1 Propositional Logic

There are several reasons that one studies computer-oriented deductive logics. Such logics have played an important role in natural language understanding systems, in intelligent database theory, in expert systems, and in automated reasoning systems, for example. Deductive logics have even been used as the basis of programming languages, a topic we consider in this book. Of course, deductive logics have been important long before the invention of computers, being key to the foundations of mathematics as well as a guide in the general realm of rational discourse.

What is a deductive logic? It has two major components, a formal language and a notion of deduction. A formal language has a syntax and semantics, just like a natural language. The first formal language we present is the propositional logic.

We first give the syntax of the propositional logic.

- Alphabet
  1. Statement Letters: $P, Q, R, P_1, Q_1, R_1, \ldots$.
  2. Logical Connectives: $\wedge, \vee, \neg, \rightarrow, \leftrightarrow$.
  3. Punctuation: (,).
- Well-formed formulas (wffs), or statements
  1. Statement letters are wffs (*atomic wffs* or *atoms*).
  2. If A and B are wffs, then so are $(\neg A)$, $(A \vee B)$, $(A \wedge B)$, $(A \rightarrow B)$, and $(A \leftrightarrow B)$.

Convention. For any inductively defined class of entities we will assume a closure clause, meaning that only those items created by (often repeated use of) the defining rules are included in the defined class.

We will denote statement letters with the letters P, Q, R, possibly with subscripts, and wffs by letters from the first part of the alphabet unless explicitly noted otherwise. Statement letters are considered to be *non-logical symbols*.

We list the logical connectives with their English labels. (The negation symbol doesn't connect anything; it only has one argument. However, for convenience we ignore that detail and call all the listed logical symbols "connectives.")

$\neg$	negation
$\wedge$	and
$\vee$	or
$\rightarrow$	implies
$\leftrightarrow$	equivalence

Example. $(((\neg P) \wedge Q) \rightarrow (P \rightarrow Q))$.

A formal language, like a natural language, has an alphabet and a notion of grammar. Here the grammar is simple; the wffs (statements) are our sentences. As even our simple example shows, the parentheses make statements very messy. We usually simplify matters by forgoing technically correct statements in favor of sensible abbreviations. We suppress some parentheses by agreeing on a hierarchy of connectives. However, it is always correct to retain parentheses to aid quick readability even if rules permit further elimination of parentheses. We also will use brackets or braces on occasion to enhance readability. They are to be regarded as parentheses formally.

We give the hierarchy with the tightest binding connective on top.

$$\neg$$

$$\vee, \wedge$$

$$\rightarrow, \leftrightarrow$$

We adopt association-to-the-right.

$$A \rightarrow B \rightarrow C \quad \text{is} \quad (A \rightarrow (B \rightarrow C)).$$

Place the parentheses as tightly as possible consistent with the existing bindings, beginning at the top of the hierarchy.

Example. The expression $\neg P \wedge Q \rightarrow P \rightarrow Q$ represents the wff

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

A more readable expression, also called a wff for convenience, is

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

1.1 Propositional Logic Semantics

We first consider the semantics of wffs by seeing their source as natural language statements.

English to formal notation.

Example 1. An ongoing hurricane implies no class meeting, so if no hurricane is ongoing then there is a class meeting.

Use: H: there is an ongoing hurricane
C: there is a class meeting

Formal representation.

(1) A
(2) $(H \to \neg C) \to (\neg H \to C)$
(3) $(H \to \neg C) \wedge [(H \to \neg C) \to (\neg H \to C)]$

The first representation is technically correct (ignoring the "use" instruction), but useless. The idea is to formalize a sentence in as fine-grained an encoding as is possible with the logic at hand. The second and third representations do this.

Notice that there are three different English expressions encoded by the implies symbol. "If . . . then" and "implies" directly translate to the formal implies connective. "So" is trickier. If "so" implies the truth of the first clause, then wff (3) is preferred as the clause is asserted and connected by the logical "and" symbol to the implication. The second interpretation simply reflects that the antecedent of "so" (supposedly) implies the consequent. This illustrates that formalizing natural language statements is often an exercise in guessing the intent of the natural language statement. The task of formalizing informal statements is often best approached by creating a candidate wff and then directly replacing the statement letters by the assigned English meanings and judging how successfully the wff captures the intent of the informal statement. One does this by asserting the truth of the wff and then considering the truth status of the component parts. For example, asserting the truth of wff (3) forces the truth of the two components of the logical "and" by the meaning of logical "and." We consider the full "meaning" of implication shortly. One should try alternate wffs if there is any question that the wff first tried is not clearly satisfactory. Doing this in this case yields representations (2) and (3). Which is better depends on one's view of the use of "so" in this sentence.

Aside: For those who suspect that this is not a logically true formula, you are correct. We deal with this aspect later.

Example 2. Either I study logic now or I go to the game. If I study logic now I will pass the exam but if I go to the game I will fail the exam. Therefore, I will go to the game and drop the course.

Use: L: I study logic now
G: I go to the game
P: I pass the exam
D: I drop the course

Formal representation.

(1) $(L \vee G) \wedge [((L \rightarrow P) \wedge (G \rightarrow \neg P)) \rightarrow G \wedge D]$
or
(2) $((L \vee G) \wedge (L \rightarrow P) \wedge (G \rightarrow \neg P)) \wedge [((L \rightarrow P) \wedge (G \rightarrow \neg P)) \rightarrow G \wedge D]$.

"Therefore" is another English trigger for the logical "implies" connective in the wff that represents the English sentences. Again, it is a question of whether the sentences preceding the "therefore" are intended as facts or only as part of the conditional statement. Two possibilities are given here. As before, the logical "and" forces the assertion of truth of its two components when the full statement is asserted to be true. Note that the association-to-the-right is used here to avoid one pair of added parentheses in representation (2). We do group the first three subformulas (conjuncts) together reflecting the English sense.

As illustrated above, logic studies the truth value of natural language sentences and their associated wffs. The truth values are limited to true (T) and false (F) in classical logic. All interpreted wffs have truth values. To determine the truth values of interpreted wffs we need to define the meaning of the logical connectives. This is done by a *truth table*, which defines the logical symbols as functions from truth values to truth values. (We often refer to these logical functions as the logical connectives of propositional logic.)

Definition. The following truth tables are called the *defining truth tables* for the given functions.

Although there appears to be only one "table" displayed here, it is because we have combined the defining truth table for each connective into one compound truth table.

To illustrate the use of the truth tables, consider the "and" function. Suppose we have a wff (possibly a subformula) of form $A \wedge B$ where we know that A and B both have truth value F. The first two columns give the arguments of the logical functions so we choose the fourth row. $A \wedge B$ has value F in that row so we know that $A \wedge B$ has value F when each of A and B has truth value F. All the logical functions except "implies" are as you have usually understood the meaning of those functions. Notice that the "or" function is the inclusive or. We discuss the function "implies" shortly, but a key concept is that this definition forces the falsity of $A \rightarrow B$ to mean both that A is true and B is false.

A	B	$\neg A$	$A \wedge B$	$A \vee B$	$A \rightarrow B$	$A \leftrightarrow B$
T	T	F	T	T	T	T
T	F	F	F	T	F	F
F	T	T	F	T	T	F
F	F	T	F	F	T	T

The idea of logic semantics is that the user specifies the truth value of the statement letters, or atoms, and the logical structure of the wff as determined by the connectives supplies the truth value of the entire wff. We capture this "evaluation" of the wffs by defining a *valuation* function.

We will need a number of definitions that deal with propositional logic semantics and we give them together for easier reference later, followed by an illustration of their use. Here and hereafter we use "iff" for "if and only if."

Definition. An *interpretation* associates a T or F with each statement letter.

Definition. A *valuation function* $\mathcal{V}$ is a mapping from wffs and interpretations to T and F, determined by the defining truth tables of the logical connectives. We write $\mathcal{V}_I[A]$ for the value of the valuation of wff A under interpretation I of A.

Definition. A wff A is a *tautology* (*valid* wff) iff $\mathcal{V}_I[A] = \mathsf{T}$ for all interpretations I.

Example. The wff $P \vee \neg P$ is a tautology.

We now consider further the choice we made in the defining truth table for implication. We have chosen a truth table for implication that yields the powerful result that the theorems of the deductive logic we study in this part of the book are precisely the valid wffs. That is, we are able to prove precisely the set of wffs that are true in all interpretations. The implication function chosen here is called *material* implication. Other meanings (i.e., semantics) for implication are considered in Part 3 of this book.

Definition. A wff A is *satisfiable* iff there exists an interpretation I such that $\mathcal{V}_I[A] = \mathsf{T}$.

We also say "I makes A true" and "I satisfies A."

Definition. A wff A is *unsatisfiable* iff the wff is not satisfiable; i.e., there does NOT exist an interpretation I such that $\mathcal{V}_I[A] = \mathsf{T}$.

Definition. Interpretation I is a *model* of wff A iff $\mathcal{V}_I[A] = \mathsf{T}$. I is a model of a set $\mathcal{S}$ of wffs iff I is a model of each wff of $\mathcal{S}$. If A is satisfiable, then *A has a model*.

Definition. A is a *logical consequence* of a set $\mathcal{S}$ of wffs iff every model of $\mathcal{S}$ is a model of A.

Notation. We will have occasion to explicitly represent finite sets. One way this is done is by listing the members. Thus, if A_1, A_2, and A_3 are the (distinct) members of a set then we can present the set as $\{A_1, A_2, A_3\}$. $\{A_i | 1 \leq i \leq n\}$ represents a set with n members $A_1, A_2, \ldots, A_n$. $\{A\}$ represents a set with one member, A.

Notation. $\mathcal{S} \models A$ denotes the relation that A is a logical consequence of the set $\mathcal{S}$ of wffs. We write $\models A$ iff A is a tautology. We write $\not\models A$ for the negation of $\models A$. We sometimes write $P \models Q$ for $\{P\} \models Q$ where P and Q are wffs.

Definition. A and B are *logically equivalent* wffs iff A and B have exactly the same models.

We illustrate the valuation function.

Example. Determine the truth value of the following sentence under the given interpretation.

An ongoing hurricane implies no class meeting, so if no hurricane is ongoing then there is a class meeting.

Use: H: there is an ongoing hurricane
C: there is a class meeting

Interpretation I is as follows.

$$I(H) = \mathsf{T} \qquad\qquad I(C) = \mathsf{T}$$

Then

$$\mathcal{V}_I[(H \to \neg C) \to (\neg H \to C)] = ??$$

We determine $\mathcal{V}_I$. We omit the subscript on $\mathcal{V}$ when it is understood.

$I(H)$	$I(C)$	$\mathcal{V}[H \to \neg C]$	$\mathcal{V}[\neg H \to C]$	$\mathcal{V}[(H \to \neg C) \to (\neg H \to C)]$
T	T	F	T	T

The above gives the truth value of the statement under the chosen interpretation. That is, the meaning and truth value of the statement is determined by the interpretation and the correct row in the truth table that corresponds to the intended interpretation. Thus,

$$\mathcal{V}_I[(H \to \neg C) \to (\neg H \to C)] = \mathsf{T}.$$

Now let us check as to whether the statement is a tautology. Following the tradition of presentation of truth tables, we omit reference to the interpretation and valuation function in the column headings, even though the entries T,F are determined by the interpretation and valuation function in the rows as above. Of course, the interpretation changes with each row.

We see that the statement A is not a tautology because $\mathcal{V}_I[A] = \mathsf{F}$ under $I(H) = \mathsf{F}$ and $I(C) = \mathsf{F}$.

H	C	$H \to \neg C$	$\neg H \to C$	$(H \to \neg C) \to (\neg H \to C)$
T	T	F	T	T
T	F	T	T	T
F	T	T	T	T
F	F	T	F	F

We now establish two facts that reveal important properties of some of the terms we have introduced. The first property is so basic that we do not even honor it by calling it a theorem.

Although people often find it more comfortable to skip proofs on the first reading of a mathematical text, we suggest that reading proofs often reinforces the basic concepts of the discipline. We particularly urge the reader to thoroughly understand the following two proofs as they utilize the basic definitions just given and help clarify the concepts involved.

Fact. Wff A is a tautology iff $\neg A$ is unsatisfiable.

Proof. We handle the two implications separately.

Note: The splitting of the iff (the biconditional, or equivalence) to two implications is justified by the tautology $(A \leftrightarrow B) \leftrightarrow ((A \to B) \wedge (B \to A))$. Likewise, the use of the contrapositive below is justified by a tautology, namely $(A \to B) \leftrightarrow (\neg B \to \neg A)$. The promotion of a tautology to permit the substitution of one proof approach by another warrants a comment as this illustrates a key use of logic. Consider the contrapositive. We wish to prove an implication $(A \to B)$. It is more convenient to prove the contrapositive $(\neg B \to \neg A)$ in some cases as in the case that follows this note. The second tautology above and the truth table for equivalence $\leftrightarrow$ says that $(\neg B \to \neg A)$ is true whenever $(A \to B)$ is true and only then. Thus the contrapositive proof establishes the original implication.

$\Longrightarrow$: Use the contrapositive.

Assume NOT($\neg A$ is unsatisfiable), i.e., $\neg A$ is satisfiable. Thus, we know a model I exists for $\neg A$, that is, $\mathcal{V}_I[\neg A] = \mathsf{T}$. But then, $\mathcal{V}_I[A] = \mathsf{F}$. Therefore, NOT($A$ is a tautology).

$\Longleftarrow$: See Exercise 10. □

The following theorem is very important as it goes to the heart of logic. A major reason to study logic is to understand the notion of logical consequence, namely, what follows logically from a given set of statements. This theorem states that given a (finite) set of statements, treated as facts, and a statement B that follows logically from those facts, then that entailment is captured by the logical implies symbol. Thus, to test for the logical consequence of a set of wffs, we need only test whether the related wff that "replaces" the logical consequence by the implies sign is a tautology.

Theorem 1. If $\mathcal{S} = \{A_i | 1 \leq i \leq n\}$ then the following equivalence holds:

$$\mathcal{S} \models B$$
iff
$$\models A_1 \wedge A_2 \wedge \cdots \wedge A_n \to B.$$

Proof. $\Longrightarrow$: We prove the contrapositive.
If $\not\models A_1 \wedge \cdots \wedge A_n \to B$ then there exists some interpretation I such that $\mathcal{V}_I[A_1 \wedge \cdots \wedge A_n \to B] = \mathsf{F}$ which forces $\mathcal{V}_I[A_1 \wedge \cdots \wedge A_n] = \mathsf{T}$ and $\mathcal{V}_I[B] = \mathsf{F}$. But then $\mathcal{V}_I[A_i] = \mathsf{T}$ all i, by repeated use of the truth table for $\wedge$. So I is a model of the set $\mathcal{S}$. Yet $\mathcal{V}_I[B] = \mathsf{F}$, so $\mathcal{S} \not\models B$. The contrapositive is shown.

$\Longleftarrow$: Again, we prove the contrapositive.

If not $\mathcal{S} \models B$ then there exists an interpretation I such that for all i, $1 \le i \le n$, $\mathcal{V}_I[A_i] = \mathsf{T}$, and $\mathcal{V}_I[B] = \mathsf{F}$. Then $\mathcal{V}_I[A_1 \wedge \cdots \wedge A_n] = \mathsf{T}$. Thus, $\mathcal{V}_I[A_1 \wedge \cdots \wedge A_n \to B] = \mathsf{F}$ so $A_1 \wedge \cdots \wedge A_n \to B$ is not a tautology. □

We next review the notion of replacement of one subformula by another, passing the validity of a wff to its offspring when appropriate. This is a powerful tool that we will use frequently.

Theorem 2 (Replacement Theorem). Let wff F contain subformula B and F_1 be the result of replacing one or more occurrences of B by wff C in F.

Then

$$\models B \leftrightarrow C$$

implies

(*) $$\models F \leftrightarrow F_1.$$

Note that (*) implies that for all interpretations I, $\mathcal{V}_I[F] = \mathsf{T}$ iff $\mathcal{V}_I[F_1] = \mathsf{T}$.

Proof. The proof is by induction on the number of connective occurrences in F. (We provide a review of induction techniques in an appendix.) We present a full proof below but first we present the key idea of the proof, which is quite intuitive.

Recall that $\mathcal{V}_I[F]$ is computed from atomic formulas outward. Only the truth value of B, not the content of B, influences $\mathcal{V}_I[E]$ for any subformula E of F containing B. Thus, replacement of B by C where $\models B \leftrightarrow C$ will mean

$$\mathcal{V}_I[F] = \mathcal{V}_I[F_1], \text{ any } I.$$

Example.

F: $P \vee (P \to Q)$
B: $P \to Q$
C: $\neg P \vee Q$

(Note: $\models (P \to Q) \leftrightarrow (\neg P \lor Q)$)
F_1: $P \lor (\neg P \lor Q)$
$\models F \leftrightarrow F_1$ by the Replacement Theorem

Aside:
$\models P \lor (\neg P \lor Q) \leftrightarrow (P \lor \neg P) \lor Q$, by associativity of $\lor$. But $(P \lor \neg P) \lor Q$ is a tautology. Therefore, $P \lor (P \to Q)$ is a tautology by reference to the example above.

We now give the formal proof. We use a form of induction proof often called complete induction that is frequently used in proofs in mathematical logic.

P(n). The theorem holds when the wff F contains n occurrences of logical connectives.

P(0). F is an atomic wff so the substitution of one or more occurrences of B by C means that F is B and so F_1 is C. Since we are given that $\models B \leftrightarrow C$ we immediately have $\models F \leftrightarrow F_1$.

We now show the induction step. For each $n \geq 0$ we show:
Assume P(k) for all $0 \leq k \leq n$, to show P(n + 1).

Case (a). F is completely replaced; i.e., F is B. This is the case P(0) already treated.

Case (b). F is of form $\neg G$ and one or more occurrences of B are replaced in G by C to create G_1. By induction hypothesis $\models G \leftrightarrow G_1$ because G has fewer occurrences of connectives than F. Thus, for any arbitrary interpretation I, $\mathcal{V}_I[G] = \mathcal{V}_I[G_1]$, so $\mathcal{V}_I[\neg G] = \mathcal{V}_I[\neg G_1]$. Thus, $\models \neg G \leftrightarrow \neg G_1$ by the truth table for equivalence. That is, $\models F \leftrightarrow F_1$.

Case (c). F is of form $G \lor H$ and one or more occurrences of B are replaced in either or both G and H by C to create G_1 and H_1, respectively. By induction hypothesis $\models G \leftrightarrow G_1$ and $\models H \leftrightarrow H_1$. (Here we need complete induction as each of G and H may have considerably fewer connective occurrences than F.) We must show $\models G \leftrightarrow G_1$ and $\models H \leftrightarrow H_1$ imply $\models F \leftrightarrow F_1$. Let I be an arbitrary interpretation for G, G_1, H, and H_1. By $\models G \leftrightarrow G_1$ and $\models H \leftrightarrow H_1$ we know that $\mathcal{V}_I[G] = \mathcal{V}_I[G_1]$ and $\mathcal{V}_I[H] = \mathcal{V}_I[H_1]$. Therefore, $\mathcal{V}_I[G \lor H] = \mathcal{V}_I[G_1 \lor H_1]$ because the first (resp., second) argument of the **or** logical function has the same truth value on both sides of the equation. To repeat, the preceding equation holds because the arguments of the **or** function "are the same" on both sides of the equation, not because of the truth table for **or**. Since this proof argument holds for any interpretation I of G, G_1, H, and H_1 we have $\models (G \lor H) \leftrightarrow (G_1 \lor H_1)$, or $\models F \leftrightarrow F_1$.

Case (d), Case (e), Case (f). F is of form $G \land H$, F is of form $G \to H$, and F is of the form $G \leftrightarrow H$, respectively. These cases were actually argued in Case (c), as the proof argument for each case is independent of the truth table for the connective, as noted in Case (c).

This establishes that for all n, $n \geq 0$, P(n) holds from which the theorem immediately follows. □

1.2 Syntax: Deductive Logics

Definition (informal). A *formal (deductive) logic* consists of

- a formal language,
- axioms, and
- rules of inference.

Definition (informal). An *(affirmative) deduction of wff A from axioms $\mathcal{A}$ and hypotheses $\mathcal{H}$* is a sequence of wffs of which each is either an

- axiom,
- hypothesis, or
- inferred wff from previous steps by a rule of inference

and with *A* as the last wff in the sequence.

Definition (informal). A *hypothesis* is an assumption (wff) that is special to the deduction at hand.

Definition (informal). A *theorem* is the last entry in an affirmative deduction with no hypotheses.

Definition (informal). A propositional logic that has tautologies as theorems is called a *propositional calculus*.

The above definitions are informal in that they are not fully specified; indeed, they can take different meanings depending on the setting. For example, a theorem is a wff dependent on the grammar of the logic under consideration.

We will be developing a refutation-based deductive logic for the propositional calculus, one that is not in the traditional format for a deductive logic. It will not have a theorem (i.e., a tautology) as the last entry in the deduction as does an affirmative logic. Almost all deductive logics appearing in logic texts are affirmative and so the word is rarely used, and is our reason for placing the term in parentheses. However, the logic we introduce here will be able to give a proof of any tautology, just not directly. We instead refute by deduction the negation of any tautology. The only adjustment to the notion of deduction given above is that we do not derive the theorem as the last wff but rather start with the negated theorem at the beginning and derive a success symbol at the end.

We highlight a key point of the above paragraph. We can show that a wff is a tautology by refuting the negation of that wff.

Why should we consider deductive logics? Although for propositional logic we can sometimes determine tautologyhood faster by truth tables than by a deductive logic, there is no counterpart to truth tables at the first-order logic level. The set of valid first-order theorems for almost all logics is not decidable; in particular, a non-valid wff cannot always be shown to be non-valid. However, the set of valid wffs is semi-decidable. (The notion of decidability is studied in Part 2 of this book.) In fact, the standard method of showing semi-decidability is by use of deductive logics.

We study a deductive logic first at the propositional level that people often find more convenient (and sometimes faster) to apply to establish tautologyhood than by using truth tables. Also, the first-order deductive logic is a natural extension of the propositional logic, so our efforts here will be doubly utilized.

1.3 The Resolution Formal Logic

Resolution deductive logic is a refutation-based logic. As such, resolution has properties not shared with affirmative deductive logics. We list some of the differences here.

- We determine unsatisfiability, not tautologyhood (validity). We emphasize that resolution is used to establish tautologyhood by showing the unsatisfiability of the tautology's negation.
- The final wff of the deduction is not the wff in question. Instead, the final wff is a fixed *empty* wff which is an artifact of convenience.
- There are no axioms, and only one rule of inference (for the propositional case).

The resolution deductive logic has the very important properties that it is sound and complete. That means that it only indicates unsatisfiability when the wff is indeed unsatisfiable (soundness) and, moreover, it always indicates that an unsatisfiable wff is unsatisfiable (completeness).

Frequently, we are interested in whether a set of assumptions logically implies a stated conclusion. For the resolution system we pose this as a conjunction of assumptions and the negation of the conclusion. We then ask whether this wff is unsatisfiable. This will be illustrated after we introduce the resolution formal system.

- Alphabet
  - As before, with logical connectives restricted to $\neg$, $\wedge$, and $\vee$.

- Well-formed formulas

  1. Statement letters are wffs (atomic wffs, atoms).
  2. If A is an atom, then A and $\neg A$ are *literals*, and wffs.
  3. If $L_1, L_2, ..., L_n$ are distinct literals then $L_1 \vee L_2 \vee ... \vee L_n$ is a *clause*, and wff, $n \geq 1$.
  4. If $C_1, C_2, ..., C_m$ are distinct clauses then $C_1 \wedge C_2 \wedge ... \wedge C_m$ is a wff, $m \geq 1$.

  Examples of wffs.

  1. $\neg P$ is a literal, a clause, and a wff.
  2. $P \vee Q$ is a clause and a wff.
  3. $P \wedge \neg Q$ is a wff consisting of two clauses.
  4. $(\neg P \vee Q) \wedge (P \vee \neg Q \vee \neg R)$ is a two-clause wff.
  5. $P \wedge \neg P$ is a two-clause wff but $P \wedge P$ is not a wff because of the repeated clauses.

- Axioms: None.

- Rule of inference
  Let $L_i, \ldots, L'_j$ denote literals.
  Let L and L^c denote two literals that are complementary (i.e., one is an atom and the other is its negation).

  The *resolution* rule has as input two clauses, the *parent* clauses, and produces one clause, the *resolvent* clause. We present the inference rule by giving the two parents above a horizontal line and the resolvent below the line. Notice that one literal from each parent clause disappears from the resolvent but all other literals appear in the resolvent.

$$\frac{L_1 \vee L_2 \vee \ldots L_n \vee L \qquad L^c \vee L'_1 \vee L'_2 \ldots \vee L'_m}{L_1 \vee L_2 \vee \ldots L_n \vee L'_1 \vee L'_2 \ldots \vee L'_m}.$$

  Literal order is unimportant. Clauses, conjunctions of clauses, and resolvents almost always do not permit duplication and we may assume that condition to hold unless explicitly revoked.

We now can formally introduce the *empty clause*, denoted by $\Box$, as the resolvent of two one-literal clauses.

$$\frac{P \qquad \neg P}{\Box}.$$

Notation. A one-literal clause is also called a unit clause.

Definition. $\mathcal{V}_I[\Box] = F$, for all I.

The easiest way to see that the above definition is natural is to consider that for any interpretation I a clause C satisfies $\mathcal{V}_I[C] = T$ iff $\mathcal{V}_I[L] = T$ for some literal L in C. Since $\square$ contains no literals it cannot satisfy this condition, so it is consistent and reasonable to let such a clause C satisfy $\mathcal{V}_I[\square] = F$, for all I.

Definition. The clauses taken as the hypotheses of a resolution deduction are called *given* clauses.

Resolution deductions are standard logic deductions tailored to resolution; they are sequences of wffs that are either given clauses or derived from earlier clauses by use of the resolution inference rule. (In the first-order case given later we do add a second inference rule.) Though not required in general, in this book we will follow the custom of listing all the given clauses at the beginning of the deduction. This allows the viewer to see all the assumptions in one place.

Definition. When all given clauses are listed before any derived clause then the sequence of given clauses is called the *prefix* of the deduction.

The goal of resolution deductions we pursue is to deduce two one-literal clauses, one clause containing an atom and the other clause containing its complement. We will see that this emanates from an unsatisfiable wff. Rather than referring to complementary literals, whose form depends on the alphabet of the wff, we apply the resolution inference rule one step further and obtain the empty clause. Thus, we can speak definitively of the last entry of a resolution refutation demonstration as the empty clause.

Before giving a sample resolution derivation we introduce terminology to simplify our discussion of resolution deductions.

Definition. A *resolution refutation* is a resolution deduction of the empty clause.

We will hereafter drop the reference to resolution in this chapter when discussing deductions and refutations as the inference system is understood to be resolution.

A clause can be regarded as a set of literals since order within a clause is unimportant. We will present example deductions with clauses displaying the "or" connective but on occasion will refer to a clause as a set of literals.

Definition. A *resolving literal* is a literal that disappears from the resolvent but is present in a parent clause.

We now present two resolution refutations.

Example 1. We give a refutation of the wff
$(\neg A \vee B) \wedge (\neg B \vee C) \wedge A \wedge \neg C$:

1.	$\neg A \vee B$	given clause
2.	$\neg B \vee C$	given clause
3.	A	given clause
4.	$\neg C$	given clause
5.	B	resolvent of clauses 1 and 3 on literals $\neg A$ and A
6.	C	resolvent of clauses 2 and 5 on literals $\neg B$ and B
7.	$\square$	resolvent of clauses 4 and 6 on literals $\neg C$ and C

Henceforth we will indicate the clauses used in the resolution inference rule application by number. Also, we use lowercase letters, starting with a, to indicate the literal position in the clause. Thus, the justification of step 5 above is given as

5. B resolvent of 1a and 3.

As an aside, we note that the above refutation demonstrates that the wff $(\neg A \vee B) \wedge (\neg B \vee C) \wedge A \wedge \neg C$ is unsatisfiable. This assertion follows from a soundness theorem that we prove next. Note that this unsatisfiability is equivalent to stating that the first three clauses of the wff imply C. Using the tautology $(\neg A \vee B) \leftrightarrow (A \rightarrow B)$ and the Replacement Theorem we see that the above wff is equivalent to $(A \wedge (A \rightarrow B) \wedge (B \rightarrow C)) \models C$. This is seen as correct by using *modus ponens* twice to yield a traditional proof that C follows from $A \wedge (A \rightarrow B) \wedge (B \rightarrow C)$ (and uses the soundness of your favorite affirmative logic).

One might notice that the inference $(A \wedge (A \rightarrow B) \wedge (B \rightarrow C)) \models C$ is obtained directly in the resolution deduction above; that is, C appears as a derived clause. So one may wonder if this direct inference approach is always possible. If so, then the refutation construct would be unnecessarily complex. However, notice that $P \models P \vee Q$ yet the singleton clause P cannot directly infer clause $P \vee Q$ via the resolution rule. But $P \wedge \neg(P \vee Q)$ can be shown unsatisfiable via a resolution refutation. The reader can verify this by noting that $\neg(P \vee Q)$ is logically equivalent to $\neg P \wedge \neg Q$ so that $P \wedge \neg(P \vee Q)$ is unsatisfiable iff $P \wedge \neg P \wedge \neg Q$ is unsatisfiable. (Later we show a general method for finding wffs within resolution logic that are unsatisfiable iff the original wff is.) The property we have shown is sometimes characterized by the statement that resolution logic does not contain a complete inference system (as opposed to a complete deductive system).

Example 2. We give a refutation of the following clause set
$\{A \vee B \vee C,\ \neg A \vee \neg B,\ B \vee \neg C,\ \neg B \vee \neg C,\ \neg A \vee C,\ \neg B \vee C\}$:

1.	$A \vee B \vee C$	given clause
2.	$\neg A \vee \neg B$	given clause
3.	$B \vee \neg C$	given clause
4.	$\neg B \vee \neg C$	given clause
5.	$\neg A \vee C$	given clause
6.	$\neg B \vee C$	given clause
7.	$\neg C$	resolvent of 3a and 4a
8.	$\neg B$	resolvent of 6b and 7
9.	$A \vee B$	resolvent of 1c and 7
10.	A	resolvent of 8 and 9b
11.	C	resolvent of 5a and 10
12.	$\square$	resolvent of 7 and 11

Notice that clause 2 was not used. Nothing requires all clauses to be used. Also note that line 7 used the fact that $\neg C \vee \neg C$ is logically equivalent to $\neg C$.

After the Soundness Theorem is considered we give some aids to help find resolution proofs.

To show the Soundness Theorem we first show that the resolution rule preserves truth. This provides the induction step in an induction proof of the Soundness Theorem.

Lemma (Soundness Step). Given interpretation I, if

(1) $\mathcal{V}_I[L_1 \vee L_2 \vee \ldots \vee L_n] = \mathsf{T}$

and

(2) $\mathcal{V}_I[L'_1 \vee L'_2 \vee \ldots \vee L'_m] = \mathsf{T}$

and

$$L'_1 = L_1^c$$

then resolvent satisfies $\quad L_2 \vee \ldots \vee L_n \vee L'_2 \vee \ldots \vee L'_m$

(3) $\mathcal{V}_I[L_2 \vee \ldots \vee L_n \vee L'_2 \vee \ldots \vee L'_m] = \mathsf{T}$.

Proof. Expression (1) implies $\mathcal{V}_I[L_i] = \mathsf{T}$ for some i, $1 \le i \le n$. Likewise, expression (2) implies $\mathcal{V}_I[L'_j] = \mathsf{T}$ for some j, $1 \le j \le m$. Since $L_1^c = L'_1$, one of $\mathcal{V}_I[L_1] = \mathsf{F}$ or $\mathcal{V}_I[L'_1] = \mathsf{F}$. Thus one of the subclauses $L_2 \vee \ldots \vee L_n$

and $L'_2 \vee \ldots \vee L'_m$ must retain a literal with valuation T. Hence (3) holds. □

Theorem 3 (Soundness Theorem for Propositional Resolution). If there is a refutation of a given clause set $\mathcal{S}$, then $\mathcal{S}$ is an unsatisfiable clause set.

Proof. We prove the contrapositive. To do this we use the following lemma. Recall that wff □ has no model. □

Lemma. If I is a model of clause set $\mathcal{S}$ then I is a model of every clause of any resolution deduction from $\mathcal{S}$.

Proof. By complete induction on the length of the resolution deduction.

Induction predicate.

P(n): If A is the nth clause in a resolution deduction from $\mathcal{S}$ and if I is a model of $\mathcal{S}$ then $\mathcal{V}_I[A] = \mathsf{T}$.

We show:

P(1). The first clause D_1 of a deduction must be a given clause. Thus $\mathcal{V}_I[D_1] = \mathsf{T}$.

For each $n \geq 1$, we show:

Assume P(k) holds for all k, $1 \leq k \leq n$, to show $P(n+1)$.

Case 1. D_{n+1} is a given clause. Then $\mathcal{V}_I[D_{n+1}] = \mathsf{T}$.

Case 2. D_{n+1} is a resolvent clause. Then the parent clauses appear earlier in the deduction, say at steps k_1 and k_2, and we know by the induction hypothesis that $\mathcal{V}_I[D_{k_1}] = \mathcal{V}_I[D_{k_2}] = \mathsf{T}$. But then $\mathcal{V}_I[D_{k_{n+1}}] = \mathsf{T}$ by the Soundness Step.

We have shown $P(n+1)$ under the induction conditions so we can conclude that $\forall n P(n)$, from which the lemma follows.

We now can conclude the proof of the theorem. Suppose $\mathcal{S}$ is satisfiable. Let I be a model of $\mathcal{S}$. Then I is a model of every clause of the refutation, including □. But $\mathcal{V}_I[\square] = \mathsf{F}$. Contradiction. Thus no such model I of $\mathcal{S}$ can exist and the theorem is proved. □

The last two sentences of the proof may seem a cheat as we defined □ to be false for all interpretations and then used that fact decisively to finish the proof. That is a fair criticism and so we give a more comfortable argument for the unsatisfiability of □. Recall that □ is created in a deduction only when for some atom Q we have already derived clause $\{Q\}$ and also clause $\{\neg Q\}$. But for no interpretation I can both $\mathcal{V}_I[Q] = \mathsf{T}$ and $\mathcal{V}_I[\neg Q] = \mathsf{T}$; i.e., one derived clause is false in interpretation I. So, by the above lemma, when $\Box$ is derived we know that we already have demonstrated the unsatisfiability of the given clause set.

Although we will not focus on proof search techniques for resolution several search aids are so useful that they should be presented. We do not prove here that these devices are safe but a later exercise will cover this.

Definition. A clause C θ-*subsumes* clause D iff D contains all the literals of C. Clause D is the θ-subsumed clause and clause C is the θ-subsuming clause.

We use the term θ-subsumes rather than subsumes as the latter term has different meaning within resolution logic. The reference to θ will be clear when we treat first-order (logic) resolution. The definition above is modified for the propositional case; we give the full and correct definition when we treat first-order resolution.

We use the term *resolution restriction* to refer to a resolution logic whose deductions are a proper subset of general resolution. (Recall that a proper subset of a set is any subset not the full set itself.) The first two search aids result in resolution restrictions in that some deductions are disallowed. Neither rule endangers the existence of a refutation if the original clause set is unsatisfiable.

Aid 1. A clause D that is θ-subsumed by clause C can be neglected in the presence of C in the search for a refutation.

Plausibility argument. The resolution inference rule eliminates only one literal from any parent clause. Thus the literals in D in some sense are eliminated one literal at a time. Those parts of the proof that eliminate the literals in D cover all the literals of C so a refutation involving D also would be a refutation involving C (and usually longer). (See Exercise 4, Chapter 3, for a suggested way to prove the safety of this search restriction.)

The plausibility argument suggests that the refutation using the θ-subsuming clause is shorter than the refutation using the θ-subsumed clause. This is almost always the case. We demonstrate the relationship between refutations with an example.

Example. We give a refutation of the following clause set.

$$\{A \vee B \vee C,\ \neg A \vee \neg B,\ B \vee \neg C,\ \neg B \vee \neg C,\ \neg A \vee C,\ C\}.$$

This example is an alteration of Example 2 on page 18 where clause 6 is here replaced by clause C that θ-subsumes the original clause $\neg B \vee C$.

Note that the sequence of resolvents given in lines 8–10 of the original refutation, needed to derive clause C at line 11, is absent here:

1.	$A \vee B \vee C$	given clause
2.	$\neg A \vee \neg B$	given clause
3.	$B \vee \neg C$	given clause
4.	$\neg B \vee \neg C$	given clause
5.	$\neg A \vee C$	given clause
6.	C	given clause
7.	$\neg C$	resolvent of 3a and 4a
8.	$\square$	resolvent of 6 and 7

Note that clause C θ-subsumes clauses 1 and 5 so those clauses should not be used in the search for a refutation (and they were not).

As another example of θ-subsumption of a clause, again using Example 2 on page 18, consider replacing clause 1 by the clause $A \vee B$. Notice that the saving is much less, though present.

Aid 2. Ignore tautologies.

Plausibility argument. Consider an example inference using a tautology.

$$\frac{P \vee \neg P \vee Q \qquad \neg P \vee R}{\neg P \vee Q \vee R}.$$

By Aid 1, ignore the resolvent as the right parent is better.

Aid 3. Favor use of shorter clauses, one-literal clauses in particular.

Plausibility argument. Aid 1 gives a general reason. The use of one-literal clauses is particularly good for that reason, but also because the resolvent θ-subsumes the other parent and rules out any need to view that clause further. In general, recall that we seek two complementary one-literal clauses to conclude the refutation, so we want to keep generating shorter clauses to move towards generating one-literal clauses.

However, it is necessary to use even the longest clause of the deduction on occasion. This is particularly true of long given clauses but also includes inferred clauses that are longer than any given clause. We therefore stress the word "favor" in Aid 3.

We now prove the Completeness Theorem for propositional resolution. Recall that this is the converse of the Soundness Theorem; now we assert that when a clause set is unsatisfiable then there is a refutation of that clause set. At the propositional level the result is stronger: Whether the clause set is satisfiable or not, there is a point at which the proof search can stop. (It is possible to generate all possible resolvents in the propositional case, although that may be a large number of clauses. This property does not of itself guarantee completeness, of course.)

As for almost all logics, the Completeness Theorem is much more difficult to prove than the Soundness Theorem. A nice property of resolution logic is that the essence of the propositional Completeness Theorem can be presented quite pictorially. However, we do need to define some terms first. We list the definitions together but suggest that they just be scanned now and addressed seriously only when they are encountered in the proof.

In the proof that follows we make use of a data structure called a binary tree, a special form of another data structure called a graph. A (data structure) graph is a structure with *arcs*, or line segments, and *nodes*, which occur at the end points of arcs. We use a directed graph, where the arcs are often represented by arrows to indicate direction. We omit the arrowheads as the direction is always downward in the tree we picture. A binary tree is defined in the first definition below. For an example of a binary tree see Figure 1.1 given in the proof of the Completeness Theorem.

In the following definitions $\mathcal{S}$ is a set of (propositional) clauses and $\mathcal{A}$ is the set of atoms named in $\mathcal{S}$. (Recall: $A \in \mathcal{A}$ means A is a member of the set $\mathcal{A}$.)

Definition. A *(labeled) binary tree* is a directed graph where every node except the single root node and the leaf nodes has one incoming arc and two outgoing arcs. The root node has no incoming arcs and the leaf nodes have no outgoing arcs. Nodes and arcs may have labels.

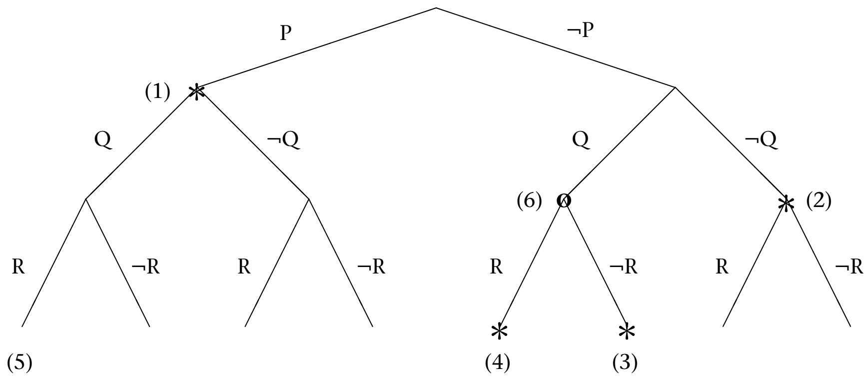
Definition. A *branch* is a connected set of arcs and associated nodes between the root and a leaf node.

Definition. An *ancestor* of (node, arc, label) B is a (node, arc, label) closer to the root than B along the branch containing B. The root node is ancestor to all nodes except itself.

Definition. A *successor* (node, arc, label) to (node, arc, label) B is a (node, arc, label) for which B is an ancestor. An *immediate successor* is an adjacent successor.

Definition. The *level* of a node, and its outgoing arcs and their labels, is the number of ancestor nodes from the root to this node on this branch. Thus the root node is level zero.

$$\mathcal{S} = \{\neg P, P \vee Q, \neg Q \vee R, \neg R\}$$



∗ Failure nodes

(1) failure node for $\neg P$

(2) failure node for $P \vee Q$

(3) failure node for $\neg Q \vee R$

(4) failure node for $\neg R$

Note that (5) is not a failure node

o Inference nodes for $\mathcal{S}$

(6) an inference node

Figure 1.1 Semantic tree for $\mathcal{S}$.

Definition. A *semantic tree* for $\mathcal{S}$ is a labeled binary tree where the two outgoing arcs of a node are labeled A and $\neg A$, respectively, for some $A \in \mathcal{A}$ and all labels at the same level are from the same atom.

Definition. Given node N the *initial list* $I(N)$ for node N is the list of ancestor labels to node N. This is a list of the literals that appear above node N as we draw our binary trees.

Definition. A semantic tree for $\mathcal{S}$ is *complete* iff for every branch and every atom $A \in \mathcal{A}$ either A or $\neg A$ is a label on that branch.

Definition. A *failure node for clause C* is a node N of a semantic tree for $\mathcal{S}$ with $C \in \mathcal{S}$ such that $I(N)$ falsifies C but for no ancestor node N does $I(N)$ falsify some clause in $\mathcal{S}$. Note that failure node N lies immediately "below" the last label needed to falsify clause C.

Definition. An *inference node* is a node where both immediate successor nodes are failure nodes.

Definition. A *closed semantic tree* has a failure node on every branch.

Definition. The expression *size (closed tree)* denotes the number of nodes above and including each failure node.

We now state and prove the propositional resolution completeness theorem. The resolution logic was defined and shown sound and complete by J. A. Robinson. The truly innovative part of resolution logic occurs in the first-order logic case; the propositional form had precursors that were close to this form.

Theorem 4 (Completeness Theorem: Robinson). If a clause set $\mathcal{S}$ is unsatisfiable then there exists a refutation of $\mathcal{S}$.

Proof. We give the proof preceded by an example to motivate the proof idea and to introduce the definitions. □

Figure 1.1 contains the labeled binary tree of our illustration. (It is traditional to picture mathematical trees upside down.) The reader should consult the definitions as we progress through the example.

Let our clause set $\mathcal{S}$ be given by $\mathcal{S} = \{\neg P, P \vee Q, \neg Q \vee R, \neg R\}$. A key property of the complete semantic tree for $\mathcal{S}$ is that each branch defines an interpretation of $\mathcal{S}$ by assigning T to each literal named on the branch. Moreover, each possible interpretation is represented by a single branch. Consider the interpretation where each atom is assigned T. This is represented by the leftmost branch of the tree in Figure 1.1. This branch is not a model of $\mathcal{S}$ because the clause $\neg P$ is false in this interpretation. This fact is represented by the failure node for $\neg P$ denoted by (1) that labels the node at level 1 on this branch. The number next to a failure node identifies the failed clause by position in the clause listing of clause set $\mathcal{S}$. Failure nodes have an $*$ through the node. Node (5) is not a failure node only because there is a failure node above it on the same branch; a failure node must be as close to the root as possible.

(The reader might place failure nodes on the complete semantic tree for the set $\mathcal{S}' = \{P \vee R,\ Q \vee R,\ \neg P \vee R,\ \neg Q \vee R,\ \neg R\}$ to further his/her understanding of these definitions. Here the associated complete semantic tree has more failure nodes than clauses and the same node can be a failure node for more than one clause. We assume that the semantic tree is drawn as for the example presented here, with the atom R associated with level three.)

The node (6) is an inference node because its two immediate successor nodes are failure nodes. An inference node is always associated with a resolvent clause whose parent clauses are the clauses associated with the failure nodes that immediately succeed it. Here the clauses that cause nodes (4) and (3) to be failure nodes are $\neg R$ and $\neg Q \vee R$, respectively. The resolvent of these two clauses is $\neg Q$, which is falsified by the ancestor list for node (6), i.e., $I((6))$. While here the node (6) is the failure node for the new clause, that is not always the case. The failure node of the resolvent could be an ancestor node of the inference node if the literal just above the inference node does not appear (complemented) in the resolvent.

The reader should understand the definitions and the example before proceeding from here.

Lemma. If a clause set is unsatisfiable, then the associated complete semantic tree is closed.

Proof. We prove the contrapositive. If some branch does not contain a failure node, then the literals labeling the branch define a model for that clause set. □

Lemma. Every closed semantic tree has an inference node.

Proof. Suppose that there exists a closed semantic tree without an inference node. This means that every node has at least one of its two immediate successor nodes not a failure node. If we begin at the root and always choose (one of) the immediate successor node(s) not a failure node then we reach a leaf node without encountering a failure node. This violates the closed semantic tree condition as we have found an "open" branch.

We now give the completeness argument.

Given a (finite) propositional unsatisfiable clause set $\mathcal{S}$:

(1) We have a closed complete semantic tree for $\mathcal{S}$ by the first lemma.

(2) Thus, we have an inference node in the tree for $\mathcal{S}$ by the second lemma.
(3) The inference node N_{inf} has two immediate successor nodes that are failure nodes by definition of inference node. Let A be an atom, and let C_P and C_N be clauses such that the failure node clauses are represented by $C_P \vee A$ and $C_N \vee \neg A$. That A and $\neg A$ appear in their respective clauses is assured by the fact that the failure nodes are immediate successors of the inference node and that the literal of the label immediately above the failure node must be complemented in the clause by definition of failure node.
(4) The resolution rule applies to $C_P \vee A$ and $C_N \vee \neg A$. The resolvent is $C_{Res} = C_P \vee C_N$.
(5) A failure node exists for resolvent C_{Res} at N_{inf} or an ancestor node for N_{inf} because all literals for C_{Res} are complements of literals in $I(N_{inf})$.
(6) Size (closed tree for $\mathcal{S} \cup \{C_{Res}\}$) $<$ size (closed tree for $\mathcal{S}$). This holds as at least the failure nodes below N_{inf} are missing from the closed tree for $\mathcal{S} \cup \{C_{Res}\}$.

The steps (1) through (6) are now to be repeated with $\{C_{Res}\} \cup \mathcal{S}$ as the new clause set. Each time we perform steps (1)–(6) with the new resolvent added the resulting tree is smaller.

The argument is valid for any closed semantic tree for clause set $\mathcal{S}$ such that size (closed tree for $\mathcal{S}$) $>$ 1. Size one fails because one needs failure nodes. The argument thus ends when a tree of size one is reached. The reader might consider the case $\mathcal{S} = \{P, \neg P\}$ to better understand the "end game" of the reduction argument. This has a tree of size three associated with it and leads to a tree of size one in one round. A size one tree is just the root node and has the clause $\square$ associated with it. Thus one has derived the empty clause. The proof is complete.

One can formalize the reduction argument by use of the least number principle if a more formal treatment is desired. That is left to the reader. $\square$

1.4 Handling Arbitrary Propositional Wffs

Now that we have established that the resolution logic is truthful (soundness) and can always confirm that a clause set is unsatisfiable (completeness) we need to broaden its usefulness. We show that any wff of propositional logic can be tested for unsatisfiability. This is done by giving an algorithm (i.e., a mechanical procedure) which associates with any propositional wff a surrogate wff within the resolution logic that is unsatisfiable iff the original wff is unsatisfiable.

Definition. A wff is in *conjunctive normal form* (CNF) iff the wff is a conjunction of clauses (i.e., if the wff is a wff of the resolution formal system).

Theorem 5. Given an arbitrary propositional wff F we can find a wff F_1 such that F_1 is in conjunctive normal form and F is unsatisfiable iff F_1 is unsatisfiable.

We actually will prove a stronger theorem although Theorem 5 is all we need to make F_1 a usable surrogate for F. The stronger assertion we show is that F and F_1 have the same models.

Proof. We use the Replacement Theorem repeatedly. It is left to the reader to see all replacements come from logical equivalences.

We will illustrate the sequential uses of the Replacement Theorem by an example. However, the prescription for the transformation from the original wff to the surrogate wff applies generally; the example is for illustration purposes only.

We test the following wff for tautologyhood.

$$[(P \vee (\neg Q \wedge R)) \wedge (R \to Q) \wedge (Q \vee R \vee \neg P)] \to (P \wedge Q).$$

Step 0. Negate the wff to obtain a wff to be tested for unsatisfiability.

New wff.

$$\neg\{[(P \vee (\neg Q \wedge R)) \wedge (R \to Q) \wedge (Q \vee R \vee \neg P)] \to (P \wedge Q)\}.$$

Step 1. Eliminate $\to$ and $\leftrightarrow$.
(Replacement Theorem direction: $\longrightarrow$)

Use
$$(A \to B) \longrightarrow \neg A \vee B,$$
$$(A \leftrightarrow B) \longrightarrow (A \vee \neg B) \wedge (\neg A \vee B),$$
$$\neg(A \to B) \longrightarrow A \wedge \neg B.$$

New wff.

$$(P \vee (\neg Q \wedge R)) \wedge (\neg R \vee Q) \wedge (Q \vee R \vee \neg P) \wedge \neg(P \wedge Q).$$

Step 2. Move $\neg$ inward as far as possible.

$$\text{Use} \quad \begin{aligned} \neg(A \wedge B) &\longrightarrow \neg A \vee \neg B, \\ \neg(A \vee B) &\longrightarrow \neg A \wedge \neg B, \\ \neg\neg A &\longrightarrow A. \end{aligned}$$

New wff.

$$(P \vee (\neg Q \wedge R)) \wedge (\neg R \vee Q) \wedge (Q \vee R \vee \neg P) \wedge (\neg P \vee \neg Q).$$

Step 3. Place the wff in conjunctive normal form (CNF).

$$\text{Use} \quad \begin{aligned} A \vee (B \wedge C) &\longrightarrow (A \vee B) \wedge (A \vee C), \\ (B \wedge C) \vee A &\longrightarrow (B \vee A) \wedge (C \vee A), \\ (A \wedge B) \vee (A \wedge C) &\longrightarrow A \wedge (B \vee C), \\ (B \wedge A) \vee (C \wedge A) &\longrightarrow (B \vee C) \wedge A. \end{aligned}$$

New wff.

$$(P \vee \neg Q) \wedge (P \vee R) \wedge (\neg R \vee Q) \wedge (Q \vee R \vee \neg P) \wedge (\neg P \vee \neg Q).$$

We now have the new wff with the same models as the given wff and the new wff is in conjunctive normal form (CNF). It is common for users of resolution logic to be imprecise and say "the given wff is now in conjunctive normal form."

The theorem is proven. □

We list the clauses that come from the CNF wff generated in the theorem.

Clauses:

1. $P \vee \neg Q$
2. $P \vee R$
3. $\neg R \vee Q$
4. $Q \vee R \vee \neg P$
5. $\neg P \vee \neg Q$

We leave to the reader the exercise of determining if this clause set, and hence the given wff, is unsatisfiable.

Finally, we might mention that resolution is not used in the fastest algorithms for testing if a propositional wff is unsatisfiable. Some of the fastest algorithms for unsatisfiability testing are based on the DPLL (Davis-Putnam-Logemann-Loveland) procedure. These algorithms are used in many applications, such as testing "correctness" of computer chips. An interested reader can find more on this procedure and its applications by entering "DPLL" in an Internet search engine. Note that

DPLL is usable only at the propositional level, whereas resolution is applicable also at the first-order logic level.

Exercises

1. Give a formal representation of the following sentence using as few statement letters as possible. Give the intended meaning of each statement letter.

   If John owes one dollar to Steve and Steve owes one dollar to Tom then if Tom owes one dollar to John we have that John doesn't owe a dollar to Steve, Steve doesn't owe a dollar to Tom nor does Tom owe a dollar to John.

2. Give a formal representation of the following sentence using the statement letters given.

   If it doesn't rain, I swim and so get wet, but if it rains, I get wet so in any event I get wet.

   Use: R: it rains
   S: I swim
   W: I get wet

3. Give a resolution refutation of the following clause set.

$$\neg A,\ B \vee C,\ \neg B \vee C,\ B \vee \neg C,\ C \vee D,\ \neg B \vee \neg C \vee D,\ A \vee \neg B \vee \neg C \vee \neg D.$$

4. Give a resolution refutation of the following clause set.

$$P \vee Q,\ \ P \vee \neg Q,\ \ R \vee S,\ \ \neg R \vee \neg S,\ \ \neg P \vee R \vee \neg S,\ \ \neg P \vee \neg R \vee S.$$

5. Give a resolution refutation of the following clause set. It is not necessary that all clauses be used in a refutation.

$$M \vee P,\ P \vee Q,\ \neg P \vee R \vee S,\ \neg M \vee Q \vee \neg S,\ \neg M \vee R,\ \neg Q \vee R \vee S,$$
$$\neg M \vee \neg S,\ \neg Q \vee \neg R,\ \neg P \vee \neg R,\ M \vee \neg R,\ M \vee \neg P \vee R \vee \neg S.$$

6. The following clause set is satisfiable. Find a model of the clause set. How can you use the lemma in the Soundness Theorem to help locate models of the clause set?

$$P \vee Q \vee R,\ \neg Q \vee R,\ \neg P \vee \neg R,\ \neg P \vee Q,\ P \vee \neg Q \vee \neg R.$$

7. Show that the following wff is a tautology using resolution.

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

8. Show that the following wff is a tautology using resolution.

$$(P \vee Q) \wedge (P \rightarrow R) \rightarrow \neg Q \rightarrow R.$$

9. Determine if the following wff is a tautology. If it is a tautology give a resolution proof of its negation. If it is not a tautology show that its negation is satisfiable.

$$(S \vee \neg(P \rightarrow R)) \rightarrow ((S \wedge (R \rightarrow P)) \vee \neg(S \vee \neg P)).$$

10. Provide the proof for the "if" case for the FACT given in Section 1.1. Argue by the contrapositive.

11. Consider an unsatisfiable clause set with a unit (one-literal) clause. Consider a new clause set obtained from the old set by resolving the unit clause against every possible clause and then removing all the θ-subsumed clauses and the unit clause. Prove that the resulting clause set is still unsatisfiable. Hint: Suppose that the new clause set is satisfiable.

12. (Generalization of Exercise 11) Consider an unsatisfiable clause set with an arbitrary clause C. Choose an arbitrary literal L in C and create a new clause set obtained from the old set by resolving L against every occurrence of L^c in the clause set and then removing all the θ-subsumed clauses and the clause C. Prove that the resulting clause set is still unsatisfiable. Hint: Solve Exercise 11 first.

13. (Harder) The *unit resolution* restriction uses the resolution inference rule that requires a unit clause as one parent clause. This is not a complete resolution deductive system over all of propositional logic. (Why?) However, consider an unsatisfiable clause set for which no clause has more than one positive literal. Show that this clause set has a unit resolution refutation. Hint: First prove the preceding exercise.