First-Order Logic

Raymond M. Smullyan

City University of New York
and Indiana University

Dover Publications, Inc.
New York

Chapter I

Preliminaries

§ 0. Foreword on Trees

Trees will play an important role throughout this work, so we shall commence with some pertinent definitions:

By an *unordered tree*, $\mathscr{T}$, we shall mean a collection of the following items:

(1) A set S of elements called *points*.

(2) A function, ℓ, which assigns to each point x a positive integer $\ell(x)$ called the *level* of x.

(3) A relation xRy defined in S, which we read "x is a *predecessor* of y" or "y is a *successor* of x". This relation must obey the following conditions:

C_1: There is a unique point a_1 of level 1. This point we call the *origin* of the tree.

C_2: Every point other than the origin has a unique predecessor.

C_3: For any points x, y, if y is a successor of x, then $\ell(y)=\ell(x)+1$.

We shall call a point x an *end point* if it has no successors; a *simple point* if it has exactly one successor, and a *junction point* if it has more than one successor. By a *path* we mean any finite or denumerable sequence of points, beginning with the origin, which is such that each term of the sequence (except the last, if there is one) is the predecessor of the next. By a *maximal path* or *branch* we shall mean a path whose last term is an end point of the tree, or a path which is infinite.

It follows at once from C_1, C_2, C_3 that for any point x, there exists a unique path P_x whose last term is x. If y lies on P_x, then we shall say that y *dominates* x, or that x is *dominated* by y. If x dominates y and $x \neq y$, then we shall say that x is (or lies) *above* y, or that y lies *below* x. We shall say that x is *comparable* with y if x dominates y or y dominates x. We shall say that y is *between* x and z if y is above one of the pair $\{x,z\}$ and below the other.

By an *ordered tree*, $\mathscr{T}$, we shall mean an unordered tree together with a function θ which assigns to each junction point z a sequence $\theta(z)$ which contains no repetitions, and whose set of terms consists of all the successors of z. Thus, if z is a junction point of an ordered tree, we can speak of the $1^{st}, 2^{nd}, \ldots, n^{th}, \ldots$ successors of z (for any n up to the number of successors of z) meaning, of course, the $1^{st}, 2^{nd}, \ldots, n^{th}, \ldots$ terms of $\theta(z)$.

For a simple point x, we shall also speak of the successor of x as the *sole* successor of x.

We shall usually display ordered trees by placing the origin at the top and the successor(s) of each point x below x, and in the order, from left to right, in which they are ordered in the tree. And we draw a line segment from x to y to signify that y is a successor of x.

We shall have occasion to speak of adding "new" points as successors of an end point x of a given tree $\mathscr{T}$. By this we mean more precisely the following: For any element y outside $\mathscr{T}$, by the adjunction of y as the sole successor of x, we mean the tree obtained by adding y to the set S, and adding the ordered pair $\langle x, y\rangle$ to the relation R (looked at as a set of ordered pairs), and extending the function ℓ by defining $\ell(y)=\ell(x)+1$. For any distinct elements $y_1, \ldots, y_n$, each outside S, by the adjunction of $y_1, \ldots, y_n$ as respective $1^{st}, 2^{nd}, \ldots, n^{th}$ successors of x, we mean the tree obtained by adding the y_i to S, adding the pairs $\langle x, y_i\rangle$ to R and extending ℓ by setting $\ell(y_1)=\ldots=\ell(y_n)=\ell(x)+1$, and extending the function θ by defining $\theta(x)$ to be the sequence $(y_1, \ldots, y_n)$. [It is obvious that the extended structure obtained is really a tree].

A tree is called *finitely generated* if each point has only finitely many successors. A tree, $\mathscr{T}$, is called *finite* if $\mathscr{T}$ has only finitely many points, otherwise the tree is called *infinite*. Obviously, a finitely generated tree may be infinite.

We shall be mainly concerned with ordered trees in which each junction point has exactly 2 successors. Such trees are called *dyadic* trees. For such trees we refer to the first successor of a junction point as the *left successor*, and the second successor as the *right successor*.

[*Exercise:* In a dyadic tree, define x to be to the left of y if there is a junction point whose left successor dominates x and whose right successor dominates y. Prove that if x is to the left of y and y is to the left of z, then x is to the left of z].

§ 1. Formulas of Propositional Logic

We shall use for our undefined logical connectives the following 4 symbols:

(1) $\sim$ [read "not"], (2) $\wedge$ [read "and"],
(3) $\vee$ [read "or"], (4) $\supset$ [read "implies"].

These symbols are respectively called the *negation*, *conjunction*, *disjunction*, and *implication* symbols. The last 3 are collectively called *binary* connectives, the first ($\sim$) the *unary* connective.

Other symbols shall be:

(i) A denumerable set $p_1, p_2, \ldots, p_n, \ldots$ of symbols called *propositional variables*.

(ii) The two symbols (,), respectively called the *left parenthesis* and the *right parenthesis* (they are used for purposes of punctuation). Until we come to *First-Order Logic*, we shall use the word "variable" to mean *propositional variable*.

We shall use the letters "p", "q", "r", "s" to stand for any of the variables $p_1, p_2, \ldots, p_n, \ldots$. The notion of *formula* is given by the following recursive rules, which enable us to obtain new formulas from those already constructed:

F_0: Every propositional variable is a formula.
F_1: If A is a formula so is $\sim A$.
F_2, F_3, F_4: If A, B are formulas so are $(A \wedge B)$, $(A \vee B)$, $(A \supset B)$.

This recursive definition of "formula" can be made explicit as follows. By a *formation sequence* we shall mean any finite sequence such that each term of the sequence is either a propositional variable or is of the form $\sim A$, where A is an earlier term of the sequence, or is of one of the forms $(A \wedge B)$, $(A \vee B)$, $(A \supset B)$, where A, B are earlier terms of the sequence. Now we can define A to be a *formula* if there exists a formation sequence whose last term is A. And such a sequence is also called a *formation sequence* for A.

For any formula A, by the *negation* of A we mean $\sim A$. It will sometimes prove notationally convenient to write A' in place of $\sim A$. For any 2 formulas A, B, we refer to $(A \wedge B)$, $(A \vee B)$, $(A \supset B)$ as the *conjunction*, *disjunction*, *conditional* of A, B respectively. In a conditional formula $(A \supset B)$, we refer to A as the *antecedent* and B as the *consequent*.

We shall use the letters "A", "B", "C", "X", "Y", "Z" to denote formulas. We shall use the symbol "b" to denote any of the binary connectives $\wedge$, $\vee$, $\supset$; and when "b" respectively denotes $\wedge$, $\vee$, $\supset$ then $(X\,b\,Y)$ shall respectively mean $(X \wedge Y)$, $(X \vee Y)$, $(X \supset Y)$. We can thus state the formation rules more succinctly as follows:

F_0: Every propositional variable is a formula.
F_1: If X is a formula so is $\sim X$.
F_2: If X, Y are formulas, then for each of the binary connectives b, the expression $(X\,b\,Y)$ is a formula.

In displaying formulas by themselves (i.e. not as parts of other formulas), we shall omit outermost parentheses (since no ambiguity can result). Also, for visual perspicuity, we use square brackets [ ] interchangeably with parentheses, and likewise braces { }. Usually we shall use square brackets as exterior to parentheses, and braces as exterior to square brackets.

Example. Consider the following formula:

$$((((p \supset q) \wedge (q \vee r)) \supset (p \vee r)) \supset \sim(q \vee s)).$$

It is easier to read if displayed as follows:

$$\{[(p \supset q) \wedge (q \vee r)] \supset (p \vee r)\} \supset \sim(q \vee s).$$

Biconditional – we use "$X \leftrightarrow Y$" as an abbreviation for $(X \supset Y) \wedge (Y \supset X)$. The formula $X \leftrightarrow Y$ is called the *biconditional* of X, Y. It is read "X if and only if Y" or "X is equivalent to Y".

Uniqueness of Decomposition. It can be proved that every formula can be formed in only one way—i. e. for every formula X, one *and only one* of the following conditions holds:

(1) X is a propositional variable.

(2) There is a *unique* formula Y such that $X = \sim Y$.

(3) There is a *unique* pair X_1, X_2 and a *unique* binary connective b such that $X = (X_1 b X_2)$.

Thus no conjunction can also be a disjunction, or a conditional; no disjunction can also be a conditional. Also none of these can also be a negation. And, e. g., $(X_1 \wedge X_2)$ can be identical with $(Y_1 \wedge Y_2)$ only if $X_1 = Y_1$ and $X_2 = Y_2$ (and similarly with the other binary connectives). We shall not prove this here; perfectly good proofs can be found, e. g. in CHURCH [1] or KLEENE [1].

In our discussion below, we shall consider a more abstract approach in which this combinatorial lemma can be circumvented.

***Discussion.** First we wish to mention that some authors prefer the following formation rules for formulas:

F_0': Same as F_0.

F_1': If X is a formula, so is $\sim(X)$.

F_2': If X, Y are formulas, so is $(X)b(Y)$.

This second set of rules has the advantage of eliminating, at the outset, outermost parentheses, but has the disadvantage of needlessly putting parentheses around variables.

It seems to us that the following set of formation rules, though a bit more complicated to state, combines the advantages of the two preceding formulations, and involves using neither more nor less parentheses than is necessary to prevent ambiguity:

F_0'': Same as before.

F_1'': If X is a formula but not a propositional variable and p is a propositional variable, $\sim(X)$ and $\sim p$ are formulas.

F_2'': If X, Y are both formulas, but neither X nor Y is a propositional variable, and if p, q are propositional variables, then the following expressions are all formulas:

(a) $(X)b(Y)$,
(b) $(X)bq$,
(c) $pb(Y)$,
(d) pbq.

In all the above 3 approaches, one needs to prove the unique decomposition lemma for many subsequent results. Now let us consider yet another scheme (of a radically different sort) which avoids this.

First of all, we delete the parentheses from our basic symbols. We now define the *negation* of X, not as the symbol $\sim$ followed by the first symbol of X, followed by the second symbol of X, etc. but simply as the *ordered pair* whose first term is "$\sim$" and whose second term is X. And we define the *conjunction* of X, Y as the *ordered triple* whose first term is X, whose second term is "$\wedge$" and whose third term is Y. [In contrast, the conjunction of X and Y, as previously defined, is a sequence of $n+m+3$ terms, where n, m are the respective number of terms of X, Y. The "3" additional terms are due to the left parenthesis, right parenthesis and "$\wedge$"]. Similarly we define the disjunction (conditional) of X, Y as the ordered triple $\langle X, b, Y \rangle$ where b is the binary connective in question.

Under this plan, a formula is either a (propositional) variable, an ordered pair (if it is a negation) or an ordered triple. Now, no ordered pair can also be an ordered triple, and neither one can be a single symbol. Furthermore, an ordered pair uniquely determines its first and second elements, and an ordered triple uniquely determines its first, second and third elements. Thus the fact that a formula can be formed in "only one way" is now immediate.

We remark that with this plan, we can (and will) still *use* parentheses to describe formulas, but the parentheses are *not* parts of the formula. For example, we write $X \wedge (Y \vee Z)$ to denote the ordered triple whose first term is X, whose second term is "$\wedge$", and whose third term is itself the ordered triple whose first, second and third terms are respectively, Y, $\vee$, Z. But (under this plan) the parentheses themselves do not belong to the object language[1]) but only to our metalanguage[1]).

The reader can choose for himself his preferred notion of "formula", since subsequent developments will not depend upon the choice.

---

[1]) The term *object language* is used to denote the language talked *about* (in this case the set of formal expressions of propositional logic), and the term *metalanguage* is used to denote the language in which we are talking about the object language (in the present case English augmented by various common mathematical symbols).

Subformulas. The notion of *immediate subformula* is given explicitly by the conditions:

I_0: Propositional variables have no immediate subformulas.

I_1: $\sim X$ has X as an immediate subformula and no others.

I_2-I_4: The formulas $X \wedge Y$, $X \vee Y$, $X \supset Y$ have X, Y as immediate subformulas and no others.

We shall sometimes refer to X, Y respectively as the *left immediate subformula, right immediate subformula* of $X \wedge Y$, $X \vee Y$, $X \supset Y$.

The notion of *subformula* is implicitly defined by the rules:

S_1: If X is an immediate subformula of Y, or if X is identical with Y, then X is a subformula of Y.

S_2: If X is a subformula of Y and Y is a subformula of Z, then X is a subformula of Z.

The above implicit definition can be made explicit as follows: Y is a subformula of Z iff (i.e. if and only if) there exists a finite sequence starting with Z and ending with Y such that each term of the sequence except the first is an immediate subformula of the preceding term.

The only formulas having no immediate subformulas are propositional variables. These are sometimes called *atomic* formulas. Other formulas are called *compound* formulas. We say that a variable p *occurs* in a formula X, or that p is one of the variables of X, if p is a subformula of X.

Degrees; Induction Principles. To facilitate proofs and definitions by induction, we define the *degree* of a formula as the number of occurrences of logical connectives. Thus:

D_0: A variable is of degree 0.

D_1: If X is of degree n, then $\sim X$ is of degree $n+1$.

D_2-D_4: If X, Y are of degrees n_1, n_2, then $X \wedge Y$, $X \vee Y$, $X \supset Y$ are each of degree n_1+n_2+1.

Example.

$p \wedge (q \vee \sim r)$ is of degree 3.

$p \wedge (q \vee r)$ is of degree 2.

We shall use the principle of *mathematical induction* (or of *finite descent*) in the following form. Let S be a set of formulas (S may be finite or infinite) and let P be a certain property of formulas which we wish to show holds for every element of S. To do this it suffices to show the following two conditions:

(1) Every element of S of degree 0 has the property P.

(2) If some element of S of degree >0 fails to have the property P, then some element of S of lower degree also fails to have property P.

Of course, we can also use (2) in the equivalent form:

(2)′ For every element X of S of positive degree, if all elements of S of degree less than that of X have property P, then X also has property P.

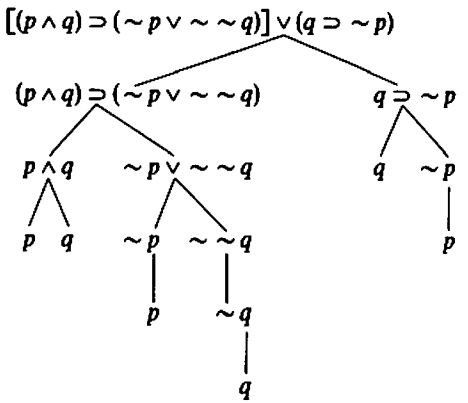
Formation Trees. It is sometimes useful to display all the subformulas of a given formula X in the form of a *dyadic tree* which we call a *formation tree* for X... which completely shows the pedigree of X. We start the tree with the formula X at the origin, and each node of the tree which is not a propositional variable "branches" into its immediate subformulas. More precisely, a formation tree for X is an ordered dyadic tree $\mathcal{T}$ whose points are formulas (or rather occurrences of formulas, since the same formula may have several different occurrences on the tree) and whose origin is (an occurrence of) X, and such that the following 3 conditions hold:

(i) Each end point is (an occurrence of) a propositional variable.

(ii) Each simple point is of the form $\sim Y$ and has (an occurrence of) Y as its sole successor.

(iii) Each junction point is of the form $X\ b\ Y$ and has (occurrences of) X, Y as respective left and right successors.

As an example the following is a formation tree for the formula $[(p \wedge q) \supset (\sim p \vee \sim\sim q)] \vee (q \supset \sim p)$:



We might remark that the subformulas of a given formula X are precisely those formulas which appear somewhere on the formation tree for X.

§ 2. Boolean Valuations and Truth Sets

Now we consider, in addition to the formulas of propositional logic, a set $\{t, f\}$ of two *distinct* elements, t, f. We refer to t, f as *truth-values*. For any set S of formulas, by a *valuation* of S, we mean a function v from S into the set $\{t, f\}$—i.e. a mapping which assigns to every element X of S one of the two values t, f. The value $v(X)$ of X under v is

called the *truth value* of X under v. We say that X is *true under* v if $v(X) = t$, and *false under* v if $v(X) = f$.

Now we wish to consider valuations of the set E of all formulas of propositional logic. We are not really interested in *all* valuations of E, but only in those which are "faithful" to the usual "truth-table" rules for the logical connectives. This idea we make precise in the following definition.

Definition 1. A valuation v of E is called a *Boolean* valuation if for every X, Y in E, the following conditions hold:

B_1: The formula $\sim X$ receives the value t if X receives the value f and f if X receives the value t.

B_2: The formula $X \wedge Y$ receives the value t if X, Y both receive the value t, otherwise $X \wedge Y$ receives the value f.

B_3: The formula $X \vee Y$ receives the value t if at least one of X, Y receives the value t, otherwise $X \vee Y$ receives the value f.

B_4: The formula $X \supset Y$ receives the value f if X, Y receive the respective values t, f, otherwise $X \supset Y$ receives the value t.

This concludes our definition of a Boolean valuation. We say that two valuations *agree* on a formula X if X is either true in both valuations or false in both valuations. And we say that 2 valuations agree on a *set* S of formulas if they agree on every element of the set S.

If S_1 is a subset of S_2 and if v_1, v_2 are respective valuations of S_1, S_2, then we say that v_2 is an *extension* of v_1 if v_2, v_1 agree on the smaller set S_1.

It is obvious that if 2 Boolean valuations agree on X then they agree on $\sim X$ (why?), and if they agree on both X, Y they must also agree on each of $X \wedge Y$, $X \vee Y$, $X \supset Y$ (why?). By mathematical induction it follows that if 2 Boolean valuations of E agree on the set of all atomic elements of E (i.e., on all propositional variables) then they agree on all of E. Stated otherwise, a valuation v_0 of the set of all *atomic* elements of E can be extended to *at most* one Boolean valuation of E.

By an *interpretation* of a formula X is meant an assignment of truth values to all of the variables which occur in X. More generally, by an interpretation of a set W (of formulas) is meant an assignment of truth values to all the variables which occur in any of the elements of W. We can thus rephrase the last statement of the preceding paragraph by saying that any interpretation v_0 of E can be extended to at most one Boolean valuation of E. That v_0 can be extended to at *least* one Boolean valuation of E will be clear from the following considerations.

Consider a single formula X and an interpretation v_0 of X—or for that matter any assignment v_0 of truth values to a set of propositional variables which include at least all variables of X (and possibly others). It is easily verified by induction on the degree of X that there exists one

and only one way of assigning truth values to all *subformulas* of X such that the *atomic* subformulas of X (which are propositional variables) are assigned the same truth values as under v_0, and such that the truth value of each *compound* subformula Y of X is determined from the truth values of the immediate subformulas of Y by the truth-table rules $B_1 - B_4$. [We might think of the situation as first constructing a formation tree for X, then assigning truth values to the end points in accordance with the interpretation v_0, and then working our way up the tree, successively assigning truth values to the junction and simple points, in terms of truth values already assigned to their successors, in accordance with the truth-table rules]. In particular, X being a subformula of itself receives a truth value under this assignment; if this value is t then we say that X is true *under the interpretation* v_0, otherwise *false* under v_0. Thus we have now defined what it means for a formula X to be true under an *interpretation*.

Now consider an interpretation, v_0, for the entire set E. Each element, X, of E has a definite truth value under v_0 (in the manner we have just indicated); we let v be that valuation which assigns to each element of E its truth value under the interpretation v_0. The valuation v is on the entire set E, and it is easily verified that v is a Boolean valuation, and of course, v is an extension of v_0. Thus it is indeed the case that every interpretation of E can be extended to one (and only one) Boolean valuation of E.

Tautologies. The notion of *tautology* is perhaps the fundamental notion of propositional logic.

Definition 2. X is a *tautology* iff X is true in all Boolean valuations of E.

Equivalently, X is a *tautology* iff X is true under every *interpretation* of E. Now it is obvious that the truth value of X under an interpretation of E depends only on the truth values assigned to the variables which occur in X. Therefore, X is a tautology if and only if X is true under every interpretation of X. Letting n be the number of variables which occur in X, there are exactly 2^n distinct interpretations of X. Thus the task of determining whether X is or is not a tautology is purely a finite and mechanical one—just evaluate its truth value under each of its 2^n interpretations (which is tantamount to the familiar truth-table analysis).

Definition 3. A formula X is called (truth-functionally) *satisfiable* iff X is true in at least one Boolean valuation. A set S of formulas is said to be (simultaneously) truth-functionally *satisfiable* iff there exists at least one Boolean valuation in which every element of S is true. Such a valuation is said to *satisfy* S.

Definition 4. A set S *truth-functionally implies* a formula X, or X is *truth-functionally implied* by S, or is a *truth-functional consequence* of S if X is true in every Boolean valuation which satisfies S. We also say that Y is truth-functionally implied by X if Y is truth functionally implied by the unit set $\{X\}$ ... i.e. if Y is true in every Boolean valuation in which X is true.

Definition 5. Two formulas X, Y are called *truth functionally equivalent* iff X, Y are true in the same Boolean valuations. [The reader should note that X truth-functionally implies Y iff $X \supset Y$ is a tautology, and that X is truth-functionally equivalent to Y iff the formula $X \leftrightarrow Y$ is a tautology].

Truth Sets. Let v be a Boolean valuation, and let S be the set of all formulas which are *true* under v. It is immediate from the definition of a Boolean valuation that the set S obeys the following conditions (for every X, Y):

S_1: Exactly one of the pair $(X, \sim X)$ belongs to S. Stated otherwise $(\sim X) \in S$ iff $X \notin S$.

S_2: $(X \wedge Y)$ is in S iff X, Y are both in S.

S_3: $(X \vee Y)$ is in S iff $X \in S$ or $Y \in S$.

S_4: $(X \supset Y)$ is in S iff $X \notin S$ or $Y \in S$.

A set S obeying the above conditions will be called *saturated* or will be said to be a *truth set*. Thus for any Boolean valuation, the set of all sentences true under the valuation is saturated. Indeed, if v is an arbitrary valuation, and if S is the set of all sentences which are true under v, then the following 2 conditions are *equivalent*:

(1) v is a Boolean valuation,

(2) S is *saturated*.

Now suppose that we start with a set S, and we define v_s to be that valuation which assigns t to every member of S, and f to every formula outside S. [The function v_s is sometimes referred to as the *characteristic* function of the set S.] It is again obvious that S is *saturated* iff v_s is a Boolean valuation.

Now the set of all sentences true under v_s is obviously S itself. Thus a set is saturated iff it is the set of all sentences true under some Boolean valuation. Thus a formula X is a *tautology* iff it is an element of every truth set; stated otherwise, the set of tautologies is the *intersection* of all truth sets and a formula X is *satisfiable* iff it is an element of some truth set. Stated otherwise, the set of satisfiable sentences is the *union* of all truth sets. Likewise a set S truth-functionally implies X iff X belongs to every truth set which includes S.

We thus see that we really do not need to "import" these "foreign" elements t, f in order to define our basic semantic notions. In some

contexts it is technically more convenient to use t and f and Boolean valuations; in other it is simpler to use truth sets.

Exercise 1 [Truth Functional Equivalence]. We shall use "$\simeq$" in our *metalanguage* and write $X \simeq Y$ to mean that X is *equivalent* to Y—i.e. that the formula $X \leftrightarrow Y$ is a tautology.

Now suppose that $X_1 \simeq X_2$. Prove the following equivalences:

$$\sim X_1 \simeq \sim X_2,$$
$$X_1 \wedge Y \simeq X_2 \wedge Y; \qquad Y \wedge X_1 \simeq Y \wedge X_2,$$
$$X_1 \vee Y \simeq X_2 \vee Y; \qquad Y \vee X_1 \simeq Y \vee X_2,$$
$$X_1 \supset Y \simeq X_2 \supset Y; \qquad Y \supset X_1 \simeq Y \supset X_2.$$

Using these facts, show that for any formula Z which contains X_1 as a part, if we replace one or more occurrences of the part X_1 by X_2, the resulting formula is equivalent to Z.

Exercise 2 – [Important for Ch. XV!]. In some formulations of propositional logic, one uses "t", "f" as symbols of the object language itself; these symbols are then called *propositional constants*. And a *Boolean valuation* is redefined by adding the condition that t must be given the value *truth* and f *falsehood*. [Thus, e.g. t by itself is a tautology; f is unsatisfiable; $X \supset t$ is a tautology; $f \supset X$ is a tautology. Also, under any Boolean valuation $t \supset Y$ has the same truth value as Y; $X \supset f$ has the opposite value to X. Thus $t \supset Y$ is a tautology iff Y is a tautology; $X \supset f$ is a tautology iff X is unsatisfiable.]

Prove the following equivalences:

(1) $X \wedge t \simeq X$; $\quad X \wedge f \simeq f$,
(2) $X \vee t \simeq t$; $\quad X \vee f \simeq X$,
(3) $X \supset t \simeq t$; $\quad t \supset X \simeq X$,
(4) $X \supset f \simeq \sim X$; $\quad f \supset X \simeq t$,
(5) $\sim t \simeq f$; $\quad \sim f \simeq t$,
(6) $X \wedge Y \simeq Y \wedge X$; $\quad X \vee Y \simeq Y \vee X$.

Using these facts show that every formula X with propositional constants is either equivalent to a formula Y which contains no propositional constants or else it is equivalent to t or to f.

Exercise 3. It is convenient to write a conjunction $(..(X_1 \wedge X_2) \wedge ... \wedge X_n)$ as $X_1 \wedge X_2 \wedge ... \wedge X_n$, and the formulas $X_1, X_2, ..., X_n$ are called the *components* of the conjunction. [Similarly we treat disjunctions.] By a *basic* conjunction is meant a conjunction with no repetitions of components such that each component is either a variable or the negation of a variable, but no variable and its negation are both components. [As an example, $p_1 \wedge \sim p_2 \wedge p_3$ is a basic conjunction—so is $\sim p_1 \wedge p_2 \wedge \sim p_3$—so is $\sim p_1 \wedge p_2 \wedge p_3$.] By a *disjunctive normal formula* is mean a formula $C_1 \vee ... \vee C_k$, where each C_i is a *basic* conjunction. [As an example the

formula $(p_1 \wedge \sim p_2 \wedge p_3) \vee (\sim p_1 \wedge p_2 \wedge p_3) \vee (\sim p_1 \wedge p_2 \wedge p_3)$ is a disjunctive normal formula.] A disjunctive normal formula is also sometimes referred to as a formula in disjunctive normal form. If we allow propositional constants t, f into our formal language, then the formula f is also said to be a disjunctive normal formula.

Prove that every formula can be put into disjunctive normal form—i.e. is equivalent to some disjunctive normal formula. [Hint: Make a truth-table for the formula. Each line of the table which comes out "T" will yield one of the basic conjunctions of the disjunctive normal form.]

Exercise 4. A binary connective C is said to be *definable* from connectives $C_1, \ldots, C_k$ if there exists a formula in two variables p, q which uses just the connectives $C_1, \ldots, C_k$ and which is equivalent to the formula pCq.

As an example, $\vee$ is definable from $\{\sim, \wedge\}$, because the formula $\sim(\sim p \wedge \sim q)$ is equivalent to $p \vee q$.

Prove: $\wedge$ is definable from $\{\sim, \vee\}$,
$\supset$ is definable from $\{\sim, \wedge\}$,
$\supset$ is definable from $\{\sim, \vee\}$,
$\wedge$ is definable from $\{\sim, \supset\}$,
$\vee$ is definable from $\{\sim, \supset\}$.

Exercise 5. Let us introduce Sheffer's *stroke* symbol "|" as a binary connective for propositional logic, and add the formation rule "If X, Y are formulas, so is $(X|Y)$". [We read "$X|Y$" as "X is *incompatible* with Y" or "either X or Y is false".] A Boolean valuation is then re-defined by adding the conditions "$X|Y$ is true under v iff at least one of X, Y is false under v":

(a) Show that | is definable from the other connectives.

(b) Show that *all* the other connectives are definable from | ($\sim$ is definable from | in the sense that there is a formula $\varphi(p)$ involving just the stroke connective and one propositional variable p such that $\varphi(p)$ is equivalent to $\sim p$).

Do the same for the *joint denial connective* $\downarrow$ (where $X \downarrow Y$ is read "both X, Y are false". Show that all other connectives are definable from $\downarrow$.

It can be shown that |, $\downarrow$ are the *only* binary connectives which each suffice to define all other connectives. [This is not easy! The "virtuoso" reader might wish to try his hand at this as an exercise.]

Chapter II

Analytic Tableaux

We now describe an extremely elegant and efficient proof procedure for propositional logic which we will subsequently extend to first order logic, and which shall be basic to our entire study. This method, which we term *analytic tableaux*, is a variant of the "semantic tableaux" of Beth [1], or of methods of Hintikka [1]. (Cf. also Anderson and Belnap [1].) Our present formulation is virtually that which we introduced in [1]. Ultimately, the whole idea derives from Gentzen [1], and we shall subsequently study the relation of analytic tableaux to the original methods of Gentzen.

§ 1. The Method of Tableaux

We begin by noting that under any interpretation the following eight facts hold (for any formulas X, Y):

1) a) If $\sim X$ is true, then X is false.
   b) If $\sim X$ is false, then X is true.
2) a) If a conjunction $X \wedge Y$ is true, then X, Y are both true.
   b) If a conjunction $X \wedge Y$ is false, then either X is false or Y is false.
3) a) If a disjunction $X \vee Y$ is true, then either X is true or Y is true.
   b) If a disjunction $X \vee Y$ is false, then both X, Y are false.
4) a) If $X \supset Y$ is true, then either X is false or Y is true.
   b) If $X \supset Y$ is false, then X is true and Y is false.

These eight facts provide the basis of the tableau method.

Signed Formulas. At this stage it will prove useful to introduce the symbols "T", "F" to our object language, and define a *signed* formula as an expression TX or FX, where X is a (unsigned) formula. (Informally, we read "TX" as "X is true" and "FX" as "X is false".)

Definition. Under any interpretation, a signed formula TX is called *true* if X is true, and *false* if X is false. And a signed formula FX is called *true* if X is false, and *false* if X is true.

Thus the truth value of TX is the same as that of X; the truth value of FX is the same as that of $\sim X$.

By the *conjugate* of a signed formula we mean the result of changing "T" to "F" or "F" to "T" (thus the conjugate of TX is FX; the conjugate of FX is TX).

Illustration of the Method of Tableaux. Before we state the eight rules for the construction of tableaux, we shall illustrate the construction with an example.