CHAPTER 7

Propositional and Predicate Logic

If, dear Reader, you will faithfully observe these Rules, and so give my little book a really fair trial, I promise you, most confidently, that you will find Symbolic Logic to be one of the most, if not the most, fascinating of mental recreations!

—Lewis Carroll, from the Introduction to Symbolic Logic

If it was so, it might be; and if it were so, it would be: but as it isn't, it ain't. That's logic.

—Lewis Carroll, from *Through The Looking Glass*

Of science and logic he chatters
As fine and as fast as he can;
Though I am no judge of such matters,
I'm sure he's a talented man.

—Winthrop Mackworth Praed, from *The Talented Man*

7.1 Introduction

In this chapter, we introduce propositional calculus and first-order predicate calculus, the languages of logic. We introduce methods that can be used to carry out deductions and prove whether or not a conclusion follows from a set of premises.

We introduce the ideas of logical equivalence, tautologies, and satisfiability. This chapter also discusses some important properties of logical systems, including soundness, completeness, monotonicity, and decidability.

This chapter assumes no previous knowledge of logic, so readers who are already familiar with the ideas of propositional logic and the predicate calculus may wish to skim this chapter.

7.2 What Is Logic?

Logic is concerned with reasoning and the validity of arguments. In general, in logic, we are not concerned with the *truth* of statements, but rather with their **validity**. That is to say, although the following argument is clearly logical, it is not something that we would consider to be true:

All lemons are blue

Mary is a lemon

Therefore, Mary is blue

This set of statements is considered to be **valid** because the conclusion (Mary is blue) follows logically from the other two statements, which we often call the **premises**.

The reason that validity and truth can be separated in this way is simple: a piece of a reasoning is considered to be valid if its conclusion is true in cases where its premises are also true. Hence, a valid set of statements such as the ones above can give a false conclusion, provided one or more of the premises are also false.

We can say: *a piece of reasoning is valid if it leads to a true conclusion in every situation where the premises are true.*

Logic is concerned with **truth values**. The possible truth values are true and false. These can be considered to be the fundamental units of logic, and almost all logic is ultimately concerned with these truth values.

7.3 Why Logic Is Used in Artificial Intelligence

Logic is widely used in computer science, and particularly in Artificial Intelligence. Logic is widely used as a representational method for Artificial Intelligence. Unlike some other representations (such as **frames**, which are described in detail in Chapter 3), logic allows us to easily reason about negatives (such as, "this book is not red") and disjunctions ("or"—such as, "He's either a soldier or a sailor").

Logic is also often used as a representational method for communicating concepts and theories within the Artificial Intelligence community. In addition, logic is used to represent language in systems that are able to understand and analyze human language.

As we will see, one of the main weaknesses of traditional logic is its inability to deal with **uncertainty**. Logical statements must be expressed in terms of truth or falsehood—it is not possible to reason, in classical logic, about possibilities. We will see different versions of logic such as **modal logics** that provide some ability to reason about possibilities, and also probabilistic methods and fuzzy logic that provide much more rigorous ways to reason in uncertain situations.

7.4 Logical Operators

In reasoning about truth values, we need to use a number of **operators**, which can be applied to truth values. We are familiar with several of these operators from everyday language:

I like apples **and** oranges.

You can have an ice cream **or** a cake.

If you come from France, **then** you speak French.

I am **not** stupid!

Here we see the four most basic logical operators being used in everyday language. The operators are:

- and
- or
- not
- if . . . then . . . (usually called **implies**)

These operators work more or less as we expect them to. One important point to note is that *or* is slightly different from the way we usually use it. In the sentence, "You can have an icecream or a cake," the mother is usually suggesting to her child that he can only have one of the items, but not both. This is referred to as an **exclusive-or** in logic because the case where both are allowed is excluded. The version of *or* that is used in logic is called **inclusive-or** and allows the case with both options.

The operators are usually written using the following symbols, although other symbols are sometimes used, according to the context:

and	$\wedge$
or	$\vee$
not	$\neg$
implies	$\rightarrow$
iff	$\leftrightarrow$

Iff is an abbreviation that is commonly used to mean "if and only if." We see later that this is a stronger form of implies that holds true if one thing implies another, and also the second thing implies the first.

For example, "you can have an ice-cream if and only if you eat your dinner." It may not be immediately apparent why this is different from "you can have an icecream if you eat your dinner." This is because most mothers really mean *iff* when they use *if* in this way.

7.5 Translating between English and Logic Notation

To use logic, it is first necessary to convert facts and rules about the real world into logical expressions using the logical operators described in Section 7.4. Without a reasonable amount of experience at this translation, it can seem quite a daunting task in some cases.

Let us examine some examples.

First, we will consider the simple operators, $\wedge$, $\vee$, and $\neg$.

Sentences that use the word *and* in English to express more than one concept, all of which is true at once, can be easily translated into logic using the AND operator, $\wedge$. For example:

"It is raining and it is Tuesday."

might be expressed as:

$R \wedge T$

Where *R* means "it is raining" and *T* means "it is Tuesday." Note that we have been fairly arbitrary in our choice of these terms. This is all right, as long as the terms are chosen in such a way that they represent the problem adequately. For example, if it is not necessary to discuss where it is raining, *R* is probably enough. If we need to write expressions such as "it is raining

in New York" or "it is raining heavily" or even "it rained for 30 minutes on Thursday," then R will probably not suffice.

To express more complex concepts like these, we usually use predicates. Hence, for example, we might translate "it is raining in New York" as:

N(R)

We might equally well choose to write it as:

R(N)

This depends on whether we consider the rain to be a property of New York, or vice versa. In other words, when we write $N(R)$, we are saying that a property of the rain is that it is in New York, whereas with $R(N)$ we are saying that a property of New York is that it is raining.

Which we use depends on the problem we are solving. It is likely that if we are solving a problem about New York, we would use $R(N)$, whereas if we are solving a problem about the location of various types of weather, we might use $N(R)$.

Let us return now to the logical operators. The expression "it is raining in New York, and I'm either getting sick or just very tired" can be expressed as follows:

$\mathrm{R(N)} \wedge (\mathrm{S(I)} \vee \mathrm{T(I)})$

Here we have used both the $\wedge$ operator, and the $\vee$ operator to express a collection of statements. The statement can be broken down into two sections, which is indicated by the use of parentheses. The section in the parentheses is $S(I) \vee T(I)$, which means "I'm either getting sick OR I'm very tired". This expression is "AND'ed" with the part outside the parentheses, which is $R(N)$.

Finally, the $\neg$ operator is applied exactly as you would expect—to express negation. For example,

It is not raining in New York,

might be expressed as

$\neg\mathrm{R(N)}$

It is important to get the $\neg$ in the right place. For example: "I'm either not well or just very tired" would be translated as

$\neg\mathrm{W(I)} \vee \mathrm{T(I)}$

The position of the ¬ here indicates that it is bound to $W(I)$ and does not play any role in affecting $T(I)$. This idea of **precedence** is explained further in Section 7.7.

Now let us see how the → operator is used. Often when dealing with logic we are discussing rules, which express concepts such as "if it is raining then I will get wet."

This sentence might be translated into logic as

$$R \rightarrow W(I)$$

This is read "R implies W(I)" or "IF R THEN W(I)". By replacing the symbols R and $W(I)$ with their respective English language equivalents, we can see that this sentence can be read as

"raining implies I'll get wet"

or "IF it's raining THEN I'll get wet."

Implication can be used to express much more complex concepts than this. For example, "Whenever he eats sandwiches that have pickles in them, he ends up either asleep at his desk or singing loud songs" might be translated as

$$S(y) \wedge E(x, y) \wedge P(y) \rightarrow A(x) \vee (S(x, z) \wedge L(z))$$

Here we have used the following symbol translations:

$S(y)$ means that y is a sandwich.

$E(x, y)$ means that x (the man) eats y (the sandwich).

$P(y)$ means that y (the sandwich) has pickles in it.

$A(x)$ means that x ends up asleep at his desk.

$S(x, z)$ means that x (the man) sings z (songs).

$L(z)$ means that z (the songs) are loud.

In fact, there are better ways to express this kind of sentence, as we will see when we examine the quantifiers ∃ and ∀ in Section 7.13.

The important thing to realize is that the choice of variables and predicates is important, but that you can choose any variables and predicates that map well to your problem and that help you to solve the problem. For example, in the example we have just looked at, we could perfectly well have used instead

$$S \rightarrow A \vee L$$

where S means "he eats a sandwich which has pickles in it," A means "he ends up asleep at his desk," and L means "he sings loud songs."

The choice of granularity is important, but there is no right or wrong way to make this choice. In this simpler logical expression, we have chosen to express a simple relationship between three variables, which makes sense if those variables are all that we care about—in other words, we don't need to know anything else about the sandwich, or the songs, or the man, and the facts we examine are simply whether or not he eats a sandwich with pickles, sleeps at his desk, and sings loud songs. The first translation we gave is more appropriate if we need to examine these concepts in more detail and reason more deeply about the entities involved.

Note that we have thus far tended to use single letters to represent logical variables. It is also perfectly acceptable to use longer variable names, and thus to write expressions such as the following:

$$\text{Fish}(x) \wedge \text{living}(x) \rightarrow \text{has\_scales}(x)$$

This kind of notation is obviously more useful when writing logical expressions that are intended to be read by humans but when manipulated by a computer do not add any value.

7.6 Truth Tables

We can use **variables** to represent possible truth values, in much the same way that variables are used in algebra to represent possible numerical values. We can then apply logical operators to these variables and can reason about the way in which they behave.

It is usual to represent the behavior of these logical operators using **truth tables**. A truth table shows the possible values that can be generated by applying an operator to truth values.

7.6.1 Not

First of all, we will look at the truth table for not, ¬.

Not is a **unary** operator, which means it is applied only to one variable. Its behavior is very simple:

¬ true is equal to false

¬ false is equal to true

If variable *A* has value *true*, then ¬*A* has value *false*.

If variable *B* has value *false*, then ¬*B* has value *true*.

These can be represented by a truth table,

A	¬ A
true	false
false	true

7.6.2 And

Now, let us examine the truth table for our first **binary** operator—one which acts on two variables:

A	B	A ∧ B
false	false	false
false	true	false
true	false	false
true	true	true

∧ is also called the **conjunctive operator.** $A \wedge B$ is the **conjunction** of A and B.

You can see that the only entry in the truth table for which $A \wedge B$ is *true* is the one where A is *true* **and** B is *true*. If A is *false*, or if B is *false*, then $A \wedge B$ is *false*. If both A and B are *false*, then $A \wedge B$ is also *false*.

What do A and B mean? They can represent any statement, or **proposition**, that can take on a truth value. For example, A might represent "It's sunny," and B might represent "It's warm outside." In this case, $A \wedge B$ would mean "It is sunny **and** it's warm outside," which clearly is true only if the two component parts are true (i.e., if it is true that it is sunny and it is true that it is warm outside).

7.6.3 Or

The truth table for the *or* operator, ∨, should need little explanation.

A	B	A ∨ B
false	false	false
false	true	true
true	false	true
true	true	true

$\vee$ is also called the **disjunctive operator**. $A \vee B$ is the **disjunction** of A and B.

Clearly $A \vee B$ is true for any situation except when both A and B are *false*. If A is *true*, or if B is *true*, or if both A and B are *true*, $A \vee B$ is true.

You should notice that this table represents the inclusive-or operator. A table to represent exclusive-or would have *false* in the final row. In other words, while $A \vee B$ is *true* if A and B are both *true*, A *EOR* B (A exclusive-or B) is *false* if A and B are both *true*.

You may also notice a pleasing symmetry between the truth tables for $\wedge$ and $\vee$. This will become useful later, as will a number of other symmetrical relationships.

7.6.4 Implies

The truth table for *implies* ($\rightarrow$) is a little less intuitive.

A	B	A $\rightarrow$ B
false	false	true
false	true	true
true	false	false
true	true	true

(This form of implication is also known as **material implication**.)

In the statement $A \rightarrow B$, A is the **antecedent**, and B is the **consequent**.

The bottom two lines of the table should be obvious. If A is *true* and B is *true*, then $A \rightarrow B$ seems to be a reasonable thing to believe. For example, if A means "you live in France" and B means "You speak French," then $A \rightarrow B$ corresponds to the statement "if you live in France, then you speak French." Clearly, this statement is true ($A \rightarrow B$ is *true*) if I live in France and I speak French (A is *true* and B is *true*).

Similarly, if I live in France, but I don't speak French (A is *true*, but B is *false*), then it is clear that $A \rightarrow B$ is not *true*.

The situations where A is *false* are a little less clear. If I do not live in France (A is not *true*), then the truth table tells us that regardless of whether I speak French or not (the value of B), the statement $A \rightarrow B$ is *true*.

$A \rightarrow B$ is usually read as "A implies B" but can also be read as "If A then B" or "If A is true then B is true." Hence, if A is *false*, the statement is not really

saying anything about the value of B, so B is free to take on any value (as long as it is *true* or *false*, of course!).

This can lead to some statements being valid that might at first glance appear absurd. All of the following statements are valid:

$5^2 = 25 \rightarrow 4 = 4$ (true $\rightarrow$ true)
$9 \times 9 = 123 \rightarrow 8 > 3$ (false $\rightarrow$ true)
$52 = 25 \rightarrow 0 = 2$ (false $\rightarrow$ false)

In fact, in the second and third examples, the consequent could be given *any* meaning, and the statement would still be true. For example, the following statement is valid:

$52 = 25 \rightarrow$ Logic is weird

Notice that when looking at simple logical statements like these, there does not need to be any real-world relationship between the antecedent and the consequent. For logic to be useful, though, we tend to want the relationships being expressed to be meaningful as well as being logically true.

7.6.5 iff

The truth table for *iff* (if and only if $\{\leftrightarrow\}$) is as follows:

A	B	A $\leftrightarrow$ B
false	false	true
false	true	false
true	false	false
true	true	true

It can be seen that $A \leftrightarrow B$ is *true* as long as A and B have the same value. In other words, if one is *true* and the other *false*, then $A \leftrightarrow B$ is *false*. Otherwise, if A and B have the same value, $A \leftrightarrow B$ is *true*.

7.7 Complex Truth Tables

Truth tables are not limited to showing the values for single operators. For example, a truth table can be used to display the possible values for $A \wedge (B \vee C)$.

A	B	C	A ∧ (B ∨ C)
false	false	false	false
false	false	true	false
false	true	false	false
false	true	true	false
true	false	false	false
true	false	true	true
true	true	false	true
true	true	true	true

Note that for two variables, the truth table has four lines, and for three variables, it has eight. In general, a truth table for n variables will have 2^n lines.

The use of brackets in this expression is important. $A \wedge (B \vee C)$ is not the same as $(A \wedge B) \vee C$.

To avoid ambiguity, the logical operators are assigned precedence, as with mathematical operators. The order of precedence that is used is as follows:

$$\neg, \wedge, \vee, \rightarrow, \leftrightarrow$$

Hence, in a statement such as

$$\neg A \vee \neg B \wedge C$$

the $\neg$ operator has the greatest precedence, meaning that it is most closely tied to its symbols. $\wedge$ has a greater precedence than $\vee$, which means that the sentence above can be expressed as

$$(\neg A) \vee ((\neg B) \wedge C)$$

Similarly, when we write

$$\neg A \vee B$$

this is the same as

$$(\neg A) \vee B$$

rather than

$$\neg(A \vee B)$$

In general, it is a good idea to use brackets whenever an expression might otherwise be ambiguous.

7.8 Tautology

Consider the following truth table:

A	$A \vee \neg A$
false	true
true	true

This truth table has a property that we have not seen before: the value of the expression $A \vee \neg A$ is *true* regardless of the value of *A*. An expression like this that is always *true* is called a **tautology**.

If *A* is a tautology, we write:

$$\models A$$

Tautologies may seem like rather uninteresting entities, but in fact they are extremely useful for logic, as we see later.

A logical expression that is a tautology is often described as being **valid**. A valid expression is defined as being one that is true under any **interpretation**. In other words, no matter what meanings and values we assign to the variables in a valid expression, it will still be true. For example, the following sentences are all valid:

If wibble is true, then wibble is true.

Either wibble is true, or wibble is not true.

In the language of logic, we can replace *wibble* with the symbol *A*, in which case these two statements can be rewritten as

$$A \rightarrow A$$

$$A \vee \neg A$$

If an expression is *false* in any interpretation, it is described as being **contradictory**. The following expressions are contradictory:

$$A \wedge \neg A$$

$$(A \vee \neg A) \rightarrow (A \wedge \neg A)$$

It doesn't matter what *A* means in these expressions, the result cannot be *true*.

Some expressions are **satisfiable**, but not valid. This means that they are true under some interpretation, but not under all interpretations. The following expressions are satisfiable:

$$A \vee B$$

$$(A \wedge B \vee \neg C) \rightarrow (D \wedge E)$$

A contradictory expression is clearly not satisfiable and so is described as being **unsatisfiable**.

7.9 Equivalence

Consider the following two expressions:

$$A \wedge B$$

$$B \wedge A$$

It should be fairly clear that these two expressions will always have the same value for a given pair of values for *A* and B. In other words, we say that the first expression is **logically equivalent** to the second expression. We write this as

$$A \wedge B \equiv B \wedge A$$

This means that the $\wedge$ operator is **commutative**.

Note that this is not the same as implication:

$$A \wedge B \rightarrow B \wedge A$$

although this second statement is also true. The difference is that if for two expressions *e*1 and *e*2:

$$e1 \equiv e2$$

then *e*1 will always have the same value as *e*2 for a given set of variables. On the other hand, as we have seen, *e*1 $\rightarrow$ *e*2 is *true* if *e*1 is *false* and *e*2 is *true*.

There are a number of logical equivalences that are extremely useful. The following is a list of a few of the most common:

$$A \vee A \equiv A$$

$$A \wedge A \equiv A$$

$$A \wedge (B \wedge C) \equiv (A \wedge B) \wedge C \qquad (\wedge \text{ is } \textbf{associative})$$

$$A \vee (B \vee C) \equiv (A \vee B) \vee C \qquad (\vee \text{ is } \textbf{associative})$$
$$A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C) \qquad (\wedge \text{ is } \textbf{distributive} \text{ over } \vee)$$
$$A \wedge (A \vee B) \equiv A$$
$$A \vee (A \wedge B) \equiv A$$
$$A \wedge \text{true} \equiv A$$
$$A \wedge \text{false} \equiv \text{false}$$
$$A \vee \text{true} \equiv \text{true}$$
$$A \vee \text{false} \equiv A$$

All of these equivalences can be proved by drawing up the truth tables for each side of the equivalence and seeing if the two tables are the same. You may want to try this to satisfy yourself that all of the equivalences are correct, particularly for some of the less intuitive ones.

The following is a very important equivalence:

$$A \rightarrow B \equiv \neg A \vee B$$

You can verify this by checking the truth tables. The reason that this is useful is that it means we do not need to use the $\rightarrow$ symbol at all—we can replace it with a combination of $\neg$ and $\vee$. Similarly, the following equivalences mean we do not need to use $\wedge$ or $\leftrightarrow$:

$$A \wedge B \equiv \neg(\neg A \vee \neg B)$$
$$A \leftrightarrow B \equiv \neg(\neg(\neg A \vee B) \vee \neg (\neg B \vee A))$$

In fact, *any* binary logical operator can be expressed using $\neg$ and $\vee$. This is a fact that is employed in electronic circuits, where *nor* gates, based on an operator called *nor*, are used. *Nor* is represented by $\downarrow$, and is defined as follows:

$$A \downarrow B \equiv \neg(A \vee B)$$

Finally, the following equivalences are known as **DeMorgan's Laws** and will be used later in this chapter:

$$A \wedge B \equiv \neg(\neg A \vee \neg B)$$
$$A \vee B \equiv \neg(\neg A \wedge \neg B)$$

By using these and other equivalences, logical expressions can be **simplified**. For example,

$$(C \wedge D) \vee ((C \wedge D) \wedge E)$$

can be simplified using the following rule:

$$A \vee (A \wedge B) \equiv A$$

hence,

$$(C \wedge D) \vee ((C \wedge D) \wedge E) \equiv C \wedge D$$

In this way, it is possible to eliminate subexpressions that do not contribute to the overall value of the expression.

7.10 Propositional Logic

There are a number of possible systems of logic. The system we have been examining so far in this chapter is called **propositional logic**. The language that is used to express propositional logic is called the **propositional calculus** (although in practice, many people use the expressions *logic* and *calculus* interchangeably in this context).

A logical system can be defined in terms of its syntax (the alphabet of symbols and how they can be combined), its semantics (what the symbols mean), and a set of rules of **deduction** that enable us to derive one expression from a set of other expressions and thus make arguments and proofs.

7.10.1 Syntax

We have already examined the syntax of propositional calculus. The alphabet of symbols, Σ is defined as follows

$$\Sigma = \{\text{true}, \text{false}, \neg, \rightarrow, (,), \wedge, \vee, \leftrightarrow, p_1, p_2, p_3, \ldots, p_n, \ldots\}$$

Here we have used **set notation** to define the possible values that are contained within the alphabet Σ. Note that we allow an infinite number of **proposition letters**, or **propositional symbols**, p_1, p_2, p_3, . . . , and so on. More usually, we will represent these by capital letters *P*, *Q*, *R*, and so on, although if we need to represent a very large number of them, we will use the subscript notation (e.g., p_1).

An expression is referred to as a **well-formed formula** (often abbreviated as wff) or a **sentence** if it is constructed correctly, according to the rules of the syntax of propositional calculus, which are defined as follows. In these

rules, we use *A*, *B*, *C* to represent sentences. In other words, we define a sentence recursively, in terms of other sentences. The following are well-formed sentences:

P, Q, R. . .

true, false

(A)

$\neg$A

A $\wedge$ B

A $\vee$ B

A $\rightarrow$ B

A $\leftrightarrow$ B

Hence, we can see that the following is an example of a wff:

P $\wedge$ Q $\vee$ (B $\wedge$ $\neg$C) $\rightarrow$ A $\wedge$ B $\vee$ D $\wedge$ ($\neg$E)

This is not to make any claims about the validity or otherwise of the expression, simply that it is allowed within the syntax of propositional calculus.

7.10.2 Semantics

The semantics of the operators of propositional calculus can be defined in terms of truth tables. As we have seen, the meaning of $P \wedge Q$ is defined as "*true* when *P* is *true* and *Q* is also *true*."

The meaning of symbols such as *P* and *Q* is arbitrary and could be ignored altogether if we were reasoning about pure logic. In other words, reasoning about sentences such as $P \vee Q \wedge \neg R$ is possible without considering what *P*, *Q*, and *R mean*.

Because we are using logic as a representational method for artificial intelligence, however, it is often the case that when using propositional logic, the meanings of these symbols are very important. The beauty of this representation is that it is possible for a computer to reason about them in a very general way, without needing to know much about the real world.

In other words, if we tell a computer, "I like ice cream, and I like chocolate," it might represent this statement as $A \wedge B$, which it could then use to reason with, and, as we will see, it can use this to make deductions.

7.11 Deduction

If we have a set of assumptions $\{A_1, A_2, \ldots, A_n\}$, and from those assumptions we are able to derive a conclusion, C, then we say that we have **deduced** C from the assumptions, which is written

$$\{A_1, A_2, \ldots, A_n\} \vdash C$$

If C can be concluded without any assumptions, then we write

$$\vdash C$$

To derive a conclusion from a set of assumptions, we apply a set of **inference rules**. To distinguish an inference rule from a sentence, we often write $A \vdash B$ as follows:

$$\frac{A}{B}$$

Some of the most useful inference rules for propositional logic are as follows. In these rules, A, B, and C stand for any logical expressions.

7.11.1 ∧-Introduction

$$\frac{A \quad B}{A \wedge B}$$

This rule is very straightforward. It says: Given A and B, we can deduce $A \wedge B$. This follows from the definition of $\wedge$.

7.11.2 ∧-Elimination

$$\frac{A \wedge B}{A}$$

Similarly,

$$\frac{A \wedge B}{B}$$

These rules say that given $A \wedge B$, we can deduce A and we can also deduce B separately. Again, these follow from the definition of $\wedge$.

7.11.3 Or-Introduction

$$\frac{A}{A \vee B}$$

$$\frac{B}{A \vee B}$$

These rules say that from *A* we can deduce the disjunction of *A* with *any* expression. For example, from the statement "I like logic," we can deduce expressions such as "I like logic or I like cheese," "I like logic or I do not like logic," "I like logic or fish can sing," "I like logic or 2 + 2 = 123," and so on. This follows because *true* ∨ *B* is *true* for any value of *B*.

7.11.4 → Elimination

This rule is usually known as **modus ponens** and is one of the most commonly used rules in logical deduction. It is expressed as follows:

$$\frac{A \quad A \rightarrow B}{B}$$

In other words, if *A* is *true* and *A* implies *B* is *true*, then we know that *B* is *true*.

For example, if we replace *A* with "it is raining" and *B* with "I need an umbrella," then we produce the following:

> It is raining. If it's raining, I need an umbrella. Therefore, I need an umbrella.

This kind of reasoning is clearly valid.

7.11.5 Reductio Ad Absurdum

We need to introduce a new notation for this rule:

$$\begin{array}{c} \neg A \\ \vdots \\ \bot \\ \hline A \end{array}$$

The symbol ⊥ is called **falsum,** which is used to indicate an absurdity, or a contradiction. For example, ⊥ can be deduced from $A \wedge \neg A$.

The reductio ad absurdum rule simply says that if we assume that A is *false* ($\neg A$) and this leads to a contradiction ($\bot$), then we can deduce that A is *true*. This is known as **proof by contradiction**.

As we will see, this is an extremely powerful concept and is widely used in logical systems.

7.11.6 $\rightarrow$ Introduction

$$\frac{\begin{array}{c}\mathrm{A}\\ \vdots \\ \mathrm{C}\end{array}}{\mathrm{A} \rightarrow \mathrm{C}}$$

This rule shows that if in carrying out a proof we start from an assumption A and derive a conclusion C, then we can conclude that $A \rightarrow C$.

7.11.7 $\neg\neg$ Elimination

$$\frac{\neg\neg \mathrm{A}}{\mathrm{A}}$$

This rule states that if we have a sentence that is negated twice, we can conclude the sentence itself, without the negation. Clearly, this rule follows from the definition of $\neg$.

7.11.8 Example 1

To carry out a proof that one set of sentences follows logically from another, we selectively apply the rules presented above to the assumptions until we arrive at the conclusions.

For example, it would be useful to prove the following:

$$\{\mathrm{A}, \neg \mathrm{A}\} \vdash \bot$$

In other words, if we start from the set of assumptions A and $\neg A$, we can conclude falsum.

First, note that

$$\neg \mathrm{A} \equiv \mathrm{A} \rightarrow \bot$$

This can be seen by comparing the truth tables for $\neg A$ and for $A \rightarrow \bot$.

Hence, we can take as our set of assumptions

$$\{A, A \rightarrow \bot\}$$

Thus, our proof using modus ponens (the $\rightarrow$ ELIMINATION rule presented in Section 7.11.2) is as follows:

$$\frac{A \quad A \rightarrow \bot}{\bot}$$

7.11.9 Example 2

Let us prove the following:

$$\{A \wedge B\} \vdash A \vee B$$

The proof is as follows:

$\underline{A \wedge B}$	assumption
$\underline{\quad A}$	by $\wedge$ elimination
$A \vee B$	by $\vee$ introduction

7.11.10 Example 3

We will use reductio ad absurdum to prove the following:

$$\vdash (\neg A \rightarrow B) \rightarrow (\neg B \rightarrow A)$$

The usual method for carrying out such proofs is based on the idea that in order to prove something of the form $A \rightarrow B$, it is a good idea to start by assuming A.

We will start with two assumptions: $\neg A$ and $(\neg A \rightarrow B)$. After the first step, which uses modus ponens, on our original assumptions to prove B, we introduce a new assumption, which is $\neg B$. The proof is as follows:

$\underline{\neg A \quad \neg A \rightarrow B}$	assumptions
$\underline{B \quad \neg B}$	modus ponens
$\underline{B \quad B \rightarrow \bot}$	rewriting $\neg B$
$\underline{\bot}$	modus ponens
$\underline{A}$	reductio ad absurdum
$\underline{\neg B \rightarrow A}$	$\rightarrow$ introduction
$(\neg A \rightarrow B) \rightarrow (\neg B \rightarrow A)$	$\rightarrow$ introduction

In carrying out this proof, we have used the relationship between $\neg B$ and $B \rightarrow \perp$ as we did in Example 1. We have also used reductio absurdum to show that if we start by assuming $\neg A$, we end up with a contradiction ($\perp$), and therefore our initial assumption, $\neg A$, was *false*. Hence, A must be true.

7.11.11 Example 4

Let us now aim to prove the following:

$$\vdash (A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow ((C \rightarrow D) \rightarrow (A \rightarrow D)))$$

To prove this, we will need to make a series of assumptions. We will start by making two assumptions, A and $A \rightarrow B$. Hence, our proof is as follows:

$$\begin{array}{rl}
\underline{A \quad A \rightarrow B} & \text{assumptions} \\
\underline{B \quad B \rightarrow C} & \text{modus ponens} \\
\underline{C \quad C \rightarrow D} & \text{modus ponens} \\
\underline{D} & \text{modus ponens} \\
\underline{A \rightarrow D} & \rightarrow \text{introduction} \\
\underline{(C \rightarrow D) \rightarrow (A \rightarrow D)} & \rightarrow \text{introduction} \\
\underline{(B \rightarrow C) \rightarrow ((C \rightarrow D) \rightarrow (A \rightarrow D))} & \rightarrow \text{introduction} \\
(A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow ((C \rightarrow D) \rightarrow (A \rightarrow D))) & \rightarrow \text{introduction}
\end{array}$$

7.12 The Deduction Theorem

A useful rule known as the **deduction theorem** provides us with a way to make propositional logic proofs easier. The rule is as follows:

$$\text{if } A \bigcup \{B\} \vdash C \text{ then } A \vdash (B \rightarrow C)$$

Here A is a set of wff's, which makes up our assumptions. Note that this rule is *true* even if A is the empty set. $A \bigcup \{B\}$ means the **union** of the set A with the set consisting of one element, B.

The rule also holds in reverse:

$$\text{if } A \vdash (B \rightarrow C) \text{ then } A \bigcup \{B\} \vdash C$$

Let us see an example of a proof using the deduction theorem.

Our aim is to prove the following:

$$\{A \rightarrow B\} \vdash A \rightarrow (C \rightarrow B)$$

Recall the axiom that was presented earlier:

$$A \rightarrow (B \rightarrow A)$$

Because propositional logic is monotonic (see Section 7.18), we can add in an additional assumption, that *A* is *true*:

$$A$$

Now, by applying modus ponens to this assumption and our hypothesis, $A \rightarrow B$, we arrive at

$$B$$

We can now apply our axiom

$$B \rightarrow (C \rightarrow B)$$

And by modus ponens on the above two lines, we get

$$C \rightarrow B$$

Hence, we have shown that

$$\{A \rightarrow B\} \bigcup A \vdash (C \rightarrow B)$$

And, therefore, by the deduction theorem

$$\{A \rightarrow B\} \vdash A \rightarrow (C \rightarrow B)$$

7.13 Predicate Calculus

7.13.1 Syntax

Predicate calculus allows us to reason about properties of objects and relationships between objects. In propositional calculus, we could express the English statement "I like cheese" by *A*. This enables us to create constructs such as $\neg A$, which means "I do not like cheese," but it does not allow us to extract any information about the cheese, or me, or other things that I like.

In predicate calculus, we use **predicates** to express properties of objects. So the sentence "I like cheese" might be expressed as

$$L(\text{me}, \text{cheese})$$

where *L* is a predicate that represents the idea of "liking." Note that as well as expressing a property of *me*, this statement also expresses a relationship between *me* and *cheese*. This can be useful, as we will see, in describing environments for robots and other **agents.** For example, a simple agent may

be concerned with the location of various blocks, and a statement about the world might be

$$T(A,B)$$

which could mean: Block *A* is on top of Block *B*.

Thus far we have expressed ideas about specific objects. It is also possible to make more general statements using the predicate calculus. For example, to express the idea that everyone likes cheese, we might say

$$(\forall x)(P(x) \rightarrow L(x, C))$$

The symbol $\forall$ is read "for all," so the statement above could be read as "for every *x* it is true that if property *P* holds for *x*, then the relationship *L* holds between *x* and *C*," or in plainer English: "every *x* that is a person likes cheese." (Here we are interpreting $P(x)$ as meaning "*x* is a person" or, more precisely, "*x* has property *P*.")

Note that we have used brackets rather carefully in the statement above. This statement can also be written with fewer brackets:

$$\forall x\ P(x) \rightarrow L(x, C)$$

$\forall$ is called the **universal quantifier**.

The quantifier $\exists$ can be used to express the notion that some values do have a certain property, but not necessarily all of them:

$$(\exists x)(L(x,C))$$

This statement can be read "there exists an *x* such that *x* likes cheese." This does not make any claims about the possible values of *x*, so *x* could be a person, or a dog, or an item of furniture. When we use the **existential quantifier** in this way, we are simply saying that there is *at least one* value of *x* for which $L(x,C)$ holds.

Note, therefore, that the following is true:

$$(\forall x)(L(x,C)) \rightarrow (\exists x)(L(x,C))$$

but the following is not:

$$(\exists x)(L(x,C)) \rightarrow (\forall x)(L(x,C))$$

7.13.2 Relationships between $\forall$ and $\exists$

It is also possible to combine the universal and existential quantifiers, such as in the following statement:

$$(\forall x)\ (\exists y)\ (L(x,y))$$

This statement can be read "for all x, there exists a y such that L holds for x and y," which we might interpret as "everyone likes something."

A useful relationship exists between $\forall$ and $\exists$. Consider the statement "not everyone likes cheese." We could write this as

$$\neg(\forall x)(P(x) \rightarrow L(x,C)) \tag{1}$$

As we have already seen, $A \rightarrow B$ is equivalent to $\neg A \vee B$. Using DeMorgan's laws, we can see that this is equivalent to $\neg(A \wedge \neg B)$. Hence, the statement (1) above, can be rewritten:

$$\neg(\forall x)\neg(P(x) \wedge \neg L(x,C)) \tag{2}$$

This can be read as "It is not true that for all x the following is not true: x is a person and x does not like cheese." If you examine this rather convoluted sentence carefully, you will see that it is in fact the same as "there exists an x such that x is a person and x does not like cheese." Hence we can rewrite it as

$$(\exists x)(P(x) \wedge \neg L(x,C)) \tag{3}$$

In making this transition from statement (2) to statement (3), we have utilized the following equivalence:

$$\exists x \equiv \neg(\forall x)\neg$$

In an expression of the form $(\forall x)(P(x, y))$, the variable x is said to be **bound**, whereas y is said to be **free**. This can be understood as meaning that the variable y could be replaced by any other variable because it is free, and the expression would still have the same meaning, whereas if the variable x were to be replaced by some other variable in $P(x,y)$, then the meaning of the expression would be changed:

$$(\forall x)(P(y, z))$$

is not equivalent to $(\forall x)(P(x, y))$, whereas $(\forall x)(P(x, z))$ is. Note that a variable can occur both bound and free in an expression, as in

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

In this expression, x is bound throughout, and z is free throughout; y is free in its first occurrence but is bound in $(\exists y)(Q(y,z))$. (Note that both occurrences of y are bound here.)

Making this kind of change is known as **substitution**. Substitution is allowed of any free variable for another free variable.

7.13.3 Functions

In much the same way that **functions** can be used in mathematics, we can express an object that relates to another object in a specific way using functions. For example, to represent the statement "my mother likes cheese," we might use

L(m(me),cheese)

Here the function $m(x)$ means the mother of x. Functions can take more than one argument, and in general a function with n arguments is represented as

$f(x_1, x_2, x_3, \ldots, x_n)$

7.14 First-Order Predicate Logic

The type of predicate calculus that we have been referring to is also called **first-order predicate logic** (**FOPL**). A first-order logic is one in which the quantifiers $\forall$ and $\exists$ can be applied to objects or **terms**, but not to predicates or functions. So we can define the syntax of FOPL as follows. First, we define a term:

A constant is a term.

A variable is a term.

$f(x_1, x_2, x_3, \ldots, x_n)$ is a term if $x_1, x_2, x_3, \ldots, x_n$ are all terms.

Anything that does not meet the above description cannot be a term. For example, the following is not a term: $\forall x\, P(x)$. This kind of construction we call a sentence or a well-formed formula (wff), which is defined as follows. In these definitions, P is a predicate, $x_1, x_2, x_3, \ldots, x_n$ are terms, and A,B are wff's. The following are the acceptable forms for wff's:

$P(x_1, x_2, x_3, \ldots, x_n)$

$\neg A$

$A \wedge B$

$A \vee B$

$A \rightarrow B$

$A \leftrightarrow B$

$(\forall x)A$

$(\exists x)A$

An **atomic formula** is a wff of the form $P(x_1, x_2, x_3, \ldots, x_n)$.

Higher order logics exist in which quantifiers can be applied to predicates and functions, and where the following expression is an example of a wff:

$$(\forall P)(\exists x)P(x)$$

In this book, we will stick with first-order logics, in which quantifiers can only be applied to variables, not predicates or functions.

7.15 Soundness

We have seen that a logical system such as propositional logic consists of a syntax, a semantics, and a set of rules of deduction. A logical system also has a set of fundamental truths, which are known as axioms. The axioms are the basic rules that are known to be true and from which all other **theorems** within the system can be proved.

An axiom of propositional logic, for example, is

$$A \rightarrow (B \rightarrow A)$$

A theorem of a logical system is a statement that can be proved by applying the rules of deduction to the axioms in the system.

If *A* is a theorem, then we write

$$\vdash A$$

A logical system is described as being **sound** if every theorem is logically valid, or a tautology.

It can be proved by induction that both propositional logic and FOPL are sound.

7.16 Completeness

A logical system is **complete** if every tautology is a theorem—in other words, if every valid statement in the logic can be proved by applying the rules of deduction to the axioms. Both propositional logic and FOPL are complete. The proofs that these systems are complete are rather complex.

7.17 Decidability

A logical system is decidable if it is possible to produce an algorithm that will determine whether any wff is a theorem. In other words, if a logical system is decidable, then a computer can be used to determine whether logical expressions in that system are valid or not.

We can prove that propositional logic is decidable by using the fact that it is complete. Thanks to the completeness of propositional logic, we can prove that a wff A is a theorem by showing that it is a tautology. To show if a wff is a tautology, we simply need to draw up a truth table for that wff and show that all the lines have *true* as the result. This can clearly be done algorithmically because we know that a truth table for n values has 2^n lines and is therefore finite, for a finite number of variables.

FOPL, on the other hand, is not decidable. This is due to the fact that it is not possible to develop an algorithm that will determine whether an arbitrary wff in FOPL is logically valid.

7.18 Monotonicity

A logical system is described as being monotonic if a valid proof in the system cannot be made invalid by adding additional premises or assumptions. In other words, if we find that we can prove a conclusion C by applying rules of deduction to a premise B with assumptions A, then adding additional assumptions A' and B' will not stop us from being able to deduce C.

Both propositional logic and FOPL are monotonic. Elsewhere in this book, we learn about probability theory, which is not a monotonic system.

Monotonicity of a logical system can be expressed as follows:

> If we can prove $\{A, B\} \vdash C$,
>
> then we can also prove: $\{A, B, A', B'\} \vdash C$.

Note that A' and B' can be anything, including $\neg A$ and $\neg B$. In other words, even adding contradictory assumptions does not stop us from making the proof in a monotonic system. In fact, it turns out that adding contradictory assumptions allows us to prove *anything*, including invalid conclusions.

This makes sense if we recall the line in the truth table for $\rightarrow$, which shows that *false* $\rightarrow$ *true*. By adding a contradictory assumption, we make our assumptions *false* and can thus prove any conclusion.

7.19 Abduction and Inductive Reasoning

The kind of reasoning that we have seen so far in this chapter has been **deductive reasoning**, which in general is based on the use of modus ponens and the other deductive rules of reasoning. This kind of reasoning assumes

that we are dealing with certainties and does not allow us to reason about things of which we are not certain. As we see elsewhere in this book, there is another kind of reasoning, **inductive reasoning**, which does not have the same logical basis but can be extremely powerful for dealing with situations in which we lack certainty.

Strangely, another form of reasoning, **abduction**, is based on a common fallacy, which can be expressed as

$$\frac{B \qquad A \rightarrow B}{A}$$

Note that abduction is very similar to modus ponens but is not logically sound. A typical example of using this rule might be "When Jack is sick, he doesn't come to work. Jack is not at work today. Therefore Jack is sick."

In fact, Jack may be having a holiday, or attending a funeral, or it may be Sunday or Christmas Day.

Given that this type of reasoning is invalid, why are we discussing it here? It turns out that although abduction does not provide a logically sound model for reasoning, it does provide a model that works reasonably well in the real world because it allows us to observe a phenomenon and propose a possible explanation or cause for that phenomenon without complete knowledge. Abductive reasoning is discussed in more detail in Chapter 17.

Inductive reasoning enables us to make predictions about what will happen, based on what has happened in the past. Humans use inductive reasoning all the time without realizing it. In fact, our entire lives are based around inductive reasoning, for example, "the sun came up yesterday and the day before, and every day I know about before that, so it will come up again tomorrow." It's possible it won't, but it seems fairly unlikely. This kind of reasoning becomes more powerful when we apply probabilities to it, as in "I've noticed that nearly every bird I see is a swallow. Therefore, it's quite likely that that bird is a swallow."

As we will see, these kinds of reasoning are extremely useful for dealing with uncertainty and are the basis of most of the learning techniques used in Artificial Intelligence.

7.20 Modal Logics and Possible Worlds

The forms of logic that we have dealt with so far deal with facts and properties of objects that are either true or false. In these **classical logics**, we do not consider the possibility that things change or that things might not always be as they are now.

Modal logics are an extension of classical logic that allow us to reason about possibilities and certainties. In other words, using a modal logic, we can express ideas such as "although the sky is usually blue, it isn't always" (for example, at night). In this way, we can reason about possible worlds. A possible world is a universe or scenario that could logically come about.

The following statements may not be true in our world, but they are possible, in the sense that they are not illogical, and could be true in a possible world:

> Trees are all blue.
>
> Dogs can fly.
>
> People have no legs.

It is possible that some of these statements will become true in the future, or even that they were true in the past. It is also possible to imagine an alternative universe in which these statements are true now. The following statements, on the other hand, cannot be true in any possible world:

$$A \wedge \neg A$$

$$(x > y) \wedge (y > z) \wedge (z > x)$$

The first of these illustrates the **law of the excluded middle**, which simply states that a fact must be either true or false: it cannot be both true and false. It also cannot be the case that a fact is neither true nor false. This is a law of classical logic, and as we see in Chapter 18, it is possible to have a logical system without the law of the excluded middle, and in which a fact can be both true *and* false.

The second statement cannot be true by the laws of mathematics. We are not interested in possible worlds in which the laws of logic and mathematics do not hold.

A statement that may be true or false, depending on the situation, is called **contingent**. A statement that must always have the same truth value,

regardless of which possible world we consider, is **noncontingent**. Hence, the following statements are contingent:

$A \wedge B$

$A \vee B$

I like ice cream.

The sky is blue.

The following statements are noncontingent:

$A \vee \neg A$

$A \wedge \neg A$

If you like all ice cream, then you like this ice cream.

Clearly, a noncontingent statement can be either true or false, but the fact that it is noncontingent means it will always have that same truth value.

If a statement A is contingent, then we say that *A* is **possibly** true, which is written

$\Diamond A$

If A is noncontingent, then it is **necessarily** true, which is written

$\Box A$

7.20.1 Reasoning in Modal Logic

It is not possible to draw up a truth table for the operators $\Diamond$ and $\Box$. (Consider the four possible truth tables for unary operators—it should be clear that none of these matches these operators.) It is possible, however, to reason about them.

The following rules are examples of the axioms that can be used to reason in this kind of modal logic:

$\Box A \rightarrow \Diamond A$

$\Box \neg A \rightarrow \neg \Diamond A$

$\Diamond A \rightarrow \neg \Box A$

Although truth tables cannot be drawn up to prove these rules, you should be able to reason about them using your understanding of the meaning of the $\Box$ and $\Diamond$ operators.

Modal logic, and other nonclassical logics, are discussed in Chapter 17.

7.21 Dealing with Change

As we have seen, classical logics do not deal well with change. They assume that if an object has a property, then it will always have that property and always has had it. Of course, this is not true of very many things in the real world, and a logical system that allows things to change is needed. The situation and event calculi are covered in more detail in Chapters 17 and 19.

7.22 Chapter Summary

- Logic is primarily concerned with the logical validity of statements, rather than with truth.
- Logic is widely used in Artificial Intelligence as a representational method.
- Abduction and inductive reasoning are good at dealing with uncertainty, unlike classical logic.
- The main operators of propositional logic are $\wedge$, $\vee$, $\neg$, $\rightarrow$, and $\leftrightarrow$ (and, or, not, implies, and iff).
- The behavior of these logical operators can be expressed in truth tables. Truth tables can also be used to solve complex problems.
- Propositional logic deals with simple propositions such as "I like cheese." First-order predicate logic allows us to reason about more complex statements such as "All people who eat cheese like cats," using the quantifiers $\forall$ and $\exists$ ("for all", and "there exists").
- A statement that is always true in any situation is called a tautology. $A \vee \neg A$ is an example of a tautology.
- Two statements are logically equivalent if they have the same truth tables.
- First-order predicate logic is sound and complete, but not decidable. Propositional logic is sound, complete, and decidable.
- Modal logics allow us to reason about certainty.

7.23 Review Questions

7.1 Explain the meanings of the following terms in the context of logic:

a. truth

b. validity

c. equivalent
d. uncertainty
e. tautology
f. satisfiable
g. sound
h. complete
i. decidable
j. modal logic

7.2 "Inductive reasoning is a reasonable way to think about everyday life, but it does not provide the logical structure that propositional logic does." Discuss.

7.3 Explain what is meant by the following: "Classical logics are not good at dealing with uncertainty."

7.4 Explain why the addition of the quantifiers $\forall$ and $\exists$ makes predicate calculus so powerful.

7.5 Explain the rule of modus ponens. Explain how it is used in everyday life.

7.6 Explain in layman's terms what the law of the excluded middle means. What difficulties might you encounter in logical deduction if you ignored the law of the excluded middle?

7.7 Assume the law of the excluded middle is not true, and use this to prove the equality $1 = 0$.

7.8 What does it mean to say that a logic is monotonic? Is propositional logic monotonic? What complexities do you think nonmonotonicity would add to the process of logical deduction? Would modus ponens still hold in a nonmonotonic logic?

7.24 Exercises

7.1 Translate the following sentences into logical statements, using either propositional or predicate logic as appropriate:

a. I like apples and pears.
b. When I eat apples and pears, I usually like to have a walk.
c. Every apple that I have ever eaten has been delicious.
d. The fact that some pears are not delicious will not stop me eating them.

e. I can only eat an apple if I have first eaten a pear, and I can only eat a pear if I eat an apple immediately afterward.

f. There exists a book that includes details of every book.

g. There exists somewhere in the world a book that lists every single person who doesn't appear in any other book.

h. If you haven't read the book that lists all other books, then you haven't read any book, unless you've read the book that lists books that do not exist, in which case you've read every book.

7.2 Draw a truth table for the following expression:

$\neg A \wedge (A \vee B) \wedge (B \vee C)$

7.3 (Hard) Prove that propositional logic and first-order predicate logic are sound and complete.

7.4 Write expressions in propositional calculus to represent the following statements:

a. If you go to Mexico, you will be far away.

b. I cannot hear you when you are far away.

c. When I can't hear you, I forget what you look like.

d. If I come to Mexico, and I don't know what you look like, I won't be able to find you.

e. Therefore, if you go to Mexico, and I follow you, I won't be able to find you.

Prove whether the conclusion follows from the premises or not.

7.5 Write expressions in first-order predicate logic to represent the following statements, and prove whether the conclusion follows from the premises or not:

a. All dancers love to dance.

b. Everyone who sings and plays an instrument loves to dance.

c. Therefore, all dancers sing and play an instrument.

7.6 Prove the following:

a. $\vdash A \rightarrow A$

b. $\vdash ((\neg A \rightarrow \neg B) \rightarrow A) \rightarrow ((\neg B \rightarrow \neg A) \rightarrow \neg B)$

c. $\vdash (\neg \neg \neg A \rightarrow \neg \neg \neg B) \rightarrow (\neg A \rightarrow \neg B)$

7.25 Further Reading

Most textbooks on Artificial Intelligence provide good coverage of logic. An excellent, short introduction to logic that provides more detail on most of the subject than has been provided here is Kelly (1997). Lewis Carroll's work, though over 100 years old, still makes for an interesting and relevant read on the subject of logic and reasoning, although his approach was rather different from that usually found today.

The idea of abduction was introduced by C. S. Peirce, in his 1878 paper *How to Make Our Ideas Clear*, published in *Popular Science Monthly*.

Francis Bacon introduced the idea of inductive reasoning in 1620. His writings on the subject can be found in *The New Organon, and Related Writings*, published in 1960.

Francis Bacon: The New Organon by Francis Bacon, edited by Lisa Jardine and Michael Silverthorne (2002 – Cambridge University Press)

Propositional Logic: Deduction and Algorithms by Hans Kleine Büning and Theodor Lettmann (1999 – Cambridge University Press)

Symbolic Logic and *Game of Logic* by Lewis Carroll (published in one volume – 1958 – Dover Books).

Predicate Logic: The Semantic Foundations of Logic by Richard L. Epstein (2000 – Wadsworth Publishing)

Propositional Logics: The Semantic Foundations of Logic by Richard L. Epstein (2000 – Wadsworth Publishing)

The Essence of Logic by John Kelly (1997 – Prentice Hall)

Introduction to Logic: Propositional Logic by Howard Pospesel (1999 – Prentice Hall)

Logic for Computer Science by Steve Reeves and Michael Clarke (1993 – Addison Wesley)

Logical Forms: An Introduction to Philosophical Logic by Mark Sainsbury (1991 – Blackwell)

Logic and Prolog by Richard Spencer-Smith (1991 – Harvester Wheatsheaf)

CHAPTER 8

Inference and Resolution for Problem Solving

Early work in theorem proving programs for quantified logics culminated in 1965 with Alan Robinson's development of machine-oriented formulation of first-order logic called Resolution (Robinson, 1965). There followed an immensely productive period of exploration of resolution-based theorem-proving.

—Alan Newell, *The Knowledge Level*

When you have eliminated the impossible, whatever remains, no matter how improbable, must be the truth.

—Sir Arthur Conan Doyle, *The Sign of Four*

At thirty a man suspects himself a fool;
Knows it at forty, and reforms his plan;
At fifty chides his infamous delay,
Pushes his prudent purpose to resolve;
In all the magnanimity of thought
Resolves; and re-resolves; then dies the same.

—Edward Young, *Night Thoughts*

8.1 Introduction

This chapter introduces the main ideas behind automated reasoning, or theorem proving. The method of resolution is discussed in some detail because it pertains to both propositional logic and first-order predicate logic (FOPL).

To explain resolution, ideas such as unification, normal forms, and Herbrand universes are introduced. This chapter is somewhat more advanced