I

Propositional Logic

1 Orders and Trees

Before starting on the basic material of this book, we introduce a general representation scheme which is one of the most important types of structures in logic and computer science: trees. We expect that most readers will be familiar with this type of structure at least informally. (For the set–theoretic background see Chapter VI.) A tree is something that looks like Figure 1:

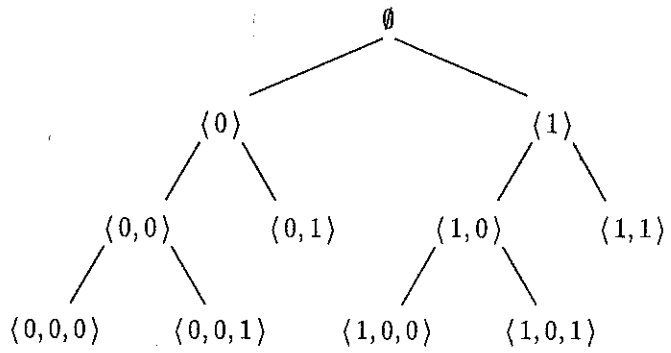

FIGURE 1.

It has nodes (in this example, binary sequences or strings) arranged in a partial order (extension as sequences means lower down in the picture of the tree). There is typically a single node (the empty set (or sequence), $\emptyset$) at the top (above all the others) which is called the root of the tree. (We draw our trees growing downwards to conform to common practice but the ordering on the tree is arranged to mirror the situation given by extension of sequences. Thus, the root is the smallest (first) element in the ordering and the nodes get larger as we travel down the tree.) A node on a tree may have one or more incomparable immediate successors. If it has more than one, we say the tree branches at that node. If each node has at most n immediate successors, the tree is n–ary or n–branching. (The one in the picture is a 2–ary, or as we usually say, a *binary*

tree.) A terminal node, that is, one with no successors, is called a leaf. Other terminology such as a path on a tree or the levels of a tree should have intuitively clear meanings. For those who wish to be formal, we give precise definitions. This material can be omitted for now and referred to as needed later. In particular König's lemma (Theorem 1.4) is not needed until §4.

Definition 1.1:

(i) A *partial order* is a set S with a binary relation called "less than", and written $<$, on S which is *transitive* and *irreflexive*:

$$x < y \text{ and } y < z \Rightarrow x < z \text{ and}$$
$$x \text{ is not less than } x \text{ for any } x.$$

(ii) The partial order $<$ is a *linear order* (or simply an *order*) if it also satisfies the *trichotomy law*:

$$x < y \text{ or } x = y \text{ or } y < x.$$

(iii) A linear order S is *well ordered* if every nonempt y subset A of S has a least element, i.e., there is an $x \in A$ such that for no $y \in A$ is $y < x$. This property easily implies that the order has no infinite descending chain, i.e., there is no set of elements $x_0, x_1, \ldots$ of S such that

$$\ldots < x_2 < x_1 < x_0.$$

(If there were such a descending chain then the set A consisting of all the x_n in the chain would have no least element.) For the converse of this implication see Exercise 6.

(iv) We use the usual notational conventions for orderings:

$$x \leq y \Leftrightarrow x < y \text{ or } x = y.$$
$$x > y \Leftrightarrow y < x.$$

Note that antisymmetry for partial orderings, $x < y \Rightarrow$ it is not the case that $y < x$, follows immediately from Definition 1.1 (i). (We follow standard mathematical practice in using "$\Rightarrow$" and "$\Leftrightarrow$" as abbreviations for "implies" and "if and only if", respectively.)

Note (for those with some knowledge of set theory): Except in Chapter VI on the elements of set theory, each set A we consider is finite or countably infinite, i.e., each such A is finite or there is a one-to-one map from the natural numbers $\mathcal{N}$ onto A and so we may assume that we have a listing of A as either $\{a_i | \ i < n\}$ for some $n \in \mathcal{N}$ or as $\{a_i | \ i \in \mathcal{N}\}$.

led a leaf. Other
d have intuitively
definitions. This
ter. In particular

"less than", and

if it also satisfies

set A of S has a
A is $y < x$. This
nding chain, i.e.,

sisting of all the
converse of this

he case that $y <$
rd mathematical
and "if and only

a Chapter VI on
untably infinite,
natural numbers
ither $\{a_i | \ i < n\}$

Definition 1.2: A *tree* is a set T (whose elements are called *nodes*) partially ordered by $<_T$, with a unique least element called the *root*, in which the predecessors of every node are well ordered by $<_T$.

A *path* on a tree T is a maximal linearly ordered subset of T.

Definition 1.3:

(i) The *levels* of a tree T are defined by induction. The 0th level of T consists precisely of the root of T. The $k+1^{\text{st}}$ level of T consists of the immediate successors of the nodes on the kth level of T. (We say that x is a successor of y in T if $y \leq_T x$. It is an immediate successor if $y <_T x$ and there is no z such that $y <_T < z <_T x$.)

(ii) The *depth* of a tree T is the maximum n such that there is a node of level n in T. If there are nodes of level n for every natural number n, we say the depth of T is infinite or ω.

(iii) If each node has at most n immediate successors, the tree is *n-ary* or *n-branching*. If each node has finitely many immediate successors, we say that the tree is *finitely branching*. A node with no successors is called a *leaf* or a *terminal node*.

Note: We only consider trees such that every node is on level n of the tree for some natural number n (and as such has precisely n many predecessors in the tree order).

The crucial fact about finitely branching trees is König's lemma:

Theorem 1.4 (König's lemma): *If a finitely branching tree T is infinite, it has an infinite path.*

Proof: Suppose T is infinite. We define the sequence of elements $x_0, x_1, \ldots$, constituting a path P on T by induction (or recursion). The first element x_0 of P is, of course, the root of T. It has infinitely many successors in T by the assumption that T is infinite. Suppose that we have defined the first n elements of P to be $x_0, x_1, \ldots, x_{n-1}$ on levels $0, 1, \ldots, n-1$ of T, respectively, so that each x_i has infinitely many successors in T. By hypothesis, x_{n-1} has only finitely many immediate successors. As it has infinitely many successors all together, (at least) one of its immediate successors, say y, also has infinitely many successors. We now set $x_n = y$. x_n is on level n of T and has infinitely many successors in T and so we may continue our definition of P. □

König's lemma is a version of the compactness theorem in propositional logic and in topology. See §6 (and especially Theorem 6.13 and Exercise 6.11) for the former and Exercises 10 and 11 for its relations to the latter.

Frequently it is just the shape of the tree that is important and not the nodes themselves. To facilitate talking about the arrangement of different materials

into the same shape and to allow the same component to be used at different places in this assemblage, we talk about labeled trees. We attach labels to the nodes of the tree. Again the picture should be clear.

Definition 1.5: A *labeled tree* T is a tree T with a function (the labeling function) that associates some object with every node. This object is called the *label* of the node.

In fact, after the first exposure or two to labeled trees we at times simply drop the word. We draw our trees already labeled and let the concerned reader adjust the formalities.

Another way of putting more structure on a tree is by adding a linear ordering on the entire tree. Consider the case of the standard binary tree of finite sequences of 0's and 1's: The underlying set is the set $S = \{0,1\}^*$ of such sequences. We think of a binary sequence or string σ of length n as a map from $\{0,1,\ldots,n-1\}$ into $\{0,1\}$. (As a technical aside, we note that the usual set–theoretic representation of numbers identifies 0 with the empty set $\emptyset$. It also identifies functions with sets of ordered pairs by saying that $\langle x,y\rangle \in f$ if and only if $f(x) = y$. Thus, the sequence of length 0 is a function that is not defined anywhere. As a set of ordered pairs, it too is then identified with $\emptyset$. For the interested reader, these and similar matters are explained in Chapter VI, Elements of Set Theory.) We use "$^\frown$" to denote *concatenation*. Thus, for example, $\langle 0,1\rangle^\frown 0$ is $\langle 0,1,0\rangle$ while $\langle 0,1,0\rangle^\frown\langle 0,1\rangle$ is $\langle 0,1,0,0,1\rangle$. (Note that we frequently abuse notation by identifying 0 with $\langle 0\rangle$ and 1 with $\langle 1\rangle$ in such situations.) The tree ordering $<_S$ is given by extension as functions $\sigma < \tau \Leftrightarrow \sigma \subset \tau$, i.e., $\sigma(n) = \tau(n)$ for every n at which σ is defined. (In this case we also say that the string or sequence σ is a *proper initial segment* of τ. Of course, σ is simply an *initial segment* of τ if $\sigma \subset \tau$ or $\sigma = \tau$.) The additional linear order usually associated with this tree is the *lexicographic ordering on sequences*: For two sequences σ and τ we say that $\sigma <_L \tau$ if $\sigma \subset \tau$ or if $\sigma(n)$, the nth entry in σ, is less than $\tau(n)$ where n is the first entry at which the sequences differ (otherwise, as one can easily see, $\tau <_L \sigma$ or $\sigma = \tau$). The same procedure can be applied to any tree to produce a linear order of all the nodes. We begin by defining a linear order on each level of the tree. This order is usually called $<_L$ and described as a *left to right ordering* for the obvious pictorial reason. (This corresponds to ordering the binary strings of each fixed length by $\sigma <_L \tau$ if, at the first place σ and τ differ, σ is 0 and τ is 1. We then say that σ is *to the left of* τ.) The left–right orderings of each level are then extended to a linear ordering (also designated $<_L$) of all the nodes of the tree: Given two nodes x and y, we say that $x <_L y$ if $x <_T y$. If x and y are incomparable in the tree ordering, we find the largest predecessors x' and y' of x and y, respectively, that are on the same level of T. We then order x and y in the same way that x' and y' were ordered by $<_L$ on their own level: $x <_L y$ iff $x' <_L y'$. Any such total ordering of the nodes of a tree is also referred to as the *lexicographic ordering of the nodes*.

Exercises

1. Give an example of a finitely branching tree that is not n–branching for any n.

2. Give an example of an infinite tree of depth 3.

3. Prove that the notion of the level of a node in a tree is well defined, i.e., every node in a tree T is on exactly one level.

4. Prove that every node of a tree other than the root has exactly one immediate predecessor.

5. Let T be a tree. We say that two nodes x and y of T are *adjacent* if one is an immediate predecessor of the other, i.e., $x <_T y$ or $y <_T x$ and there is no node strictly between them. Prove that there is no sequence of nodes $x_1, \ldots, x_n$ $(n > 3)$ such that each x_i is adjacent to x_{i+1}, $x_1 = x_n$ but there are no other duplications on the list. (In graph–theoretic terms (see Exercise 6.8) this says that if we define the edges of a graph to be the adjacent nodes of a tree, then the graph is acyclic.) Hint: Use Exercise 3.

6. Prove that if there is no infinite descending chain in a linear order $<$ on S then S is well ordered. (Those with some knowledge of set theory should note that we are assuming that S is countable as we do for all sets except in Chapter VI on set theory.)

7. Prove that the lexicographic ordering $<_L$ of pairs from the natural numbers $\mathcal{N}$ is well ordered.

8. a) Prove that the lexicographic ordering $<_L$ of n–tuples of natural numbers is well ordered for each n.

   b) Prove that the lexicographic ordering of the set of all n–tuples (with $n < m$ for any $m \in \mathcal{N}$) is well ordered.

9. Consider the set of all finite sequences of natural numbers. Define an ordering $<$ as follows: $\sigma < \tau$ iff either σ is shorter than τ or, if not, $\sigma <_L \tau$. Prove that $<$ is a well ordering.

The next two exercises are for those readers familiar with the topological notions of product topologies and compactness.

10. Show that König's lemma for binary trees is equivalent to the compactness of the topological space $C = \{0,1\}^\omega$ where $\{0,1\}$ is given the discrete topology and C the product topology.

11. Show that König's lemma for all finitely branching trees is equivalent to the compactness of all spaces $\prod X_i$ for every sequence of finite sets X_i, $i \in \mathcal{N}$, where each X_i has the discrete topology.

2 Propositions, Connectives and Truth Tables

Propositions are just statements and propositional logic describes and studies the ways in which statements are combined to form other statements. This is what is called the syntactic part of logic, the one which deals with statements as just strings (i.e., sequences) of symbols. We also are concerned with ascribing meaning to the symbols in various ways. This part of language is called semantics and a major theme in the development of logic is the relationship between these two aspects of language. The analysis of the internal structure of statements is left to a later time and a subject called predicate logic. For now we consider some of the ways in which one builds statements from other statements in English. The construction procedures we consider are the ones basic to mathematical texts. We call the operations that combine propositions to form new ones *connectives*. The connectives one finds most frequently in a mathematical text are "or", "and", "not", "implies" and "if and only if". The meaning given to them by the working mathematician does not precisely reflect their meaning in everyday discourse; it has been changed slightly so as to become entirely unambiguous. They should be thought of simply as part of the jargon of mathematics.
We introduce formal symbols for these connectives as follows:

$\vee$	for "or"	*(disjunction)*
$\wedge$	for "and"	*(conjunction)*
$\neg$	for "not"	*(negation)*
$\rightarrow$	for "implies"	*(conditional)*
$\leftrightarrow$	for "if and only if"	*(biconditional).*

Before making the meaning of these connectives precise, we describe how they are used to form statements of propositional logic. The description of the syntax of any language begins with its alphabet. The *language of propositional logic* consists of the following symbols:

(i) *Connectives*: $\vee$, $\wedge$, $\neg$, $\rightarrow$, $\leftrightarrow$

(ii) *Parentheses*:) , (

(iii) *Propositional Letters*: A, A_1, A_2, ..., B, B_1, B_2, ..., ...

Once the symbols of our language are specified, we can describe the statements of the language of propositions. The definition that follows selects certain strings of symbols from the language and calls them propositions. It is an *inductive* definition that describes the "shortest" statements first and then describes how to build longer statements from shorter ones in accordance with certain definite rules.

Definition 2.1 (Propositions):

(i) Propositional letters are propositions.

(ii) If α and β are propositions, then $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\neg\alpha)$, $(\alpha \rightarrow \beta)$ and $(\alpha \leftrightarrow \beta)$ are propositions.

(iii) A string of symbols is a proposition if and only if it can be obtained by starting with propositional letters (i) and repeatedly applying (ii).

For example, $(A \vee B)$, C, $((A \wedge B) \rightarrow C)$, $(\neg(A \wedge B) \rightarrow C)$ are all propositions while $A \wedge \neg$, $(A \vee B$, $(\wedge \rightarrow A)$ are not. We return to these examples in 2.3 below. The importance of Clause (iii) in the above definition is that it provides us with the basis for applying induction directly to propositions. We also return to this subject after first introducing another approach to describing propositions.

Labeled (binary) trees provide us with an important way of representing propositions. It is not hard to see that each proposition φ can be represented as a finite labeled binary tree T. The leaves of T are labeled with propositional letters. If any nonterminal node of T is labeled with a proposition α, its immediate successors are labeled with propositions (one or two) that can be combined using one of the connectives to form α. The left to right ordering on the immediate successors of α is given by the syntactic position of the component propositions. This procedure can be carried out so that the original proposition φ is the label of the root of the tree T. One could take such labeled trees to define propositions and ignore the inductive definition given above. We instead offer a precise translation between these notions:

Definition 2.2: A *formation tree* is a finite tree T of binary sequences (with root $\emptyset$ and a left to right ordering given by the ordinary lexicographic ordering of sequences) whose nodes are all labeled with propositions. The labeling satisfies the following conditions:

(i) The leaves are labeled with propositional letters.

(ii) If a node σ is labeled with a proposition of the form $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\alpha \rightarrow \beta)$ or $(\alpha \leftrightarrow \beta)$, its immediate successors, $\sigma^\frown 0$ and $\sigma^\frown 1$, are labeled with α and β (in that order).

(iii) If a node σ is labeled with a proposition of the form $(\neg\alpha)$, its unique immediate successor, $\sigma^\frown 0$, is labeled with α.

The formation tree T *represents* or *is associated* with the proposition with which its root is labeled.

Example 2.3: We can depict the formation trees associated with the correctly formed propositions listed above by inserting the appropriate labels for nodes on the proper trees as in Figure 2:

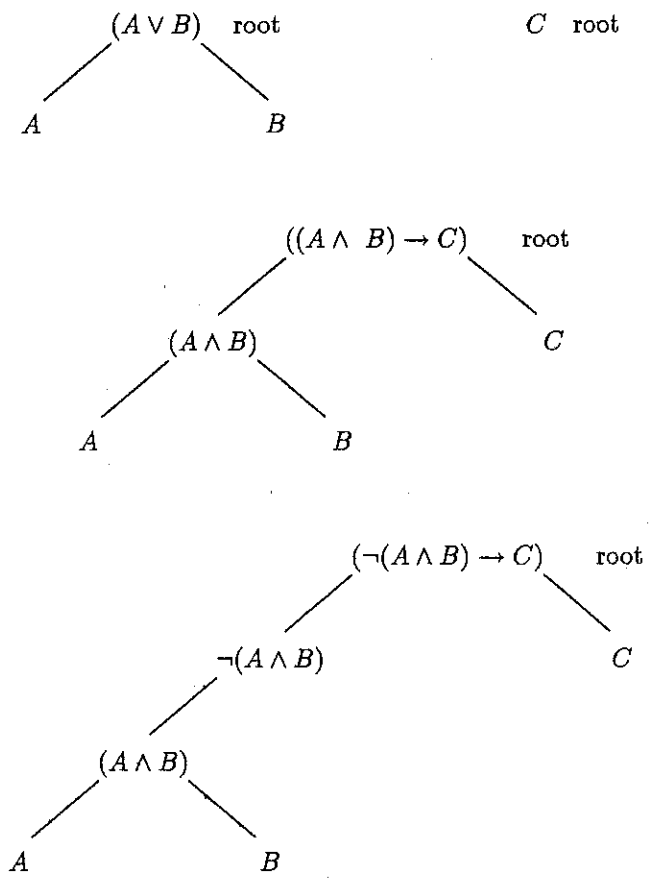

FIGURE 2.

Note that in these examples we talked of *the* formation tree associated with a given proposition. Behind this usage stands a theorem that there is, in fact, a unique formation tree associated with each proposition. The method of proof for such a theorem is induction. The definition of propositions is an inductive one: the propositional letters are the base case and the formation rules given by the various connectives constitute the inductive step. Clause (iii) of the definition says that all propositions are included in this process. Corresponding to this type of definition we have both other definitions and proofs by induction. Indeed, induction is the primary method for dealing with most of the notions we consider. Thus, for example, to define a formation tree associated with each proposition it suffices to define one for each propositional letter and to specify how to define one for a proposition constructed from others via each of the connectives in terms of the ones defined for its constituent propositions. The corresponding method of proof by induction establishes that some property P (such as there being at most

one formation tree associated with each proposition) holds for every proposition by first showing that it holds for each propositional letter (the base case) and then showing that, if it holds for propositions α and β, then it holds for each of the propositions constructed from α and β via the five basic connectives. This method is really nothing more than the usual procedure of induction on natural numbers. A translation into that terminology is supplied in Exercise 15.

Theorem 2.4: *Each proposition has a unique formation tree associated with it.*

Proof: We first show by induction that each proposition α has a formation tree associated with it. The base case is that α is a propositional letter, say A. In this case the tree consisting precisely of $\emptyset$ (its root and only node) labeled with A is the desired tree. For the induction step consider the proposition $\alpha \to \beta$. By induction there are formation trees T_α and T_β associated with α and β, respectively. The desired formation tree $T_{(\alpha\to\beta)}$ for $(\alpha \to \beta)$ has as its root $\emptyset$ labeled with $(\alpha \to \beta)$. Below this root we attach copies of T_α (on the left) and T_β (on the right). This can be described formally as letting the other nodes of $T_{(\alpha\to\beta)}$ be all sequences $0^\frown\sigma$ for every σ on T_α and $1^\frown\tau$ for every τ on T_β. The labels are the same as they were in the original trees. As both T_α and T_β were formation trees, it is clear that $T_{(\alpha\to\beta)}$ is one as well. (The labeling of nodes is inherited from the original trees except for that of the root by $(\alpha \to \beta)$. This labeling of the root is acceptable as the root's immediate successors are, by definition, labeled with α and β, respectively.) The cases for the other binary connectives are handled in exactly the same way. For $(\neg\alpha)$ we simply add on the nodes $0^\frown\sigma$ for σ in T_α (with the same label as σ) to the root $\emptyset$ which we label with $(\neg\alpha)$. Note that if the formation trees for α and β have depth n and m, respectively, the tree for any of the propositions built by applying one of the basic connectives has depth $\max\{n, m\} + 1$.

We next claim that there is at most one formation tree associated with each proposition. For the propositional letters, this claim is clear. As the root of the tree $\emptyset$ must be labeled by the propositional letter if the tree is associated with it, the definition requires that the root be a leaf, i.e., it is the entire tree. Consider now the inductive case for $(\alpha \to \beta)$ as an example. The crucial claim is that if T is a formation tree associated with $(\alpha \to \beta)$, then its root $\emptyset$ must be labeled with $(\alpha \to \beta)$. Suppose on the other hand that it is labeled with some proposition such as $(\gamma \vee \delta)$ and so $(\alpha \to \beta) = (\gamma \vee \delta)$. It is clear that, in this case, one of α and γ must be a proper initial segment of the other. This, however, would contradict the simple facts about parentheses in propositions established in Exercises 3 and 4. The other possibilities for the form of the proposition that might be the same string as $(\alpha \to \beta)$ are handled similarly (see Exercise 17). Now, by the definition of a formation tree, $\emptyset$ must have two immediate successors 0 and 1 which, given our claim, must be labeled with α and β, respectively. Every node on T below $i = 0$ or 1 must be of the form $0^\frown\sigma$ or $1^\frown\sigma$, respectively, for some binary sequence σ. For $n = 0, 1$ let $T_n = \{\sigma |\ n^\frown\sigma \in T\}$ have the standard ordering and be labeled as in T. It is clear that T_0 is a formation tree for α and T_1 for β. They are unique by induction and so T has been uniquely determined as required.

The other connectives are handled similarly and are left to Exercise 17. □

This theorem corresponds to what is often called the *unique readability* of propositions: There is only one way to parse a proposition into its component parts all the way down to the propositional letters. Along these lines, we, in informal usage, abuse our notation by omitting parentheses whenever no confusion can arise. Thus, for example, we write $\neg\alpha$ for $(\neg\alpha)$ and $\alpha \to \beta$ for $(\alpha \to \beta)$. Formally, unique readability gives us another way to define functions on the propositions and prove facts about them: induction on formation trees. Typically, we induct on the depth of the formation tree associated with a proposition. The advantage of using trees is that, if one defines a function on formation trees, one automatically has one on the associated propositions. If instead, one defined an operation directly on propositions by induction, one would not know that there is only one way of analyzing a given proposition inductively so as to guarantee that the operation is well defined. This is precisely unique readability. We see some examples of such procedures in the next section. For now, we just note that the theorem allows us to define the depth of a proposition. We can also use it to pick out the propositional letters that are "relevant" to a given proposition:

Definition 2.5:

(i) The *depth* of a proposition is the depth of the associated formation tree.

(ii) The *support* of a proposition is the set of propositional letters that occur as labels of the leaves of the associated formation tree. (That this notion corresponds to the ones that occur syntactically in the proposition is proven by another induction argument. See Exercise 16.)

*Closure operations and inductive definitions:

Another approach to the type of inductive definition given may clarify the role of Definition 2.1(iii) in guaranteeing that only those expressions generated by (i) and (ii) are propositions. We begin with the (algebraic) notion of closure. A set S is *closed* under a single (for example n–ary) operation $f(s_1, \ldots, s_n)$ iff for every $s_1, \ldots, s_n \in S$, $f(s_1, \ldots, s_n) \in S$. The *closure* of a set S under (all) the operations in a set T is the smallest set C such that

1. $S \subseteq C$ and
2. if $f \in T$ is n–ary and $s_1, \ldots, s_n \in C$, then $f(s_1, \ldots, s_n) \in C$.

To see that there is a smallest such set consider the set

$$C = \cap\{D |\ S \subseteq D \ \&\ D \text{ is closed under the operations of } T\}.$$

Of course $S \subseteq C$. Now show that C is closed under the operations of T. It is then clear that C is the smallest such set as it is contained in every set $D \supseteq S$ that is

closed under the operations of T. We could now define the set of propositions as the closure of the set of propositional letters (i) under the operations $\wedge$, $\vee$, $\neg$, $\rightarrow$ and $\leftrightarrow$ as listed in (ii).

Turning now to semantics, we take the view that the meaning of a propositional letter is simply its truth value, that is, its truth or falsity. (Remember that we are postponing the analysis of the internal structure of propositions to the next chapter.) Each proposition then has a unique *truth value* (T, for true or F, for false). The truth value of a compound proposition is determined from the truth values of its parts in accordance with the *truth tables* in Figure 3:

Definition 2.6 (Truth tables):

α	β	$(\alpha \vee \beta)$
T	T	T
T	F	T
F	T	T
F	F	F

α	β	$(\alpha \wedge \beta)$
T	T	T
T	F	F
F	T	F
F	F	F

α	β	$(\alpha \rightarrow \beta)$
T	T	T
T	F	F
F	T	T
F	F	T

α	β	$(\alpha \leftrightarrow \beta)$
T	T	T
T	F	F
F	T	F
F	F	T

α	$\neg\alpha$
T	F
F	T

FIGURE 3.

As pointed out earlier, the meaning of these connectives as specified by these truth tables is not exactly the same as in ordinary English. For $\vee$, the meaning is that of the inclusive or: $\alpha \vee \beta$ is true if either or both of α and β are true. The meaning of $\rightarrow$ is further removed from that in colloquial usage. In mathematics, $(\alpha \rightarrow \beta)$ is asserted to be false only when α is true and β is false. It is asserted to be true in all other cases.

The formal assignment of truth values to propositions based on those of the propositional letters is given in the next section. Intuitively, it should be clear from the inductive definition of propositions how, given any proposition whatsoever, we can construct a truth table for it by considering it as being built up step by step starting from propositional letters. For example, Figure 4 is the truth table for $((A \wedge B) \rightarrow C)$.

A	B	C	$(A \wedge B)$	$((A \wedge B) \rightarrow C)$
T	T	T	T	T
T	T	F	T	F
T	F	T	F	T
T	F	F	F	T
F	T	T	F	T
F	T	F	F	T
F	F	T	F	T
F	F	F	F	T

FIGURE 4.

The eight combinations of truth values for A, B, C ($2^3 = 8$) can be thought of as all possible states of the world as far as any proposition in which only the propositional letters A, B, C appear is concerned. The column for $(A \wedge B)$ is auxiliary and could be eliminated. The result would be the *abbreviated truth table* for $((A \wedge B) \rightarrow C)$. If the convention implicit in the above table for systematically listing the (eight) possible combinations of truth values for the propositional letters A, B and C is observed, then it is clear that to any proposition there corresponds a unique abbreviated truth table.

A priori, we might have begun our definition of propositions with some other list of basic connectives. In general, an *n–ary connective* is any function σ that assigns a proposition $\sigma(A_1, \ldots, A_n)$ to every n–tuple of propositions $A_1, \ldots, A_n$. So $\neg$ is 1–ary (*unary*), while $\wedge$ and $\vee$ are 2–ary (or *binary*). An n–ary connective is *truth functional* if the truth value for $\sigma(A_1, \ldots, A_n)$ is uniquely determined by the truth values for $A_1, \ldots, A_n$. Our five connectives are truth functional since their meaning was defined by truth tables. On the other hand a connective like "because" is not. For let A symbolize "I had prune juice for breakfast" and B "there was an earthquake at noon". Even in the event that both A and B have truth values T it is at least debatable whether (B because A) should have truth value T. The debate might be more or less heated in other cases depending on the content of A and B. An n–ary connective that is truth functional can be completely described by means of a truth table as in Figure 5 where each b_i, $1 \leq i \leq 2^n$ is either T or F.

Conversely, two distinct abbreviated truth tables (with the conventional listing of truth values for $A_1, \ldots, A_n$) correspond to distinct truth functional connectives. By counting we see that there are 2^{2^n} distinct n–ary truth functional connectives. (So there are $12 = 16 - 4$ binary connectives that we are not using.)

Definition 2.7: A set S of truth functional connectives is *adequate* if, given any truth functional connective σ, we can find a proposition built up from the connectives in S with the same abbreviated truth table as σ.

A_1	A_2	$\cdots$	A_n	$\sigma(A_1,\ldots,A_k)$
T	T	$\cdots$	T	b_1
T	T	$\cdots$	F	b_2
.	.	$\cdots$	.	.
.	.	$\cdots$	.	.
F	F	$\cdots$	F	.

FIGURE 5.

Theorem 2.8 (Adequacy): $\{\neg, \wedge, \vee\}$ *is adequate.*

Proof: Let $A_1, \ldots, A_k$ be distinct propositional letters and let a_{ij} denote the entry (T or F) corresponding to the ith row and j^{th} column of the truth table for $\sigma(A_1, \ldots, A_k)$ as in Figure 6. Suppose that at least one T appears in the last column.

A_1	$\cdots$	A_j	$\cdots$	A_k	$\cdots$	$\sigma(A_1,\ldots,A_k)$
						b_1
						b_2
						.
						.
		a_{ij}				b_i

FIGURE 6.

For any proposition α, let α^T be α and α^F be $(\neg\alpha)$. For the ith row denote the *conjunction* $(A_1^{a_{i1}} \wedge \ldots \wedge A_k^{a_{ik}})$ by a_i. Let $i_1, \ldots, i_m$ be the rows with a T in the last column. The desired proposition is the *disjunction* $(a_{i_1} \vee \ldots \vee a_{i_m})$. The proof that this proposition has the given truth table is left as Exercise 14. (Note that we abused our notation by leaving out a lot of parentheses in the interest of readability. The convention is that of *right associativity*, that is, $A \wedge B \wedge C$ is an abbreviation for $(A \wedge (B \wedge C))$.) We also indicate a disjunction over a set of propositions with the usual set–theoretic terminology. Thus, the disjunction just constructed would be written as $\bigvee\{a_i |\ b_i = T\}$. □

Example 2.9: The procedure given in the above proof can be illustrated by constructing a proposition built using only $\wedge$, $\vee$ and $\neg$ which has the truth table given in Figure 7.

	A	B	C	?
1	T	T	T	T
2	T	T	F	F
3	T	F	T	F
4	T	F	F	F
5	F	T	T	T
6	F	T	F	F
7	F	F	T	F
8	F	F	F	T

FIGURE 7.

We begin by looking only at rows with a T in the last column. For each such row we find a proposition which is true for that row and false for every other row. The proposition we want is the disjunction of the propositions we obtain for all relevant rows (rows 1, 5, 8 in this case). For any particular row, the proposition true for only that row is obtained by taking the conjunction of the letters having a T in that row and the negations of letters having an F in that row. In this case row 1 gives $(A \wedge B \wedge C)$ (we abuse notation again!); row 5 gives $((\neg A) \wedge B \wedge C)$; and row 8 gives $((\neg A) \wedge (\neg B) \wedge (\neg C))$. Thus, the proposition $(A \wedge B \wedge C) \vee \big((\neg A) \wedge B \wedge C\big) \vee \big((\neg A) \wedge (\neg B) \wedge (\neg C)\big)$ has the given truth table.

Clearly, given any proposition α we can construct its truth table and then follow the above procedure to find another proposition that has the same truth table and is a disjunction of conjunctions of propositional letters and their negations. A proposition of this form that has the same (abbreviated) truth table as α is called a *disjunctive normal form* (DNF) of α. There is also a *conjunctive normal form* (CNF) equivalent of α that is presented in Exercise 3.3. Another method of finding DNF and CNF equivalents of α is presented at the end of the exercises for §4.

Remark 2.10: The above procedure does not tell us what to do in case the last column consists entirely of F's. See Exercise 13.

Corollary 2.11: $\{\neg, \vee\}$ *is adequate.*

Proof: We can easily check that $(A_1 \wedge A_2)$ has the same truth table as $\neg\big((\neg(A_1)) \vee (\neg(A_2))\big)$. Thus, given any proposition α we can find a DNF of α and then eliminate any use of $\wedge$ by this substitution. The resulting proposition will still have the same truth table. □

The sets $\{\neg, \wedge\}$ and $\{\neg, \rightarrow\}$ are also shown to be adequate in the exercises. If a set is not adequate, how do you prove that? (See Exercise 10.)

Remark 2.12: By the adequacy theorems (Theorem 2.8 and Corollary 2.11) we could, in theory, get by with just the connectives $\neg$, $\vee$ and $\wedge$ or even just $\neg$ and $\vee$. The induction clause in the definition of propositions and many related definitions and proofs (such as those involving tableaux in Sections 4, 5 and 6) could then be considerably shortened. We, however, leave the list of connectives as it is but generally explicitly deal with only a couple of cases in any particular proof and leave the rest as exercises.

Exercises

1. Which of the following expressions are official (i. e., unabbreviated) propositions of propositional logic based on the propositional letters A, B, C, D?

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

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

   (c) $A \rightarrow (B \wedge C)$

   (d) $((A \leftrightarrow B) \rightarrow (\neg A))$

   (e) $((\neg A) \rightarrow B \vee C)$

   (f) $(((C \vee B) \wedge A) \leftrightarrow D)$

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

   (h) $(A \wedge (B \wedge C)))$.

2. Prove your answers to 1(a), (b) and (f) by either giving the step by step procedure producing the proposition in accordance with the inductive definition of propositions (you can simply draw a correctly labeled formation tree) or proving, by induction on statements, that there is some property enjoyed by all propositions but not by this expression.

3. Prove that every proposition other than a propositional letter begins with a left parenthesis and ends with a right parenthesis. Moreover, the number of right and left parentheses are equal in every proposition.

4. Prove that in any nonempty proper initial segment of a proposition there are more left than right parentheses.

5. Find DNF equivalents for the following propositions:

   (a) $(A \to B) \to C$

   (b) $(A \leftrightarrow B) \vee (\neg C)$.

6. Prove that $\{\neg, \wedge\}$ is an adequate set of connectives. (Hint: Express $\vee$ in terms of $\neg$ and $\wedge$.)

7. Prove that $\{\neg, \to\}$ is an adequate set of connectives.

8. Prove that the binary connective $(\alpha \mid \beta)$ ("not both ... and") called the *Sheffer stroke* whose truth table is given in Figure 8 is adequate.

α	β	$\alpha \mid \beta$
T	T	F
T	F	T
F	T	T
F	F	T

FIGURE 8.

   (Hint: Express $\neg$ and $\wedge$ in terms of $\mid$.)

9. Show that joint denial (neither α nor β), written as $\alpha \downarrow \beta$, is also adequate.

10. Prove that $\{\wedge, \vee\}$ is not adequate.
    Hint: Show by induction that $\neg\alpha$ is not equivalent to any statement built up from α using only $\wedge$ and $\vee$.

11. Prove that $\{\vee, \to\}$ is not an adequate set of connectives.

12. Prove that $\{\vee, \to, \leftrightarrow\}$ is not an adequate set of connectives.

13. Explain how to handle the case of a column of all F's in the proof of Theorem 2.8.

14. Prove that the expressions constructed in the proof of Theorem 2.8 (including the case considered in Exercise 13) have the desired truth tables.

15. We say that all propositional letters are built at stage 0. If propositions α and β have been built by level n, we say that $(\neg\alpha)$, $(\alpha \vee \beta)$, $(\alpha \wedge \beta)$, $(\alpha \to \beta)$ and $(\alpha \leftrightarrow \beta)$ have been built by level $n+1$. Clause (iii) of Definition 2.1 says that every proposition φ is built by some level n. Explain how we can rephrase proof by induction on the definition of propositions in terms of ordinary induction on the natural numbers $\mathcal{N}$.

(Hint: Proving that all propositions have property P by induction on propositions corresponds to proving that all propositions built by level n have property P by induction on n.)

16. We say that each propositional letter A *occurs* in itself and no propositional letters other than A occur in A. The propositional letters that *occur* in $(\neg\alpha)$ are precisely the ones that occur in α. The ones that occur in $(\alpha \vee \beta)$, $(\alpha \wedge \beta)$, $(\alpha \rightarrow \beta)$ and $(\alpha \leftrightarrow \beta)$ are precisely those that occur in either α or β (or both). This notion clearly captures the idea of a syntactic occurrence of a propositional letter A in a proposition α.
Prove that the support of a proposition α is precisely the set of propositional letters that occur in α.

3 Truth Assignments and Valuations

Our view of propositional logic is that the meaning or content of a proposition is just its truth value. Thus, the whole notion of semantics for propositional logic consists of assigning truth values to propositions. We begin with the propositional letters.

Definition 3.1: A *truth assignment* $\mathcal{A}$ is a function that assigns to each propositional letter A a unique truth value $\mathcal{A}(A) \in \{T, F\}$.

The truth values of all propositions should now be determined by the assignment to the propositional letters. The determinations are made in accordance with the truth tables for the connectives given in the last section.

Definition 3.2: A *truth valuation* $\mathcal{V}$ is a function that assigns to each proposition α a unique truth value $\mathcal{V}(\alpha)$ so that its value on a compound proposition (that is, one with a connective) is determined in accordance with the appropriate truth tables. Thus, for example, $\mathcal{V}((\neg\alpha)) = T$ iff $\mathcal{V}(\alpha) = F$ and $\mathcal{V}((\alpha \vee \beta)) = T$ iff $\mathcal{V}(\alpha) = T$ or $\mathcal{V}(\beta) = T$. We say that $\mathcal{V}$ *makes* α *true* if $\mathcal{V}(\alpha) = T$.

The basic result here is that a truth assignment to the propositional letters uniquely determines the entire truth valuation on all propositions. We analyze the situation in terms of an induction on the depth of the propositions, that is, the depth of the (unique) formation tree associated with the proposition.

Theorem 3.3: *Given a truth assignment $\mathcal{A}$ there is a unique truth valuation $\mathcal{V}$ such that $\mathcal{V}(\alpha) = \mathcal{A}(\alpha)$ for every propositional letter α.*

Proof: Given a truth assignment $\mathcal{A}$, define (by induction on the depth of the associated formation tree) a valuation $\mathcal{V}$ on all propositions by first setting $\mathcal{V}(\alpha) = \mathcal{A}(\alpha)$ for all propositional letters α. This takes care of all formation trees (propositions) of depth 0. Assuming that $\mathcal{V}$ has been defined on all propositions with depth at most n, the inductive steps are simply given by the truth tables associated with each connective. For example, suppose $T_{(\alpha\to\beta)}$ is the formation tree (of depth $n+1$) for $(\alpha \to \beta)$. (It is built from T_α and T_β (with the maximum of their depths being exactly n) as in Theorem 2.4.) $\mathcal{V}((\alpha \to \beta))$ is then defined to be F iff $\mathcal{V}(\alpha) = T$ and $\mathcal{V}(\beta) = F$. The valuation is defined on α and β by induction since they have depth at most n.

Clearly $\mathcal{V}$ has been defined so as to be a valuation and it does extend $\mathcal{A}$. It remains to show that any two valuations $\mathcal{V}_1$, $\mathcal{V}_2$ both extending $\mathcal{A}$ must coincide. We prove this by induction on the depth of propositions:

(i) $\mathcal{V}_1(\alpha) = \mathcal{V}_2(\alpha)$ for all propositional letters α (depth 0) since $\mathcal{V}_1$, $\mathcal{V}_2$ both extend $\mathcal{A}$.

(ii) Suppose $\mathcal{V}_1(\alpha) = \mathcal{V}_2(\alpha)$ for all propositions α of depth at most n and that α and β have depth at most n. Thus, $\mathcal{V}_1(\alpha) = \mathcal{V}_2(\alpha)$ and $\mathcal{V}_1(\beta) = \mathcal{V}_2(\beta)$ by induction. $\mathcal{V}_1((\alpha \wedge \beta))$ and $\mathcal{V}_2((\alpha \wedge \beta))$ are then both given by the truth table for $\wedge$ and so are equal. The same argument works for all the other connectives and so $\mathcal{V}_1$ and $\mathcal{V}_2$ agree on every proposition. □

Note that, by induction again on the depth of α, the definition of $\mathcal{V}(\alpha)$ in this construction only depends on the values of $\mathcal{A}$ on the support of α (the propositional letters occurring in α). Thus, the proof of the theorem actually proves:

Corollary 3.4: *If $\mathcal{V}_1$ and $\mathcal{V}_2$ are two valuations that agree on the support of α, the finite set of propositional letters used in the construction of the proposition α, then $\mathcal{V}_1(\alpha) = \mathcal{V}_2(\alpha)$.* □

Definition 3.5: A proposition σ of propositional logic is said to be *valid* if for any valuation $\mathcal{V}$, $\mathcal{V}(\sigma) = T$. Such a proposition is also called a *tautology*.

Definition 3.6: Two propositions α and β such that, for every valuation $\mathcal{V}$, $\mathcal{V}(\alpha) = \mathcal{V}(\beta)$ are called *logically equivalent*. We denote this by $\alpha \equiv \beta$.

Example 3.7:

(i) $(A \vee (\neg A))$ and $(((A \to B) \to A) \to A)$ (Law of the excluded middle and Peirce's law) are tautologies. See Exercise 1.

(ii) For any proposition α and any DNF β of α, $\alpha \equiv \beta$.

(iii) We could rephrase the adequacy theorem (2.8) to say that, given any proposition α, we can find a β that uses only $\neg$, $\vee$, $\wedge$ and such that $\alpha \equiv \beta$.

Although Corollary 3.4 allows us to check whether a given proposition is a tautology or not using Definition 2.5, it also tells us that we can answer the same question by finding out whether the last column of the corresponding truth table has all T's or not. We choose not to develop these proofs by truth tables further because they do not generalize to proofs for predicate logic which we study shortly. We close this section with some definitions and notations which are important later and generalize nicely to the corresponding concepts in the logic of predicates.

Definition 3.8: Let Σ be a (possibly infinite) set of propositions. We say that σ *is a consequence of* Σ (and write $\Sigma \vDash \sigma$) if, for any valuation $\mathcal{V}$,

$$(\mathcal{V}(\tau) = \mathrm{T} \text{ for all } \tau \in \Sigma) \Rightarrow \mathcal{V}(\sigma) = \mathrm{T}.$$

Note that, if Σ is empty, $\Sigma \vDash \sigma$ (or just $\vDash \sigma$) iff σ is valid. We also write this as $\vDash \sigma$. This definition gives a semantic notion of consequence. We see several syntactic notions in the coming sections that correspond to different proof procedures. A major result is the equivalence of the syntactic and semantic notions of consequence that is embodied in the soundness and completeness theorems (§5).

Definition 3.9: We say that a valuation $\mathcal{V}$ is a *model* of Σ if $\mathcal{V}(\sigma) = T$ for every $\sigma \in \Sigma$. We denote by $\mathcal{M}(\Sigma)$ the set of all models of Σ.

Notation: Rather than writing "implies" and "if and only if" in our definitions, theorems, etc., we often use $\Rightarrow$ and $\Leftrightarrow$ instead. These are not symbols of the language of propositional logic but of the language (or metalanguage) in which we discuss propositional logic.

Proposition 3.10: *Let Σ, Σ_1, Σ_2 be sets of propositions. Let $Cn(\Sigma)$ denote the set of consequences of Σ and Taut the set of all tautologies.*

(i) $\Sigma_1 \subseteq \Sigma_2 \Rightarrow Cn(\Sigma_1) \subseteq Cn(\Sigma_2)$.

(ii) $\Sigma \subseteq Cn(\Sigma)$.

(iii) $Taut \subseteq Cn(\Sigma)$ *for all* Σ.

(iv) $Cn(\Sigma) = Cn(Cn(\Sigma))$.

(v) $\Sigma_1 \subseteq \Sigma_2 \Rightarrow \mathcal{M}(\Sigma_2) \subseteq \mathcal{M}(\Sigma_1)$.

(vi) $Cn(\Sigma) = \{\sigma |\ \mathcal{V}(\sigma) = \mathrm{T}$ *for all* $\mathcal{V} \in \mathcal{M}(\pm)\}$.

(vii) $\sigma \in Cn(\{\sigma_1, \ldots, \sigma_n\}) \Leftrightarrow \sigma_1 \rightarrow (\sigma_2 \ldots \rightarrow (\sigma_n \rightarrow \sigma) \ldots) \in Taut$.

We leave the proof of this proposition as Exercise 5.

The last assertion of Proposition 3.10 tells us that testing whether σ is a consequence of a finite set Σ of propositions (sometimes called "premises") can be done in finitely many steps by checking (for example, by using truth tables) whether the proposition on the right–hand side of (vii) is a tautology. But what do we do if Σ is infinite? We had better learn how to *prove* that σ is a consequence. The first method we consider is that of tableaux.

Exercises

1. Prove that the propositions in Example 3.7 (i) are tautologies by checking directly, using Corollary 3.4, that they are true under all valuations.

2. Prove De Morgan's laws for any propositions $\alpha_1, \ldots, \alpha_n$, i.e.,

   (a) $\neg(\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n) \equiv \neg\alpha_1 \wedge \neg\alpha_2 \wedge \ldots \wedge \neg\alpha_n$

   (b) $\neg(\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n) \equiv \neg\alpha_1 \vee \neg\alpha_2 \vee \ldots \vee \neg\alpha_n$.

   Hint: Do not write out the truth tables. Argue directly from the truth conditions for disjunctions and conjunctions.

3. A proposition is a *literal* if it is a propositional letter or its negation. A proposition α is in *conjunctive normal form* (CNF) if there are literals $\alpha_{1,1}, \ldots, \alpha_{1,n_1}, \alpha_{2,1}, \ldots, \alpha_{2,n_2}, \ldots, \alpha_{k,1}, \ldots, \alpha_{2,n_k}$ such that α is

   $$(\alpha_{1,1} \vee \alpha_{1,2} \vee \ldots \vee \alpha_{1,n_1}) \wedge (\alpha_{2,1} \vee \alpha_{2,2} \vee \ldots \vee \alpha_{2,n_2}) \wedge \ldots \wedge (\alpha_{k,1} \vee \ldots \vee \alpha_{k,n_k}).$$

   Prove that every proposition is equivalent to one in CNF (i.e., one that has the same truth table). (Hint: Consider a DNF (of $\neg\alpha$) and use Exercise 2.)

4. Find a CNF for each of the following propositions:

   (a) $(A \wedge B \wedge C) \rightarrow D$.

   (b) $(A \wedge B) \rightarrow (C \vee D)$.

5. Supply the (short) proofs from the appropriate definitions for (i)–(vii) of Proposition 3.10.

4 Tableau Proofs in Propositional Calculus

We describe a system for building proofs of propositions. The proofs are labeled binary trees called tableaux. The labels on the trees are *signed propositions*, that is, a proposition preceded by either a T or an F (which we can think of as indicating an assumed truth value for the proposition). We call the labels of the nodes the *entries of the tableau*. Formally, we define (or describe how to build) tableaux for propositions inductively by first specifying certain (labeled binary) trees as tableaux (the so–called atomic tableaux) and then giving a development rule defining tableaux for compound propositions from tableaux for simple propositions.

The plan of the procedure is to start with some entry, i.e., some signed proposition such as $F\big(\neg(A \wedge (B \vee C))\big)$, and analyze it into its components. We say that an entry is correct if our assumption about the truth value of the given proposition is correct. For our current example, $F\big(\neg(A \wedge (B \vee C))\big)$, this would mean that $\neg(A \wedge (B \vee C))$ is false. The guiding principle for the analysis is that, if an entry is correct, then (at least) one of the sets of entries into which we analyze it contains only correct entries. In our sample case, we would analyze $F\big(\neg(A \wedge (B \vee C))\big)$ first into $T(A \wedge (B \vee C))$. (If $\neg(A \wedge (B \vee C))$ is false, then $(A \wedge (B \vee C))$ is true.) We would then analyze $T(A \wedge (B \vee C))$ into TA and $T(B \vee C)$. (If $(A \wedge (B \vee C))$ is true, then so are both A and $(B \vee C)$.) Next we would analyze $T(B \vee C)$ into either TB or TC. (If $(B \vee C)$ is true, then so is one of B or C.)

The intent of the procedure, as a way of producing proofs of propositions, is to start with some signed proposition, such as $F\alpha$, as the root of our tree and to analyze it into its components in such a way as to see that any analysis leads to a contradiction. We then conclude that we have refuted the original assumption that α is false and so have a proof of α. Suppose, for example, that we start with $F(\neg(A \wedge \neg A))$ and proceed as in the above analysis (replacing $(B \vee C)$ by $\neg A$). We reach TA and $T\neg A$ and then analyze $T\neg A$ into FA. We now have entries saying both that A is true and that it is false. This is the desired contradiction and we would conclude that we have a proof of the valid proposition $\neg(A \wedge \neg A)$.

The base case of our inductive definition of tableaux consists of the (labeled binary) trees displayed in Figure 9 as the *atomic tableaux* for any propositions α and β and propositional letter A. It then proceeds as in Definition 4.1.

Definition 4.1 (Tableaux): A *finite tableau* is a binary tree, labeled with signed propositions called entries, that satisfies the following inductive definition:

(i) All atomic tableaux are finite tableaux.

(ii) If τ is a finite tableau, P a path on τ, E an entry of τ occurring on P and τ' is obtained from τ by adjoining the unique atomic tableau with root entry E to τ at the end of the path P, then τ' is also a finite tableau.

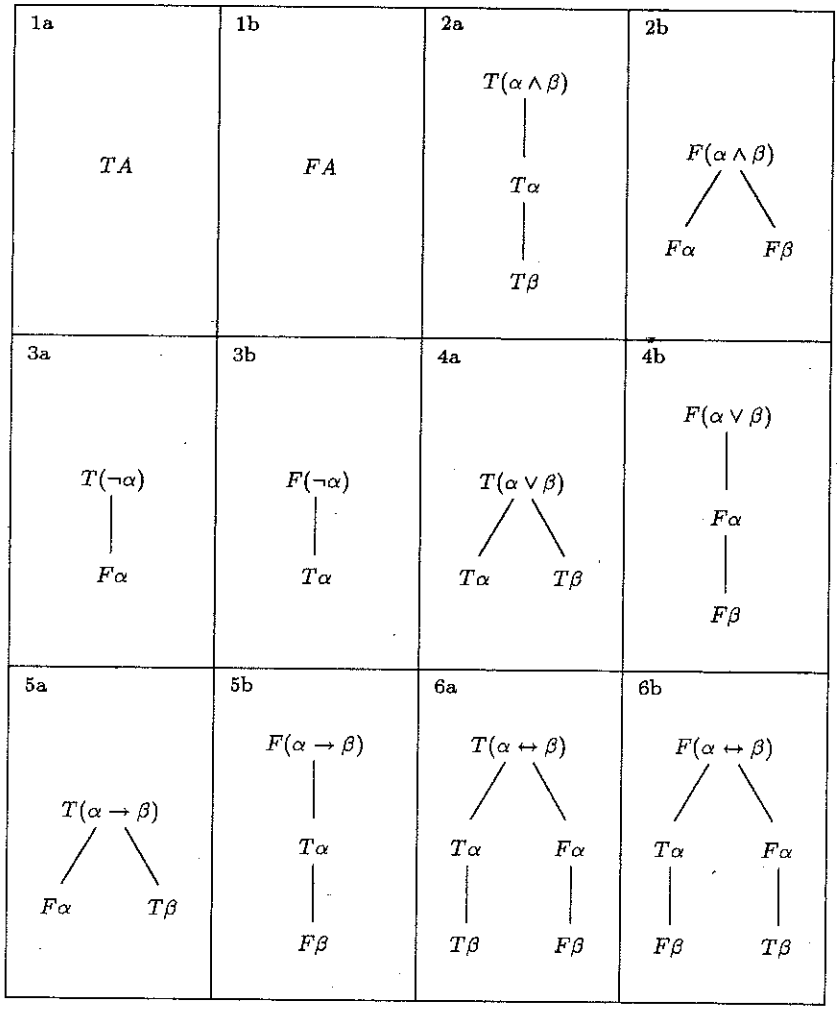
1a
TA
1b
FA
2a
$T(\alpha \wedge \beta)$
$T\alpha$
$T\beta$
2b
$F(\alpha \wedge \beta)$
$F\alpha$ $F\beta$
3a
$T(\neg\alpha)$
$F\alpha$
3b
$F(\neg\alpha)$
$T\alpha$
4a
$T(\alpha \vee \beta)$
$T\alpha$ $T\beta$
4b
$F(\alpha \vee \beta)$
$F\alpha$
$F\beta$
5a
$T(\alpha \to \beta)$
$F\alpha$ $T\beta$
5b
$F(\alpha \to \beta)$
$T\alpha$
$F\beta$
6a
$T(\alpha \leftrightarrow \beta)$
$T\alpha$ $F\alpha$
$T\beta$ $F\beta$
6b
$F(\alpha \leftrightarrow \beta)$
$T\alpha$ $F\alpha$
$F\beta$ $T\beta$

FIGURE 9.

If $\tau_0, \tau_1, \ldots, \tau_n, \ldots$ is a (finite or infinite) sequence of finite tableaux such that, for each $n \geq 0$, τ_{n+1} is constructed from τ_n by an application of (ii), then $\tau = \cup\tau_n$ is a *tableau*.

This definition describes all possible tableaux. In this section, we could get by with finite tableaux (see Theorem 4.11) but would necessarily be driven to infinite ones in predicate logic and even in the next section where we consider propositional logic with (infinitely many) premises.

Each tableau is a way of analyzing a proposition. The intent is that, if it is all right to assume that all the signs on entries on a path down to some entry E in a tableau are correct, then one of the paths of the tableau that continue on through E to the next level of the tree is also correct.

To see that this intention is realized, it suffices to consider the atomic tableaux. Consider, for example, (5a). If $\alpha \to \beta$ is true, then so is one of the branches through it: α is false or β is true. Similarly for (4a), if $\alpha \vee \beta$ is true, then so is one of α or β. The other atomic tableaux can be analyzed in the same way. This intuition is developed formally in the next section as the soundness theorem for tableaux. The other major theorem about tableaux is the completeness theorem. It is connected with the idea that we can show that if α is valid, then all possible analyses of a given signed proposition $F\alpha$ lead to contradictions. This constitutes a proof α. In order to do this, we have to develop a systematic method for generating a tableau with a given root which includes all possible procedures. First, however, some examples of tableaux.

Example 4.2: We wish to begin a tableau with the signed proposition $F\big(((\alpha \to \beta) \vee (\gamma \vee \delta)) \wedge (\alpha \vee \beta)\big)$. There is only one atomic tableau that has this entry as its root — the appropriate instance of the atomic tableau of type (2b) given in Figure 10.

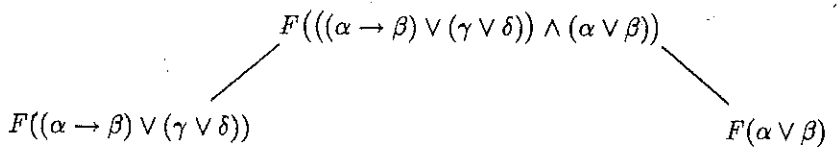

FIGURE 10.

Now this tableau has two entries other than its root either of which could be chosen for use in the induction clause to build a bigger tableau. (We could legally use the root entry again but that would not be very interesting.) The two possibilities are given in Figure 11 A and B.

We could also do each of these steps in turn to get the tableau given in Figure 11 C.

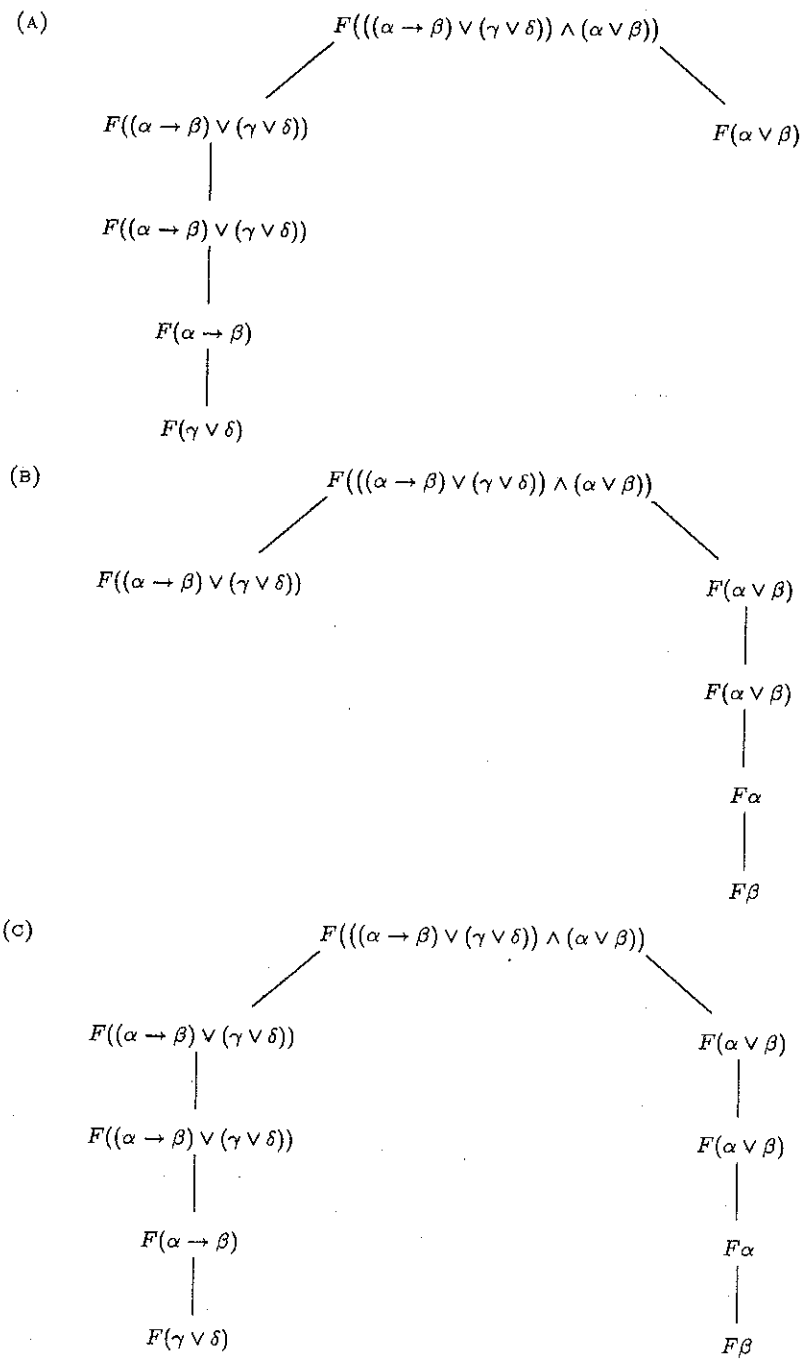
(A)
$F(((\alpha \to \beta) \vee (\gamma \vee \delta)) \wedge (\alpha \vee \beta))$
$F((\alpha \to \beta) \vee (\gamma \vee \delta))$
$F(\alpha \vee \beta)$
$F((\alpha \to \beta) \vee (\gamma \vee \delta))$
$F(\alpha \to \beta)$
$F(\gamma \vee \delta)$
(B)
$F(((\alpha \to \beta) \vee (\gamma \vee \delta)) \wedge (\alpha \vee \beta))$
$F((\alpha \to \beta) \vee (\gamma \vee \delta))$
$F(\alpha \vee \beta)$
$F(\alpha \vee \beta)$
$F\alpha$
$F\beta$
(C)
$F(((\alpha \to \beta) \vee (\gamma \vee \delta)) \wedge (\alpha \vee \beta))$
$F((\alpha \to \beta) \vee (\gamma \vee \delta))$
$F(\alpha \vee \beta)$
$F((\alpha \to \beta) \vee (\gamma \vee \delta))$
$F(\alpha \vee \beta)$
$F(\alpha \to \beta)$
$F\alpha$
$F(\gamma \vee \delta)$
$F\beta$

FIGURE 11.

In this last tableau we could (ignoring duplications) choose either $F(\alpha \to \beta)$ or $F(\gamma \vee \delta)$ as the entry to develop. $F(\gamma \vee \delta)$ is the end of the only path in the tableau that contains either of these entries. Thus, in either case the appropriate atomic tableau would be appended to that path. Choosing $F(\alpha \to \beta)$ would give the tableau of Figure 12.

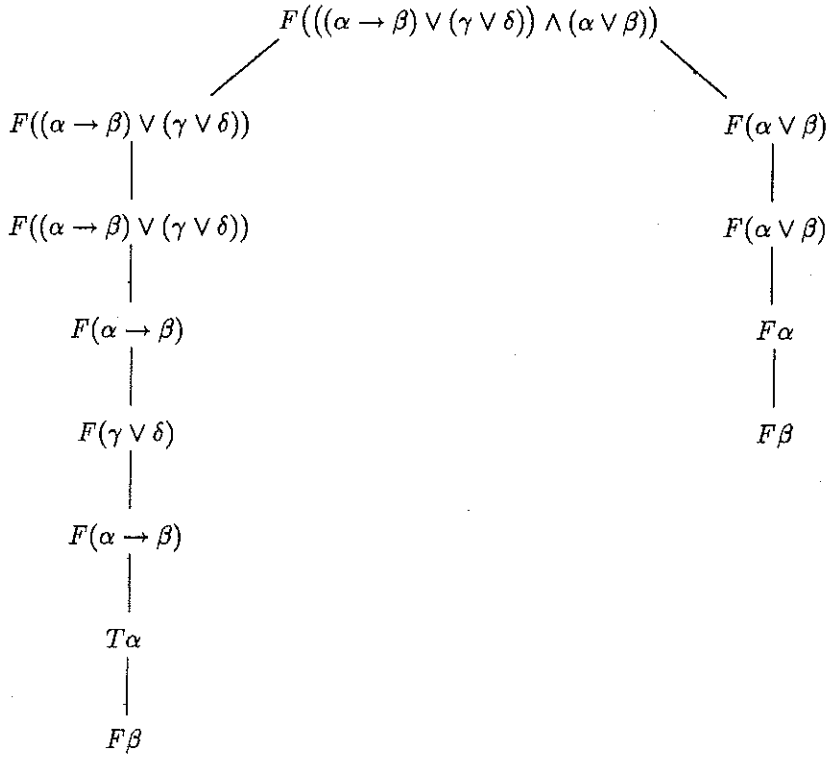

FIGURE 12.

As the reader must have noticed, each time we select an entry it gets repeated at the end of the chosen path as part of the atomic tableau that we affix to the end of the path. As a notational convenience we often omit this second occurrence when we draw tableaux although it remains part of the formal definition. (They actually are needed when we consider the predicate calculus and so we included them in our formal definition.)

We now wish to describe those tableaux that will constitute proofs and a systematic procedure for generating them from given signed propositions. We need a number of auxiliary notions:

Definition 4.3: Let τ be a tableau, P a path on τ and E an entry occurring on P.

(i) E has been *reduced* on P if all the entries on one path through the atomic

tableau with root E occur on P. (For example, TA and FA are reduced for every propositional letter A. $T\neg\alpha$ and $F\neg\alpha$ are reduced (on P) if $F\alpha$ and $T\alpha$, respectively, appear on P. $T(\alpha \vee \beta)$ is reduced if either $T\alpha$ or $T\beta$ appears on P. $F(\alpha \vee \beta)$ is reduced if both $F\alpha$ and $F\beta$ appear on P.)

(ii) P is *contradictory* if, for some proposition α, $T\alpha$ and $F\alpha$ are both entries on P. P is *finished* if it is contradictory or every entry on P is reduced on P.

(iii) τ is *finished* if every path through τ is finished.

(iv) τ is *contradictory* if every path through τ is contradictory. (It is, of course, then finished as well.)

Example 4.4: Figure 13 shows a finished tableau with three paths. The leftmost path is contradictory; the other two are not.

We can now define tableau proofs of α as ones that show that the assumption that α is false always leads to a contradiction.

Definition 4.5: A *tableau proof* of a proposition α is a contradictory tableau with root entry $F\alpha$. A proposition is *tableau provable*, written $\vdash \alpha$, if it has a tableau proof.

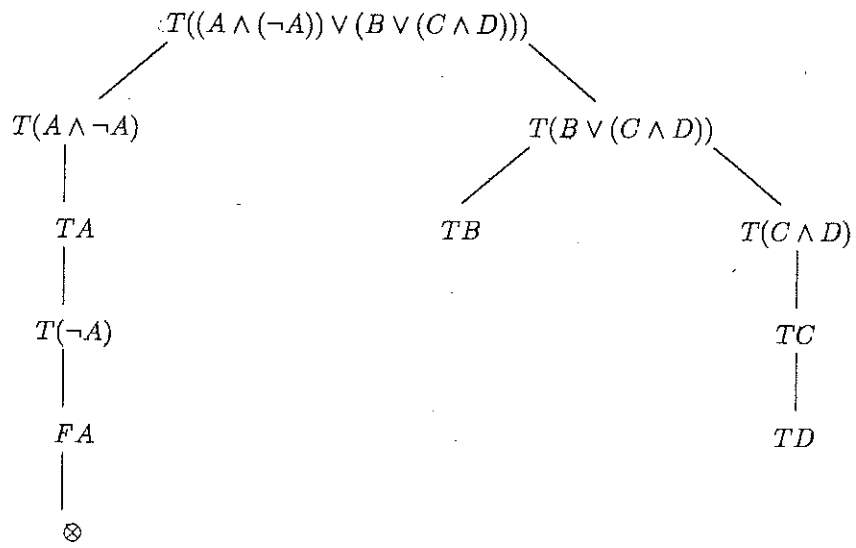

FIGURE 13.

A *tableau refutation* for a proposition α is a contradictory tableau starting with $T\alpha$. A proposition is *tableau refutable* if it has a tableau refutation.

Example 4.6: Peirce's law. Figure 14 gives a tableau proof of an instance of Peirce's law. Remember that we don't actually recopy the entries that we are reducing. We put $\otimes$ at the end of a path to denote that it is contradictory.

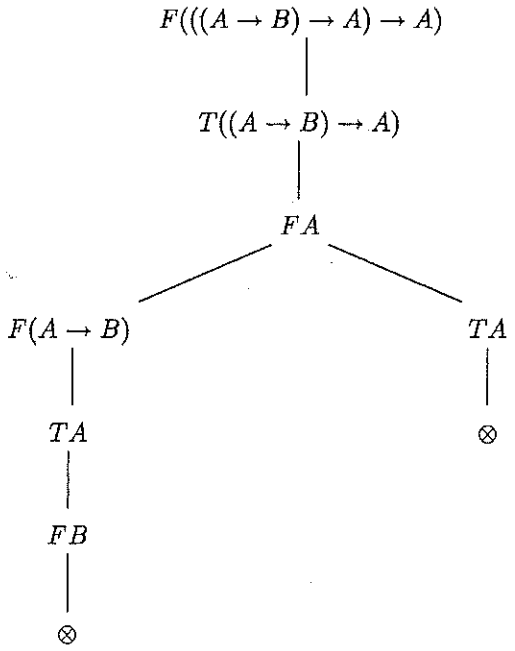

FIGURE 14.

In much of what follows, for every definition or theorem dealing with a tableau proof or a logical truth (or both), there is a dual definition or theorem dealing with a tableau refutation and a logical falsehood, respectively. It is left to the reader to provide these dual propositions.

The next step in producing proofs is to see that there is a finished tableau starting with any given signed proposition as root entry. We describe a simple systematic procedure for producing such a tableau. Theorem 4.11 shows that the tableaux produced by this procedure are always finite.

Definition 4.7 (Complete systematic tableaux): Let R be a signed proposition. We define the *complete systematic tableau* (CST) with root entry R by induction. We begin the construction by letting τ_0 be the unique atomic tableau with R

at its root. Assume that τ_m has been defined. Let n be the smallest level of τ_m containing an entry that is unreduced on some noncontradictory path in τ_m and let E be the leftmost such entry of level n. We now let τ_{m+1} be the tableau gotten by adjoining the unique atomic tableau with root E to the end of every noncontradictory path of τ_m on which E is unreduced. The union of the sequence τ_m is our desired complete systematic tableau.

Theorem 4.8: *Every* CST *is finished.*

Proof: Consider any entry E that occurs at some level n of the CST τ and lies on a noncontradictory path P in τ. There are at most finitely many entries on τ at or above level n. Thus, all the entries at level n or above on τ must be in place by some point of the construction. That is, there is an m_0 such that for every $m \geq m_0$, τ_m through level n is the same as τ through level n. Now, for $m \geq m_0$, the restriction of P to τ_m is a path in τ_m containing E. At each step $m \geq m_0$ in the construction of the CST we reduce the entry on the lexicographically least node labeled with an unreduced entry that is on some noncontradictory path in the tableau τ_m. If E is not already reduced on P by stage m_0, we can proceed for at most finitely many steps in this construction before E would become the lexicographically least unreduced entry. At this point in the construction we would reduce E. □

In allowing infinite tableaux, we seem to be in conflict with the intuition that proofs should be finite objects. However, by König's lemma, we can restrict our attention to finite contradictory tableaux.

Theorem 4.9: *If $\tau = \cup\tau_n$ is a contradictory tableau, then for some m, τ_m is a finite contradictory tableau. Thus, in particular, if a* CST *is a proof, it is a finite tableau.*

Proof: τ is a finitely branching tree. Consider the subset of all nodes of τ with no contradiction above them. If this set is infinite, it has an infinite path by König's lemma. As this contradicts the assumption that every path in τ is contradictory, there are only finitely many such nodes. They must all appear by some level n of τ. Thus, every node at level $n+1$ of τ has a contradiction above it. Once again, as τ through level $n+1$ is finite, there is an m such that τ_m is the same as τ through level $n+1$. Now every path P in τ_m is either a path in τ (ending with a leaf of level $\leq n$) or a path containing a node of level $n+1$. In the first case, P is contradictory by our assumption that τ is contradictory. In the second, P is contradictory by our choice of n and m. Thus, τ_m is the desired contradictory tableau.

Note that if $\tau = \cup\tau_n$ is as in the definition of a CST and m is least such that τ_m is contradictory, then we cannot extend τ_m in the construction of τ. In this case $\tau = \tau_m$. □

In the next section, we consider the import of this argument for the semantic as well as syntactic versions of the compactness theorem.

We conclude this section with a proof that each CST is actually finite. The proof involves an induction based on a measure of the complexity of propositions that we call the *degree of a proposition.*

Definition 4.10: We define $d(\alpha)$, the *degree of a proposition* α by induction.

(i) If α is a propositional letter, then $d(\alpha) = 0$.

(ii) If α is $\neg\beta$, then $d(\alpha) = d(\beta) + 1$.

(iii) If α is $\beta \vee \gamma$, $\beta \wedge \gamma$, $\beta \rightarrow \gamma$ or $\beta \leftrightarrow \gamma$, then $d(\alpha) = d(\beta) + d(\gamma) + 1$.

The degree of a signed proposition $T\alpha$ or $F\alpha$ is the degree of α. If P is a path in a tableau τ, then $d(P)$ the degree of P is the sum of the signed propositions on P that are not reduced on P.

Theorem 4.11: *Every* CST *is finite.*

Proof: Let $\tau = \cup\tau_m$ be any CST as described in Definition 4.7. We prove that every path on τ is finite (indeed has length at most the degree of the root of τ) and so, by König's lemma, τ itself is finite. Consider any path P on τ. It is the union of paths P_m on τ_m. A change occurs between P_m and P_{m+1} when, according to the Definition 4.7, we add the atomic tableau σ with root E to the end of P_m for some entry E that is unreduced on a path in τ_m. We claim that $d(P_{m+1}) < d(P_m)$. Of course, this immediately implies that we can add an atomic tableau to the end of P_m at most finitely often (indeed at most $d(\alpha)$ many times where α is the proposition at the root of τ). Thus, P is finite as desired. To verify the claim first note that adding σ to the end of our path reduces the entry E on P. This subtracts $d(E)$ from the degree of the path while adding on the degrees of the signed propositions other than E occurring on the path of σ that is added on to P_m to form P_{m+1}. Thus, it suffices to check that the sum of the degrees of the signed formulas (excluding the root) on each path through any atomic tableau σ is less than the degree of the root of σ. This is immediate from the definition of degree and the list of atomic tableaux in Figure 9. □

Exercises

Give tableau proofs of each of the propositions listed in (1) – (9) below.

1. Idempotence and Commutativity of $\wedge$, $\vee$

   (a) $(\alpha \vee \alpha) \leftrightarrow \alpha$

   (b) $(\alpha \wedge \alpha) \leftrightarrow \alpha$

(c) $(\alpha \wedge \beta) \leftrightarrow (\beta \wedge \alpha)$

(d) $(\alpha \vee \beta) \leftrightarrow (\beta \vee \alpha)$.

2. Associativity and Distributivity of $\wedge$, $\vee$

(a) $((\alpha \wedge \beta) \wedge \gamma) \leftrightarrow (\alpha \wedge (\beta \wedge \gamma))$

(b) $((\alpha \vee \beta) \vee \gamma) \leftrightarrow (\alpha \vee (\beta \vee \gamma))$

(c) $(\alpha \vee (\beta \wedge \gamma)) \leftrightarrow ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$

(d) $(\alpha \wedge (\beta \vee \gamma)) \leftrightarrow ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$.

3. Pure Implication Laws

(a) $\alpha \rightarrow \alpha$

(b) $\alpha \rightarrow (\beta \rightarrow \alpha)$

(c) $(\alpha \rightarrow \beta) \rightarrow ((\beta \rightarrow \gamma) \rightarrow (\alpha \rightarrow \gamma))$

(d) $(\alpha \rightarrow (\beta \rightarrow \gamma)) \rightarrow ((\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \gamma))$.

4. Introduction and Elimination of $\wedge$

(a) $(\alpha \rightarrow (\beta \rightarrow \gamma)) \rightarrow ((\alpha \wedge \beta) \rightarrow \gamma)$

(b) $(\alpha \wedge \beta) \rightarrow \gamma) \rightarrow ((\alpha \rightarrow (\beta \rightarrow \gamma))$.

5. De Morgan's Laws

(a) $\neg(\alpha \vee \beta) \leftrightarrow (\neg\alpha \wedge \neg\beta)$

(b) $\neg(\alpha \wedge \beta) \leftrightarrow (\neg\alpha \vee \neg\beta)$.

6. Contrapositive
$(\alpha \rightarrow \beta) \leftrightarrow (\neg\beta \rightarrow \neg\alpha)$.

7. Double Negation
$\alpha \leftrightarrow \neg\neg\alpha$.

8. Contradiction
$\neg(\alpha \wedge \neg\alpha)$.

9. (a) $(\neg\alpha \vee \beta) \rightarrow (\alpha \rightarrow \beta)$

(b) $(\alpha \rightarrow \beta) \rightarrow (\neg\alpha \vee \beta)$.

Conjunctive and disjunctive normal forms

Recall from Exercise 3.3 that a conjunctive normal form (CNF) for a proposition α is a conjunct of disjuncts of literals (propositional letters or their negations) which is equivalent to α. Similarly, a disjunctive normal form (DNF) for α is a disjunct of conjuncts of literals which is equivalent to α. For any proposition α, we can find equivalent conjunctive and disjunctive normal forms by the following procedure:

(i) Eliminate all uses of $\leftrightarrow$ in the formation (tree) of α by replacing any step going from β and γ to $\beta \leftrightarrow \gamma$ by one going to $(\beta \rightarrow \gamma) \wedge (\gamma \rightarrow \beta)$. This produces a proposition α_1 equivalent to α in which $\leftrightarrow$ does not occur.

(ii) Eliminate all uses of $\rightarrow$ in the formation of α_1 by replacing any step going to $\beta \rightarrow \gamma$ by one going to $\neg\beta \vee \gamma$. This produces an α_2 equivalent to α in which the only connectives are $\neg$, $\vee$ and $\wedge$.

(iii) Get a third equivalent proposition α_3 in which, in addition, $\neg$ appears only immediately before propositional letters by replacing in α_2 all occurrences of $\neg\neg\beta$ by β, of $\neg(\beta \vee \gamma)$ by $\neg\beta \wedge \neg\gamma$ and of $\neg(\beta \wedge \gamma)$ by $\neg\beta \vee \neg\gamma$.

(iv) Now use the associativity and distributivity laws above to get equivalents of α_3 which are either conjuncts of disjuncts of literals (CNF) or disjuncts of conjuncts of literals (DNF).

We provide an example of this procedure by finding both normal forms for the proposition $\alpha = (A \rightarrow B) \leftrightarrow \neg C$:

$(A \rightarrow B) \leftrightarrow \neg C$ (i)

$((A \rightarrow B) \rightarrow \neg C) \wedge (\neg C \rightarrow (A \rightarrow B))$ (ii)

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

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

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

We can now apply step (iv) to get a CNF for α:

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

We can also use distributivity to produce a DNF for α:

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

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

This last line is a DNF for α. It can, however, be simplified by using some of the other rules proved above and simple truth table considerations. In particular, contradictions such as $C \wedge \neg C$ can be eliminated from disjuncts and tautologies

such as $C \vee \neg C$ can be eliminated from conjuncts. Applying these procedures simplifies the DNF derived for α to the following:

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

10. Use the procedure described above to find CNF and DNF equivalents for the following propositions:

    (a) $(A \rightarrow B) \leftrightarrow (A \rightarrow C)$

    (b) $(A \leftrightarrow B) \rightarrow (C \vee D)$.

11. Use the laws provided in the above exercises to prove that each step of the above procedure produces a proposition equivalent to the original proposition α.

5 Soundness and Completeness of Tableau Proofs

We are going to prove the equivalence of the semantic notion of validity ($\models$) and the syntactic notion of provability ($\vdash$). Thus, we show that all tableau provable propositions are valid (soundness of the proof method) and that all valid propositions are tableau provable (completeness of the method).

Theorem 5.1 (Soundness): *If α is tableau provable, then α is valid, i.e.,* $\vdash \alpha \Rightarrow \models \alpha$.

Proof: We prove the contrapositive. Suppose α is not valid. By definition there is a valuation $\mathcal{V}$ assigning F to α. We say that the valuation $\mathcal{V}$ *agrees with* a signed proposition E in two situations: if E is $T\alpha$ and $\mathcal{V}(\alpha) = T$ or if E is $F\alpha$ and $\mathcal{V}(\alpha) = F$. We show (Lemma 5.2) that if any valuation $\mathcal{V}$ agrees with the root node of a tableau, then there is a path P in the tableau such that $\mathcal{V}$ agrees with every entry on P. As no valuation can agree with any path on a contradictory tableau there can be no tableau proof of α. □

If $\mathcal{V}$ is a valuation that agrees with the root entry of a given tableau τ given as in Definition 4.1 as $\cup\tau_n$, then τ has a path P every entry of which agrees with $\mathcal{V}$.

Lemma 5.2: *If $\mathcal{V}$ is a valuation that agrees with the root entry of a given tableau τ given as in Definition 4.1 as $\cup\tau_n$, then τ has a path P every entry of which agrees with $\mathcal{V}$.*

Proof: We prove by induction that there is a sequence $\langle P_n \rangle$ such that, for every n, P_n is contained in P_{n+1} and P_n is a path through τ_n such that $\mathcal{V}$ agrees with every entry on P_n. The desired path P through τ will then simply be the union of the P_n. The base case of the induction is easily seen to be true by the assumption

that $\mathcal{V}$ agrees with the root of τ. As an example, consider (6a) with root entry $T(\alpha \leftrightarrow \beta)$. If $\mathcal{V}(\alpha \leftrightarrow \beta) = T$, then either $\mathcal{V}(\alpha) = T$ and $\mathcal{V}(\beta) = T$ or $\mathcal{V}(\alpha) = F$ and $\mathcal{V}(\beta) = F$ by the truth table definition for $\leftrightarrow$. We leave the verifications for the other atomic tableaux as Exercise 1.

For the induction step, suppose that we have constructed a path P_n in τ_n every entry of which agrees with $\mathcal{V}$. If τ_{n+1} is gotten from τ_n without extending P_n, then we let $P_{n+1} = P_n$. If P_n is extended in τ_{n+1}, then it is extended by adding on to its end an atomic tableau with root E for some entry E appearing on P_n. As we know by induction that $\mathcal{V}$ agrees with E, the same analysis as used in the base case shows that $\mathcal{V}$ agrees with one of the extensions of P_n to a path P_{n+1} in τ_{n+1}. □

Theorem 5.3 (Completeness): *If α is valid, then α is tableau provable, i.e., $\vDash \alpha \Rightarrow \vdash \alpha$. In fact, any finished tableau with root entry $F\alpha$ is a proof of α and so, in particular, the complete systematic tableaux with root $F\alpha$ is such a proof.*

The crucial idea in the proof of the completeness theorem is embodied in Lemma 5.4: We can always define a valuation that agrees with all entries on any noncontradictory path of any finished tableau.

Lemma 5.4: *Let P be a noncontradictory path of a finished tableau τ. Define a truth assignment $\mathcal{A}$ on all propositional letters A as follows:*

$\mathcal{A}(A) = T$ if TA is an entry on P.

$\mathcal{A}(A) = F$ otherwise.

If $\mathcal{V}$ is the unique valuation (Theorem 3.3) extending the truth assignment $\mathcal{A}$, then $\mathcal{V}$ agrees with all entries of P.

Proof: We proceed by induction on the depth of propositions on P.

(i) If α is a propositional letter and $T\alpha$ occurs on P, then $\mathcal{V}(\alpha) = T$ by definition and we are done. If $F\alpha$ occurs on P, then, as P is noncontradictory, $T\alpha$ does not and $\mathcal{V}(\alpha) = F$.

(ii) Suppose $T(\alpha \wedge \beta)$ occurs on the noncontradictory path P. Since τ is a finished tableau, both $T(\alpha)$ and $T(\beta)$ occur on P. By the induction hypothesis $\mathcal{V}(\alpha) = T = \mathcal{V}(\beta)$ and so $\mathcal{V}(\alpha \wedge \beta) = T$ as required.

(iii) Suppose $F(\alpha \wedge \beta)$ occurs on the noncontradictory path P. Again by the definition of a finished tableau, either $F\alpha$ or $F\beta$ must occur on P. Whichever it is, the induction hypothesis tells us that it agrees with $\mathcal{V}$ and so either $\mathcal{V}(\alpha) = F$ or $\mathcal{V}(\beta) = F$. In either case $\mathcal{V}(\alpha \wedge \beta) = F$ as required.

The remaining connectives are treated like one of these two cases depending on whether or not the corresponding atomic tableau branches. The details are left as Exercise 2. □

Proof (of Theorem 5.3): Suppose that α is valid and so $\mathcal{V}(\alpha) = T$ for every valuation $\mathcal{V}$. Consider any finished tableau τ with root $F\alpha$. (The CST with root $F\alpha$ is one by Theorem 4.8.) If τ had a noncontradictory path P, there would be, by Lemma 5.2, a valuation $\mathcal{V}$ that agrees with all its entries and so in particular with $F\alpha$. This would give us a valuation with $\mathcal{V}(\alpha) = F$ contradicting the validity of α. Thus, every path on τ is contradictory and τ is a tableau proof of α. □

It is clear from the proof of the completeness theorem (in fact from Lemma 5.4) that if you try to construct a tableau proof for α (i.e., one starting with $F\alpha$) and you do your best by constructing a finished tableau with root $F\alpha$ but fail to produce a proof of α (i.e., the finished tableau has at least one noncontradictory path), then the valuation defined by this noncontradictory path as in Lemma 5.4 gives us a counterexample to the assertion that α is valid. As we can always produce a finished tableau with any given root, we must, for every proposition, be able to get either a tableau proof or a counterexample to its validity!

It is this dichotomy (albeit expressed at the level of more complicated fragments of predicate logic) that forms the basis for constructive solutions to many problems. It is also the underlying rationale of PROLOG and of the implementation of other constructive theorem provers as programming languages. One starts with an assumption such as "there is no x such that $\mathcal{P}(x)$" and one either proves it true or finds a counterexample, that is, one actually produces an x such that $\mathcal{P}(x)$. We consider these matters in II.5 and, in more detail, in Chapter III.

Exercises

1. Verify the remaining cases of atomic tableaux in Lemma 5.2.
2. Verify the cases for the remaining connectives in Lemma 5.4.

Reformulate and prove the analogs of the results of this section for tableau refutations and satisfiability:

3. If α is tableau refutable, i.e., there is a contradictory tableau with root $T\alpha$, then α is *unsatisfiable*, i.e., there is no valuation $\mathcal{V}$ such that $\mathcal{V}(\alpha) = T$.
4. If α is unsatisfiable, then there is a tableau refutation of α.

6 Deductions from Premises and Compactness

Recall the treatment at the end of §3 of the consequences of a set Σ of propositions (which we called premises). A proposition σ is a consequence of Σ ($\Sigma \models \sigma$) if every valuation that is a model of Σ is also one of σ, i.e., every valuation that

makes all the elements of Σ true also makes σ true. (See Definitions 3.2 and 3.8.) This notion of consequence and the associated one of a proof from given premises (which we are about to define) reflect common usage in mathematical arguments. A theorem is typically stated as an implication of the form $\alpha \to \beta$. The proof of the theorem, however, is generally presented in a format that begins by assuming that the hypotheses (α) are true and then argues that the conclusion (β) must be true. Viewed syntactically in terms of proofs, we might describe this procedure as "assuming" α and then "deducing" β. The semantic notion of consequence captures the first view of such an argument. We now want to capture the syntactic or proof–theoretic version by defining what it means to prove a proposition from a set of premises. Once we have developed the appropriate notions, a formal version of the informal mathematical method of argument described above is (easily) provided by the deduction theorem (Exercise 6). We now turn to the abstract formulation of the notions needed to express this result.

We begin our analysis with the definition of tableaux with premises from a set of sentences. It differs from the basic definition only in that we are allowed to add on entries of the form $T\alpha$ for premises α. This variation reflects the intuition that working from a set of premises means that we are assuming them to be true.

Definition 6.1 (Tableaux from premises): Let Σ be a (possibly infinite) set of propositions. We define the *finite tableaux with premises from* Σ (or just *from* Σ for short) by induction:

(i) Every atomic tableau is a finite tableau from Σ.

(ii) If τ is a finite tableau from Σ and $\alpha \in \Sigma$, then the tableau formed by putting $T\alpha$ at the end of every noncontradictory path not containing it is also a finite tableau from Σ.

(iii) If τ is a finite tableau from Σ, P a path in τ, E an entry of τ occurring on P and τ' is obtained from τ by adjoining the unique atomic tableau with root entry E to the end of the path P, then τ' is also a finite tableau from Σ.

If $\tau_0, \tau_1, \ldots, \tau_n, \ldots$ is a (finite or infinite) sequence of finite tableaux from Σ such that, for each $n \geq 0$, τ_{n+1} is constructed from τ_n by an application of (ii) or (iii), then $\tau = \cup\tau_n$ is a *tableau from* Σ.

We can now define tableau proofs as before.

Definition 6.2: A *tableau proof of a proposition* α *from* Σ (or *with premises from* Σ) is a tableau from Σ with root entry $F\alpha$ that is contradictory, that is, one in which every path is contradictory. If there is such a proof we say that α *is provable from* Σ and write it as $\Sigma \vdash \alpha$.

Example 6.3: Figure 15 gives a tableau proof of A from a set consisting of two premises, $\{\neg B, (A \vee B)\}$.

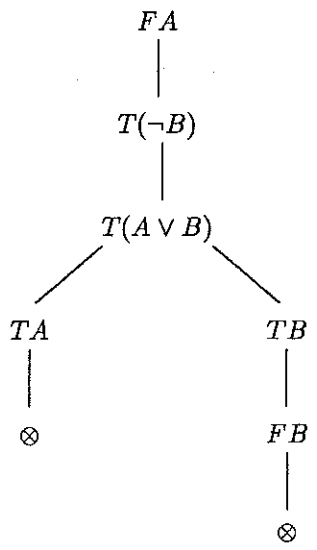

FIGURE 15.

We can now mimic the development of the last section to prove the soundness and completeness theorems for deductions from premises. The only changes are in the definition of a finished tableau and the CST. A *finished tableau from* Σ is a tableau from Σ that is a finished tableau in the sense of Definition 4.3 and has an occurrence of $T\alpha$ on every noncontradictory path for every $\alpha \in \Sigma$. The idea here is again that we are incorporating the truth of the premises into the analysis.

Similarly we must take steps in the construction of the CST *from* Σ to guarantee the appearance of these premises. We list the elements of Σ as α_m, $m \in \mathcal{N}$, and revise the definition of the CST by simply adding on one step to the definition of τ_{m+1}. If our new construction has produced τ_m we let τ'_{m+1} be the next tableau that would be defined by the standard CST procedure. (If that procedure would now terminate because every path is contradictory, we also terminate the current construction.) We now add on $T\alpha_m$ to the end of every noncontradictory path in τ'_{m+1} that does not already contain $T\alpha$ to form our new τ_{m+1}.

Theorem 6.4: *Every* CST *from a set of premises is finished.*

Proof: Exercise 1. □

The proofs of the soundness and completeness theorems can now be carried out as before with the caveat that we must always see to it that the propositions

in Σ are true in the relevant valuations. We state the appropriate lemmas and theorems and leave most of their proofs in this setting as exercises.

Lemma 6.5: *If a valuation $\mathcal{V}$ makes every $\alpha \in \Sigma$ true and agrees with the root of a tableau τ from Σ, then there is a path in τ every entry of which agrees with $\mathcal{V}$.*

Proof: Exercise 2. □

Theorem 6.6 (Soundness of deductions from premises): *If there is a tableau proof of α from a set of premises Σ, then α is a consequence of Σ, i.e., $\Sigma \vdash \alpha \Rightarrow \Sigma \models \alpha$.*

Proof: If not, there is a valuation that makes β true for every $\beta \in \Sigma$ but makes α false. Continue now as in the proof of Theorem 5.1. □

Lemma 6.7: *Let P be a noncontradictory path in a finished tableau τ from Σ. Define a valuation $\mathcal{V}$ as in Lemma 5.4. $\mathcal{V}$ then agrees with all entries on P and so in particular makes every proposition $\beta \in \Sigma$ true (as $T\beta$ must appear on P for every $\beta \in \Sigma$ by definition of a finished tableau from Σ).*

Proof: Exercise 3. □

Theorem 6.8 (Completeness of deductions from premises): *If α is a consequence of a set Σ of premises, then there is a tableau deduction of α from Σ, i.e., $\Sigma \models \alpha \Rightarrow \Sigma \vdash \alpha$.*

Proof: If $\Sigma \models \alpha$, every valuation $\mathcal{V}$ that makes every proposition in Σ true also makes α true. Consider the CST from Σ with root $F\alpha$. It is finished by Theorem 6.4. Now apply Lemma 6.7. □

Again we consider the problem of finiteness of proofs. If Σ is finite, then an argument similar to that of Theorem 4.11 shows that the CST from Σ itself is also finite. (See Exercise 12.) For infinite sets of premises, the argument for Theorem 4.9 using König's lemma works just as before.

Theorem 6.9: *If $\tau = \cup\tau_n$ is a contradictory tableau from Σ, then, for some m, τ_m is a finite contradictory tableau from Σ. In particular, if a CST from Σ is a proof, it is finite.*

Proof: Exercise 4. □

Thus, we know that if α is provable from Σ, then there is a finite tableau proof of it. This can be viewed as a syntactic version of the compactness theorem. Using the completeness and soundness theorems it can be converted into a semantic one.

Theorem 6.10 (Compactness): *α is a consequence of Σ iff α is a consequence of some finite subset of Σ.*

Proof: Exercise 5. □

We have left the indirect proof, via completeness and soundness, of the semantic version of the compactness theorem as an exercise. However, a direct proof of this result is also available. The compactness theorem is genuinely deeper than the others we have proven and deserves two proofs. An advantage of the direct approach is that the completeness theorem can be proved from the compactness theorem and without recourse to infinite tableaux. The direct approach also shows that compactness is simply a consequence of König's lemma.

Definition 6.11: A set Σ of propositions is called *satisfiable* if it has a model, i.e., there is a valuation $\mathcal{V}$ such that $\mathcal{V}(\alpha) = T$ for every $\alpha \in \Sigma$. We also say that such a valuation *satisfies* Σ.

Example 6.12:

(i) $\{A_1,\ A_2,\ (A_1 \wedge A_2),\ A_3,\ (A_1 \wedge A_3),\ A_4,\ (A_1 \wedge A_4),\ \ldots\}$ is a satisfiable infinite set of propositions.

(ii) $\{A_1,\ A_2,\ (A_1 \rightarrow A_3),\ (\neg A_3)\}$ is a finite set of propositions that is not satisfiable nor is any set containing it.

Theorem 6.13 (Compactness): *Let $\Sigma = \{\alpha_i |\ i \in \omega\}$ be an infinite set of propositions. Σ is satisfiable if and only if every finite subset Γ of Σ is satisfiable.*

Proof: Note that the "only if" direction of the theorem is trivially true; the other direction is not (not *trivially* that is). The problem is that finding different valuations which satisfy longer and longer initial segments does not necessarily mean that there is a single valuation satisfying the whole sequence. Building such a valuation is essentially an application of König's lemma.

Let $\langle C_i |\ i \in \omega \rangle$ be a list of all the propositional letters. We define a tree T whose nodes are binary sequences ordered by extension. We use $lth(\sigma)$ to denote the length of a sequence σ and set $T = \{\sigma |$ there is a valuation $\mathcal{V}$ such that, for $i \leq lth(\sigma)$, $\mathcal{V}(\alpha_i) = T$ and $\mathcal{V}(C_i) = T$ iff $\sigma(i) = 1\}$. What this definition says is that we put σ on the tree unless interpreting it as an assignment of truth values to the propositional letters C_i $(i \leq lth(\sigma))$ already forces one of the α_i to be false for $i \leq lth(\sigma)$.

Claim: There is an infinite path in T if and only if Σ is satisfiable.

Proof of Claim: If $\mathcal{V}$ satisfies Σ, then, by definition, the set of all σ such that $\sigma(i) = 1$ iff $\mathcal{V}(C_i) = T$ is a path on T. On the other hand, suppose that

$\langle \sigma_j |\ j \in \mathcal{N} \rangle$ is an infinite path on T. Let $\mathcal{V}$ be the unique valuation extending the assignment determined by the σ_j, i.e., the one for which C_i is true iff $\sigma_j(i) = 1$ for some j (or equivalently, as the σ_i are linearly ordered by extension, iff $\sigma_j(i) = 1$ for every i such that $i \leq lth(\sigma_j)$). If $\mathcal{V} \not\models \Sigma$, then there is some $\alpha_j \in \Sigma$ such that $\mathcal{V}(\alpha_j) = F$. Now by Corollary 3.4 this last fact depends on the truth values assigned by $\mathcal{V}$ to only finitely many propositional letters. Let us suppose it depends only on those C_i with $i \leq n$. It is then clear from the definition of T that no σ with length $\geq n$ can be on T at all. As there are only finitely many binary sequences σ with length $\leq n$, we have contradicted the assumption that the sequence $\langle \sigma_j \rangle$ is an infinite path on T and so $\mathcal{V} \models \Sigma$ as claimed.

The next claim is that there is, for every n, a σ of length n in T. By assumption every finite subset of Σ is satisfiable. Thus, for each n, there is a valuation $\mathcal{V}_n$ that makes α_i true for each $i \leq n$. The string σ given by $\sigma(i) = 1$ iff $\mathcal{V}_n(C_i) = T$ for $i \leq n$ is then on T by definition.

König's lemma (Theorem 1.4) now tells us that there is an infinite path in T and so Σ is satisfiable as required. □

The connection between this version of the compactness theorem for propositional logic and the compactness theorem of topology is considered in Exercises 9 and 10. Other applications of the compactness theorem can be found in Exercises 7 and 8.

Exercises

1. Prove Theorem 6.4.
2. Prove Lemma 6.5.
3. Follow the proof of Lemma 5.4 to prove Lemma 6.7.
4. Follow the proof of Theorem 4.9 to prove Theorem 6.9.
5. Deduce Theorem 6.10 from the results preceding it.
6. **Deduction Theorem**: Let Σ be a finite set of propositions and $\bigwedge \Sigma$ the conjunction of its members. Prove that for any proposition α the following are equivalent:
   (a) $\Sigma \models \alpha$.
   (b) $\models \bigwedge \Sigma \rightarrow \alpha$.
   (c) $\Sigma \vdash \alpha$.
   (d) $\vdash \bigwedge \Sigma \rightarrow \alpha$.

Applications of Compactness

For Problems 7 and 8, use the compactness theorem for propositional logic or König's lemma. The key point in each case is to faithfully translate the given problem into an appropriate set of propositions (or an appropriate tree). One then applies compactness or König's lemma. Finally, one must translate the result of this application back into the terms of the problem. These problems are treated in predicate logic in Exercises II.7.5.

7. A partial order has *width at most* n if every set of pairwise incomparable elements has size at most n. A chain in a partial order $<$ is simply a subset of the order that is linearly ordered by $<$. Prove that an infinite partial order of width at most 3 can be divided into three chains (not necessarily disjoint) if every finite partial suborder of width at most 3 can be so divided.
   Hint (Using compactness): Let the elements of the order be $\{p_n | \ n \in \mathcal{N}\}$. Consider propositions Rp_ip_j, Ap_i, Bp_i and Cp_i for $i, j \in \mathcal{N}$. Think of Rp_ip_j as saying that $p_i < p_j$. Think of Ap_i as saying that p_i is in chain A and similarly for Bp_i and Cp_i. Now write down the *sets* of propositions expressing the desired conclusions: Each of A, B and C is a chain; every element is in A, B or C; the order has width 3.
   Note: Dilworth's theorem states that any partial order of width at most n can be divided into n chains. Thus, Dilworth's theorem for infinite orders follows from the theorem for finite orders by compactness. As the finite case is proved by induction on the size of the given order, this is a nontrivial application of compactness.

8. A *graph* G is a set of elements $\{a_0, a_1, \ldots\}$ called nodes and a set of pairs of nodes $\{a_i, a_j\}$ called edges. We say that G is *n–colorable* if we can label its nodes with n colors $C_1, \ldots, C_n$ so that no two nodes in a single edge of G have the same color. Suppose every finite subgraph of G (a finite subset of the nodes and the edges between them) is four–colorable. Prove that G is four–colorable.
   Hint (Using König's lemma): Define a tree of 4–ary sequences ordered by extension. Put a sequence σ of length $n+1$ on the tree if and only if it defines a four–coloring of the nodes $a_0, a_1, \ldots, a_n$ by coloring a_j with color $C_{\sigma(j)}$.
   Note: The four–color theorem says that every planar graph is four–colorable. By this exercise, it suffices to prove the theorem for finite graphs, as a graph is planar if and only if all its finite subgraphs are planar.

Connections with topological compactness and König's lemma

The compactness theorem for propositional logic can be connected to the topology on the set $\mathcal{T}$ of all possible truth valuations that is determined by letting the open sets be generated by those of the form $\{\mathcal{V} : (\exists \alpha \in \Sigma)\ (\mathcal{V} \not\models \alpha)\}$ for any set Σ of propositions.