2.12 BOOLEAN ALGEBRA

In 1854, the English mathematician George Boole published his magnum opus, *An Investigation of the Laws of Thought*. In this work he applied mathematical methods to the study of logic as it apparently guides our thinking (even when we are not aware of using it), and he introduced an algebra for the manipulation of symbols that represent statements. The formulation of this work that we shall give adheres to modern usage and differs somewhat from Boole's original schema, but the basic ideas are the same.

A "statement" (or "proposition") in logic, unlike its use as an equivalent of "instruction" in many programming languages, is a verbal or symbolic (or both) expression that we can assert to be either true or false. Thus: "some roses are red," "$2 + 3 = 4$," "computers can be used for either good or evil,"

" 'VENUS' follows 'ADONIS' in alphabetic ordering," are examples of statements. Every statement has a "truth value" of either TRUE or FALSE. Or, as in many of those programming languages that include facilities for the handling of such expressions, we shall say that the value of the statement is 1 if it is true, 0 if it is false.

Such expressions as "$5 + 6 * 7$," "thou shalt not follow a multitude to do evil," "twas brillig and the slithy toves did tyre and gimble in the wabe," "who is Sylvia, what is she?", are not statements for it is not meaningful to speak of their being either true or false.

In some cases it may be extremely difficult, or perhaps impossible, to determine if a given statement is true or false, although we may concede that one or the other possibility must hold. Such statements as "there are no positive whole numbers x, y, z, n with $n > 2$ that satisfy the equation $x^n + y^n = z^n$," or "there are five consecutive 7's somewhere in the infinite decimal representation of π" are of this kind.

In still other instances it may be quite questionable as to whether an apparent statement can even be said to have truth value, even if we ignore the matter of determining whether it be true or false. Such declarations as "life has purpose," "God exists," "there exist real numbers that can not be individually described or individually characterized or referred to in any manner whatsoever,"[1] may be in this category. We leave to the reader the adjudication of this matter for each of our sample declarations. In fact, Niels Bohr has noted that often a statement that appears to be "profound" is characterized by the fact that both that statement *and* its contradiction seem to possess some deep validity. Our last examples may be in this category.

The conditional statements we use in writing programs, however, involve arithmetic operations and are finitely verifiable, not subject to any uncertainty because of a possibly clouded meaning of truth or falsity.

Relations become mathematical statements when particular numbers are substituted for the arguments. Thus, the relation "$y \geq x + 2$" becomes for $x = 6$, $y = 3$ the false statement "$3 \geq 8$." Such relations on the natural numbers are also known as *predicates*.

We shall now take the first steps toward defining a formal system for the study and manipulation of statements.

1. We denote arbitrary statements, or propositions, by the letters A, B, $C, \ldots$. Each such symbol represents a "Boolean variable" which can take on the value 0 for FALSE or 1 for TRUE—just as in elementary algebra we let

1. In fact, there is a general belief by mathematicians that in a certain sense more of these surprisingly unapproachable numbers exist than of the other kind. See the discussion of transfinite numbers in Section 3.7.

symbols like x, y, z represent variables that can take on values that are real numbers.

We shall also use the symbols "$\neg$" for NOT, "|" for OR, and "&" for AND. If A is the statement "…," then "$\neg A$" shall denote the statement "it is not true that '…'," or "'…' is false," or simply "NOT A." $\neg A$ is also called "the negation of A." The symbol "$\neg$" is a *unary* connective, for it applies to a single statement. $\neg$A is TRUE (or 1) if A is FALSE (0), and FALSE if A is TRUE. Table 2.1 shows the values of $\neg A$ for the two possible values of A.

TABLE 2.1. TRUTH TABLE FOR $\neg A$

A	$\neg A$
0	1
1	0

If B is the statement "xxx" then "$A \mid B$" shall denote the statement "either '…' or 'xxx' is true (and perhaps both are true)." $A \mid B$ is also known as the "disjunction" of A and B. Again, we can tablulate the values of $A \mid B$ (giving rise to a "truth table") for each of the four possible combinations of values of A and B.

TABLE 2.2. TRUTH TABLE FOR $A \mid B$

A	B	$A \mid B$
0	0	0
0	1	1
1	0	1
1	1	1

"$A \,\&\, B$" shall denote the statement "both '…' and 'xxx' are true." $A \,\&\, B$ may also be called the "conjunction of A and B." Both "|" and "&" are binary connectives.

TABLE 2.3. TRUTH TABLE FOR $A \,\&\, B$

A	B	$A \,\&\, B$
0	0	0
0	1	0
1	0	0
1	1	1

Other symbols that are sometimes used for these same connectives are as follows:

$$\neg A\text{: } -A, \overline{A}, \sim A, A', \text{NOT } A$$
$$A \mid B\text{: } A \vee B, A + B, A \oplus B, A \text{ OR } B$$
$$A \,\&\, B\text{: } A \wedge B, A \cdot B, AB, A * B, A \text{ AND } B.$$

2. The wff's, or "statements" of our system are those expressions or symbolic strings that we can form using the symbols for Boolean variables and applying any number of times the three connectives. To avoid ambiguity, we shall also have to use the left and right parentheses (,) just as in elementary algebra we use parentheses to make clear the interpretation of complex expressions. Let "*BE*" represent this class of wff's or Boolean expressions. We can now define, using the BNF notation, how the symbolic strings that represent the members of this class are formed.

$$<\text{Boolean variable}> ::= A \mid B \mid C \mid \ldots \mid Z$$

(We can assume an alphabet of any size.)

$$<BE> ::= <\text{Boolean variable}> \mid (<BE> \mid <BE>) \mid \neg <BE> \mid (<BE> \,\&\, <BE>).$$

Every Boolean expression defines a Boolean function that yields a value of 0 or 1 for every possible set of truth values assigned to the Boolean variables of which it is composed. The value for each combination of arguments can be computed by repeated reference to the truth tables for $\neg A, A \mid B, A \,\&\, B$.

Thus, the truth table for the Boolean expression

$$((A \,\&\, (B \mid C)) \mid \neg B)$$

can be computed as shown in Table 2.4.

TABLE 2.4. TRUTH TABLE FOR $((A \,\&\, (B \mid C)) \mid \neg B)$

A	B	C	$(B \mid C)$	$(A \,\&\, (B \mid C))$	$\neg B$	$((A \,\&\, (B \mid C)) \mid \neg B)$
0	0	0	0	0	1	1
0	0	1	1	0	1	1
0	1	0	1	0	0	0
0	1	1	1	0	0	0
1	0	0	0	0	1	1
1	0	1	1	1	1	1
1	1	0	1	1	0	1
1	1	1	1	1	0	1

We note that $((A \,\&\, (B \mid C)) \mid \neg B)$ takes on the same truth values for different combinations of arguments as the much simpler expression $(A \mid \neg B)$.

3. In this particular formal system, the wff's, or the statements about propositions or statements are themselves statements, as we shall see, and there-

fore do not comprise new objects in the system. We shall introduce two new symbols involved in making statements about statements.

The symbol "=" appearing between two Boolean expressions as in $(A \mid \neg B) = ((A \,\&\, (B \mid C)) \mid \neg B)$ shall mean not that the two expressions appearing on both sides of the "=" are identical but, rather, that for given values of the Boolean variables which appear in them they have the same truth value—one is not true while the other is false. (In mathematics, the symbol "≡" is often used for "is identical with," as opposed to the "=" that is usually used in a sense similar to that we have just given.) This is consistent with our use of the "=" in elementary algebra. For example, when we write

$$x^2 - y^2 - 2x + 1 = (x + y - 1)(x - y - 1),$$

we mean that the two forms on both sides of the "=" take on the same values for all admissible values of x, y.

We note, however, that the "=" is expressible in terms of the other connectives; the truth table for $A = B$ is identical with that for $(\neg A \,\&\, \neg B \mid (A \,\&\, B)$.

TABLE 2.5

A	B	$A = B$	$(\neg A \,\&\, \neg B) \mid (A \,\&\, B)$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	1	1

Therefore, $A = B$ can always be replaced by this other expression which involves the connectives $\neg$, &, |.

The symbol "⊃" between two expressions, as in $A \supset B$, denotes "implies," and this expression is read as "A implies B" or "if '…' then 'xxx'." The arrow "→" is often used for this purpose as well, $A \to B$ having the same meaning as $A \supset B$. By definition, $A \supset B$ has the same meaning as $(\neg A \mid B)$ or "either A is false or B is true" (to see the reasonableness of this definition, note that if "either A is false or B is true" then "if A is true, B must be true," and conversely). From this definition, the truth table for $A \supset B$ is as shown in Table 2.6.

TABLE 2.6 TRUTH TABLE FOR $A \supset B$

A	B	$A \supset B$
0	0	1
0	1	1
1	0	0
1	1	1

What we have done so far constitutes the definition of the *syntax* of this formal language, the rules for forming those well-formed symbolic strings that represent the objects and formulas of the system. It remains to attach "meaning" to, or to define the *semantics* of, these expressions (cf. the discussion of the syntax and semantics of formal languages in Chapter 9). This can be done through our use of truth tables. Every Boolean expression defines a truth table and this table implies the meaning of that expression. For example, if we keep in mind the association of "1" with TRUE and "0" with FALSE, we see that the table for "$A \mid B$" indicates that this expression denotes a statement that is TRUE when either A or B (possibly both) is TRUE. Similarly, "$A \rightarrow B$" is an expression that represents a statement which is TRUE if A is FALSE or if B is TRUE.

If we are given the "double implication," both $A \rightarrow B$ and $B \rightarrow A$ (i.e., $(A \rightarrow B)$ & $(B \rightarrow A)$), this is often written $A \leftrightarrow B$. This is then the same as $A = B$, for it is quickly seen that the truth table for this conjunction is identical with that for $A = B$. In other words, we can say that two statements are "equivalent" if each implies the other. We note from the table that $A \supset B$ is true when A is false, independently of the value of B. Or, in terms of the verbal interpretation of these logical symbols, a false statement implies any statement, whether that statement be true or false. It may not be clear at this point in our discussion that this means we can "prove" the truth of any statement at all if we accept as hypothesis a false statement, but such is the case. Some years ago, Bertrand Russell, after having mentioned this fact in a lecture, was challenged by a member of the audience to deduce from the false statement "$2 + 2 = 3$" that he was the Pope. "That is easy," Russell immediately replied. "From $2 + 2 = 3$ we conclude, subtracting 2 from both sides, that $2 = 1$. I and the Pope are two; therefore, we are one." However, this theorem that a false hypothesis implies any conclusion whatsoever has probably been used more widely by politicians than by mathematicians.

If we have $A \rightarrow B$, we often say that "B is a necessary condition for A." That is, A cannot be true unless B is true. We also say that "A is a sufficient condition for B." That is, it suffices for A to be true in order for B to be true.

Thus, if A is the statement "Mary is a good programmer," and B is the statement "Mary includes ample comments in her program documentation," A is a sufficient condition for B, and B is a necessary condition for A.

Tautologies. Some Boolean expressions take on the constant value 1 for all possible assignments of values to the Boolean variables of which they are composed (they are always true). Consider the following expression S,

$$(A \supset B) \supset ((C \mid A) \supset (C \mid B)).$$

The truth table for this expression is computed as in Table 2.7.

TABLE 2.7

A	B	C	$(A \supset B)$	$(C \mid A)$	$(C \mid B)$	$((C \mid A) \supset (C \mid B))$	S
0	0	0	1	0	0	1	1
0	0	1	1	1	1	1	1
0	1	0	1	0	1	1	1
0	1	1	1	1	1	1	1
1	0	0	0	1	0	0	1
1	0	1	0	1	1	1	1
1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1

We see that S is identically 1; that is, it is always true, no matter what the truth values of the statements A, B, C may be. Such statements are called "tautologies." They comprise the theorems of our system.

The method of truth tables can be used to verify quickly that the following are tautologies:

i. $(A \,\&\, B) = (B \,\&\, A)$
$((A \,\&\, B) \,\&\, C) = (A \,\&\, (B \,\&\, C))$,

i.e., the connective AND is commutative and associative. Thus, we can write $A \,\&\, B \,\&\, C$ without parentheses.

ii. $(A \mid B) = (B \mid A)$
$((A \mid B) \mid C) = (A \mid (B \mid C))$,

i.e., the connective OR is commutative and associative. Thus, we can write $A \mid B \mid C$ without parentheses.

iii. $(A \,\&\, (B \mid C)) = (A \,\&\, B) \mid (A \,\&\, C)$,

i.e., AND is distributive over OR.

iv. $(A \mid (B \,\&\, C)) = (A \mid B) \,\&\, (A \mid C)$,

i.e., OR is distributive over AND.

We note the analogy with the laws of elementary algebra if the "&" is thought of as ordinary multiplication and the "|" as addition (with the exception of the last tautology, for addition is not distributive over multiplication). This explains the use of the symbols $*$, $\cdot$ for AND and of $+$ for OR.

4. The technique of using truth tables to determine if a given Boolean statement is a tautology makes it unnecessary in this exposition to postulate a set of basic axioms of the system and to describe how theorems are to be

derived from these axioms. It is quite possible to do this, to enunciate a set o one, two, three, four, or more axioms and to define rules for deducin theorems (modus ponens, substitution, cf. below and the exercises), but th truth tables enable us to determine by a straightforward computation (a effective procedure) if a given statement is always true, i.e., is a tautolog However, while the truth table technique provides an algorithm to verif that a given statement is a theorem, it provides at best a very awkwar mechanism to *discover* new theorems or tautologies. A formulation employ ing axioms and rules of inference would be better for this purpose.

2.13 AN AXIOMATIC FORMULATION OF THE PROPOSITIONAL CALCULUS

One particular axiomatic formulation for a calculus of propositions, or fo the "propositional calculus," is the following:

Axioms:

1. $(P \mid P) \rightarrow P$
2. $P \rightarrow (P \mid Q)$
3. $(P \mid Q) \rightarrow (Q \mid P)$
4. $(P \rightarrow Q) \rightarrow ((R \mid P) \rightarrow (R \mid Q))$

Rules:

1. *Substitution.* If a given Boolean expression is true (a tautology), it wil remain true if any Boolean variable appearing within it is replaced in all it occurrences by an arbitrary Boolean expression.

2. *Modus ponens.* If $X \rightarrow Y$ is a tautology, where X, Y are arbitrary Boolear expressions and X is a tautology, then Y is a tautology.

Definitions: Note that the symbols $\neg$, $=$, and & do not appear in the axioms. Meaning is assigned to expressions containing these symbols ir accordance with the following definitions, which are consistent with ou earlier usage. We shall write "$\overset{\text{Df}}{=}$" between two expressions to mean that the expression to its left "is equal by definition to" the expression to its right

1. $(P = Q) \overset{\text{Df}}{=} (P \rightarrow Q) \,\&\, (Q \rightarrow P)$
2. $(\neg P \mid Q) \overset{\text{Df}}{=} (P \rightarrow Q)$
3. $(P \,\&\, Q) \overset{\text{Df}}{=} \neg(\neg P \mid \neg Q)$

In this formulation, the tautologies include the axioms and all additiona tautologies, or theorems, that can be obtained from the axioms by repeated application of the rules of substitution and modus ponens.

We illustrate the use of these methods to prove several propositions. We shall follow a format similar to that used for proofs in elementary Euclidean geometry. Parentheses are omitted in several places where no confusion results from such omission.

Example 1. Prove the theorem T_1: $(Q \to R) \to ((P \to Q) \to (P \to R))$.

Step		*Reason*
(1) $(Q \to R) \to ((\neg P \mid Q) \to (\neg P \mid R))$		Use rule (1) in axiom (4). Replace P by Q, Q by R, R by $\neg P$.
(2) $(Q \to R) \to ((P \to Q) \to (P \to R))$	QED	Definition (2) used in step (1).

Example 2. Prove the law of the excluded middle T_2: $\neg P \mid P$.

Step		*Reason*
(1) $P \to (P \mid P)$		Use rule (1) in axiom (2). Replace Q by P.
(2) $(P \to (P \mid P)) \to (P \to P)$		Use rule (1) in T_1. Replace Q by $(P \mid P)$, R by P. Then use axiom (1) and rule (2) to assert the right-hand side as a theorem.
(3) $P \to P$		Step (2) and rule (2).
(4) $\neg P \mid P$	QED	Definition (2).

All the theorems of this calculus of statements can be proven in this manner. Some of the proofs may be rather difficult to formulate, unlike the straightforward use of the method of truth tables. The latter is an effective, easily programmable, "decision procedure" for determining the truth or falsity of statements in the propositional calculus.

2.14 THE METHOD OF "REDUCTIO AD ABSURDUM"

Every student of high school plane geometry has seen the use of this method, which is used widely in all of mathematics, to prove propositions. It is applied in the following way to prove a statement P. We assume that P is false (i.e., we assume $\neg P$). Then, under this hypothesis we proceed to prove some absurdity; usually we prove the self-contradicting conclusion that the hypothesis is false. That is, we prove P to be true after having assumed its contradiction $\neg P$. We then conclude that $\neg P$ cannot be true, the assump-

tion of $\neg P$ having been "reduced to the absurd." Therefore, P must be true, for by the "law of the excluded middle" either P or $\neg P$ must be true.

Using the language of the Boolean algebra of logical propositions we are, in applying this method, using the following theorem of logic:

$$(\neg P \rightarrow P) \leftrightarrow P.$$

This theorem is readily proven by the method of truth tables.

We illustrate use of the method by giving the proof, attributed to Euclid, that $\sqrt{2}$ is *not* a rational number, that is, it cannot be expressed in the form p/q where p, q are natural numbers. Assume the contradiction of this proposition, that is, assume $\sqrt{2}$ *is* a rational number p/q. We can further assume that p/q is written in lowest terms or that any common factor has been canceled from numerator and denominator. We then have $\sqrt{2} = p/q$ or, squaring both sides and clearing of fractions, $p^2 = 2q^2$. Hence p^2 is an even number and so must p also be even, say $p = 2m$. Therefore, $(2m)^2 = 2q^2$ or, dividing through by 2, $2m^2 = q^2$ and q is also seen to be an even number.

But now we have arrived at the conclusion that *both* p and q are even, contradicting our hypothesis that they have no common factors. We therefore conclude that our hypothesis must be false and $\sqrt{2}$ can*not* be expressed in the form p/q.

2.15 FORMING A BOOLEAN EXPRESSION WHOSE TRUTH TABLE IS GIVEN

The truth table for a given Boolean expression is, using our earlier terminology, the graph of the function defined by that expression. We have seen that more than one Boolean expression can correspond to a given truth table, there being, in fact, an infinite number of such Boolean expressions.

It is a simple matter to write a particular Boolean expression having a given truth table as its graph. We show this in the following by describing the "disjunctive normal form" of a Boolean function.

Suppose, for example, we are given Table 2.8 for a Boolean function that we tentatively call F.

TABLE 2.8

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

We seek a Boolean expression for F with this given truth table. We note that F is true if and only if (1) A is false, B is false, C is true, or if (2) A is false, B is true, C is false, or if (3) A is true, B is true, C if false. That is, F is true if (1) $(\neg A \,\&\, \neg B \,\&\, C)$, or if (2) $(\neg A \,\&\, B \,\&\, \neg C)$, or if (3) $(A \,\&\, B \,\&\, \neg C)$. Using the connector "|" we can write this last statement as

$$F = (\neg A \,\&\, \neg B \,\&\, C) \mid (\neg A \,\&\, B \,\&\, \neg C) \mid (A \,\&\, B \,\&\, \neg C).$$

This is now a Boolean expression for F formed directly from the truth table. It is called the disjunctive normal form since F is written as a disjunction of terms. Every Boolean function can thus be written as a disjunction of conjunctions of Boolean variables and their negations.

2.16 ON "POLISH NOTATION"

The original use of the parenthesis-free notation bearing the surprisingly nationalistic name that is in common use today was in the propositional calculus. The Polish mathematician, J. Lukasiewicz, a member of the pre-World War II "Great Polish School of Mathematics" (a school closed largely through the efforts of A. Hitler and his followers) introduced the scheme of writing $p \to q$, or $p \supset q$, as Cpq. With this writing of the implication symbol as a prefix, no parentheses are needed in writing complex expressions. For example, the statement

$$(p \supset q) \supset ((q \supset r) \supset (p \supset r))$$

becomes in Polish notation:

$$CCpqCCqrCpr.$$

More generally, any binary operator can be written in front of its two operands. If we have an arithmetic expression involving several different operations such as

1. $((a + b) \div (c \times d)) - (e - f)$

this becomes, in "Polish notation"

2. $- \div + ab \times cd - ef.$

This prefix notation has wide application in compiling. It is a comparatively straightforward task to scan a string like (2) and compile the program that computes it. The handling of (1) is a more complicated affair.

The compelling advantage of the prefix notation is that it is completely unambiguous, requiring no parentheses or additional conventions. An expression like $a + b * c$ is not well-defined unless parentheses are inserted or some precedence relationship is stated about the arithmetic operations (such as, do multiplications before additions). However, when written in prefix

notation, no such conventions are needed. Thus, the two possible interpretations of this expression would be in prefix notation the two different strings, $+ a * bc$ and $* + abc$, according respectively to whether the multiplication or the addition is done first.

A slight variant of this notation is "postfix" notation in which the operator is written directly after its two operands. A number of hand calculator models are designed to handle arithmetic in this manner.

EXERCISES

2.1 If $f(x) = x^2 - 3x + 2$ then $f(2y + z - 1) = ?$

2.2 Let $A = \{1, 2, 3,\}$, $B = \{8, 9\}$, $C = \{2, 3\}$. The graph of $f: A \to B$ is given as $f = \{(1, 9), (2, 9), (3, 8)\}$.

a. What are the domain and the range of f?

b. Is f a one-to-one mapping of A to B?

c. Is f an onto mapping?

d. Is f invertible?

2.3 Let $f(x) = x + 2$, $g(x) = 2x + 3$ be functions defined on the integers (positive, negative, zero).

a. What is $f(g(x))$? $g(f(x))$?

b. Give the values of: $f(4)$, $g(0)$, $f^{-1}(6)$, $g^{-1}(7)$. Is $g^{-1}(8)$ defined?

c. Is f a one-to-one mapping from the integers into the integers? Is g?

d. Are f and g total functions on the integers?

2.4 If N is the set of nonnegative integers, give an example of a total function f: $N \to N$ which is invertible and whose inverse is total, but do not use as an example $f(x) = x$.

2.5 Do the following equations provide enough information to define a function $f(x)$ on the nonnegative integers? Compute $f(4)$, $f(5)$.

$$\begin{cases} f(0) = 2 \\ f(1) = 4 \\ f(x + 2) = x \cdot f(x). \end{cases}$$

2.6 Do the following equations provide a proper definition of a function $f(m, n)$ of two arguments? The arguments are assumed to be nonnegative integers.

$$\begin{cases} f(0, n) = n + 1 \\ f(m, 0) = f(m - 1, 1) \\ f(m, n) = f(m - 1, f(m, n - 1)). \end{cases}$$

Find $f(2, 2)$.

2.7 Let a relation xRy be defined by $x \leq y + 2$. Is R reflexive? symmetric? transitive?

2.8 Define a relation R on the set $\{a, b, c, d, e\}$ as follows. $R = \{(a, a), (b, b), (c, c),$

$(d,d), (b,c), (c,d)\}$. R is not an equivalence. Why not? What ordered pairs should be added to make it one, using the smallest possible number of additional pairs?

2.9 A relation R is defined by: xRy if x and y are positive integers, and x divides y, and $x^2 + y^2 < 25$. Write the graph of R.

2.10 Is the relation "x divides y" reflexive? symmetric? transitive?

2.11 How many functions f: $S \to S$ with domain equal to S (f if total on S) and range in S are there if S has n elements? How many such one-to-one functions are there?

2.12 Let f be a function that maps each n-bit binary word into either 0 or 1. How many such functions are there?

2.13 Write a PL/1 recursive procedure (or use any other programming language with this capability) to compute the nth Fibonacci number. These numbers $\{F_k, k = 0, 1, 2, 3, \ldots\}$ are defined by the equations:

$$\begin{cases} F_0 = 0 \\ F_1 = 1 \\ F_k = F_{k-1} + F_{k-2} \quad (k \geq 2). \end{cases}$$

2.14 What is the value of $(\lambda x(3 + x^2))(3 + y)$?

2.15 Describe, using the BNF, the symbolic strings that represent the nonnegative binary integers written with no leading zeros (except for 0 itself) and no + sign.

2.16 Consider all finite strings formed from the letters of the alphabet A, B, ..., Z. Letting this class of strings be denoted by <STRING>, define <STRING> using the BNF. Extend this and define, with the BNF, all strings (call these <WORD>) that contain the substring CAT.

2.17 Which of the following are logical statements? (Those readers unfamiliar with the notation of (c), (d), (e) should look at Section 3.4, Miscellaneous Special Notation.)

a. $2x + 3y$

b. GO TO 1000

c. $\exists\, x, y(x + y = 5)$

d. $\forall\, x, y(y > x + 2)$

e. $\exists\, x(\forall z(2x + 3z))$

2.18 The relation on the natural numbers x, y $\{((x + y) \leq 10) \,\&\, (y \leq 4x)\}$ becomes a true statement for which pairs of natural numbers?

2.19 Show that $A \mid (B \,\&\, C) = (A \mid B) \,\&\, (A \mid C)$ is a theorem by computing the truth tables of the Boolean expressions on either side of the "=" sign.

2.20 Compute the truth tables for the following statements and indicate which are tautologies:

$$(A \,\&\, B) \to (A \mid B)$$
$$(A \mid B) \to (A \,\&\, B)$$
$$((A \,\&\, B) \,\&\, (\neg A)) \to B$$
$$(A \to \neg B) \to (A \to B).$$

2.21 The Boolean algebra of statements or propositions has been defined in terms of three basic operations: $\neg$, &, |. Show that A & B can be expressed in terms of $\neg$, | and, therefore, that these two operations $\neg$, | are sufficient for all Boolean expressions. Show, similarly, that $\neg$, & are adequate to replace the symbols $\neg$, &, |.

2.22 Determine by truth table if

$$(P \to Q) \to ((\neg R \to \neg Q) \to (P \to R))$$

is a tautology.

2.23 Let us write $A \,!\, B$ to denote the "exclusive OR" of A, B; i.e., $A \,!\, B$ is true if and only if precisely one of A, B is true. Write the truth table for $A \,!\, B$ and express it in terms of &, |, $\neg$.

2.24 We define "A NOR B" written as "$A \uparrow B$" to mean the statement, "neither A nor B is true." Write the truth table for $A \uparrow B$. Express $A \uparrow B$ in terms of &, |, $\neg$. Also, express $\neg A$, A & B, $A \mid B$ using only the symbol "$\uparrow$" as a connector.

2.25 Write a PL/I program (or use any other appropriate programming language) that will test whether

$$(P \to Q) \to ((Q \to R) \to (P \to R))$$

is a tautology.

2.26 Prove, by truth tables, DeMorgan's laws:

$$(A \mid B) = \neg(\neg A \,\&\, \neg B)$$
$$(A \,\&\, B) = \neg(\neg A \mid \neg B).$$

State the generalizations of these laws for n statements $A_1, A_2, \ldots, A_n$.

2.27 Show that there are an infinite number of Boolean expressions equivalent to any given one.

2.28 Write a segment of PL/I code using the BOOL operation[1] (or use any other appropriate programming language) that will, from two given bit strings

$$X: 0111010101,\ Y: 1101010010,$$

form a string Z of the same length as X, Y where each bit Z_i ($i = 1, 2, \ldots, 10$) is the NAND ("not and") of the corresponding bits in X,Y (i.e., $Z_i = \neg(X_i \,\&\, Y_i)$).

---

[1] In PL/I the BOOL operation is used as follows. The statement Z = BOOL(X, Y, W); where X, Y are bit strings of the same length and W is a four-bit binary string, results in the computation of a bit string Z of the same length as X, Y. Each bit Z_i in the string Z is obtained from the corresponding bits X_i, Y_i in X, Y by applying a binary Boolean operation whose truth table is defined by W; e.g., if we write Z = BOOL(X, Y, W), where X = 10111011, Y = 11011101, W = 1010, then W defines the following truth table

X_i	Y_i	Z_i	
0	0	1	The word W read from top to bottom 1010
0	1	0	
1	0	1	
1	1	0	

The result of this operation will be, then, Z = 00100010.

2.29 Given the PL/I statement

$$Z = \text{BOOL}(X, Y, W);$$

where $W =$ '1011'B, each bit Z_i in Z is equal to what Boolean function of the corresponding bits X_i, Y_i in X, Y? (Use the disjunctive normal form.)

2.30 Another standard form for Boolean functions is the "conjunctive normal form" in which a Boolean function is expressed as a conjunction of disjunctions; e.g., $F = (A \mid \neg B) \& (\neg A \mid B)$. Show that such a form can be given for any statement F by (i) writing $\neg F$ in disjunctive normal form; (ii) taking the negation of the result of (i) by using DeMorgan's laws (Exercise 2.26).

2.31 Prove the tautologies:

a. $(P \rightarrow Q) \rightarrow ((R \rightarrow P) \rightarrow (R \rightarrow Q))$

b. $P \mid \overline{P}$

c. $P \rightarrow (\neg P \rightarrow Q)$.

Do not use truth tables. Give deductive proofs, based upon the given axioms for the propositional calculus, by applying the rules of substitution and of modus ponens.

REFERENCES

1. Berkling, K., "A Compiling Machine Based on Tree Structures and the Lambda Calculus." *IEEE Transactions on Computers*, **C-20**, 4 (April 1971).
2. Church, A., "The Calculi of Lambda Conversion," in *Annals of Mathematics Studies*, Vol. 6. Princeton, New Jersey: Princeton University Press, 1941.
3. IBM Corporation. *OS PL/I Optimizing and Checkout Compilers: Language Reference Manual.* Order No. GC 33-0009.
4. Kleene, S. C., Foundations of Mathematics." *Encyclopedia Britannica*, 1971.
5. *Lambda Calculus and Computer Science Theory*, Proceedings of the Symposium held in Rome, March 25–27, 1975. IAC–CNR Istituto per la Applicazioni del Calcolo "Mauro Picone," Consiglio Nazionale delle Richercho. Edited by C. Boehm. New York: Springer, 1975 (Lecture Notes in Computer Science, Vol. 37).
6. Lindsey, C. H., and S. G. Van Der Meulen, *Informal Introduction to ALGOL 68.* Amsterdam: North-Holland Publishing Company, 1973.
7. Naur, P., ed. "Revised Report on the Algorithmic Language ALGOL 60." *Comm. Assn. Comp. Mach.*, **6**, 1 (January 1963).
8. Schwartz, J. T., *Set Theory as a Language for Program Specification and Programming.* New York: Courant Institute of Mathematical Sciences, Lecture Notes.
9. Schwartz, J. T., *An Interim Report on the SETL Project, Installment 1: Generalities.* New York: Courant Institute of Mathematical Sciences, Lecture Notes, February 1973.
10. Van Wijngaarden, A., B. Mailloux, J. Peck, and C. Koster, "Report on the Algorithmic Language ALGOL 68." *Numer. Math.*, **14**, 2 (1969).