, we conclude that F is inconsistent if and only if S is inconsistent. Q.E.D.

Let S be a standard form of a formula F. If F is inconsistent, then by Theorem 4.1, $F = S$. If F is not inconsistent, we note that, in general, F is not equivalent to S. For example, let $F \triangleq (\exists x)P(x)$ and $S \triangleq P(a)$. Clearly, S is a standard form of F. However, let I be the interpretation defined below:

Domain: $D = \{1, 2\}$.

Assignment for a:

a
1

Assignment for P:

$P(1)$	$P(2)$
F	T

Then, clearly, F is true in I, but S is false in I. Therefore $F \neq S$.

It is noted that a formula may have more than one standard form. For the sake of simplicity, when we transform a formula F into a standard form S, we should replace existential quantifiers by Skolem functions that are as simple as possible. That is, we should use Skolem functions with the least number of arguments. This means that we should move existential quantifiers to the left as far as possible. Furthermore, if we have $F = F_1 \wedge \cdots \wedge F_n$, we can separately obtain a set S_i of clauses, where each S_i represents a standard form of F_i, $i = 1, \ldots, n$. Then, let $S = S_1 \cup \cdots \cup S_n$. By arguments similar to those given in the proof of Theorem 4.1, it is not difficult to see that F is inconsistent if and only if S is inconsistent.

Example 4.3

In this example, we shall show how to express the following theorem in a standard form:

If $x \cdot x = e$ for all x in group G, where $\cdot$ is a binary operator and e is the identity in G, then G is commutative.

We shall first symbolize the above theorem together with some basic axioms in group theory and then represent the negation of the above theorem by a set of clauses.

We know that group G satisfies the following four axioms:

A_1: $x, y \in G$ implies that $x \cdot y \in G$ (closure property);

A_2: $x, y, z \in G$ implies that $x \cdot (y \cdot z) = (x \cdot y) \cdot z$ (associativity property);

A_3: $x \cdot e = e \cdot x = x$ for all $x \in G$ (identity property);

A_4: for every $x \in G$ there exists an element $x^{-1} \in G$ such that $x \cdot x^{-1} = x^{-1} \cdot x = e$ (inverse property).

Let $P(x, y, z)$ stand for $x \cdot y = z$ and $i(x)$ for x^{-1}. Then the above axioms can be represented by

A_1': $(\forall x)(\forall y)(\exists z)\, P(x, y, z)$

A_2': $(\forall x)(\forall y)(\forall z)(\forall u)(\forall v)(\forall w)(P(x, y, u) \wedge P(y, z, v) \wedge P(u, z, w) \rightarrow P(x, v, w))$
$\wedge (\forall x)(\forall y)(\forall z)(\forall u)(\forall v)(\forall w)(P(x, y, u) \wedge P(y, z, v) \wedge P(x, v, w) \rightarrow P(u, z, w))$

A_3': $(\forall x)\, P(x, e, x) \wedge (\forall x)\, P(e, x, x)$

A_4': $(\forall x)\, P(x, i(x), e) \wedge (\forall x)\, P(i(x), x, e)$.

The conclusion of the theorem is

B: If $x \cdot x = e$ for all $x \in G$, then G is commutative, i.e., $u \cdot v = v \cdot u$ for all $u, v \in G$.

B can be represented by

B': $(\forall x)\, P(x, x, e) \rightarrow ((\forall u)(\forall v)(\forall w)(P(u, v, w) \rightarrow P(v, u, w)))$.

Now, the entire theorem is represented by the formula $F = A_1' \wedge \cdots \wedge A_4' \rightarrow B'$. Thus, $\sim F = A_1' \wedge A_2' \wedge A_3' \wedge A_4' \wedge \sim B'$. To obtain a set S of clauses for $\sim F$, we first obtain a set S_i of clauses for each axiom A_i', $i = 1, 2, 3, 4$ as follows.

S_1: $\{P(x, y, f(x, y))\}$

S_2: $\{\sim P(x, y, u) \vee \sim P(y, z, v) \vee \sim P(u, z, w) \vee P(x, v, w),$
$\sim P(x, y, u) \vee \sim P(y, z, v) \vee \sim P(x, v, w) \vee P(u, z, w)\}$

S_3: $\{P(x, e, x), P(e, x, x)\}$.

S_4': $\{P(x, i(x), e), P(i(x), x, e)\}$.

Since

$$\begin{aligned}
\sim B' &= \sim\big((\forall x)P(x,x,e) \rightarrow ((\forall u)(\forall v)(\forall w)(P(u,v,w) \rightarrow P(v,u,w)))\big) \\
&= \sim\big(\sim(\forall x)P(x,x,e) \vee ((\forall u)(\forall v)(\forall w)(\sim P(u,v,w) \vee P(v,u,w)))\big) \\
&= (\forall x)P(x,x,e) \wedge \sim((\forall u)(\forall v)(\forall w)(\sim P(u,v,w) \vee P(v,u,w))) \\
&= (\forall x)P(x,x,e) \wedge (\exists u)(\exists v)(\exists w)(P(u,v,w) \wedge \sim P(v,u,w)),
\end{aligned}$$

a set of clauses for $\sim B'$ is given below.

T: $\{P(x,x,e),$
$P(a,b,c),$
$\sim P(b,a,c)\}.$

Thus, the set $S = S_1 \cup S_2 \cup S_3 \cup S_4 \cup T$ is the set consisting of the following clauses:

(1) $P(x,y,f(x,y))$

(2) $\sim P(x,y,u) \vee \sim P(y,z,v) \vee \sim P(u,z,w) \vee P(x,v,w)$

(3) $\sim P(x,y,u) \vee \sim P(y,z,v) \vee \sim P(x,v,w) \vee P(u,z,w)$

(4) $P(x,e,x)$

(5) $P(e,x,x)$

(6) $P(x,i(x),e)$

(7) $P(i(x),x,e)$

(8) $P(x,x,e)$

(9) $P(a,b,c)$

(10) $\sim P(b,a,c).$

In Example 4.3, we have shown how to obtain a set S of clauses for the formula $\sim F$. By Theorems 2.2 and 4.1, we know that F is valid if and only if S is inconsistent. As we said at the beginning of this section, we shall use refutation procedures to prove theorems. Thus, from here on, we shall assume that the input to a refutation procedure is always a set of clauses, such as the set S obtained in the above example. Furthermore, we shall use "unsatisfiable" ("satisfiable"), instead of "inconsistent" ("consistent"), for sets of clauses.

4.3 THE HERBRAND UNIVERSE OF A SET OF CLAUSES

By definition, a set S of clauses is unsatisfiable if and only if it is false under all interpretations over all domains. Since it is inconvenient and impossible

to consider all interpretations over all domains, it would be nice if we could fix on one special domain H such that S is unsatisfiable if and only if S is false under all the interpretations over this domain. Fortunately, there does exist such a domain, which is called the *Herbrand universe* of S, defined as follows.

Definition Let H_0 be the set of constants appearing in S. If no constant appears in S, then H_0 is to consist of a single constant, say $H_0 = \{a\}$. For $i = 0, 1, 2, \ldots$, let H_{i+1} be the union of H_i and the set of all terms of the form $f^n(t_1, \ldots, t_n)$ for all n-place functions f^n occurring in S, where t_j, $j = 1, \ldots, n$, are members of the set H_i. Then each H_i is called the *i-level constant set* of S, and H_∞, or $\lim_{i\to\infty} H_i$, is called the *Herbrand universe* of S.

Example 4.4

Let $S = \{P(a), \sim P(x) \vee P(f(x))\}$. Then

$$H_0 = \{a\}$$

$$H_1 = \{a, f(a)\}$$

$$H_2 = \{a, f(a), f(f(a))\}$$

$$\vdots$$

$$H_\infty = \{a, f(a), f(f(a)), f(f(f(a))), \ldots\}.$$

Example 4.5

Let $S = \{P(x) \vee Q(x), R(z), T(y) \vee \sim W(y)\}$. Since there is no constant in S, we let $H_0 = \{a\}$. There is no function symbol in S, hence $H = H_0 = H_1 = \cdots = \{a\}$.

Example 4.6

Let $S = \{P(f(x), a, g(y), b)\}$. Then

$$H_0 = \{a, b\}$$

$$H_1 = \{a, b, f(a), f(b), g(a), g(b)\}$$

$$H_2 = \{a, b, f(a), f(b), g(a), g(b), f(f(a)), f(f(b)), f(g(a)), f(g(b)), g(f(a)), g(f(b)), g(g(a)), g(g(b))\}$$

$$\vdots$$

In the sequel, by expression we mean a term, a set of terms, an atom, a set of atoms, a literal, a clause, or a set of clauses. When no variable appears in an expression, we sometimes call the expression a *ground expression* to emphasize this fact. Thus we may use a ground term, a ground atom, a ground literal, and a ground clause to mean that no variable occurs

in the respective expressions. Furthermore, a *subexpression* of an expression E is an expression that occurs in E.

Definition Let S be a set of clauses. The set of ground atoms of the form $P^n(t_1, \ldots, t_n)$ for all n-place predicates P^n occurring in S, where $t_1, \ldots, t_n$ are elements of the Herbrand universe of S, is called the *atom set*, or the *Herbrand base* of S.

Definition A *ground instance* of a clause C of a set S of clauses is a clause obtained by replacing variables in C by members of the Herbrand universe of S.

Example 4.7

Let $S = \{P(x), Q(f(y)) \vee R(y)\}$. $C = P(x)$ is a clause in S and $H = \{a, f(a), f(f(a)), \ldots\}$ is the Herbrand universe of S. Then $P(a)$ and $P(f(f(a)))$ are both ground instances of C.

We now consider interpretations over the Herbrand universe. Let S be a set of clauses. As discussed in Chapter 3, an interpretation over the Herbrand universe of S is an assignment of constants, function symbols, and predicate symbols occurring in S. In the following, we shall define a special interpretation over the Herbrand universe of S, called the H-interpretation of S.

Definition Let S be a set of clauses; H, the Herbrand universe of S; and I, an interpretation of S over H. I is said to be an *H-interpretation* of S if it satisfies the following conditions:

1. I maps all constants in S to themselves.
2. Let f be an n-place function symbol and $h_1, \ldots, h_n$ be elements of H. In I, f is assigned a function that maps $(h_1, \ldots, h_n)$ (an element in H^n) to $f(h_1, \ldots, h_n)$ (an element in H).

There is no restriction on the assignment to each n-place predicate symbol in S. Let $A = \{A_1, A_2, \ldots, A_n, \ldots\}$ be the atom set of S. An H-interpretation I can be conveniently represented by a set

$$I = \{m_1, m_2, \ldots, m_n, \ldots\}$$

in which m_j is either A_j or $\sim A_j$ for $j = 1, 2, \ldots$. The meaning of this set is that if m_j is A_j, then A_j is assigned "true"; otherwise, A_j is assigned "false."

Example 4.8

Consider the set $S = \{P(x) \vee Q(x), R(f(y))\}$. The Herbrand universe H of S is $H = \{a, f(a), f(f(a)), \ldots\}$. There are three predicate symbols: P, Q, and R. Hence the atom set of S is

$$A = \{P(a), Q(a), R(a), P(f(a)), Q(f(a)), R(f(a)), \ldots\}.$$

Some H-interpretations for S are as follows:

$$I_1 = \{P(a), Q(a), R(a), P(f(a)), Q(f(a)), R(f(a)), \ldots\}$$

$$I_2 = \{\sim P(a), \sim Q(a), \sim R(a), \sim P(f(a)), \sim Q(f(a)), \sim R(f(a)), \ldots\}$$

$$I_3 = \{P(a), Q(a), \sim R(a), P(f(a)), Q(f(a)), \sim R(f(a)), \ldots\}.$$

An interpretation of a set S of clauses does not necessarily have to be defined over the Herbrand universe of S. Thus an interpretation may not be an H-interpretation. For example, let $S = \{P(x), Q(y, f(y, a))\}$. If the domain is $D = \{1, 2\}$, then the following is an interpretation of S.

$D = \{1, 2\}$.

a	$f(1,1)$	$f(1,2)$	$f(2,1)$	$f(2,2)$
2	1	2	2	1

$P(1)$	$P(2)$	$Q(1,1)$	$Q(1,2)$	$Q(2,1)$	$Q(2,2)$
T	F	F	T	F	T

For an interpretation such as the one defined above, we can define an H-interpretation I^* corresponding to I. We use the above example to illustrate this point. First, we find the atom set of S,

$$A = \{P(a), Q(a, a), P(f(a, a)), Q(a, f(a, a)), Q(f(a, a), a), Q(f(a, a), f(a, a)), \ldots\}.$$

Next, we evaluate each member of A by using the above table.

$$P(a) = P(2) = F$$

$$Q(a, a) = Q(2, 2) = T$$

$$P(f(a, a)) = P(f(2, 2)) = P(1) = T$$

$$Q(a, f(a, a)) = Q(2, f(2, 2)) = Q(2, 1) = F$$

$$Q(f(a, a), a) = Q(f(2, 2), 2) = Q(1, 2) = T$$

$$Q(f(a, a), f(a, a)) = Q(f(2, 2), f(2, 2)) = Q(1, 1) = F$$

$$\vdots$$

Therefore, the H-interpretation I^* corresponding to I is

$$I^* = \{\sim P(a), Q(a, a), P(f(a, a)), \sim Q(a, f(a, a)), Q(f(a, a), a), \sim Q(f(a, a), f(a, a)), \ldots\}.$$

In case there is no constant in S, the element a that is used to initiate the Herbrand universe of S can be mapped into any element of the domain

D. In this case, if there is more than one element in D, then there is more than one H-interpretation corresponding to I. For example, let $S = \{P(x), Q(y, f(y,z))\}$ and let an interpretation I for S be as follows:

$$D = \{1, 2\}$$

$f(1,1)$	$f(1,2)$	$f(2,1)$	$f(2,2)$
1	2	2	1

$P(1)$	$P(2)$	$Q(1,1)$	$Q(1,2)$	$Q(2,1)$	$Q(2,2)$
T	***F***	***F***	***T***	***F***	***T***

Then the two H-interpretations corresponding to I are

$$I^* = \{\sim P(a), Q(a,a), P(f(a,a)), \sim Q(a, f(a,a)), Q(f(a,a),a), \sim Q(f(a,a), f(a,a)), \ldots\} \quad \text{if} \quad a = 2,$$

$$I^* = \{P(a), \sim Q(a,a), P(f(a,a)), \sim Q(a, f(a,a)), \sim Q(f(a,a),a), \sim Q(f(a,a), f(a,a)), \ldots\} \quad \text{if} \quad a = 1.$$

We can formalize the concepts mentioned above as follows:

Definition Given an interpretation I over a domain D, an *H-interpretation I^* corresponding to I* is an H-interpretation that satisfies the following condition:

Let $h_1, \ldots, h_n$ be elements of H (the Herbrand universe of S). Let every h_i be mapped to some d_i in D. If $P(d_1, \ldots, d_n)$ is assigned $T(F)$ by I, then $P(h_1, \ldots, h_n)$ is also assigned $T(F)$ in I^*.

In fact, it is not hard to prove the following lemma. The proof is left as an exercise.

Lemma 4.1 If an interpretation I over some domain D satisfies a set S of clauses, then any one of the H-interpretations I^* corresponding to I also satisfies S.

Theorem 4.2 A set S of clauses is unsatisfiable if and only if S is false under all the H-interpretations of S.

Proof ($\Rightarrow$) The first half of the above theorem is obvious since, by definition, S is unsatisfiable if and only if S is false under all the interpretations over *any* domain.

($\Leftarrow$) To prove the second half of the above theorem, assume that S is false under all the H-interpretations of S. Suppose S is not unsatisfiable. Then there is an interpretation I over some domain D such that S is true

under I. Let I^* be an H-interpretation corresponding to I. According to Lemma 4.1, S is true under I^*. This contradicts the assumption that S is false under all the H-interpretations of S. Therefore, S must be unsatisfiable. Q.E.D.

Thus we have obtained the objective stated at the beginning of this section. That is, we need consider only interpretations over the Herbrand universe, or more strongly, H-interpretations, for checking whether or not a set of clauses is unsatisfiable. Because of the above theorem, from here on, whenever we mention an interpretation, we mean an H-interpretation.

Let $\varnothing$ denote the empty set. Each of the following observations is obvious. We shall leave their proofs to the reader.

1. A ground instance C' of a clause C is satisfied by an interpretation I if and only if there is a ground literal L' in C' such that L' is also in I, that is, $C' \cap I \neq \varnothing$.
2. A clause C is satisfied by an interpretation I if and only if every ground instance of C is satisfied by I.
3. A clause C is falsified by an interpretation I if and only if there is at least one ground instance C' of C such that C' is not satisfied by I.
4. A set S of clauses is unsatisfiable if and only if for every interpretation I, there is at least one ground instance C' of some clause C in S such that C' is not satisfied by I.

Example 4.9

1. Consider the clause $C = \sim P(x) \vee Q(f(x))$. Let I_1, I_2, and I_3 be defined as follows:

$$I_1 = \{\sim P(a), \sim Q(a), \sim P(f(a)), \sim Q(f(a)), \sim P(f(f(a))), \sim Q(f(f(a))), \ldots\}$$

$$I_2 = \{\ P(a),\ \ Q(a),\ \ P(f(a)),\ \ Q(f(a)),\ \ P(f(f(a))),\ \ Q(f(f(a))), \ldots\}$$

$$I_3 = \{\ P(a), \sim Q(a),\ \ P(f(a)), \sim Q(f(a)),\ \ P(f(f(a))), \sim Q(f(f(a))), \ldots\}.$$

The reader should be able to see that C is satisfied by I_1 and I_2, but falsified by I_3.

2. Consider $S = \{P(x), \sim P(a)\}$. There are only two H-interpretations:

$$I_1 = \{P(a)\} \quad \text{and} \quad I_2 = \{\sim P(a)\}.$$

S is falsified by both H-interpretations, and therefore is unsatisfiable.

4.4 SEMANTIC TREES

Having introduced the Herbrand universe, we now consider semantic trees [Robinson, 1968a; Kowalski and Hayes, 1969]. It will be seen in the

sequel that finding a proof for a set of clauses is equivalent to generating a semantic tree.

Definition If A is an atom, then the two literals A and $\sim A$ are said to be each other's *complement*, and the set $\{A, \sim A\}$ is called a *complementary pair*.

We note that a clause is a tautology if it contains a complementary pair. In the sequel, when we use "tautology," we shall specifically mean a clause that is a tautology.

Definition Given a set S of clauses, let A be the atom set of S. A *semantic tree* for S is a (downward) tree T, where each link is attached with a finite set of atoms or negations of atoms from A in such a way that:

i. For each node N, there are only finitely many immediate links $L_1, \ldots, L_n$ from N. Let Q_i be the conjunction of all the literals in the set attached to $L_i, i = 1, \ldots, n$. Then $Q_1 \vee Q_2 \vee \cdots \vee Q_n$ is a valid propositional formula.

ii. For each node N, let $I(N)$ be the union of all the sets attached to the links of the branch of T down to and including N. Then $I(N)$ does not contain any complementary pair.

Definition Let $A = \{A_1, A_2, \ldots, A_k, \ldots\}$ be the atom set of a set S of clauses. A semantic tree for S is said to be *complete* if and only if for every tip node N of the semantic tree, that is, a node that has no links sprouting from it, $I(N)$ contains either A_i or $\sim A_i$ for $i = 1, 2, \ldots$.

Example 4.10

Let $A = \{P, Q, R\}$ be the atom set of a set S of clauses. Then each one of the two trees in Fig. 4.1 is a complete semantic tree for S. (See p. 58.)

Example 4.11

Consider $S = \{P(x), P(a)\}$. The atom set of S is $\{P(a)\}$. A complete semantic tree for S is shown in Fig. 4.2. (See p. 58.)

Example 4.12

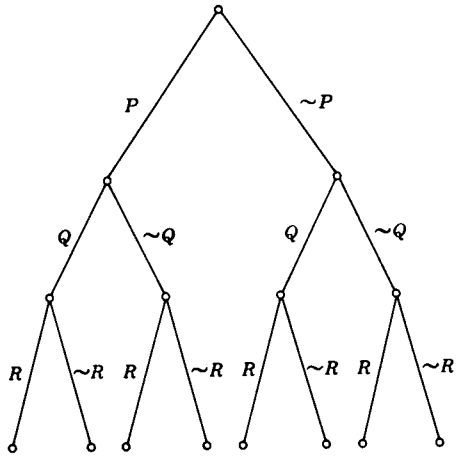
Consider $S = \{P(x), Q(f(x))\}$. The atom set of S is

$$\{P(a), Q(a), P(f(a)), Q(f(a)), P(f(f(a))), Q(f(f(a))), \ldots\}.$$

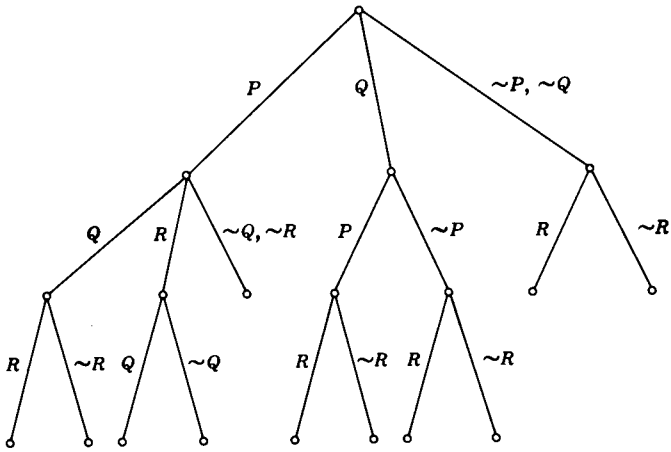
Fig. 4.3 shows a semantic tree for S.

It is noted that for each node N in a semantic tree for S, $I(N)$ is a subset of some interpretation for S. For this reason, $I(N)$ will be called a *partial interpretation* for S.

When the atom set A of a set S of clauses is infinite, any complete semantic tree for S will be infinite. As is easily seen, a complete semantic



(a)



(b)

Figure 4.1

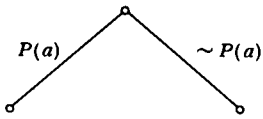


Figure 4.2

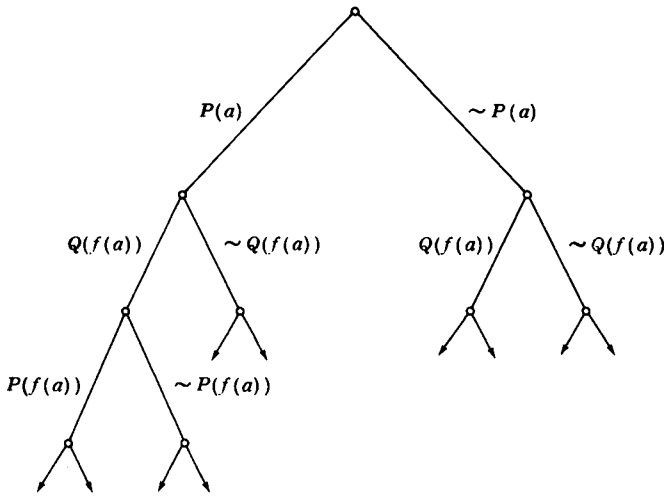


Figure 4.3

tree for S corresponds to an exhaustive survey of all possible interpretations for S. If S is unsatisfiable, then S fails to be true in each of these interpretations. Thus, we may stop expanding nodes from a node N if $I(N)$ falsifies S. This motivates the following definitions.

Definition A node N is a *failure node* if $I(N)$ falsifies some ground instance of a clause in S, but $I(N')$ does not falsify any ground instance of a clause in S for every ancestor node N' of N.

Definition A semantic tree T is said to be *closed* if and only if every branch of T terminates at a failure node.

Definition A node N of a closed semantic tree is called an *inference node* if all the immediate descendant nodes of N are failure nodes.

Example 4.13

Let $S = \{P, Q \vee R, \sim P \vee \sim Q, \sim P \vee \sim R\}$. The atom set of S is $A = \{P, Q, R\}$. Figure 4.4a is a complete semantic tree for S, while Fig. 4.4b is a closed semantic tree for S.

Example 4.14

Consider $S = \{P(x), \sim P(x) \vee Q(f(x)), \sim Q(f(a))\}$. The atom set of S is

$$A = \{P(a), Q(a), P(f(a)), Q(f(a)), \ldots\}.$$

Figure 4.5 shows a closed semantic tree for S.

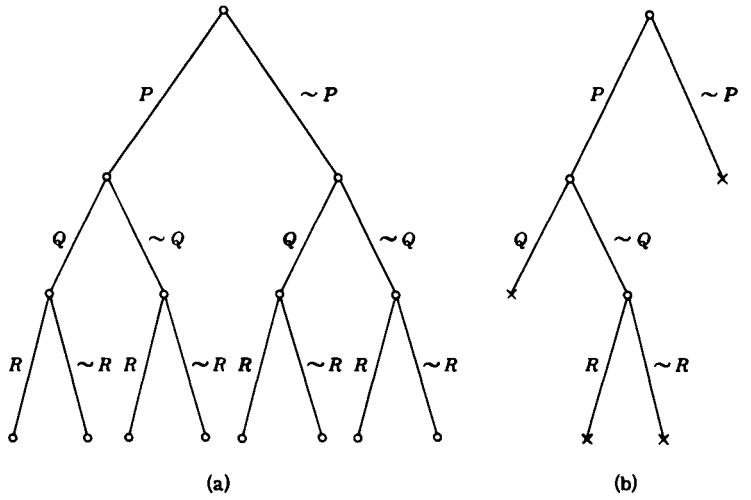


Figure 4.4

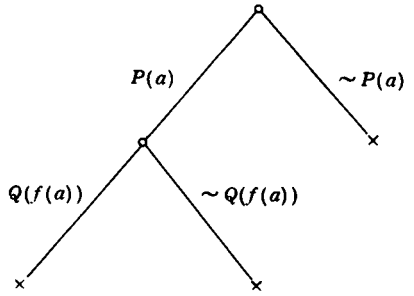


Figure 4.5

4.5 HERBRAND'S THEOREM

Herbrand's theorem is a very important theorem in symbolic logic; it is a base for most modern proof procedures in mechanical theorem proving. Herbrand's theorem is closely related to Theorem 4.2 given in Section 4.3. That is, to test whether a set S of clauses is unsatisfiable, we need consider only interpretations over the Herbrand universe of S. If S is false under all interpretations over the Herbrand universe of S, then we can conclude that S is unsatisfiable. Since there are usually many, possibly an infinite number, of these interpretations, we should organize them in some systematic way.

This can be done by using a semantic tree. We shall give two versions of Herbrand's theorem. The one stated most often in the literature is the second version; however, the first version is useful in this book.

Theorem 4.3 (Herbrand's Theorem, Version I) A set S of clauses is unsatisfiable if and only if corresponding to every complete semantic tree of S, there is a finite closed semantic tree.

Proof ($\Rightarrow$) Suppose S is unsatisfiable. Let T be a complete semantic tree for S. For each branch B of T, let I_B be the set of all literals attached to all links of the branch B. Then I_B is an interpretation for S. Since S is unsatisfiable, I_B must falsify a ground instance C' of a clause C in S. However, since C' is finite, there must exist a failure node N_B (which is a finite number of links away from the root node) on the branch B. Since every branch of T has a failure node, there is a closed semantic tree T' for S. Furthermore, since only a finite number of links are connected to each node of T', T' must be finite (that is, the number of nodes in T' is finite), for otherwise, by Konig's lemma [Knuth, 1968], we could find an infinite branch containing no failure node. Thus we complete the proof of the first half of the theorem.

($\Leftarrow$) Conversely, if corresponding to every complete semantic tree T for S there is a finite closed semantic tree, then every branch of T contains a failure node. This means that every interpretation falsifies S. Hence S is unsatisfiable. This completes the proof of the second half of the theorem.

Theorem 4.4 (Herbrand's Theorem, Version II). A set S of clauses is unsatisfiable if and only if there is a finite unsatisfiable set S' of ground instances of clauses of S.

Proof ($\Rightarrow$) Suppose S is unsatisfiable. Let T be a complete semantic tree for S. Then, by Herbrand's theorem (version I), there is a finite closed semantic tree T' corresponding to T. Let S' be the set of all the ground instances of clauses that are falsified at all the failure nodes of T'. S' is finite since there are a finite number of failure nodes in T'. Since S' is false in every interpretation of S', S' is unsatisfiable.

($\Leftarrow$) Suppose there is a finite unsatisfiable set S' of ground instances of clauses in S. Since every interpretation I of S contains an interpretation I' of S', if I' falsifies S', then I must also falsify S'. However, S' is falsified by every interpretation I'. Consequently, S' is falsified by every interpretation I of S. Therefore, S is falsified by every interpretation of S. Hence, S is unsatisfiable. Q.E.D.

Example 4.15

Let $S = \{P(x), \sim P(f(a))\}$. This set S is unsatisfiable. Hence, by Herbrand's theorem, there is a finite unsatisfiable set S' of ground instances of clauses in S. We have found that one of these sets is $S' = \{P(f(a)), \sim P(f(a))\}$.

Example 4.16

Let $S = \{\sim P(x) \vee Q(f(x), x), P(g(b)), \sim Q(y, z)\}$. This set S is unsatisfiable. One of the unsatisfiable sets of ground instances of clauses in S is

$$S' = \{\sim P(g(b)) \vee Q(f(g(b)), g(b)), P(g(b)), \sim Q(f(g(b)), g(b))\}.$$

Example 4.17

Let the set S consist of the following clauses:

$$\begin{aligned} S = \{&\sim P(x, y, u) \vee \sim P(y, z, v) \vee \sim P(x, v, w) \vee P(u, z, w), \\ &\sim P(x, y, u) \vee \sim P(y, z, v) \vee \sim P(u, z, w) \vee P(x, v, w), \\ &P(g(x, y), x, y), P(x, h(x, y), y), P(x, y, f(x, y)), \\ &\sim P(k(x), x, k(x))\}. \end{aligned}$$

This set S is also unsatisfiable. However, it is not very easy to find by hand a finite unsatisfiable set S' of ground instances of clauses in S. One way to find such a set S' is to generate a closed semantic tree T' for S. Then the set S' of all the ground instances falsified at all the failure nodes of T' is such a desired set. The following is a desired set S'. The reader may want to check that each ground clause in S' is a ground instance of some clause in S, and that S' is unsatisfiable.

$$\begin{aligned} S' = \{&P(a, h(a, a), a), \\ &\sim P(k(h(a, a)), h(a, a), k(h(a, a))), \\ &P(g(a, k(h(a, a))), a, k(h(a, a))), \\ &\sim P(g(a, k(h(a, a))), a, k(h(a, a))) \vee \sim P(a, h(a, a), a) \\ &\quad \vee \sim P(g(a, k(h(a, a))), a, k(h(a, a))) \vee P(k(h(a, a)), h(a, a), k(h(a, a)))\}. \end{aligned}$$

4.6 IMPLEMENTATION OF HERBRAND'S THEOREM

The second version of Herbrand's theorem suggests a refutation procedure. That is, given an unsatisfiable set S of clauses to prove, if there is a mechanical procedure that can successively generate sets $S_1', \ldots, S_n', \ldots$ of ground instances of clauses in S and can successively test $S_1', S_2', \ldots$ for unsatisfiability, then, as guaranteed by Herbrand's theorem, this procedure can detect a finite N such that S_N' is unsatisfiable.

Gilmore was one of the first men to implement the above idea [Gilmore, 1960]. In 1960, he wrote a computer program that successively generated sets $S_0', S_1', \ldots$, where S_i' is the set of all the ground instances obtained by replacing the variables in S by the constants in the i-level constant set H_i of S. Since each S_i' is a conjunction of ground clauses, one can use any method

available in the propositional logic to check its unsatisfiability. Gilmore used the multiplication method. That is, as each S_i' is produced, S_i' is multiplied out into a disjunctive normal form. Any conjunction in the disjunctive normal form containing a complementary pair is removed. Should some S_i' be empty, then S_i' is unsatisfiable and a proof is found.

Example 4.18

Consider

$$S = \{P(x), \sim P(a)\}.$$

$$H_0 = \{a\}$$

$$S_0' = P(a) \wedge \sim P(a) = \square.$$

Thus S is proved to be unsatisfiable.

Example 4.19

Consider

$$S = \{P(a), \sim P(x) \vee Q(f(x)), \sim Q(f(a))\}.$$

$$H_0 = \{a\}.$$

$$\begin{aligned} S_0' &= P(a) \wedge (\sim P(a) \vee Q(f(a))) \wedge \sim Q(f(a)) \\ &= (P(a) \wedge \sim P(a) \wedge \sim Q(f(a))) \vee (P(a) \wedge Q(f(a)) \wedge \sim Q(f(a))) \\ &= \square \vee \square = \square. \end{aligned}$$

Thus S is proved to be unsatisfiable.

The multiplication method used by Gilmore is inefficient. As is easily seen, even for a small set of ten two-literal ground clauses, there are 2^{10} conjunctions. To overcome this inefficiency, Davis and Putnam [1960] introduced a more efficient method for testing the unsatisfiability of a set of ground clauses. We shall now describe their method with some modification.

The Method of Davis and Putnam

Let S be a set of ground clauses. Essentially, the method consists of the following four rules.

I. *Tautology Rule* Delete all the ground clauses from S that are tautologies. The remaining set S' is unsatisfiable if and only if S is.

II. *One-Literal Rule* If there is a unit ground clause L in S, obtain S' from S by deleting those ground clauses in S containing L. If S' is empty,

S is satisfiable. Otherwise, obtain a set S'' from S' by deleting $\sim L$ from S'. S'' is unsatisfiable if and only if S is. Note that if $\sim L$ is a ground unit clause, then the clause becomes $\square$ when $\sim L$ is deleted from the clause.

III. *Pure-Literal Rule* A literal L in a ground clause of S is said to be *pure* in S if and only if the literal $\sim L$ does not appear in any ground clause in S. If a literal L is pure in S, delete all the ground clauses containing L. The remaining set S' is unsatisfiable if and only if S is.

IV. *Splitting Rule* If the set S can be put into the form

$$(A_1 \vee L) \wedge \cdots \wedge (A_m \vee L) \wedge (B_1 \vee \sim L) \wedge \cdots \wedge (B_n \vee \sim L) \wedge R,$$

where A_i, B_i, and R are free of L and $\sim L$, then obtain the sets $S_1 = A_1 \wedge \cdots \wedge A_m \wedge R$ and $S_2 = B_1 \wedge \cdots \wedge B_n \wedge R$. S is unsatisfiable if and only if $(S_1 \vee S_2)$ is unsatisfiable, that is, both S_1 and S_2 are unsatisfiable.

We can now show that the above rules are sound. That is, if the original set S is unsatisfiable, then the remaining set after one of the rules is applied is still unsatisfiable, and vice versa.

For Rule I Since a tautology is satisfied by every interpretation, S' is unsatisfiable if and only if S is.

For Rule II If S' is empty, then all the ground clauses in S contain L. Hence any interpretation containing L can satisfy S. Therefore S is satisfiable. We still have to show that S'' is unsatisfiable if and only if S is unsatisfiable. Suppose S'' is unsatisfiable. If S is satisfiable, then there is a model M of S containing L. For S'', M must satisfy all the clauses which do not contain L. Furthermore, since M falsifies $\sim L$, M must satisfy all the clauses that originally contain $\sim L$. Therefore, M must satisfy S''. This contradicts the assumption that S'' is unsatisfiable. Hence, S must be unsatisfiable. Conversely, suppose S is unsatisfiable. If S'' is satisfiable, then there is a model M'' of S''. Thus any interpretation of S containing M'' and L must be a model of S. This contradicts the assumption that S has no model. Hence S'' must be unsatisfiable. Therefore, S'' is unsatisfiable if and only if S is.

For Rule III Suppose S' is unsatisfiable. Then S must be unsatisfiable since S' is a subset of S. Conversely, suppose S is unsatisfiable. If S' is satisfiable, then there is a model M of S'. Since neither L nor $\sim L$ is in S', neither L nor $\sim L$ is in M. Thus any interpretation of S that contains M and L is a model of S. This contradicts the assumption that S has no model. Hence S' must be unsatisfiable. Therefore, S' is unsatisfiable if and only if S is unsatisfiable.

For Rule IV Suppose S is unsatisfiable. If $(S_1 \vee S_2)$ is satisfiable, then either S_1 or S_2 has a model. If $S_1(S_2)$ has a model M, then any interpretation

of S containing $\sim L(L)$ is a model of S. This contradicts the assumption that S has no model. Hence $(S_1 \vee S_2)$ is unsatisfiable. Conversely, suppose $(S_1 \vee S_2)$ is unsatisfiable. If S is satisfiable, S must have a model M. If M contains $\sim L(L)$, M can satisfy $S_1(S_2)$. This contradicts the assumption that $(S_1 \vee S_2)$ is unsatisfiable. Hence S must be unsatisfiable. Therefore, S is unsatisfiable if and only if $(S_1 \vee S_2)$ is.

The above rules are all very important. We shall see in the subsequent chapters that these rules have many extensions. We now give some examples to show how these rules can be used.

Example 4.20

Show that $S = (P \vee Q \vee \sim R) \wedge (P \vee \sim Q) \wedge \sim P \wedge R \wedge U$ is unsatisfiable.

(1) $(P \vee Q \vee \sim R) \wedge (P \vee \sim Q) \wedge \sim P \wedge R \wedge U$

(2) $(Q \vee \sim R) \wedge (\sim Q) \wedge R \wedge U$ Rule II on $\sim P$

(3) $\sim R \wedge R \wedge U$ Rule II on $\sim Q$

(4) $\square \wedge U$ Rule II on $\sim R$.

Since the last formula contains the empty clause $\square$, S is unsatisfiable.

Example 4.21

Show that $S = (P \vee Q) \wedge \sim Q \wedge (\sim P \vee Q \vee \sim R)$ is satisfiable.

(1) $(P \vee Q) \wedge \sim Q \wedge (\sim P \vee Q \vee \sim R)$

(2) $P \wedge (\sim P \vee \sim R)$ Rule II on $\sim Q$

(3) $\sim R$ Rule II on P

(4) ■ Rule II on $\sim R$.

The last set is an empty set. Hence S is satisfiable.

Example 4.22

Show that $S = (P \vee \sim Q) \wedge (\sim P \vee Q) \wedge (Q \vee \sim R) \wedge (\sim Q \vee \sim R)$ is satisfiable.

(1) $(P \vee \sim Q) \wedge (\sim P \vee Q) \wedge (Q \vee \sim R) \wedge (\sim Q \vee \sim R)$

(2) $(\sim Q \wedge (Q \vee \sim R) \wedge (\sim Q \vee \sim R))$
$\vee (Q \wedge (Q \vee \sim R) \wedge (\sim Q \vee \sim R))$ Rule IV on P

(3) $\sim R \vee \sim R$ Rule II on $\sim Q$ and Q

(4) ■ $\vee$ ■ Rule II on $\sim R$.

Since both of the split sets are satisfiable, S is satisfiable.

Example 4.23

Show that $S = (P \vee Q) \wedge (P \vee \sim Q) \wedge (R \vee Q) \wedge (R \vee \sim Q)$ is satisfiable.

(1) $(P \vee Q) \wedge (P \vee \sim Q) \wedge (R \vee Q) \wedge (R \vee \sim Q)$

(2) $(R \vee Q) \wedge (R \vee \sim Q)$ Rule III on P

(3) ■ Rule III on R.

Thus, S is satisfiable.

The above method for testing the unsatisfiability (inconsistency) is more efficient than the multiplication method. This method can be applied to any formula in the propositional logic. That is, first transform the given propositional formulas into a conjunctive normal form, and then apply the above four rules on the conjunctive normal form. The reader is encouraged to apply this method to Example 2.13 of Chapter 2.

REFERENCES

Anderson, R., and W. W. Bledsoe (1970): A linear format for resolution with merging and a new technique for establishing completeness, *J. Assoc. Comput. Mach.* **17** 525–534.

Boyer, R. S. (1971): "Locking: A Restriction of Resolution," Ph.D. Thesis, University of Texas at Austin, Texas.

Chang, C. L. (1970a): The unit proof and the input proof in theorem proving, *J. Assoc. Comput. Mach.* **17** 698–707.

Church, A. (1936): An unsolvable problem of number theory, *Amer. J. Math.* **58** 345–363.

Davis, M. (1963): Eliminating the irrelevant from mechanical proofs, *Proc. Symp. Appl. Math.* **15** 15–30.

Davis, M. and H. Putnam (1960): A computing procedure for quantification theory, *J. Assoc. Comput. Mach.* **7** 201–215.

Gilmore, P. C. (1960): A proof method for quantification theory: Its justification and realization, *IBM J. Res. Develop.* 28–35.

Knuth, D. E. (1968): "The Art of Computer Programming," Addison-Wesley, Reading, Massachusetts.

Kowalski, R. and P. Hayes (1969): Semantic trees in automatic theorem proving, *in*: "Machine Intelligence," vol. 4 (B. Meltzer and D. Michie, eds.), American Elsevier, New York, pp. 87–101.

Kowalski, R. and D. Keuhner (1970): Linear resolution with selection function, Metamathematics Unit, Edinburgh University, Scotland.

Loveland, D. W. (1970a): A linear format for resolution, *Proc. IRIA Symp. Automatic Demonstration, Versailles, France, 1968*, Springer-Verlag, New York, pp. 147–162.

Loveland, D. W. (1970b): Some linear Herbrand proof procedures: An analysis, Dept. Computer Science, Carnegie-Mellon University.

Luckham, D. (1970): Refinements in resolution theory, *Proc. IRIA Symp. Automatic Demonstration, Versailles, France, 1968*, Springer-Verlag, New York, pp. 163–190.

Meltzer, B. (1966): Theorem-proving for computers: Some results on resolution and renaming, *Computer J.* **8** 341–343.

Prawitz, D., H. Prawitz, and N. Voghera (1960): A mechanical proof procedure and its realization in an electronic computer, *J. Assoc. Comput. Mach.* **7** 102–128.

Reiter, R. (1971): Two results on ordering for resolution with merging and linear format, *J. Assoc. Comput. Mach.* **18** 630–646.

Robinson, J. A. (1965a): A machine oriented logic based on the resolution principle, *J. Assoc. Comput. Mach.* **12** 23–41.

Robinson, J. A. (1965b): Automatic deduction with hyper-resolution, *Internat. J. Comput. Math.* **1** 227–234.

Robinson, J. A. (1968a): The generalized resolution principle, *in:* "Machine Intelligence," vol. 3, (D. Michie, ed.), American Elsevier, New York, pp. 77–94.

Slagle, J. R. (1967): Automatic theorem proving with renamable and sematic resolution, *J. Assoc. Comput. Mach.* **14** 687–697.

Turing, A. M. (1936): On computable numbers, with an application to the entschiedungs-problem, *Proc. London Math. Soc.* 2, 42, pp. 230–265.

Wang, H. (1960b): Towards mechanical mathematics, *IBM J. Res. Develop.* **4** 224–268.

Wos, L., D. Carson, and G. A. Robinson (1964): The unit preference strategy in theorem proving, *Proc. AFIPS 1964 Fall Joint Computer Conf.* **26** pp. 616–621.

Wos, L., G. A. Robinson, and D. F. Carson (1965): Efficiency and completeness of the set of support strategy in theorem proving, *J. Assoc. Comput. Mach.* **12** 536–541.

Yates, R., B. Raphael, and T. Hart (1970): Resolution graphs, *Artificial Intelligence* **1** 257–290.

EXERCISES

Section 4.2

1. Find a standard form for each of the following formulas:

   (a) $\sim((\forall x)\,P(x) \rightarrow (\exists y)(\forall z)\,Q(y, z))$

   (b) $(\forall x)\big((\sim E(x, 0) \rightarrow ((\exists y)\,(E(y, g(x)) \wedge (\forall z)\,(E(z, g(x)) \rightarrow E(y, z))))\big)$

   (c) $\sim((\forall x)\,P(x) \rightarrow (\exists y)\,P(y))$.

2. Suppose $(\exists x)(\forall y)\,M[x, y]$ is a prenex normal form of a formula F, where $M[x, y]$ is the matrix that contains only variables x and y. Let f be a function symbol not occurring in $M[x, y]$. Prove that F is valid if and only if $(\exists x)\,M[x, f(x)]$ is valid.
3. Let S_1 and S_2 be standard forms of formulas F_1 and F_2, respectively. If $S_1 = S_2$, is it true that $F_1 = F_2$? Explain.
4. Consider the following statements:

   F_1: Everyone who saves money earns interst.
   F_2: If there is no interest, then nobody saves money.

   Let $S(x, y)$, $M(x)$, $I(x)$, and $E(x, y)$ represent "x saves y," "x is money," "x is interest," and "x earns y," respectively.

   (1) Symbolize F_1 and F_2.
   (2) Find the standard forms of F_1 and $\sim F_2$.

Section 4.3

5. Let $S = \{P(f(x), a, g(f(x), b))\}$.
   (1) Find H_0 and H_1.
   (2) Find all the ground instances of S over H_0.
   (3) Find all the ground instances of S over H_1.
6. Prove Lemma 4.1 of this chapter.
7. Let F denote a formula. Let S be a standard form of $\sim F$. Find the necessary and sufficient condition for F such that the Herbrand universe of S is finite.
8. Consider the following clause C and interpretation I:

   C: $P(x) \vee Q(x, f(x))$

   I: $\{\sim P(a), \sim P(f(a)), \sim P(f(f(a))), \ldots,$

   $\sim Q(a, a), Q(a, f(a)), \sim Q(a, f(f(a))), \ldots,$

   $\sim Q(f(a), a), Q(f(a), f(a)), \sim Q(f(a), f(f(a))), \ldots\}$.

   Does I satisfy C?
9. Consider the following set S of clauses:

$$S = \begin{cases} P(x) \\ Q(f(y)). \end{cases}$$

   Let an interpretation I be defined as below:

   $I = \{P(a), P(f(a)), P(f(f(a))), \ldots,$

   $Q(a), \sim Q(f(a)), Q(f(f(a))), \ldots\}$.

   Does I satisfy S?
10. Consider $S = \{P(x), \sim P(f(y))\}$.

    1. Give H_0, H_1, H_2, and H_3 of S.
    2. Is it possible to find an interpretation that satisfies S? If yes, give one. If no, why?

Section 4.4

11. Let $S = \{P, \sim P \vee Q, \sim Q\}$. Give a closed semantic tree of S.
12. Consider $S = \{P(x), \sim P(x) \vee Q(x, a), \sim Q(y, a)\}$.

    (a) Give the atom set of S.
    (b) Give a complete semantic tree of S.
    (c) Give a closed semantic tree of S.

Section 4.5

13. Consider $S = \{P(x, a, g(x, b)), \sim P(f(y), z, g(f(a), b))\}$. Find an unsatisfiable set S' of ground instances of clauses in S.
14. Let $S = \{P(x), Q(x, f(x)) \vee \sim P(x), \sim Q(g(y), z)\}$. Find an unsatisfiable set S' of ground instances of clauses in S.

Section 4.6

15. Use the rules suggested by Davis and Putnam to prove that the following formulas are unsatisfiable.

    (1) $(\sim P \vee Q) \wedge \sim Q \wedge P$

    (2) $(P \vee Q) \wedge (R \vee Q) \wedge \sim R \wedge \sim Q$

    (3) $(P \vee Q) \wedge (\sim P \vee Q) \wedge (\sim R \vee \sim Q) \wedge (R \vee \sim Q)$.

16. Use the rules suggested by Davis and Putnam to prove that the following formulas are satisfiable.

    (1) $P \wedge Q \wedge R$

    (2) $(P \vee Q) \wedge (\sim P \vee Q) \wedge R$

    (3) $(P \vee Q) \wedge \sim Q$.

Chapter 5

The Resolution Principle

5.1 INTRODUCTION

In the previous chapters, we have considered Herbrand's theorem. Based upon this theorem, we have given a refutation procedure. However, Herbrand's procedure has one major drawback: It requires the generation of sets S_1', S_2',... of ground instances of clauses. For most cases, this sequence grows exponentially. To see this, let us give one simple example. Consider

$$S = \{P(x, g(x), y, h(x, y), z, k(x, y, z)), \sim P(u, v, e(v), w, f(v, w), x)\}.$$

Since

$$\begin{aligned} H_0 &= \{a\}, \\ H_1 &= \{a, g(a), h(a, a), k(a, a, a), e(a), f(a, a)\} \\ &\vdots\ , \end{aligned}$$

the number of elements in S_0', S_1', ... is 2, 1512, ..., respectively. It is interesting to note that the earliest set that is unsatisfiable is S_5'. However, H_5' has of the order of 10^{64} elements. Consequently, S_5' has of the order of 10^{256} elements. With current computer technology, it is impossible even to store S_5' in a computer, not to mention test its unsatisfiability.

In order to avoid the generation of sets of ground instances as required

in Herbrand's procedure, we shall introduce in this chapter the resolution principle due to Robinson [1965]. It can be applied directly to any set S of clauses (not necessarily ground clauses) to test the unsatisfiability of S.

The essential idea of the resolution principle is to check whether S contains the empty clause $\square$. If S contains $\square$, then S is unsatisfiable. If S does not contain $\square$, the next thing to check is whether $\square$ can be derived from S. It will be clear later that, by Herbrand's theorem (version I), checking for the presence of $\square$ is equivalent to counting the number of nodes of a closed semantic tree for S. By Theorem 4.3, S is unsatisfiable if and only if there is a finite closed semantic tree T for S. Clearly S contains $\square$ if and only if T consists of only one node—the root node. If S does not contain $\square$, T must contain more than one node. However, if we can reduce the number of nodes in T to one, eventually $\square$ can be forced to appear. This is what the resolution principle does. Indeed, we can view the resolution principle as an inference rule that can be used to generate new clauses from S. If we put these new clauses into S, some nodes of the original T can be forced to become failure nodes. Thus the number of nodes in T can be reduced and the empty clause $\square$ will be eventually derived.

In this chapter, we shall first consider the resolution principle for the propositional logic. Then we shall extend it to the first-order logic.

5.2 THE RESOLUTION PRINCIPLE FOR THE PROPOSITIONAL LOGIC

The resolution principle is essentially an extension of the *one-literal rule* of Davis and Putnam, given in Section 4.6 of Chapter 4. Consider the following clauses:

C_1: P

C_2: $\sim P \vee Q$.

Using the one-literal rule, from C_1 and C_2 we can obtain a clause

C_3: Q.

What the one-literal rule has required us to do is first to examine whether there is a complementary pair of a literal (for example, P) in C_1 and a literal (for example, $\sim P$) in C_2, then to delete this pair from C_1 and C_2 to obtain clause C_3, which is Q.

Extending the above rule and applying it to any pair of clauses (not necessarily unit clauses), we have the following rule, which is called the *resolution principle: For any two clauses C_1 and C_2, if there is a literal L_1 in C_1 that is complementary to a literal L_2 in C_2, then delete L_1 and L_2 from C_1 and C_2, respectively, and construct the disjunction of the remaining clauses. The constructed clause is a resolvent of C_1 and C_2.*

Example 5.1

Consider the following clauses:

C_1: $P \vee R$

C_2: $\sim P \vee Q$.

Clause C_1 has the literal P, which is complementary to $\sim P$ in C_2. Therefore, by deleting P and $\sim P$ from C_1 and C_2, respectively, and constructing the disjunction of the remaining clauses R and Q, we obtain a resolvent $R \vee Q$.

Example 5.2

Consider clauses

C_1: $\sim P \vee Q \vee R$

C_2: $\sim Q \vee S$.

The resolvent of C_1 and C_2 is $\sim P \vee R \vee S$.

Example 5.3

Consider clauses

C_1: $\sim P \vee Q$

C_2: $\sim P \vee R$.

Since there is no literal in C_1 that is complementary to any literal in C_2, there is no resolvent of C_1 and C_2.

An important property of a resolvent is that any resolvent of two clauses C_1 and C_2 is a logical consequence of C_1 and C_2. This is stated in the following theorem.

Theorem 5.1 *Given two clauses C_1 and C_2, a resolvent C of C_1 and C_2 is a logical consequence of C_1 and C_2.*

Proof Let C_1, C_2, and C be denoted as follows: $C_1 = L \vee C_1'$, $C_2 = \sim L \vee C_2'$, and $C = C_1' \vee C_2'$, where C_1' and C_2' are disjunctions of literals. Suppose C_1 and C_2 are true in an interpretation I. We want to prove that the resolvent C of C_1 and C_2 is also true in I. To prove this, note that either L or $\sim L$ is false in I. Assume L is false in I. Then C_1 must not be a unit clause, for otherwise, C_1 would be false in I. Therefore, C_1' must be true in I. Thus, the resolvent C, that is, $C_1' \vee C_2'$, is true in I. Similarly, we can show that if $\sim L$ is false in I, then C_2' must be true in I. Therefore, $C_1' \vee C_2'$ must be true in I. Q.E.D.

If we have two unit clauses, then the resolvent of them, if there is one, is the empty clause $\square$. More importantly, if a set S of clauses is unsatisfiable, we can use the resolution principle to generate $\square$ from S. This result will be presented as a theorem in Section 5.6 of this chapter. Meanwhile, we define the notion of deduction.

Definition Given a set S of clauses, a (resolution) *deduction* of C from S is a finite sequence $C_1, C_2, \ldots, C_k$ of clauses such that each C_i either is a clause in S or a resolvent of clauses preceding C_i, and $C_k = C$. A deduction of $\square$ from S is called a *refutation*, or a *proof* of S.

We say that a clause C can be *deduced* or *derived* from S if there is a deduction of C from S. We shall now present several examples to show how the resolution principle can be used to prove the unsatisfiability of a set of clauses.

Example 5.4

Consider the set

$$\left.\begin{array}{ll} (1) & \sim P \vee Q \\ (2) & \sim Q \\ (3) & P \end{array}\right\} S.$$

From (1) and (2), we can obtain a resolvent

(4) $\sim P$.

From (4) and (3), we can obtain a resolvent

(5) $\square$.

Since $\square$ is derived from S by resolution, according to Theorem 5.1 $\square$ is a logical consequence of S. However, $\square$ can only be a logical consequence of an unsatisfiable set of clauses. Hence, S is unsatisfiable.

Example 5.5

For the set

$$\left.\begin{array}{ll} (1) & P \vee Q \\ (2) & \sim P \vee Q \\ (3) & P \vee \sim Q \\ (4) & \sim P \vee \sim Q \end{array}\right\} S$$

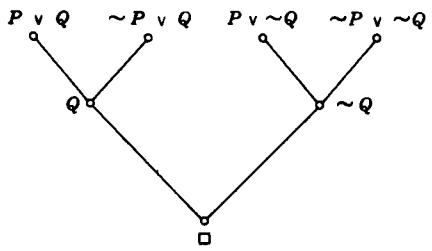


Figure 5.1

we generate the following resolvents

(5) Q from (1) and (2)

(6) $\sim Q$ from (3) and (4)

(7) $\square$ from (5) and (6).

Since $\square$ is derived, S is unsatisfiable. The above deduction can be represented by the tree in Fig. 5.1, called a *deduction tree.*

The resolution principle is a very powerful inference rule. In the sequel, we shall define this principle in the context of the first-order logic. We shall also show that the resolution principle is *complete* in proving the unsatisfiability of a set of clauses. That is, given a set S of clauses, S is unsatisfiable if and only if there is a resolution deduction of the empty clause $\square$ from S. In Section 5.7, we shall apply the resolution principle to several examples so that the reader can appreciate the usefulness of this inference rule.

5.3 SUBSTITUTION AND UNIFICATION

In the last section, we considered the resolution principle for the propositional logic. We shall extend it to the first-order logic in later sections. In Section 5.2, we noted that the most important part of applying the resolution principle is finding a literal in a clause that is complementary to a literal in another clause. For clauses containing no variables, this is very simple. However, for clauses containing variables, it is more complicated. For example, consider the clauses:

C_1: $P(x) \vee Q(x)$

C_2: $\sim P(f(x)) \vee R(x)$.

There is no literal in C_1 that is complementary to any literal in C_2. However, if we substitute $f(a)$ for x in C_1 and a for x in C_2, we obtain

C_1': $P(f(a)) \vee Q(f(a))$

C_2': $\sim P(f(a)) \vee R(a)$.

We know that C_1' and C_2' are ground instances of C_1 and C_2, respectively, and $P(f(a))$ and $\sim P(f(a))$ are complementary to each other. Therefore, from C_1' and C_2', we can obtain a resolvent

C_3': $Q(f(a)) \vee R(a)$.

More generally, if we substitute $f(x)$ for x in C_1, we obtain

C_1^*: $P(f(x)) \vee Q(f(x))$.

Again, C_1^* is an *instance* of C_1. This time the literal $P(f(x))$ in C_1^* is complementary to the literal $\sim P(f(x))$ in C_2. Therefore, we can obtain a resolvent from C_1^* and C_2,

C_3: $Q(f(x)) \vee R(x)$.

C_3' is an instance of clause C_3. By substituting appropriate terms for the variables in C_1 and C_2 as above, we can generate new clauses from C_1 and C_2. Furthermore, clause C_3 is the *most general clause* in the sense that all the other clauses which can be generated by the above process are instances of C_3. C_3 will also be called a resolvent of C_1 and C_2. In the sequel, we shall consider how to generate resolvents from clauses (possibly containing variables). Since obtaining resolvents from clauses frequently requires substitutions for variables, we give the following definitions.

Definition A *substitution* is a finite set of the form $\{t_1/v_1, \ldots, t_n/v_n\}$, where every v_i is a variable, every t_i is a term different from v_i, and no two elements in the set have the same variable after the stroke symbol. When $t_1, \ldots, t_n$ are ground terms, the substitution is called a *ground substitution*. The substitution that consists of no elements is called the *empty substitution* and is denoted by ε. We shall use Greek letters to represent substitutions.

Example 5.6

The following two sets are substitutions:

$$\{f(z)/x, y/z\}, \qquad \{a/x, g(y)/y, f(g(b))/z\}.$$

Definition Let $\theta = \{t_1/v_1, \ldots, t_n/v_n\}$ be a substitution and E be an expression. Then $E\theta$ is an expression obtained from E by replacing simultaneously each occurrence of the variable v_i, $1 \leqslant i \leqslant n$, in E by the term t_i. $E\theta$ is called an *instance* of E. (We note that the definition of an instance is compatible with that of a ground instance of a clause, defined in Chapter 4.)

Example 5.7

Let $\theta = \{a/x, f(b)/y, c/z\}$ and $E = P(x, y, z)$. Then $E\theta = P(a, f(b), c)$.

Definition Let $\theta = \{t_1/x_1, \ldots, t_n/x_n\}$ and $\lambda = \{u_1/y_1, \ldots, u_m/y_m\}$ be two substitutions. Then the *composition* of θ and λ is the substitution, denoted by $\theta \circ \lambda$, that is obtained from the set

$$\{t_1\lambda/x_1, \ldots, t_n\lambda/x_n, u_1/y_1, \ldots, u_m/y_m\}$$

by deleting any element $t_j\lambda/x_j$ for which $t_j\lambda = x_j$, and any element u_i/y_i such that y_i is among $\{x_1, x_2, \ldots, x_n\}$.

Example 5.8

Let

$$\theta = \{t_1/x_1, t_2/x_2\} = \{f(y)/x, z/y\}$$

$$\lambda = \{u_1/y_1, u_2/y_2, u_3/y_3\} = \{a/x, b/y, y/z\}.$$

Then

$$\{t_1\lambda/x_1, t_2\lambda/x_2, u_1/y_1, u_2/y_2, u_3/y_3\} = \{f(b)/x, y/y, a/x, b/y, y/z\}.$$

However, since $t_2\lambda = x_2, t_2\lambda/x_2$, i.e., y/y, should be deleted from the set. In addition, since y_1 and y_2 are among $\{x_1, x_2, x_3\}$, u_1/y_1 and u_2/y_2, that is, a/x and b/y, should be deleted. Thus we obtain

$$\theta \circ \lambda = \{f(b)/x, y/z\}.$$

It should be noted that the composition of substitutions is associative, and that the empty substitution ε is both a left and a right identity. That is, $(\theta \circ \lambda) \circ \mu = \theta \circ (\lambda \circ \mu)$ and $\varepsilon \circ \theta = \theta \circ \varepsilon$ for all θ, λ, and μ.

In the resolution proof procedure, in order to identify a complementary pair of literals, very often we have to unify (match) two or more expressions. That is, we have to find a substitution that can make several expressions identical. Therefore, we now consider the unification of expressions.

Definition A substitution θ is called a *unifier* for a set $\{E_1, \ldots, E_k\}$ if and only if $E_1\theta = E_2\theta = \cdots = E_k\theta$. The set $\{E_1, \ldots, E_k\}$ is said to be *unifiable* if there is a unifier for it.

Definition A unifier σ for a set $\{E_1, \ldots, E_k\}$ of expressions is a *most general unifier* if and only if for each unifier θ for the set there is a substitution λ such that $\theta = \sigma \circ \lambda$.

Example 5.9

The set $\{P(a, y), P(x, f(b))\}$ is unifiable since the substitution $\theta = \{a/x, f(b)/y\}$ is a *unifier* for the set.

5.4 UNIFICATION ALGORITHM

In this section, we shall give a unification algorithm for finding a most general unifier for a finite unifiable set of nonempty expressions. When the set is not unifiable, the algorithm will also detect this fact.

Consider $P(a)$ and $P(x)$. These two expressions are not identical: The disagreement is that a occurs in $P(a)$ but x in $P(x)$. In order to unify $P(a)$ and $P(x)$, we first have to find the disagreement, and then try to eliminate it. For $P(a)$ and $P(x)$, the disagreement is $\{a,x\}$. Since x is a variable, x can be replaced by a, and thus the disagreement can be eliminated. This is the idea upon which the unification algorithm is based.

Definition The disagreement set of a nonempty set W of expressions is obtained by locating the first symbol (counting from the left) at which not all the expressions in W have exactly the same symbol, and then extracting from each expression in W the subexpression that begins with the symbol occupying that position. The set of these respective subexpressions is the *disagreement set* of W.

Example 5.10

If W is $\{P(x, \underline{f(y,z)}), P(x,\underline{a}), P(x,\underline{g(h(k(x)))})\}$, then the first symbol position at which not all atoms in W are exactly the same is the fifth, since they all have the first four symbols $P(x,$ in common. Thus, the disagreement set consists of the respective subexpressions (underlined terms) that begin in symbol position number five, and it is in fact the set $\{f(y,z),a,g(h(k(x)))\}$.

Unification Algorithm

Step 1 Set $k = 0$, $W_k = W$, and $\sigma_k = \varepsilon$.

Step 2 If W_k is a singleton, stop; σ_k is a most general unifier for W. Otherwise, find the disagreement set D_k of W_k.

Step 3 If there exist elements v_k and t_k in D_k such that v_k is a variable that does not occur in t_k, go to Step 4. Otherwise, stop; W is not unifiable.

Step 4 Let $\sigma_{k+1} = \sigma_k\{t_k/v_k\}$ and $W_{k+1} = W_k\{t_k/v_k\}$. (Note that $W_{k+1} = W\sigma_{k+1}$.)

Step 5 Set $k = k+1$ and go to Step 2.

Example 5.11

Find a most general unifier for $W = \{P(a, x, f(g(y))), P(z, f(z), f(u))\}$.

1. $\sigma_0 = \varepsilon$ and $W_0 = W$. Since W_0 is not a singleton, σ_0 is not a most general unifier for W.

2. The disagreement set $D_0 = \{a,z\}$. In D_0, there is a variable $v_0 = z$ that does not occur in $t_0 = a$.

3. Let

$$\begin{aligned}\sigma_1 &= \sigma_0 \circ \{t_0/v_0\} = \varepsilon \circ \{a/z\} = \{a/z\},\\ W_1 &= W_0\{t_0/v_0\}\\ &= \{P(a, x, f(g(y))), P(z, f(z), f(u))\}\{a/z\}\\ &= \{P(a, x, f(g(y))), P(a, f(a), f(u))\}.\end{aligned}$$

4. W_1 is not a singleton, hence we find the disagreement set D_1 of W_1:

$$D_1 = \{x, f(a)\}.$$

5. From D_1, we find that $v_1 = x$ and $t_1 = f(a)$.
6. Let

$$\begin{aligned}\sigma_2 &= \sigma_1 \circ \{t_1/v_1\} = \{a/z\} \circ \{f(a)/x\} = \{a/z, f(a)/x\},\\ W_2 &= W_1\{t_1/v_1\}\\ &= \{P(a, x, f(g(y))), P(a, f(a), f(u))\}\{f(a)/x\}\\ &= \{P(a, f(a), f(g(y))), P(a, f(a), f(u))\}.\end{aligned}$$

7. W_2 is not a singleton, hence we find the disagreement set D_2 of W_2: $D_2 = \{g(y), u\}$. From D_2, we find that $v_2 = u$ and $t_2 = g(y)$.
8. Let

$$\begin{aligned}\sigma_3 &= \sigma_2 \circ \{t_2/v_2\} = \{a/z, f(a)/x\} \circ \{g(y)/u\} = \{a/z, f(a)/x, g(y)/u\},\\ W_3 &= W_2\{t_2/v_2\}\\ &= \{P(a, f(a), f(g(y))), P(a, f(a), f(u))\}\{g(y)/u\}\\ &= \{P(a, f(a), f(g(y))), P(a, f(a), f(g(y)))\}\\ &= \{P(a, f(a), f(g(y)))\}.\end{aligned}$$

9. Since W_3 is a singleton, $\sigma_3 = \{a/z, f(a)/x, g(y)/u\}$ is a most general unifier for W.

Example 5.12

Determine whether or not the set $W = \{Q(f(a), g(x)), Q(y, y)\}$ is unifiable.

1. Let $\sigma_0 = \varepsilon$ and $W_0 = W$.
2. W_0 is not a singleton, hence we find the disagreement set D_0 of W_0: $D_0 = \{f(a), y\}$. From D_0, we know that $v_0 = y$ and $t_0 = f(a)$.
3. Let

$$\begin{aligned}\sigma_1 &= \sigma_0 \circ \{t_0/v_0\} = \varepsilon \circ \{f(a)/y\} = \{f(a)/y\},\\ W_1 &= W_0\{t_0/v_0\} = \{Q(f(a), g(x)), Q(y, y)\}\{f(a)/y\}\\ &= \{Q(f(a), g(x)), Q(f(a), f(a))\}.\end{aligned}$$

4. W_1 is not a singleton, hence we find the disagreement set D_1 of W_1: $D_1 = \{g(x), f(a)\}$.

5. However, no element of D_1 is a variable. Hence the unification algorithm is terminated and we conclude that W is not unifiable.

It is noted that the above unification algorithm will always terminate for any finite nonempty set of expressions, for otherwise there would be generated an infinite sequence $W\sigma_0, W\sigma_1, W\sigma_2, \ldots$ of finite nonempty sets of expressions with the property that each successive set contains one less variable than its predecessor (namely, $W\sigma_k$ contains v_k but $W\sigma_{k+1}$ does not). This is impossible since W contains only finitely many distinct variables.

We indicated before that if W is unifiable, the unification algorithm will always find a most general unifier for W. This is proved in the following theorem.

Theorem 5.2 (Unification Theorem) If W is a finite nonempty unifiable set of expressions, then the unification algorithm will always terminate at Step 2, and the last σ_k is a most general unifier for W.

Proof Since W is unifiable, we let θ be any unifier for W. For $k = 0, 1, \ldots$, we show that there is a substitution λ_k such that $\theta = \sigma_k \circ \lambda_k$ by induction on k. For $k = 0$, we let $\lambda_0 = \theta$. Then $\theta = \sigma_0 \circ \lambda_0$, since $\sigma_0 = \varepsilon$. Suppose $\theta = \sigma_k \circ \lambda_k$ holds for $0 \leqslant k \leqslant n$. If $W\sigma_n$ is a singleton, then the unification algorithm terminates at Step 2. Since $\theta = \sigma_n \circ \lambda_n, \sigma_n$ is a most general unifier for W. If $W\sigma_n$ is not a singleton, then the unification algorithm will find the disagreement set D_n of $W\sigma_n$. Because $\theta = \sigma_n \circ \lambda_n$ is a unifier of W, λ_n must unify D_n. However, since D_n is the disagreement set, there must be a variable in D_n. Let t_n be any other element different from v_n. Then, since λ_n unifies $D_n, v_n\lambda_n = t_n\lambda_n$. Now if v_n occurs in t_n, then $v_n\lambda_n$ occurs in $t_n\lambda_n$. However, this is impossible since v_n and t_n are distinct, and $v_n\lambda_n = t_n\lambda_n$. Therefore, v_n does not occur in t_n. Hence the unification algorithm will not terminate at Step 3, but will go to Step 4 to set $\sigma_{n+1} = \sigma_n \circ \{t_n/v_n\}$. Let $\lambda_{n+1} = \lambda_n - \{t_n\lambda_n/v_n\}$. Then, since v_n does not occur in t_n, $t_n\lambda_{n+1} = t_n(\lambda_n - \{t_n\lambda_n/v_n\}) = t_n\lambda_n$. Thus, we have

$$\begin{aligned}\{t_n/v_n\} \circ \lambda_{n+1} &= \{t_n\lambda_{n+1}/v_n\} \cup \lambda_{n+1} \\ &= \{t_n\lambda_n/v_n\} \cup \lambda_{n+1} \\ &= \{t_n\lambda_n/v_n\} \cup (\lambda_n - \{t_n\lambda_n/v_n\}) \\ &= \lambda_n.\end{aligned}$$

That is, $\lambda_n = \{t_n/v_n\} \circ \lambda_{n+1}$. Therefore,

$$\theta = \sigma_n \circ \lambda_n = \sigma_n \circ \{t_n/v_n\} \circ \lambda_{n+1} = \sigma_{n+1} \circ \lambda_{n+1}.$$

Hence, for all $k \geqslant 0$, there is a substitution λ_k such that $\theta = \sigma_k \circ \lambda_k$. Since

the unification algorithm must terminate, and since it will not terminate at Step 3, it must terminate at Step 2. Furthermore, since $\theta = \sigma_k \circ \lambda_k$ for all k, the last σ_k is a most general unifier for W. Q.E.D.

5.5 THE RESOLUTION PRINCIPLE FOR THE FIRST-ORDER LOGIC

Having introduced the unification algorithm in the last section, we can now consider the resolution principle for the first-order logic.

Definition If two or more literals (with the same sign) of a clause C have a most general unifier σ, then $C\sigma$ is called a *factor* of C. If $C\sigma$ is a unit clause, it is called a *unit factor* of C.

Example 5.13

Let $C = \underline{P(x)} \vee \underline{P(f(y))} \vee \sim Q(x)$. Then the first and the second literals (underlined) have a most general unifier $\sigma = \{f(y)/x\}$. Hence, $C\sigma = P(f(y)) \vee \sim Q(f(y))$ is a factor of C.

Definition Let C_1 and C_2 be two clauses (called *parent clauses*) with no variables in common. Let L_1 and L_2 be two literals in C_1 and C_2, respectively. If L_1 and $\sim L_2$ have a most general unifier σ, then the clause

$$(C_1\sigma - L_1\sigma) \cup (C_2\sigma - L_2\sigma)$$

is called a *binary resolvent* of C_1 and C_2. The literals L_1 and L_2 are called the *literals resolved upon.*

Example 5.14

Let $C_1 = P(x) \vee Q(x)$ and $C_2 = \sim P(a) \vee R(x)$. Since x appears in both C_1 and C_2, we rename the variable in C_2 and let $C_2 = \sim P(a) \vee R(y)$. Choose $L_1 = P(x)$ and $L_2 = \sim P(a)$. Since $\sim L_2 = P(a)$, L_1 and $\sim L_2$ have the most general unifier $\sigma = \{a/x\}$. Therefore,

$$\begin{aligned}
&(C_1\sigma - L_1\sigma) \cup (C_2\sigma - L_2\sigma) \\
&\quad = (\{P(a), Q(a)\} - \{P(a)\}) \cup (\{\sim P(a), R(y)\} - \{\sim P(a)\}) \\
&\quad = \{Q(a)\} \cup \{R(y)\} = \{Q(a), R(y)\} = Q(a) \vee R(y).
\end{aligned}$$

Thus $Q(a) \vee R(y)$ is a binary resolvent of C_1 and C_2. $P(x)$ and $\sim P(a)$ are the literals resolved upon.

Definition A *resolvent* of (parent) clauses C_1 and C_2 is one of the following binary resolvents:

1. a binary resolvent of C_1 and C_2,
2. a binary resolvent of C_1 and a factor of C_2,
3. a binary resolvent of a factor of C_1 and C_2,
4. a binary resolvent of a factor of C_1 and a factor of C_2.

Example 5.15

Let $C_1 = P(x) \vee P(f(y)) \vee R(g(y))$ and $C_2 = \sim P(f(g(a))) \vee Q(b)$. A factor of C_1 is $C_1' = P(f(y)) \vee R(g(y))$. A binary resolvent of C_1' and C_2 is $R(g(g(a))) \vee Q(b)$. Therefore, $R(g(g(a))) \vee Q(b)$ is a resolvent of C_1 and C_2.

The resolution principle, or resolution for short, is an inference rule that generates resolvents from a set of clauses. This rule was introduced in 1965 by Robinson. It is more efficient than the earlier proof procedures such as the direct implementation of Herbrand's theorem used by Gilmore and Davis and Putnam. Furthermore, resolution is *complete*. That is, it will always generate the empty clause $\square$ from an unsatisfiable set of clauses. Before we prove this statement, we first give an example in plane geometry.

Example 5.16

Show that alternate interior angles formed by a diagonal of a trapezoid are equal.

To prove this theorem, we first axiomatize it. Let $T(x,y,u,v)$ mean that $xyuv$ is a trapezoid with upper-left vertex x, upper-right vertex y, lower-right vertex u, and lower-left vertex v; let $P(x,y,u,v)$ mean that the line segment xy is parallel to the line segment uv; and let $E(x,y,z,u,v,w)$ mean that the angle xyz is equal to the angle uvw. Then we have the following axioms:

A_1:	$(\forall x)(\forall y)(\forall u)(\forall v)[T(x,y,u,v) \rightarrow P(x,y,u,v)]$	definition of a trapezoid
A_2:	$(\forall x)(\forall y)(\forall u)(\forall v)[P(x,y,u,v) \rightarrow E(x,y,v,u,v,y)]$	alternate interior angles of parallel lines are equal
A_3:	$T(a,b,c,d)$	given in Fig. 5.2.

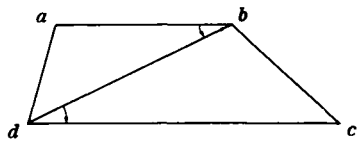


Figure 5.2

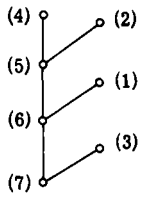


Figure 5.3

From these axioms, we should be able to conclude that $E(a,b,d,c,d,b)$ is true, that is,

$$A_1 \wedge A_2 \wedge A_3 \rightarrow E(a,b,d,c,d,b)$$

is a valid formula. Since we want to prove this by refutation, we negate the conclusion and prove that

$$A_1 \wedge A_2 \wedge A_3 \wedge \sim E(a,b,d,c,d,b)$$

is unsatisfiable. To do this, we transform it into a standard form as follows:

$$S = \{\sim T(x,y,u,v) \vee P(x,y,u,v),\ \sim P(x,y,u,v) \vee E(x,y,v,u,v,y),$$
$$T(a,b,c,d),\ \sim E(a,b,d,c,d,b)\}.$$

The above standard form S is a set of four clauses. We now show that the set S is unsatisfiable by resolution:

(1)	$\sim T(x,y,u,v) \vee P(x,y,u,v)$	a clause in S
(2)	$\sim P(x,y,u,v) \vee E(x,y,v,u,v,y)$	a clause in S
(3)	$T(a,b,c,d)$	a clause in S
(4)	$\sim E(a,b,d,c,d,b)$	a clause in S
(5)	$\sim P(a,b,c,d)$	a resolvent of (2) and (4)
(6)	$\sim T(a,b,c,d)$	a resolvent of (1) and (5)
(7)	$\square$	a resolvent of (3) and (6).

Since the last clause is the empty clause that is derived from S, we conclude that S is unsatisfiable. The above proof steps can be easily represented by a tree shown in Fig. 5.3.

The tree in Fig. 5.3 is called a deduction tree. That is, a *deduction tree* from a set S of clauses is an (upward) tree, to each initial node of which is attached a clause in S, and to each noninitial node of which is attached a resolvent of clauses attached to its immediate predecessor nodes. We call

the deduction tree a *deduction tree of R* if R is the clause attached to the root node of the deduction tree. Since a deduction tree is merely the tree representation of a deduction, in the sequel we shall use deduction and deduction tree interchangeably.

5.6 COMPLETENESS OF THE RESOLUTION PRINCIPLE

In Chapter 4, we presented the concept of semantic trees to prove Herbrand's theorem. In this section, we shall use the semantic tree to prove the completeness of the resolution principle. In fact, there is a close relationship between the semantic tree and resolution, as is demonstrated by the following example.

Example 5.17

Consider the following set S of clauses:

(1) P

(2) $\sim P \vee Q$

(3) $\sim P \vee \sim Q$.

The atom set of S is $\{P,Q\}$. Let T be a complete semantic tree as shown in Fig. 5.4a. T has a closed semantic tree T' shown in Fig. 5.4b. Node (2) in Fig. 5.4b is an inference node. Its two successors, nodes (4) and (5), are failure nodes. The clauses falsified at nodes (4) and (5) are $\sim P \vee \sim Q$ and $\sim P \vee Q$, respectively. As can be seen, these two clauses must have a complementary pair of literals, and therefore can be resolved. Resolving

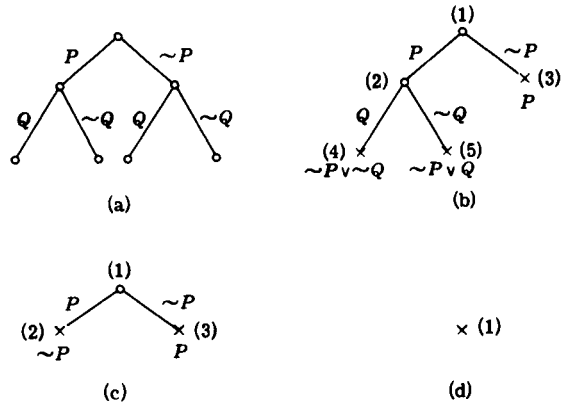


Figure 5.4 (a) T. (b) T'. ($\sim P \vee Q$) and ($\sim P \vee \sim Q$) can be resolved to give $\sim P$. (c) T''. P and $\sim P$ can be resolved to give $\square$. (d) T'''.

$\sim P \vee \sim Q$ and $\sim P \vee Q$, we obtain $\sim P$. Note that $\sim P$ is falsified by the partial interpretation corresponding to node (2). If we put $\sim P$ into S, then we can have a closed semantic tree T'' for $S \cup \{\sim P\}$, as shown in Fig. 5.4c. In Fig. 5.4c, node (1) is an inference node. This time we can obtain $\square$ by resolving P and $\sim P$. Putting $\square$ into $S \cup \{\sim P\}$, we obtain the closed semantic tree T''' for $S \cup \{\sim P\} \cup \{\square\}$ as shown in Fig. 5.4d. The above "collapsing" of the semantic tree actually corresponds to a resolution proof as follows:

(4) $\sim P$ a resolvent of (2) and (3)

(5) $\square$ a resolvent of (4) and (1).

In the following, we shall use the concept developed above to prove the completeness of the resolution principle. That is, after constructing a closed semantic tree for an unsatisfiable set of clauses, we can gradually force the tree to collapse and at the same time obtain a resolution proof. Before proving the completeness theorem, we prove a lemma called the *lifting lemma.*

Lemma 5.1 (Lifting Lemma) If C_1' and C_2' are instances of C_1 and C_2, respectively, and if C' is a resolvent of C_1' and C_2', then there is a resolvent C of C_1 and C_2 such that C' is an instance of C.

Proof We rename, if necessary, the variables in C_1 and C_2 so that variables in C_1 are all different from those in C_2. Let L_1' and L_2' be the literals resolved upon, and let

$$C' = (C_1'\gamma - L_1'\gamma) \cup (C_2'\gamma - L_2'\gamma),$$

where γ is a most general unifier of L_1' and $\sim L_2'$. Since C_1' and C_2' are instances of C_1 and C_2, respectively, there is a substitution θ such that $C_1' = C_1\theta$ and $C_2' = C_2\theta$. Let $L_i^1, \ldots, L_i^{r_i}$ be the literals in C_i corresponding to L_i' (i.e., $L_i^1\theta = \cdots = L_i^{r_i}\theta = L_i'$), $i = 1,2$. If $r_i > 1$, obtain a most general unifier λ_i for $\{L_i^1, \ldots, L_i^{r_i}\}$ and let $L_i = L_i^1\lambda_i, i = 1,2$. (Note that $L_i^1\lambda_i, \ldots, L_i^{r_i}\lambda_i$ are the same, since λ_i is a most general unifier.) Then L_i is a literal in the factor $C_i\lambda_i$ of C_i. If $r_i = 1$, let $\lambda_i = \varepsilon$ and $L_i = L_i^1\lambda_i$. Let $\lambda = \lambda_1 \cup \lambda_2$. Thus, clearly L_i' is an instance of L_i. Since L_i' and $\sim L_2'$ are unifiable, L_1 and $\sim L_2$ are unifiable. Let σ be a most general unifier of L_1 and $\sim L_2$. Let

$$\begin{aligned} C &= ((C_1\lambda)\sigma - L_1\sigma) \cup ((C_2\lambda)\sigma - L_2\sigma) \\ &= ((C_1\lambda)\sigma - (\{L_1^1, \ldots, L_1^{r_1}\}\lambda)\sigma) \cup ((C_2\lambda)\sigma - (\{L_2^1, \ldots, L_2^{r_2}\}\lambda)\sigma) \\ &= (C_1(\lambda \circ \sigma) - \{L_1^1, \ldots, L_1^{r_1}\}(\lambda \circ \sigma)) \cup (C_2(\lambda \circ \sigma) - \{L_2^1, \ldots, L_2^{r_2}\}(\lambda \circ \sigma)). \end{aligned}$$

C is a resolvent of C_1 and C_2. Clearly, C' is an instance of C since

$$\begin{aligned} C' &= (C_1'\gamma - L_1'\gamma) \cup (C_2'\gamma - L_2'\gamma) \\ &= ((C_1\theta)\gamma - (\{L_1^1, \ldots, L_1^{r1}\}\theta)\gamma) \cup ((C_2\theta)\gamma - (\{L_2^1, \ldots, L_2^{r2}\}\theta)\gamma) \\ &= (C_1(\theta \circ \gamma) - \{L_1^1, \ldots, L_1^{r1}\}(\theta \circ \gamma)) \cup (C_2(\theta \circ \gamma) - \{L_2^1, \ldots, L_2^{r2}\}(\theta \circ \gamma)) \end{aligned}$$

and $\lambda \circ \sigma$ is more general than $\theta \circ \gamma$. Thus we complete the proof of this lemma.

Theorem 5.3 (Completeness of the Resolution Principle) A set S of clauses is unsatisfiable if and only if there is a deduction of the empty clause $\square$ from S.

Proof ($\Rightarrow$) Suppose S is unsatisfiable. Let $A = \{A_1, A_2, A_3, \ldots\}$ be the atom set of S. Let T be a complete semantic tree as shown in Fig. 5.5. By Herbrand's theorem (version I), T has a finite closed semantic tree T'. If T' consists of only one (root) node, then $\square$ must be in S, for no other clause can be falsified at the root of a semantic tree. In this case, the theorem is obviously true. Assume T' consists of more than one node. Then T' has at least one inference node, for, if it did not, then every node would have at least one nonfailure descendent. We could then find an infinite branch through T', violating the fact that T' is a finite closed semantic tree. Let N be an inference node in T'. Let N_1 and N_2 be the failure nodes immediately below N. Let

$$\begin{aligned} I(N) &= \{m_1, m_2, \ldots, m_n\}, \\ I(N_1) &= \{m_1, m_2, \ldots, m_n, m_{n+1}\}, \\ I(N_2) &= \{m_1, m_2, \ldots, m_n, \sim m_{n+1}\}. \end{aligned}$$

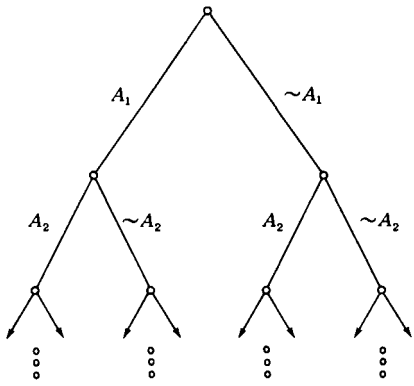


Figure 5.5

Since N_1 and N_2 are failure nodes but N is not a failure node, there must exist two ground instances C_1' and C_2' of clauses C_1 and C_2 such that C_1' and C_2' are false in $I(N_1)$ and $I(N_2)$, respectively, but both C_1' and C_2' are not falsified by $I(N)$. Therefore, C_1' must contain $\sim m_{n+1}$ and C_2' must contain m_{n+1}. Let $L_1' = \sim m_{n+1}$ and $L_2' = m_{n+1}$. By resolving upon the literals L_1' and L_2', we can obtain a resolvent C' of C_1' and C_2', namely,

$$C' = (C_1' - L_1') \cup (C_2' - L_2').$$

C' must be false in $I(N)$ since both $(C_1' - L_1')$ and $(C_2' - L_2')$ are false in $I(N)$. By Lemma 5.1, there is a resolvent C of C_1 and C_2 such that C' is a ground instance of C. Let T'' be the closed semantic tree for $(S \cup \{C\})$, obtained from T' by deleting any node or link that is below the first node where the resolvent C' is falsified. Clearly, the number of nodes in T'' is fewer than that in T'. Applying the above process on T'' again, we can obtain another resolvent of clauses in $(S \cup \{C\})$. Putting this resolvent into $(S \cup \{C\})$, we can get another smaller closed semantic tree. This process is repeated again and again until the closed semantic tree that consists of only the root node is generated. This is possible only when $\square$ is derived. Therefore, there is a deduction of $\square$ from S.

($\Leftarrow$) Conversely, suppose there is a deduction of $\square$ from S. Let R_1, $R_2, \ldots, R_k$ be the resolvents in the deduction. Assume S is satisfiable. Then there is a model M of S. However, if a model satisfies clauses C_1 and C_2, it must also satisfy any resolvent of C_1 and C_2 (Theorem 5.1). Therefore, M satisfies $R_1, R_2, \ldots, R_k$. However, this is impossible because one of these resolvents is $\square$. Hence, S must be unsatisfiable. Q.E.D.

5.7 EXAMPLES USING THE RESOLUTION PRINCIPLE

In the previous chapters, we have used many methods to prove the inconsistency of formulas. In this section we shall use examples to demonstrate how resolution can be efficiently used to prove theorems.

Example 5.18

Let us reconsider Example 2.10. We have four statements:

(1′) $P \rightarrow S$

(2′) $S \rightarrow U$

(3′) P

(4′) U.

To show that (4′) follows from (1′), (2′), and (3′), we first transform all these statements into standard forms. Thus we have

(1) $\sim P \vee S$

(2) $\sim S \vee U$

(3) P

(4) U.

We prove that U is a logical consequence of (1), (2), and (3) by refutation. We negate (4) and obtain the following proof:

(1)	$\sim P \vee S$	
(2)	$\sim S \vee U$	
(3)	P	
(4)	$\sim U$	negation of conclusion
(5)	S	a resolvent of (3) and (1)
(6)	U	a resolvent of (5) and (2)
(7)	$\square$	a resolvent of (6) and (4).

Example 5.19

Reconsider Example 2.12 in Chapter 2. We have three formulas F_1, F_2, and F_3. We want to prove that $\sim Q$ is a logical consequence of $F_1 \wedge F_2 \wedge F_3$. We negate $\sim Q$ and transform $F_1 \wedge F_2 \wedge F_3 \wedge Q$ into a standard form. This gives rise the following set of clauses:

(1)	$\sim P \vee \sim Q \vee R$	from F_1
(2)	$\sim P \vee \sim Q \vee S$	from F_1
(3)	P	from F_2
(4)	$\sim S$	from F_3
(5)	Q	negation of conclusion.

Using resolution, we obtain the following proof:

(6)	$\sim Q \vee S$	a resolvent of (3) and (2)
(7)	S	a resolvent of (6) and (5)
(8)	$\square$	a resolvent of (7) and (4).

Example 5.20

Consider the following set of formulas:

F_1: $(\forall x)(C(x) \rightarrow (W(x) \wedge R(x)))$

F_2: $(\exists x)(C(x) \wedge O(x))$

G: $(\exists x)(O(x) \wedge R(x))$.

Our problem is to show that G is a logical consequence of F_1 and F_2. We transform F_1, F_2, and $\sim G$ into standard forms and obtain the following five clauses:

(1) $\sim C(x) \vee W(x)$
(2) $\sim C(x) \vee R(x)$ } from F_1

(3) $C(a)$
(4) $O(a)$ } from F_2

(5) $\sim O(x) \vee \sim R(x)$ from $\sim G$.

The above set of the clauses is unsatisfiable. This can be proved by resolution as follows:

(6) $R(a)$ a resolvent of (3) and (2)

(7) $\sim R(a)$ a resolvent of (5) and (4)

(8) $\square$ a resolvent of (7) and (6).

Therefore, G is a logical consequence of F_1 and F_2.

Example 5.21

Reconsider Example 3.15. We have

F_1: $(\exists x)(P(x) \wedge (\forall y)(D(y) \rightarrow L(x,y)))$

F_2: $(\forall x)(P(x) \rightarrow (\forall y)(Q(y) \rightarrow \sim L(x,y)))$

G: $(\forall x)(D(x) \rightarrow \sim Q(x))$.

As usual, F_1, F_2, and $\sim G$ are transformed into the following clauses:

(1) $P(a)$
(2) $\sim D(y) \vee L(a, y)$ } from F_1

(3) $\sim P(x) \vee \sim Q(y) \vee \sim L(x, y)$ from F_2

(4) $D(b)$
(5) $Q(b)$ } from $\sim G$.

Using resolution, we obtain the following proof:

(6) $L(a,b)$ a resolvent of (4) and (2)

(7) $\sim Q(y) \vee \sim L(a,y)$ a resolvent of (3) and (1)

(8) $\sim L(a,b)$ a resolvent of (5) and (7)

(9) $\square$ a resolvent of (6) and (8).

The resolution principle is quite natural. Let us try to translate the above proof into English:

(a) From F_1, we can assume that there is a patient a who likes every doctor (clauses (1) and (2)).
(b) Let us assume that the conclusion is wrong. That is, assume that b is both a doctor and a quack (clauses (4) and (5)).
(c) Since the patient a likes every doctor, a likes b (clause (6)).
(d) Since a is a patient, a does not like any quack (clause (7)).
(e) However, b is a quack. Therefore, a does not like b ((clause (8)).
(f) This is impossible because of (c). Thus, we complete the proof.

Example 5.22

Premises: The custom officials searched everyone who entered this country who was not a VIP. Some of the drug pushers entered this country and they were only searched by drug pushers. No drug pusher was a VIP.

Conclusion: Some of the officials were drug pushers.

Let $E(x)$ mean "x entered this country," $V(x)$ mean "x was a VIP," $S(x,y)$ mean "y searched x," $C(x)$ mean "x was a custom official," and $P(x)$ mean "x was a drug pusher."

The premises are represented by the following formulas:

$$(\forall x)(E(x) \wedge \sim V(x) \rightarrow (\exists y)(S(x,y) \wedge C(y)))$$

$$(\exists x)(P(x) \wedge E(x) \wedge (\forall y)(S(x,y) \rightarrow P(y)))$$

$$(\forall x)(P(x) \rightarrow \sim V(x))$$

and the conclusion is

$$(\exists x)(P(x) \wedge C(x)).$$

Transforming premises into clauses, we obtain

(1) $\sim E(x) \vee V(x) \vee S(x, f(x))$

(2) $\sim E(x) \vee V(x) \vee C(f(x))$

(3) $P(a)$

(4) $E(a)$

(5) $\sim S(a,y) \vee P(y)$

(6) $\sim P(x) \vee \sim V(x)$.

The negation of the conclusion is

(7) $\sim P(x) \vee \sim C(x)$.

The resolution proof is as follows:

(8)	$\sim V(a)$	from (3) and (6)
(9)	$V(a) \vee C(f(a))$	from (2) and (4)
(10)	$C(f(a))$	from (8) and (9)
(11)	$V(a) \vee S(a, f(a))$	from (1) and (4)
(12)	$S(a, f(a))$	from (8) and (11)
(13)	$P(f(a))$	from (12) and (5)
(14)	$\sim C(f(a))$	from (13) and (7)
(15)	$\square$	from (10) and (14).

Thus, we have established the conclusion.

Example 5.23

The premise is that everyone who saves money earns interest. The conclusion is that if there is no interest, then nobody saves money. Let $S(x, y)$, $M(x)$, $I(x)$, and $E(x, y)$ denote "x saves y," "x is money," "x is interest," and "x earns y," respectively. Then the premise is symbolized as

$$(\forall x)((\exists y)(S(x, y) \wedge M(y)) \rightarrow (\exists y)(I(y) \wedge E(x, y)))$$

and the conclusion is

$$\sim(\exists x)\, I(x) \rightarrow (\forall x)(\forall y)(S(x, y) \rightarrow \sim M(y)).$$

Translating the premise into clauses, we have

(1) $\sim S(x, y) \vee \sim M(y) \vee I(f(x))$

(2) $\sim S(x, y) \vee \sim M(y) \vee E(x, f(x))$.

The negation of the conclusion is

(3) $\sim I(z)$

(4) $S(a, b)$

(5) $M(b)$.

The proof given in the following is indeed very simple:

(6)	$\sim S(x, y) \vee \sim M(y)$	from (3) and (1)
(7)	$\sim M(b)$	from (6) and (4)
(8)	$\square$	from (7) and (5).

Thus the conclusion is established.

Example 5.24

Premise: Students are citizens.

Conclusion: Students' votes are citizens' votes.

Let $S(x)$, $C(x)$, and $V(x,y)$ denote "x is a student," "x is a citizen," and "x is a vote of y," respectively. Then the premise and the conclusion are symbolized as follows:

$(\forall y)(S(y) \rightarrow C(y))$ (premise)

$(\forall x)((\exists y)(S(y) \wedge V(x,y)) \rightarrow (\exists z)(C(z) \wedge V(x,z)))$ (conclusion).

A standard form of the premise is

(1) $\sim S(y) \vee C(y)$.

Since

$$\begin{aligned}
&\sim\big((\forall x)((\exists y)(S(y) \wedge V(x,y)) \rightarrow (\exists z)(C(z) \wedge V(x,z)))\big) \\
&\quad = \sim\big((\forall x)((\forall y)(\sim S(y) \vee \sim V(x,y)) \vee (\exists z)(C(z) \wedge V(x,z)))\big) \\
&\quad = \sim\big((\forall x)(\forall y)(\exists z)(\sim S(y) \vee \sim V(x,y) \vee (C(z) \wedge V(x,z)))\big) \\
&\quad = (\exists x)(\exists y)(\forall z)(S(y) \wedge V(x,y) \wedge (\sim C(z) \vee \sim V(x,z)))
\end{aligned}$$

we have three clauses for the negation of the conclusion,

(2) $S(b)$

(3) $V(a,b)$

(4) $\sim C(z) \vee \sim V(a,z)$.

The proof proceeds as follows:

(5) $C(b)$ from (1) and (2)

(6) $\sim V(a,b)$ from (5) and (4)

(7) $\square$ from (6) and (3).

Again, although the above proof is quite mechanical, it is actually very natural. Indeed, we can translate the above proof into English as follows: Let us assume that b is a student, a is b's vote, and a is not a vote of any citizen. Since b is a student, b must be a citizen. Furthermore, a must not be a vote of b because b is a citizen. This is impossible. Thus we complete the proof.