2.5.3 Logical Consequence

Definition 2.48 Let U be a set of formulas and A a formula. A is a *logical consequence* of U, denoted $U \models A$, iff every model of U is a model of A. ∎

The formula A need not be true in every possible interpretation, only in those interpretations which satisfy U, that is, those interpretations which satisfy every formula in U. If U is empty, logical consequence is the same as validity.

Example 2.49 Let $A = (p \vee r) \wedge (\neg q \vee \neg r)$. Then A is a logical consequence of $\{p, \neg q\}$, denoted $\{p, \neg q\} \models A$, since A is true in all interpretations $\mathscr{I}$ such that $\mathscr{I}(p) = T$ and $\mathscr{I}(q) = F$. However, A is not valid, since it is not true in the interpretation $\mathscr{I}'$ where $\mathscr{I}'(p) = F$, $\mathscr{I}'(q) = T$, $\mathscr{I}'(r) = T$. ∎

The caveat concerning $\leftrightarrow$ and $\equiv$ also applies to $\rightarrow$ and $\models$. Implication, $\rightarrow$, is an operator in the object language, while $\models$ is a symbol for a concept in the metalanguage. However, as with equivalence, the two concepts are related:

Theorem 2.50 *$U \models A$ if and only if $\models \bigwedge_i A_i \rightarrow A$.*

Definition 2.51 $\bigwedge_{i=1}^{i=n} A_i$ is an abbreviation for $A_1 \wedge \cdots \wedge A_n$. The notation $\bigwedge_i$ is used if the bounds are obvious from the context or if the set of formulas is infinite. A similar notation $\bigvee$ is used for disjunction. ∎

Example 2.52 From Example 2.49, $\{p, \neg q\} \models (p \vee r) \wedge (\neg q \vee \neg r)$, so by Theorem 2.50, $\models (p \wedge \neg q) \rightarrow (p \vee r) \wedge (\neg q \vee \neg r)$. ∎

The proof of Theorem 2.50, as well as the proofs of the following two theorems are left as exercises.

Theorem 2.53 *If $U \models A$ then $U \cup \{B\} \models A$ for any formula B.*

Theorem 2.54 *If $U \models A$ and B is valid then $U - \{B\} \models A$.*

2.5.4 Theories *

Logical consequence is the central concept in the foundations of mathematics. Valid logical formulas such as $p \vee q \leftrightarrow q \vee p$ are of little mathematical interest. It is much more interesting to assume that a set of formulas is true and then to investigate the consequences of these assumptions. For example, Euclid assumed five formulas about geometry and deduced an extensive set of logical consequences. The formal definition of a mathematical theory is as follows.

Definition 2.55 Let $\mathscr{T}$ be a set of formulas. $\mathscr{T}$ is *closed under logical consequence* iff for all formulas A, if $\mathscr{T} \models A$ then $A \in \mathscr{T}$. A set of formulas that is closed under logical consequence is a *theory*. The elements of $\mathscr{T}$ are *theorems*. ∎

Theories are constructed by selecting a set of formulas called *axioms* and deducing their logical consequences.

Definition 2.56 Let $\mathscr{T}$ be a theory. $\mathscr{T}$ is said to be *axiomatizable* iff there exists a set of formulas U such that $\mathscr{T} = \{A \mid U \models A\}$. The set of formulas U are the *axioms* of $\mathscr{T}$. If U is finite, $\mathscr{T}$ is said to be *finitely axiomatizable*. ∎

Arithmetic is axiomatizable: There is a set of axioms developed by Peano whose logical consequences are theorems of arithmetic. Arithmetic is not finitely axiomatizable, because the induction axiom is not by a single axiom but an axiom scheme with an instance for each property in arithmetic.

2.6 Semantic Tableaux

The method of *semantic tableaux* is an efficient decision procedure for satisfiability (and by duality validity) in propositional logic. We will use semantic tableaux extensively in the next chapter to prove important theorems about deductive systems. The principle behind semantic tableaux is very simple: search for a model (satisfying interpretation) by decomposing the formula into sets of atoms and negations of atoms. It is easy to check if there is an interpretation for each set: a set of atoms and negations of atoms is satisfiable iff the set does not contain an atom p and its negation $\neg p$. The formula is satisfiable iff one of these sets is satisfiable.

We begin with some definitions and then analyze the satisfiability of two formulas to motivate the construction of semantic tableaux.

2.6.1 Decomposing Formulas into Sets of Literals

Definition 2.57 A *literal* is an atom or the negation of an atom. An atom is a *positive literal* and the negation of an atom is a *negative literal*. For any atom p, $\{p, \neg p\}$ is a *complementary pair of literals*.

For any formula A, $\{A, \neg A\}$ is a *complementary pair of formulas*. A is the *complement* of $\neg A$ and $\neg A$ is the *complement* of A. ∎

Example 2.58 In the set of literals $\{\neg p, q, r, \neg r\}$, q and r are positive literals, while $\neg p$ and $\neg r$ are negative literals. The set contains the complementary pair of literals $\{r, \neg r\}$. ∎

Example 2.59 Let us analyze the satisfiability of the formula:

$$A = p \wedge (\neg q \vee \neg p)$$

in an arbitrary interpretation $\mathscr{I}$, using the inductive rules for the evaluation of the truth value of a formula.

- The principal operator of A is conjunction, so $v_{\mathscr{I}}(A) = T$ if and only if both $v_{\mathscr{I}}(p) = T$ and $v_{\mathscr{I}}(\neg q \vee \neg p) = T$.
- The principal operator of $\neg q \vee \neg p$ is disjunction, so $v_{\mathscr{I}}(\neg q \vee \neg p) = T$ if and only if either $v_{\mathscr{I}}(\neg q) = T$ or $v_{\mathscr{I}}(\neg p) = T$.
- Integrating the information we have obtained from this analysis, we conclude that $v_{\mathscr{I}}(A) = T$ if and only if either:

1. $v_{\mathscr{I}}(p) = T$ and $v_{\mathscr{I}}(\neg q) = T$, or
2. $v_{\mathscr{I}}(p) = T$ and $v_{\mathscr{I}}(\neg p) = T$.

A is satisfiable if and only if there is an interpretation such that (1) holds or an interpretation such that (2) holds. ∎

We have reduced the question of the satisfiability of A to a question about the satisfiability of sets of literals.

Theorem 2.60 *A set of literals is satisfiable if and only if it does* not *contain a complementary pair of literals.*

Proof Let L be a set of literals that does not contain a complementary pair. Define the interpretation $\mathscr{I}$ by:

$$\mathscr{I}(p) = T \quad \text{if } p \in L,$$
$$\mathscr{I}(p) = F \quad \text{if } \neg p \in L.$$

The interpretation is well-defined—there is only one value assigned to each atom in L—since there is no complementary pair of literals in L. Each literal in L evaluates to T so L is satisfiable.

Conversely, if $\{p, \neg p\} \subseteq L$, then for any interpretation $\mathscr{I}$ for the atoms in L, either $v_{\mathscr{I}}(p) = F$ or $v_{\mathscr{I}}(\neg p) = F$, so L is not satisfiable. ∎

Example 2.61 Continuing the analysis of the formula $A = p \wedge (\neg q \vee \neg p)$ from Example 2.59, A is satisfiable if and only at least one of the sets $\{p, \neg p\}$ and $\{p, \neg q\}$ does not contain a complementary pair of literals. Clearly, only the second set does not contain a complementary pair of literals. Using the method described in Theorem 2.60, we obtain the interpretation:

$$\mathscr{I}(p) = T, \qquad \mathscr{I}(q) = F.$$

We leave it to the reader to check that for this interpretation, $v_{\mathscr{I}}(A) = T$. ∎

The following example shows what happens if a formula is unsatisfiable.

Example 2.62 Consider the formula:

$$B = (p \vee q) \wedge (\neg p \wedge \neg q).$$

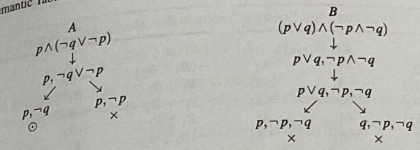


Fig. 2.7 Semantic tableaux

The analysis of the formula proceeds as follows:

- $v_{\mathscr{I}}(B) = T$ if and only if $v_{\mathscr{I}}(p \vee q) = T$ and $v_{\mathscr{I}}(\neg p \wedge \neg q) = T$.
- Decomposing the conjunction, $v_{\mathscr{I}}(B)$=T if and only if $v_{\mathscr{I}}(p \vee q) = T$ and $v_{\mathscr{I}}(\neg p) = v_{\mathscr{I}}(\neg q) = T$.
- Decomposing the disjunction, $v_{\mathscr{I}}(B) = T$ if and only if either:

1. $v_{\mathscr{I}}(p) = v_{\mathscr{I}}(\neg p) = v_{\mathscr{I}}(\neg q) = T$, or
2. $v_{\mathscr{I}}(q) = v_{\mathscr{I}}(\neg p) = v_{\mathscr{I}}(\neg q) = T$.

Both sets of literals $\{p, \neg p, \neg q\}$ and $\{q, \neg p, \neg q\}$ contain complementary pairs, so by Theorem 2.60, both set of literals are unsatisfiable. We conclude that it is impossible to find a model for B; in other words, B is unsatisfiable. ∎

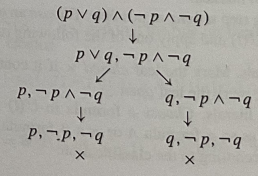
2.6.2 Construction of Semantic Tableaux

The decomposition of a formula into sets of literals is rather difficult to follow when expressed textually, as we did in Examples 2.59 and 2.62. In the method of semantic tableaux, sets of formulas label nodes of a tree, where each path in the tree represents the formulas that must be satisfied in one possible interpretation.

The initial formula labels the root of the tree; each node has one or two child nodes depending on how a formula labeling the node is decomposed. The leaves are labeled by the sets of literals. A leaf labeled by a set of literals containing a complementary pair of literals is marked ×, while a leaf labeled by a set not containing a complementary pair is marked ⊙.

Figure 2.7 shows semantic tableaux for the formulas from the examples.

The tableau construction is not unique; here is another tableau for B:



α	α_1	α_2
$\neg\neg A_1$	A_1	
$A_1 \wedge A_2$	A_1	A_2
$\neg(A_1 \vee A_2)$	$\neg A_1$	$\neg A_2$
$\neg(A_1 \rightarrow A_2)$	A_1	$\neg A_2$
$\neg(A_1 \uparrow A_2)$	A_1	A_2
$A_1 \downarrow A_2$	$\neg A_1$	$\neg A_2$
$A_1 \leftrightarrow A_2$	$A_1 \rightarrow A_2$	$A_2 \rightarrow A_1$
$\neg(A_1 \oplus A_2)$	$A_1 \rightarrow A_2$	$A_2 \rightarrow A_1$

β	β_1	β_2
$\neg(B_1 \wedge B_2)$	$\neg B_1$	
$B_1 \vee B_2$	B_1	B_2
$B_1 \rightarrow B_2$	$\neg B_1$	B_2
$B_1 \uparrow B_2$	$\neg B_1$	B_2
$\neg(B_1 \downarrow B_2)$	B_1	$\neg B_2$
$\neg(B_1 \leftrightarrow B_2)$	$\neg(B_1 \rightarrow B_2)$	B_2
$B_1 \oplus B_2$	$\neg(B_1 \rightarrow B_2)$	$\neg(B_2 \rightarrow B_1)$
		$\neg(B_2 \rightarrow B_1)$

Fig. 2.8 Classification of α- and β-formulas

It is constructed by branching to search for a satisfying interpretation for $p \vee q$ before searching for one for $\neg p \wedge \neg q$. The first tableau contains fewer nodes, showing that it is preferable to decompose conjunctions before disjunctions.

A concise presentation of the rules for creating a semantic tableau can be given if formulas are classified according to their principal operator (Fig. 2.8). If the formula is a negation, the classification takes into account both the negation and the principal operator. α-formulas are conjunctive and are satisfiable only if both subformulas α_1 and α_2 are satisfied, while β-formulas are disjunctive and are satisfied even if only one of the subformulas β_1 or β_2 is satisfiable.

Example 2.63 The formula $p \wedge q$ is classified as an α-formula because it is true if and only if both p and q are true. The formula $\neg(p \wedge q)$ is classified as a β-formula. It is logically equivalent to $\neg p \vee \neg q$ and is true if and only if either $\neg p$ is true or $\neg q$ is true. ∎

We now give the algorithm for the construction of a semantic tableau for a formula in propositional logic.

Algorithm 2.64 (Construction of a semantic tableau)
Input: A formula ϕ of propositional logic.
Output: A semantic tableau $\mathcal{T}$ for ϕ all of whose leaves are marked.

Initially, $\mathcal{T}$ is a tree consisting of a single root node labeled with the singleton set $\{\phi\}$. This node is not marked.

Repeat the following step as long as possible: *Choose* an unmarked leaf l labeled with a set of formulas $U(l)$ and apply one of the following rules.

- $U(l)$ is a set of literals. Mark the leaf *closed* $\times$ if it contains a complementary pair of literals. If not, mark the leaf *open* $\odot$.
- $U(l)$ is not a set of literals. *Choose* a formula in $U(l)$ which is not a literal. Classify the formula as an α-formula A or as a β-formula B and perform one of the following steps according to the classification:

– A is an α-formula. Create a new node l' as a child of l and label l' with:

$$U(l') = (U(l) - \{A\}) \cup \{A_1, A_2\}.$$

(In the case that A is $\neg\neg A_1$, there is no A_2.)
– B is a β-formula. Create two new nodes l' and l'' as children of l. Label l' with:

$$U(l') = (U(l) - \{B\}) \cup \{B_1\},$$

and label l'' with:

$$U(l'') = (U(l) - \{B\}) \cup \{B_2\}.$$

∎

Definition 2.65 A tableau whose construction has terminated is a *completed tableau*. A completed tableau is *closed* if all its leaves are marked closed. Otherwise (if some leaf is marked open), it is *open*. ∎

2.6.3 Termination of the Tableau Construction

Since each step of the algorithm decomposes one formula into one or two simpler formulas, it is clear that the construction of the tableau for any formula terminates, but it is worth proving this claim.

Theorem 2.66 *The construction of a tableau for any formula ϕ terminates. When the construction terminates, all the leaves are marked $\times$ or $\odot$.*

Proof Let us assume that $\leftrightarrow$ and $\oplus$ do not occur in the formula ϕ; the extension of the proof for these cases is left as an exercise.

Consider an unmarked leaf l that is chosen to be expanded during the construction of the tableau. Let $b(l)$ be the total number of binary operators in all formulas in $U(l)$ and let $n(l)$ be the total number of negations in $U(l)$. Define:

$$W(l) = 3 \cdot b(l) + n(l).$$

For example, if $U(l) = \{p \vee q, \neg p \wedge \neg q\}$, then $W(l) = 3 \cdot 2 + 2 = 8$.

Each step of the algorithm adds either a new node l' or a pair of new nodes l', l'' as children of l. We claim that $W(l') < W(l)$ and, if there is a second node, $W(l'') < W(l)$.

Suppose that $A = \neg(A_1 \vee A_2)$ and that the rule for this α-formula is applied at l to obtain a new leaf l' labeled:

$$U(l') = (U(l) - \{\neg(A_1 \vee A_2)\}) \cup \{\neg A_1, \neg A_2\}.$$

Then:

$$W(l') = W(l) - (3 \cdot 1 + 1) + 2 = W(l) - 2 < W(l),$$

because one binary operator and one negation are removed, while two negations are added.

Suppose now that $B = B_1 \vee B_2$ and that the rule for this β-formula is applied at l to obtain two new leaves l', l'' labeled:

$$\begin{aligned} U(l') &= (U(l) - \{B_1 \vee B_2\}) \cup \{B_1\}, \\ U(l'') &= (U(l) - \{B_1 \vee B_2\}) \cup \{B_2\}. \end{aligned}$$

Then:

$$W(l') \leq W(l) - (3 \cdot 1) < W(l), \qquad W(l'') \leq W(l) - (3 \cdot 1) < W(l).$$

We leave it to the reader to prove that $W(l)$ decreases for the other α- and β-formulas.

The value of $W(l)$ decreases as each branch in the tableau is extended. Since, obviously, $W(l) \geq 0$, no branch can be extended indefinitely and the construction of the tableau must eventually terminate.

A branch can always be extended if its leaf is labeled with a set of formulas that is not a set of literals. Therefore, when the construction of the tableau terminates, all leaves are labeled with sets of literals and each is marked open or closed by the first rule of the algorithm. ∎

2.6.4 *Improving the Efficiency of the Algorithm* *

The algorithm for constructing a tableau is not deterministic: at most steps, there is a choice of which leaf to extend and if the leaf contains more than one formula which is not a literal, there is a choice of which formula to decompose. This opens the possibility of applying heuristics in order to cause the tableau to be completed quickly. We saw in Sect. 2.6.2 that it is better to decompose α-formulas before β-formulas to avoid duplication.

Tableaux can be shortened by closing a branch if it contains a formula and its negation and not just a pair of complementary literals. Clearly, there is no reason to continue expanding a node containing:

$$(p \wedge (q \vee r)), \quad \neg(p \wedge (q \vee r)).$$

We leave it as an exercise to prove that this modification preserves the correctness of the algorithm.

There is a lot of redundancy in copying formulas from one node to another:

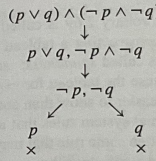
$$U(l') = (U(l) - \{A\}) \cup \{A_1, A_2\}.$$

In a variant of semantic tableaux called *analytic tableaux* (Smullyan, 1968), when a new node is created, it is labeled only with the new formulas:

$$U(l') = \{A_1, A_2\}.$$

The algorithm is changed so that the formula to be decomposed is selected from the set of formulas labeling the nodes on the branch from the root to a leaf (provided, of course, that the formula has not already been selected). A leaf is marked closed if two complementary literals (or formulas) appear in the labels of one or two nodes on a branch, and a leaf is marked open if is not closed but there are no more formulas to decompose.

Here is an analytic tableau for the formula B from Example 2.62, where the formula $p \vee q$ is not copied from the second node to the third when $p \wedge q$ is decomposed:



We prefer to use semantic tableaux because it is easy to see which formulas are candidates for decomposition and how to mark leaves.

2.7 Soundness and Completeness

The construction of a semantic tableau is a purely formal. The decomposition of a formula depends solely on its syntactical properties: its principal operator and—if it is a negation—the principal operator of the formula that is negated. We gave several examples to motivate semantic tableau, but we have not yet proven that the algorithm is correct. We have not connected the syntactical outcome of the algorithm (Is the tableau closed or not?) with the semantical concept of truth value. In this section, we prove that the algorithm is correct in the sense that it reports that a formula is satisfiable or unsatisfiable if and only if there exists or does not exist a model for the formula.

The proof techniques of this section should be studied carefully because they will be used again and again in other logical systems.

Theorem 2.67 Soundness and completeness *Let $\mathcal{T}$ be a completed tableau for a formula A. A is unsatisfiable if and only if $\mathcal{T}$ is closed.*

Here are some corollaries that follow from the theorem.

Corollary 2.68 *A is satisfiable if and only if $\mathcal{T}$ is open.*

Proof A is satisfiable iff (by definition) A is not unsatisfiable iff (by Theorem 2.67) $\mathcal{T}$ is not closed iff (by definition) $\mathcal{T}$ is open. ∎

Corollary 2.69 *A is valid if and only if the tableau for $\neg A$ closes.*

Proof A is valid iff $\neg A$ is unsatisfiable iff the tableau for $\neg A$ closes. ∎

Corollary 2.70 *The method of semantic tableaux is a decision procedure for validity in propositional logic.*

Proof Let A be a formula of propositional logic. By Theorem 2.66, the construction of the semantic tableau for $\neg A$ terminates in a completed tableau. By the previous corollary, A is valid if and only if the completed tableau is closed. ∎

The forward direction of Corollary 2.69 is called *completeness*: if A is valid, we can discover this fact by constructing a tableau for $\neg A$ and the tableau will close. The converse direction is called *soundness*: any formula A that the tableau construction claims valid (because the tableau for $\neg A$ closes) actually is valid. Informally in logic, soundness is easier to show than completeness. The reason is that while we only include in a formal system rules that are obviously sound, it is hard to be sure that we haven't forgotten some rule that may be needed for completeness. At the extreme, the following vacuous algorithm is sound but far from complete!

Algorithm 2.71 (Incomplete decision procedure for validity)
Input: A formula A of propositional logic.
Output: A is not valid. ∎

Example 2.72 If the rule for $\neg(A_1 \vee A_2)$ is omitted, the construction of the tableau is still sound, but it is not complete, because it is impossible to construct a closed tableau for the obviously valid formula $A = \neg p \vee p$. Label the root of the tableau with the negation $\neg A = \neg(\neg p \vee p)$; there is now no rule that can be used to decompose the formula. ∎

2.7.1 Proof of Soundness

The theorem to be proved is: if the tableau $\mathscr{T}$ for a formula A closes, then A is unsatisfiable. We will prove a more general theorem: if $\mathscr{T}_n$, the subtree rooted at node n of $\mathscr{T}$, closes then the set of formulas $U(n)$ labeling n is unsatisfiable. Soundness is the special case for the root.

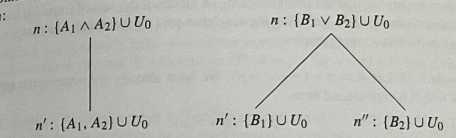
To make the proof easier to follow, we will use $A_1 \wedge A_2$ and $B_1 \vee B_2$ as representatives of the classes of α- and β-formulas, respectively.

Proof of Soundness The proof is by induction on the height h_n of the node n in $\mathscr{T}_n$. Clearly, a closed leaf is labeled by an unsatisfiable set of formulas. Recall (Definition 2.42) that a set of formulas is unsatisfiable iff for any interpretation the truth value of at least one formula is false. In the inductive step, if the children of a node n are labeled by an unsatisfiable set of formulas, then: (a) either the unsatisfiable formula also appears in the label of n, or (b) the unsatisfiable formulas in the labels of the children were used to construct an unsatisfiable formula in the label of n. Let us write out the formal proof.

For the base case, $h_n = 0$, assume that $\mathscr{T}_n$ closes. Since $h_n = 0$ means that n is a leaf, $U(n)$ must contain a complementary set of literals so it is unsatisfiable.

For the inductive step, let n be a node such that $h_n > 0$ in $\mathscr{T}_n$. We need to show that $\mathscr{T}_n$ is closed implies that $U(n)$ is unsatisfiable. By the inductive hypothesis, we can assume that for any node m of height $h_m < h_n$, if $\mathscr{T}_m$ closes, then $U(m)$ is unsatisfiable.

Since $h_n > 0$, the rule for some α- or β-formula was used to create the children of n:



Case 1: $U(n) = \{A_1 \wedge A_2\} \cup U_0$ and $U(n') = \{A_1, A_2\} \cup U_0$ for some (possibly empty) set of formulas U_0.
Clearly, $\mathscr{T}_{n'}$ is also a closed tableau and since $h_{n'} = h_n - 1$, by the inductive hypothesis $U(n')$ is unsatisfiable. Let $\mathscr{I}$ be an arbitrary interpretation. There are two possibilities:

- $v_{\mathscr{I}}(A_0) = F$ for some formula $A_0 \in U_0$. But $U_0 \subset U(n)$ so $U(n)$ is also unsatisfiable.
- Otherwise, $v_{\mathscr{I}}(A_0) = T$ for all $A_0 \in U_0$, so $v_{\mathscr{I}}(A_1) = F$ *or* $v_{\mathscr{I}}(A_2) = F$. Suppose that $v_{\mathscr{I}}(A_1) = F$. By the definition of the semantics of $\wedge$, this implies that $v_{\mathscr{I}}(A_1 \wedge A_2) = F$. Since $A_1 \wedge A_2 \in U(n)$, $U(n)$ is unsatisfiable. A similar argument holds if $v_{\mathscr{I}}(A_2) = F$.

Case 2: $U(n) = \{B_1 \vee B_2\} \cup U_0$, $U(n') = \{B_1\} \cup U_0$, and $U(n'') = \{B_2\} \cup U_0$ for some (possibly empty) set of formulas U_0.
Clearly, $\mathscr{T}_{n'}$ and $\mathscr{T}_{n''}$ are also closed tableaux and since $h_{n'} \leq h_n - 1$ and $h_{n''} \leq h_n - 1$, by the inductive hypothesis $U(n')$ and $U(n'')$ are both unsatisfiable. Let $\mathscr{I}$ be an arbitrary interpretation. There are two possibilities:

- $v_{\mathscr{I}}(B_0) = F$ for some formula $B_0 \in U_0$. But $U_0 \subset U(n)$ so $U(n)$ is also unsatisfiable.
- Otherwise, $v_{\mathscr{I}}(B_0) = T$ for all $B_0 \in U_0$, so $v_{\mathscr{I}}(B_1) = F$ (since $U(n')$ is unsatisfiable) *and* $v_{\mathscr{I}}(B_2) = F$ (since $U(n'')$ is unsatisfiable). By the definition of the semantics of $\vee$, this implies that $v_{\mathscr{I}}(B_1 \vee B_2) = F$. Since $B_1 \vee B_2 \in U(n)$, $U(n)$ is unsatisfiable. ∎

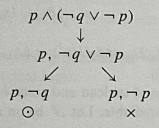
2.7.2 Proof of Completeness

The theorem to be proved is: if A is unsatisfiable then *every* tableau for A closes. Completeness is much more difficult to prove than soundness. For soundness, we had a single (though arbitrary) closed tableau for a formula A and we proved that A is unsatisfiable by induction on the structure of a tableau. Here we need to prove that no matter how the tableau for A is constructed, it must close.

Rather than prove that every tableau must close, we prove the contrapositive (Corollary 2.68): if some tableau for A is open (has an open branch), then A is satisfiable. Clearly, there is a model for the set of literals labeling the leaf of an open branch. We extend this to an interpretation for A and then prove by induction on the length of the branch that the interpretation is a model of the sets of formulas labeling the nodes on the branch, including the singleton set $\{A\}$ that labels the root.

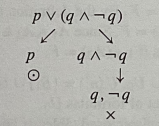
Let us look at some examples.

Example 2.73 Let $A = p \wedge (\neg q \vee \neg p)$. We have already constructed the tableau for A which is reproduced here:



The interpretation $\mathscr{I}(p) = T$, $\mathscr{I}(q) = F$ defined by assigning T to the literals labeling the leaf of the open branch is clearly a model for A. ∎

Example 2.74 Now let $A = p \vee (q \wedge \neg q)$; here is a tableau for A:



The open branch of the tableau terminates in a leaf labeled with the singleton set of literals $\{p\}$. We can conclude that any model for A must define $\mathscr{I}(p) = T$. However, an interpretation for A must also define an assignment to q and the leaf gives us no guidance as to which value to choose for $\mathscr{I}(q)$. But it is obvious that it doesn't matter what value is assigned to q; in either case, the interpretation will be a model of A. ∎

To prove completeness we need to show that the assignment of T to the literals labeling the leaf of an open branch can be extended to a model of the formula labeling the root. There are four steps in the proof:

1. Define a property of sets of formulas;
2. Show that the union of the formulas labeling nodes in an open branch has this property;
3. Prove that any set having this property is satisfiable;
4. Note that the formula labeling the root is in the set.

Definition 2.75 Let U be a set of formulas. U is a *Hintikka set* iff:

1. For all atoms p appearing in a formula of U, either $p \notin U$ or $\neg p \notin U$.
2. If $A \in U$ is an α-formula, then $A_1 \in U$ *and* $A_2 \in U$.
3. If $B \in U$ is a β-formula, then $B_1 \in U$ *or* $B_2 \in U$. ∎

Example 2.76 U, the union of the set of formulas labeling the nodes in the open branch of Example 2.74, is $\{p,\ p \vee (q \wedge \neg q)\}$. We claim that U is a Hintikka set. Condition (1) obviously holds since there is only one literal p in U and $\neg p \notin U$. Condition (2) is vacuous. For Condition (3), $B = p \vee (q \wedge \neg q) \in U$ is a β-formula and $B_1 = p \in U$. ∎

Condition (1) requires that a Hintikka set not contain a complementary pair of literals, which to be expected on an open branch of a tableau. Conditions (2) and (3) ensure that U is *downward saturated*, that is, U contains sufficient subformulas so that the decomposition of the formula to be satisfied will not take us out of U. In turn, this ensures that an interpretation defined by the set of literals in U will make all formulas in U true.

The second step of the proof of completeness is to show that the set of formulas labeling the nodes in an open branch is a Hintikka set.

Theorem 2.77 *Let l be an open leaf in a completed tableau $\mathscr{T}$. Let $U = \bigcup_i U(i)$, where i runs over the set of nodes on the branch from the root to l. Then U is a Hintikka set.*

Proof In the construction of the semantic tableau, there are no rules for decomposing a literal p or $\neg p$. Thus if a literal p or $\neg p$ appears for the first time in $U(n)$ for some n, the literal will be copied into $U(k)$ for all nodes k on the branch from n to l, in particular, $p \in U(l)$ or $\neg p \in U(l)$. This means that all literals in U appear in $U(l)$. Since the branch is open, no complementary pair of literals appears in $U(l)$, so Condition (1) holds for U.

Suppose that $A \in U$ is an α-formula. Since the tableau is completed, A was the formula selected for decomposing at some node n in the branch from the root to l. Then $\{A_1, A_2\} \subseteq U(n') \subseteq U$, so Condition (2) holds.

Suppose that $B \in U$ is an β-formula. Since the tableau is completed, B was the formula selected for decomposing at some node n in the branch from the root to l. Then either $B_1 \in U(n') \subseteq U$ or $B_2 \in U(n') \subseteq U$, so Condition (3) holds. ∎

The third step of the proof is to show that a Hintikka set is satisfiable.

Theorem 2.78 (Hintikka's Lemma) *Let U be a Hintikka set. Then U is satisfiable.*

Proof We define an interpretation and then show that the interpretation is a model of U. Let $\mathscr{P}_U$ be set of all atoms appearing in all formulas of U. Define an interpretation $\mathscr{I} : \mathscr{P}_U \mapsto \{T, F\}$ as follows:

$$\begin{aligned} \mathscr{I}(p) &= T \qquad \text{if } p \in U, \\ \mathscr{I}(p) &= F \qquad \text{if } \neg p \in U, \\ \mathscr{I}(p) &= T \qquad \text{if } p \notin U \text{ and } \neg p \notin U. \end{aligned}$$

Since U is a Hintikka set, by Condition (1) $\mathscr{I}$ is well-defined, that is, every atom in $\mathscr{P}_U$ is given exactly one value. Example 2.74 demonstrates the third case: the atom q appears in a *formula* of U so $q \in \mathscr{P}_U$, but neither the literal q nor its complement $\neg q$ appear in U. The atom is arbitrarily mapped to the truth value T.

We show by structural induction that for any $A \in U$, $v_{\mathscr{I}}(A) = T$.

- If A is an atom p, then $v_{\mathscr{I}}(A) = v_{\mathscr{I}}(p) = \mathscr{I}(p) = T$ since $p \in U$.
- If A is a negated atom $\neg p$, then since $\neg p \in U$, $\mathscr{I}(p) = F$, so $v_{\mathscr{I}}(A) = v_{\mathscr{I}}(\neg p) = T$.
- If A is an α-formula, by Condition (2) $A_1 \in U$ and $A_2 \in U$. By the inductive hypothesis, $v_{\mathscr{I}}(A_1) = v_{\mathscr{I}}(A_2) = T$, so $v_{\mathscr{I}}(A) = T$ by definition of the conjunctive operators.
- If A is β-formula B, by Condition (3) $B_1 \in U$ or $B_2 \in U$. By the inductive hypothesis, either $v_{\mathscr{I}}(B_1) = T$ or $v_{\mathscr{I}}(B_2) = T$, so $v_{\mathscr{I}}(A) = v_{\mathscr{I}}(B) = T$ by definition of the disjunctive operators. ∎

Proof of Completeness Let $\mathscr{T}$ be a completed open tableau for A. Then U, the union of the labels of the nodes on an open branch, is a Hintikka set by Theorem 2.77. Theorem 2.78 shows an interpretation $\mathscr{I}$ can be found such that U is simultaneously satisfiable in $\mathscr{I}$. A, the formula labeling the root, is an element of U so $\mathscr{I}$ is a model of A. ∎

2.8 Summary

The presentation of propositional logic was carried out in a manner that we will use for all systems of logic. First, the syntax of formulas is given. The formulas are defined as trees, which avoids ambiguity and simplifies the description of structural induction.

The second step is to define the semantics of formulas. An interpretation is a mapping of atomic propositions to the values $\{T, F\}$. An interpretation is used to give a truth value to any formula by induction on the structure of the formula, starting from atoms and proceeding to more complex formulas using the definitions of the Boolean operators.

A formula is satisfiable iff it is true in *some* interpretation and it is valid iff is true in *all* interpretations. Two formulas whose values are the same in all interpretations are logically equivalent and can be substituted for each other. This can be used to show that for any formula, there exists a logically equivalent formula that uses only negation and either conjunction or disjunction.

While truth tables can be used as a decision procedure for the satisfiability or validity of formulas of propositional logic, semantic tableaux are usually much more efficient. In a semantic tableau, a tree is constructed during a search for a model of a formula; the construction is based upon the structure of the formula. A semantic tableau is closed if the formula is unsatisfiable and open if it is satisfiable.

We proved that the algorithm for semantic tableaux is sound and complete as a decision procedure for satisfiability. This theorem connects the syntactical aspect of a formula that guides the construction of the tableau with its meaning. The central concept in the proof is that of a Hintikka set, which gives conditions that ensure that a model can be found for a set of formulas.

2.9 Further Reading

The presentation of semantic tableaux follows that of Smullyan (1968) although he uses analytic tableaux. Advanced textbooks that also use tableaux are Nerode and Shore (1997) and Fitting (1996).

2.10 Exercises

2.1 Draw formation trees and construct truth tables for

$$(p \to (q \to r)) \to ((p \to q) \to (p \to r)),$$
$$(p \to q) \to p,$$
$$((p \to q) \to p) \to p.$$

2.2 Prove that there is a unique formation tree for every derivation tree.

2.3 Prove the following logical equivalences:

$$A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C),$$
$$A \vee B \equiv \neg(\neg A \wedge \neg B),$$
$$A \wedge B \equiv \neg(\neg A \vee \neg B),$$
$$A \to B \equiv \neg A \vee B,$$
$$A \to B \equiv \neg(A \wedge \neg B).$$

2.4 Prove $((A \oplus B) \oplus B) \equiv A$ and $((A \leftrightarrow B) \leftrightarrow B) \equiv A$.