1 Propositional logic

1.1 What is propositional logic?

In propositional logic, atomic formulas are propositions. Any assertion will do. For example,

A = "Aristotle is dead,"

B = "Barcelona is on the Seine," and

C = "Courtney Love is tall"

are atomic formulas. Atomic formulas are the building blocks used to construct sentences. In any logic, a sentence is regarded as a particular type of formula. In propositional logic, there is no distinction between these two terms. We use "formula" and "sentence" interchangeably.

In propositional logic, as with all logics we study, each sentence is either true or false. A *truth value* of 1 or 0 is assigned to the sentence accordingly. In the above example, we may assign truth value 1 to formula A and truth value 0 to formula B. If we take proposition C literally, then its truth is debatable. Perhaps it would make more sense to allow truth values between 0 and 1. We could assign 0.75 to statement C if Miss Love is taller than 75% of American women. Fuzzy logic allows such truth values, but the classical logics we study do not.

In fact, the content of the propositions is not relevant to propositional logic. Henceforth, atomic formulas are denoted only by the capital letters A, B, C,... (possibly with subscripts) without referring to what these propositions actually say. The veracity of these formulas does not concern us. Propositional logic is not the study of truth, but of the relationship between the truth of one statement and that of another.

The language of propositional logic contains words for "not," "and," "or," "implies," and "if and only if." These words are represented by symbols:

$$\neg \text{ for "not," } \wedge \text{ for "and," } \vee \text{ for "or,"}$$

$$\rightarrow \text{ for "implies," and } \leftrightarrow \text{ for "if and only if."}$$

As is always the case when translating one language into another, this correspondence is not exact. Unlike their English counterparts, these symbols represent concepts that are precise and invariable. The meaning of an English word, on the other hand, always depends on the context. For example, $\wedge$ represents a concept that is similar but not identical to "and." For atomic formulas A and B, $A \wedge B$ always means the same as $B \wedge A$. This is not always true of the word "and." The sentence

She became violently sick and she went to the doctor.

does not have the same meaning as

She went to the doctor and she became violently sick.

Likewise $\vee$ differs from "or." Conversationally, the use of "A or B" often precludes the possibility of both A and B. In propositional logic $A \vee B$ always means either A or B or both A and B.

We must precisely define the symbols $\neg$, $\wedge$, $\vee$, $\rightarrow$, and $\leftrightarrow$. We are confronted with the conundrum of how to define the first word of a language (having recourse to no other words!). For this reason, we take the symbols $\neg$ and $\wedge$ as *primitives*. We define the other symbols in terms of these two symbols. Although we do not define $\neg$ and $\wedge$ in terms of the other symbols, we do describe the semantics of these symbols in an unambiguous manner.

Before describing the semantics of the language, we discuss the syntax. Whereas the semantics regards the meaning, or interpretation, of sentences in the language, the syntax regards the grammar of the language. The syntax of propositional logic tells us which strings of symbols are permissible as formulas. Naturally, any atomic formula is a formula. We also have the following two rules.

(R1) If F is a formula, then $\neg F$ is a formula.
(R2) If F and G are formulas, then $(F \wedge G)$ is a formula.

Definition 1.1 The formula $\neg F$ is the *negation* of F
and the formula $(F \wedge G)$ is the *conjunction* of F and G.

Definition 1.2 A finite string of symbols is a *formula* of propositional logic if and only if it is built up from atomic formulas by repeated application of rules (R1) and (R2).

Example 1.3 $\neg(\neg(A \wedge B) \wedge \neg C)$ is a formula and $((A\neg\wedge)B(C\neg$ is not.

Note that we have restricted the definition of formula to the primitive symbols $\neg$ and $\wedge$. If we were describing the syntax of propositional logic to a computer, then this definition of formula would suffice. However, to make formulas more palatable to humans, we include the other symbols ($\vee$, $\rightarrow$, and $\leftrightarrow$) to be defined later. We may regard formulas involving these symbols as abbreviations for more complicated formulas involving only $\neg$ and $\wedge$. The inclusion of these symbols make the formulas easier (for us humans) to read.

Also toward the aim of readability, we employ certain conventions. The use of these abbreviations and conventions alters our notion of "formula" somewhat. One of these conventions is the following:

(C1) If F or (F) is a formula, then we view F and (F) as the same formula.

That is, we may drop the outermost parentheses. This extends our definition of formula. Technically, by the above definition, $A \wedge B$ is not a formula. However, using convention (C1), we do not distinguish $A \wedge B$ from the formula $(A \wedge B)$.

The use of convention (C1) leads to some ambiguities that we presently address. Suppose that, in (R1), F denotes $A \wedge B$ (which, by (C1) is a formula). Then $\neg F$ does not represent the formula $\neg A \wedge B$. Rather, $\neg F$ denotes the formula $\neg(A \wedge B)$. As we shall see, $\neg A \wedge B$ and $\neg(A \wedge B)$ do not mean the same thing. Likewise, $F \wedge G$ denotes the formula $(F) \wedge (G)$.

The use of (C1) also requires care in defining the notion of "subformula." A subformula of a formula F (viewed as a string of symbols) is a substring of F that is itself a formula. However, because of (C1), not every such substring is a subformula. So we do not want to take this property as the definition of "subformula." Instead, we define "subformula" as follows.

Definition 1.4 The following rules define the *subformulas* of a formula.

Any formula is a subformula of itself.
Any subformula of F is also a subformula of $\neg F$.
Any subformula of F or G is also a subformula of $(F \wedge G)$.

Example 1.5 Let A and B be atomic and let F be the formula $\neg(\neg A \wedge \neg B)$.

The formula $A \wedge \neg B$ occurs as a substring of F, but it is not a subformula of F. There is no way to build the formula F from the formula $A \wedge \neg B$. The subformulas of F are A, B, $\neg A$, $\neg B$, $(\neg A \wedge \neg B)$, and $\neg(\neg A \wedge \neg B)$.

Having described the syntax of propositional logic, we now describe the semantics. That is, we say how to interpret the formulas. Not only must we describe the semantics for the symbols "$\wedge$" and "$\neg$," but we must also say how to interpret formulas in which these symbols occur together. For this we state the *order of operations*. It is the role of parentheses to dictate which subformulas are to be considered first when interpreting a formula. If no parentheses are present, then we use the following rule:

$$\neg \text{ has priority over } \wedge .$$

For example, the formula $\neg(A \wedge B)$ means "not both A and B." The parentheses tell us that the "$\wedge$" in this formula has priority over "$\neg$." The formula $\neg A \wedge B$, on the other hand, has a different interpretation. In the absence of parentheses, we use the rule that $\neg$ has priority over $\wedge$. So this formula means "both not A and B."

The semantics of propositional logic is defined by this rule along with Tables 1.1 and 1.2.

These are examples of *truth tables.* Each row of these tables assigns truth values to atomic formulas and gives resulting truth values for more complex formulas. For example, the third row of Table 1.1 tells us that if A is true and B is false, then $(A \wedge B)$ is false. We see that $(A \wedge B)$ has truth value 1 only if both A and B have truth value 1, corresponding to our notion of "and." Likewise, Table 1.2 tells us that $\neg A$ has the opposite truth value of A, corresponding to our notion of negation.

Using these two truth tables, we can find truth tables for any formula. This is because every formula is built from atomic formulas via rules (R1) and (R2). Suppose, for example, we want to find a truth table for the formula $\neg(\neg A \wedge \neg B)$. Given truth values for A and B, we can use Table 1.2 to find the truth values of $\neg A$ and $\neg B$. Given truth values for $\neg A$ and $\neg B$, we can then use Table 1.1 to find the truth value of $(\neg A \wedge \neg B)$. Finally, we can refer again to Table 1.2 to find the truth value of $\neg(\neg A \wedge \neg B)$. The resulting truth table for this formula is shown in Table 1.3.

Note that the formulas listed across the top of Table 1.3 are precisely the subformulas from Example 1.5. From this table we see that the formula $\neg(\neg A \wedge \neg B)$ has truth value 1 if and only if A or B has truth value 1. This formula corresponds to the notion of "or" discussed earlier. The symbol $\vee$ is used to denote this useful notion.

Table 1.1 Truth table for $A \wedge B$

A	B	$(A \wedge B)$
0	0	0
0	1	0
1	0	0
1	1	1

Table 1.2 Truth table for $\neg A$

A	$\neg A$
0	1
1	0

Table 1.3 Truth table for $(A \vee B)$

A	B	$\neg A$	$\neg B$	$(\neg A \wedge \neg B)$	$\neg(\neg A \wedge \neg B)$
0	0	1	1	1	0
0	1	1	0	0	1
1	0	0	1	0	1
1	1	0	0	0	1

Table 1.4 Truth table for $(A \rightarrow B)$

A	B	$\neg A$	$(B \vee \neg A)$	$(A \rightarrow B)$
0	0	1	1	1
0	1	1	1	1
1	0	0	0	0
1	1	0	1	1

Definition 1.6 The symbol $\vee$ is defined as follows: for any formulas F and G, $(F \vee G)$ is an abbreviation for $\neg(\neg F \wedge \neg G)$. The formula $(F \vee G)$ is called the *disjunction* of F and G.

Two other abbreviations that are convenient are the following.

Definition 1.7 The symbols $\rightarrow$ and $\leftrightarrow$ are defined as follows:

$(F \rightarrow G)$ abbreviates $(G \vee \neg F)$, and
$(F \leftrightarrow G)$ abbreviates $((F \rightarrow G) \wedge (G \rightarrow F))$.

We previously remarked that the symbol $\rightarrow$ corresponds to the English word "implies" and the symbol $\leftrightarrow$ corresponds to the phrase "if and only if." Again, these correspondences are merely mnemonic devices for the semantics of the symbols. For example $(A \leftrightarrow B)$ is true if A and B have the same truth values and is otherwise false. So $\leftrightarrow$ behaves exactly like the phrase "if and only if." The relationship between $\rightarrow$ and "implies" is a bit tenuous. Consider the truth table for $(A \rightarrow B)$ (Table 1.4).

We see that $(A \rightarrow B)$ is true unless A is true and B is false. In particular, $(A \rightarrow B)$ is true whenever A is false. Thus, in logic, a false statement implies anything. This differs from the colloquial use of the word "implies." We would not say "Barcelona is on the Seine implies Aristotle is dead" or, even more egregious, "Barcelona is on the Seine implies Barcelona is not on the Seine." However, $(A \rightarrow B)$ and $(A \rightarrow \neg A)$ are true statements of propositional logic whenever A is false.

Having introduced new symbols, we must determine the order of operation for these symbols. When evaluating the truth value of a formula, we must know the order in which to proceed. Rather than ranking all of the symbols in a hierarchy, we state just one rule:

$\neg$ has priority over $\wedge$, $\vee$, $\rightarrow$, and $\leftrightarrow$.

Beyond this, the parentheses dictate the order in which to proceed.

Example 1.8 Consider the formula $((\neg A \rightarrow B)\wedge C)\vee\neg(A\wedge D)$. Call this formula F. Suppose we know that the truth values for A, B, C, and D are 1, 0, 1, and 0, respectively. To evaluate the truth value for F we begin with the subformula $\neg A$ (using Table 1.2) since $\neg$ has priority. We next evaluate the subformula $(\neg A \rightarrow B)$ (Table 1.4) which is in the innermost set of parentheses. We next evaluate the truth values for $((\neg A \rightarrow B) \wedge C)$ and $(A \wedge D)$ (Table 1.1). We then find the truth values for $\neg(A \wedge D)$ (Table 1.2) and, finally, for F (Table 1.3). We obtain the following truth values.

A	B	C	D	$\neg A$	$(\neg A \rightarrow B)$	$((\neg A \rightarrow B) \wedge C)$	$(A \wedge D)$	$\neg(A \wedge D)$	F
1	0	1	0	0	1	1	0	1	1

This is just one row of a truth table for this formula. We could choose other truth values for A, B, C, and D other than 1, 0, 1, and 0. Since there are two possible values for each of these four atomic formulas, there are $2^4 = 16$ ways to assign truth values to A, B, C, and D. So the full table has 16 rows. The completion of this table is left as Exercise 1.3(d).

The role of parentheses is not only to determine the order of operations, but also to make formulas more readable. Toward this aim, we omit parentheses when they are not necessary. We have already discussed convention (C1) that allows us to drop the outermost parentheses from the formula (F). We also use the following convention:

(C2) For any formulas F, G, and H,
we view $F \wedge G \wedge H$ as the same formula as $(F \wedge G) \wedge H$
and $F \vee G \vee H$ as the same formula as $(F \vee G) \vee H$.

Since the formulas $(F \wedge G) \wedge H$ and $F \wedge (G \wedge H)$ have the same truth tables, there is no ambiguity in dropping the parentheses and simply writing $F \wedge G \wedge H$. In contrast, $F \wedge G \vee H$ is ambiguous and is not permitted as a formula of propositional logic. The formulas $(F \wedge G) \vee H$ and $F \wedge (G \vee H)$ do not have the same truth tables.

We have now completely defined propositional logic.

In summary, propositional logic, like any logic, is a language. Its dictionary contains the words $\neg$, $\wedge$, $\vee$, $\rightarrow$, and $\leftrightarrow$. (The symbols "(" and ")" are used only as punctuation.) The words $\vee$, $\rightarrow$, and $\leftrightarrow$ are defined in terms of $\neg$ and $\wedge$. The words $\neg$ and $\wedge$ are considered primitive and are listed in our hypothetical dictionary without definition. The dictionary also contains infinitely many atomic formulas that are merely listed as capital letters (with subscripts, perhaps). The grammar of this language consists of the rules (R1) and (R2) along with conventions (C1) and (C2) regarding parentheses.

Propositional logic, like any logic, also has rules for deduction. These rules follow from the semantics of the logic. The semantics of propositional logic are summarized by Tables 1.1 and 1.2 and the definitions of the symbols $\vee$, $\rightarrow$, and $\leftrightarrow$. The semantics and the rules for deduction that follow from the semantics are implicit in the words "not," "and," "or," "implies," and "if and only if" (although this correspondence is not exact). For example, if $(A \wedge B)$ is true (truth value 1), then we can deduce that both A and B are true. And if $A \rightarrow B$ and A both have truth value 1, then it follows that B also has truth value 1. We discuss these and other rules for deduction in Section 1.5.

1.2 Validity, satisfiability, and contradiction

Let $\mathcal{S} = \{A_1, \ldots, A_n\}$ be a set of atomic formulas. Let $\mathcal{F}(\mathcal{S})$ be the set of all formulas that can be built from the atomic formulas in $\mathcal{S}$.

Definition 1.9 An *assignment* of $\mathcal{S}$ is a function $\mathcal{A} : \mathcal{S} \rightarrow \{0, 1\}$.

That is, an assignment of $\mathcal{S}$ assigns truth values to each atomic formula in $\mathcal{S}$. An assignment $\mathcal{A}$ of $\mathcal{S}$ naturally extends to all of $\mathcal{F}(\mathcal{S})$. Given any formula F in $\mathcal{F}(\mathcal{S})$, an assignment $\mathcal{A}$ of $\mathcal{S}$ corresponds to a unique row of the truth table for F. We define $\mathcal{A}(F)$ to be the truth value of F in this row.

An assignment $\mathcal{A}$ of $\mathcal{S}$ also extends to certain formulas not in $\mathcal{F}(\mathcal{S})$. Suppose F_0 is a formula that is not in $\mathcal{F}(\mathcal{S})$. Let $\mathcal{S}_0$ be the set of atomic subformulas of F_0. If every extension of $\mathcal{A}$ to $\mathcal{S} \cup \mathcal{S}_0$ has the same value for F_0, then we define $\mathcal{A}(F_0)$ to be this value.

Example 1.10 Let A and B be atomic formulas. Let $\mathcal{A}$ be the assignment of $\{A, B\}$ defined by $\mathcal{A}(A) = 1$ and $\mathcal{A}(B) = 0$. Then

$$\mathcal{A}(A \wedge B) = 0,$$

$$\mathcal{A}(A \vee B) = 1,$$

$$\mathcal{A}(A \wedge (C \vee \neg C)) = 1, \text{and}$$

$$\mathcal{A}(B \vee (C \wedge \neg C)) = 0.$$

The reason $\mathcal{A}(A \wedge (C \vee \neg C)) = 1$ is that $\mathcal{A}(A) = 1$ and, no matter what truth value we assign to C, $(C \vee \neg C)$ has truth value 1. Likewise $\mathcal{A}(B \vee (C \wedge \neg C)) = 0$ because both B and $(C \wedge \neg C)$ have truth value 0, regardless of the truth value of C.

Let $\mathcal{A}$ be an assignment of $\mathcal{S}$ and let F be a formula. If $\mathcal{A}(F) = 1$, then we say F *holds* under assignment $\mathcal{A}$. Equivalently, we say $\mathcal{A}$ *models* F. We write $\mathcal{A} \models F$ to denote this concept.

Definition 1.11 A formula is *valid* if it holds under every assignment. We use $\models F$ to denote this. A valid formula is called a *tautology.*

Example 1.12 The formula $(C \vee \neg C)$ from the previous example is a tautology.

Definition 1.13 A formula is *satisfiable* if it holds under some assignment.

Definition 1.14 A formula is *unsatisfiable* if it holds under no assignment. An unsatisfiable formula is called a *contradiction.*

Example 1.15 The formula $(C \wedge \neg C)$ is a contradiction.

Suppose that we want to determine whether or not a given formula is valid. This is an example of a *decision problem.* A decision problem is any problem that, given certain input, asks a question to be answered with a "yes" or a "no." Given formula F as input, we may ask "Is F valid?" We refer to this as the *validity problem.* Likewise, we may ask "Is F satisfiable?," and refer to this the *satisfiability problem.* For propositional logic, truth tables provide a systematic approach for resolving such decision problems. If all of the truth values for F are 1s, then F is valid. If some truth value is 1, then F is satisfiable. Otherwise, if no truth values are 1s, F is unsatisfiable.

Example 1.16 Consider the formula $(A \wedge (A \rightarrow B)) \rightarrow B$. To determine whether this formula is satisfiable, we compute the following truth table.

A	B	$A \rightarrow B$	$A \wedge (A \rightarrow B)$	$(A \wedge (A \rightarrow B)) \rightarrow B$
0	0	1	0	1
0	1	1	0	1
1	0	0	0	1
1	1	1	1	1

We see that $(A \wedge (A \rightarrow B)) \rightarrow B$ has truth value 1 under any assignment. So not only is this formula satisfiable, it is valid.

Example 1.17 Consider now the formula $((A \to B) \to A) \wedge \neg A$. Suppose we want to determine whether this formula is satisfiable or not. Again we compute a truth table.

A	B	$(A \to B)$	$((A \to B) \to A)$	$\neg A$	$((A \to B) \to A) \wedge \neg A$
0	0	1	0	1	0
0	1	1	0	1	0
1	0	0	1	0	0
1	1	1	1	0	0

This formula is unsatisfiable. It is a contradiction.

Theoretically, we can determine whether any formula F is valid, satisfiable or unsatisfiable by looking at a truth table. Unfortunately, this is not always an efficient method. If F contains n atomic formulas, then there are 2^n rows to compute in the truth table for F. So if F happens to have, say, 23 atomic formulas, then computing a truth table is not feasible. One of our aims in this chapter is to find alternative methods for resolving the validity and satisfiability problems that avoid truth tables. More generally, our aim is to contrive various ways of determining whether or not a given formula is a consequence of a given set of formulas. This is a central problem of any logic.

1.3 Consequence and equivalence

We now introduce the fundamental notion of consequence. First, we define what it means for one formula to be a consequence of another. Later in this section, we similarly define what it means for a formula to be a consequence of a *set* of formulas.

Definition 1.18 Formula G is a *consequence* of formula F if for every assignment $\mathcal{A}$, if $\mathcal{A} \models F$ then $\mathcal{A} \models G$. We denote this by $F \models G$.

Note that the symbol $\models$ is used in a variety of ways. There is always a formula to the right of this symbol. When we write $\_\_ \models F$, the interpretation of "$\models$" depends on how we fill in the blank. The blank may either be filled with an assignment $\mathcal{A}$, a formula G, or not filled with the empty set. The three corresponding interpretations for $\models$ are as follows:

- $\mathcal{A} \models F$ means that $\mathcal{A}(F) = 1$. We read this as "$\mathcal{A}$ models F."
- $G \models F$ means every assignment that models G also models F. That is, F is a consequence of G.