Chapter 2
Propositional Logic

§2.1 Introduction

Propositional Logic is concerned with propositions and their interrelationships. The notion of a proposition here cannot be defined precisely. Roughly speaking, a *proposition* is a possible "condition'" of the world about which we want to say something. The condition need not be true in order for us to talk about it. In fact, we might want to say that it is false or that it is true if some other proposition is true.

In this chapter, we first look at the syntactic rules for the language of Propositional Logic. We then look at semantic interpretation for the expressions specified by these rules. Given this semantics, we define the concept of propositional entailment, which identifies for us, at least in principle, all of the logical conclusions one can draw from any set of propositional sentences.

§2.2 Syntax

In Propositional Logic, there are two types of sentences -- simple sentences and compound sentences. Simple sentences express ``atomic" propositions about the world. Compound sentences express logical relationships between the simpler sentences of which they are composed.

Simple sentences in Propositional Logic are often called *propositional constants* or, sometimes, *logical constants*. In what follows, we refer to a logical constant using a sequence of alphanumeric characters beginning with a lower case character. For example, *raining* is a logical constant, as are *rAiNiNg* and *r32aining. Raining* is not a logical constant because it begins with an upper case character. 324567 fails because it begins with a number. *raining-or-snowing* fails because it contains non-alphanumeric characters.

Compound sentences are formed from simpler sentences and express relationships among the constituent sentences. There are six types of compound sentences, viz. negations, conjunctions, disjunctions, implications, reductions, and equivalences.

A *negation* consists of the negation operator $\neg$ and a simple or compound sentence, called the *target*. For example, given the sentence p, we can form the negation of p as shown below.

$$\neg p$$

A *conjunction* is a sequence of sentences separated by occurrences of the $\wedge$ operator and enclosed in parentheses, as shown below. The constituent sentences are called *conjuncts*. For example, we can form the conjunction of p and q as follows.

$$(p \wedge q)$$

A *disjunction* is a sequence of sentences separated by occurrences of the $\vee$ operator and enclosed in parentheses. The constituent sentences are called *disjuncts*. For example, we can form the disjunction of p and q as follows.

$$(p \vee q)$$

An *implication* consists of a pair of sentences separated by the $\Rightarrow$ operator and enclosed in parentheses. The sentence to the left of the operator is called the *antecedent*, and the sentence to the right is called the *consequent*. The implication of p and q is shown below.

$$(p \Rightarrow q)$$

A *reduction* is the reverse of an implication. It consists of a pair of sentences separated by the $\Leftarrow$ operator and enclosed in parentheses. The sentence to the left of the operator is called the *consequent*, and the sentence to the right is called the *antecedent*. The reduction of p to q is shown below.

$$(p \Leftarrow q)$$

An *equivalence* is a combination of an implication and a reduction. For example, we can express the equivalence of p and q as shown below.

$$(p \Leftrightarrow q)$$

Note that the constituent sentences within any compound sentence can be either simple sentences or compound sentences or a mixture of the two. For example, the following is a legal compound sentence.

$$((p \vee q) \Rightarrow \neg r)$$

One disadvantage of our notation, as written, is that the parentheses tend to build up and need to be matched correctly. It would be nice if we could dispense with parentheses, e.g. simplifying the preceding sentence to the one shown below.

$$p \vee q \Rightarrow \neg r$$

Unfortunately, we cannot do without parentheses entirely, since then we would be unable to render certain sentences unambiguously. For example, the sentence shown above could have resulted from dropping parentheses from either of the following sentences.

$$((p \vee q) \Rightarrow \neg r)$$
$$(p \vee (q \Rightarrow \neg r))$$

The solution to this problem is the use of *operator precedence*. The following table gives a hierarchy of precedences for our operators. The $\neg$ operator has higher precedence than $\wedge$, $\wedge$ has higher precedence than $\vee$,and $\vee$ has higher precedence than $\Rightarrow$, $\Leftarrow$, and $\Leftrightarrow$.

$$\begin{array}{c} \neg \\ \wedge \\ \vee \\ \Leftarrow \Leftrightarrow \Rightarrow \end{array}$$

In unparenthesized sentences, it is often the case that an expression is flanked by operators, one on either side. In interpreting such sentences, the question is whether the operator associates with the operator on its left or the one on its right. We can use precedence to make this determination. In particular, we agree that an operand in such a situation always associates with the operator of higher precedence. When an operand is surrounded by operators of equal precedence, the operand associates to the right. The following examples show how these rules work in various cases. The expressions on the right are the fully parenthesized versions of the expressions on the left.

$\neg p \wedge q$	$(\neg p \wedge q)$
$p \wedge \neg q$	$(p \wedge \neg q)$
$p \wedge q \vee r$	$((p \wedge q) \vee r)$
$p \vee q \wedge r$	$(p \vee (q \wedge r))$
$p \Rightarrow q \Rightarrow r$	$(p \Rightarrow (q \Rightarrow r))$
$p \Rightarrow q \Leftarrow r$	$(p \Rightarrow (q \Leftarrow r))$

Note that just because precedence allows us to delete parentheses in some case does not mean that we can dispense with parentheses entirely. Consider the example shown above. Precedence eliminates the ambiguity by dictating that the unparenthesized sentence is an implication with a disjunction as antecedent. However, this makes for a problem for those cases when we want to express a disjunction with an implication as a disjunct. In such cases, we must retain at least one pair of parentheses.

§2.3 Semantics

The semantics of logic is similar to the semantics of algebra. Algebra is unconcerned with the real-world meaning of variables like *x* and *y*. What is interesting is the relationship between the variables expressed in the equations we write; and algebraic methods are designed to respect these relationships, no matter what meanings or values are assigned to the constituent variables.

In a similar way, logic itself is unconcerned with what sentences say about the world being described. What is interesting is the relationship between the truth of simple sentences and the truth of compound sentences within which the simple sentences are

contained. Also, logical reasoning methods are designed to work no matter what meanings or values are assigned to the logical "variables" used in sentences.

Although the values assigned to variables are not crucial in the sense just described, in talking *about* logic itself, it is sometimes useful to make these assignments explicit and to consider various assignments or all assignments and so forth. Such an assignment is called an interpretation.

Formally, an *interpretation* for propositional logic is a mapping assigning a truth value to each of the simple sentences of the language. In what follows, we refer to the meaning of a constant or expression under an interpretation *i* by superscripting the constant or expression with *i* as a superscipt.

The assignment shown below is an example for the case of a logical language with just three propositional constants, viz. *p*, *q*, and *r*.

$$p_i = true$$
$$q_i = false$$
$$r_i = true$$

The following assignment is another interpretation for the same language.

$$p_j = false$$
$$q_j = false$$
$$r_j = true$$

Note that the expressions above are not themselves sentences in Propositional Logic. Propositional Logic does not allow superscripts and does not use the = symbol. Rather, these are informal, metalevel statements *about* particular interpretations. Although talking about propositional logic using a notation similar to that propositional logic can sometimes be confusing, it allows us to convey meta-information precisely and efficiently. To minimize problems, in this book we use such meta-notation infrequently and only when there is little chance of confusion.

Looking at the preceding interpretations, it is important to bear in mind that, as far as logic is concerned, any interpretation is as good as any other. It does not directly fix the interpretation of individual logical constants.

On the other hand, *given* an interpretation for the logical constants of a language, logic *does* fix the interpretation for all compound sentences in that language. In fact, it is possible to determine the truth value of a compound sentence by repeatedly applying the following rules.

1. If the truth value of a sentence is *true* in an interpretation, the truth value of its negation is *false*. If the truth value of a sentence is *false*, the truth value of its negation is *true.*

2. The truth value of a conjunction is *true* under an interpretation if and only if the truth value of its conjuncts are both *true*; otherwise, the truth value is *false.*

3. The truth value of a disjunction is *true* if and only if the truth value of at least one its conjuncts is *true*; otherwise, the truth value is *false*. Note that this is the *inclusive or* interpretation of the $\vee$ operator and is differentiated from *exclusive or* in which a disjunction is true if and only if an odd number of its disjuncts are false.

4. The truth value of an implication is *false* if and only if its antecedent is *true* and is consequent is *false*; otherwise, the truth value is *true.* This is called *material implication*.

5. As with an implication, the truth value of a reduction is *false* if and only if its antecedent is *true* and is consequent is *false*; otherwise, the truth value is *true.* Of course, it is important to remember that in a reduction the antecedent and consequent are reversed.

6. An equivalence is *true* if and only if the truth values of its constituents agree, i.e. they are either both *true* or both *false*.

We say that an interpretation *i satisfies* a sentence if and only if it is *true* under that interpretation.

§2.4 Evaluation

Given the semantic definitions in the last section, we can easily determine for any given interpretation whether or not any sentence is *true* or *false* under that interpretation. The technique is simple. We substitute true and false values for the propositional constants and replace complex expressions with the corresponding values, working from the inside out.

As an example, consider the interpretation *i* show below.

$$p_i = true$$
$$q_i = false$$
$$r_i = true$$

We can see that *i* satisfies $(p \vee q) \wedge (\neg q \vee r)$. (For the sake of space and clarity, we use 1 for *true* and 0 for *false*.)

$$(p \vee q) \wedge (\neg q \vee r)$$
$$(true \vee false) \wedge (\neg false \vee true)$$
$$true \wedge (\neg false \vee true)$$
$$true \wedge (true \vee true)$$
$$true \wedge true$$
$$true$$

Now consider interpretation *j* defined as follows.

$$p_i = true$$
$$q_i = true$$
$$r_i = false$$

In this case, *j* does not satisfy $(p \vee q) \wedge (\neg q \vee r)$.

$$(p \vee q) \wedge (\neg q \vee r)$$
$$(true \vee true) \wedge (\neg true \vee false)$$
$$true \wedge (\neg true \vee false)$$
$$true \wedge (false \vee false)$$
$$true \wedge false$$
$$false$$

Using this technique, we can evaluate the truth of arbitrary sentences in our language. The cost is proportional to the size of the sentence.

§2.5 Reverse Evaluation

Reverse evaluation is the opposite of evaluation. We begin with one or more compound sentences and try to figure out which interpretations satisfy those sentences.

One way to do this is using a truth table for the language. A *truth table* for a propositional language is a table showing all of the possible interpretations for the propositional constants in the language.

The following figure shows a truth table for a propositional language with just three propositional constants. Each row corresponds to a single interpretation. The interpretations *i* and *j* correspond to the third and seventh rows of this table, respectively.

p	*q*	*r*
true	*true*	*true*
true	*true*	*false*
true	*false*	*true*
true	*false*	*false*
false	*true*	*true*
false	*true*	*false*
false	*false*	*true*
false	*false*	*false*

Note that, for a propositional language with *n* logical constants, there are *n* columns in the truth tables 2^n rows.

In doing reverse evaluation, we process input sentences in turn, for each sentence crossing out interpretations in the truth table that do not satisfy the sentence. The interpretations remaining at the end of this process are all possible interpretations of the input sentences.

§2.6 Validity, Satisfiability, Unsatisfiability

Evaluation and reverse evaluation are processes that involve specific sentences and specific interpretations. In Computational Logic, we are rarely concerned with specific interpretations; we are more interested in the properties of sentences that hold across interpretations. In particular, the notion of satisfaction imposes a classification of sentences in a language based on whether there are interpretations that satisfy that sentence.

A sentence is *valid* if and only if it is satisfied by *every* interpretation. The following sentence is valid.

$$p \vee \neg p$$

A sentence is *satisfiable* if and only if it is satisfied by at least one interpretation. We have already seen several examples of satisfiable sentences.

A sentence is *unsatisfiable* if and only if it is not satisfied by any interpretation. The following sentence is unsatisfiable. No matter what interpretation we take, the sentence is always false.

$$p \Leftrightarrow \neg p$$

In one sense, valid sentences and unsatisfiable sentences are useless. Valid sentences do not rule out any possible interpretations; unsatisfiable sentences rule out all interpretations; thus they say nothing about the world. On the other hand, from a logical perspective, they are extremely useful in that, as we shall see, they serve as the basis for legal transformations that we can perform on other logical sentences.

Note that we can easily check the validity, satisfiability, or unsatisfiability of a sentence can easily by looking at the truth table for the propositional constants in the sentence.

§2.7 Propositional Entailment

Validity, satisfiability, and unsatisfiability are properties of individual sentences. In logical reasoning, we are not so much concerned with individual sentences as we are with the relationships between sentences. In particular, we would like to know, given some sentences, whether other sentences are or are not logical conclusions. This relative property is known as *logical entailment.* When we are speaking about Propositional Logic, we use the phrase *propositional entailment.*

A set of sentences Δ *logically entails* a sentence φ (written $\Delta \models \varphi$) if and only if every interpretation that satisfies Δ also satisfies φ.

For example, the sentence p logically entails the sentence $(p \vee q)$. Since a disjunction is true whenever one of its disjuncts is true, then $(p \vee q)$ must be true whenever p is true. On the other hand, the sentence p does *not* logically entail $(p \wedge q)$. A conjunction is true if and only if *both* of its conjuncts are true, and q may be false. Of course, any set of sentences containing both p and q does logically entail $(p \wedge q)$.

Note that the relationship of logical entailment is a logical one. Even if the premises of a problem do not logically entail the conclusion, this does not mean that the conclusion is necessarily false, even if the premises are true. It just means that it is *possible* that the conclusion is false.

Once again, consider the case of $(p \wedge q)$. Although p does not logically entail this sentence, it is possible that p is true and q is true and, therefore, $(p \wedge q)$ is true. However, the logical entailment does not hold because it is also possible that q is false and, therefore, $(p \wedge q)$ is false.

Exercises

(a) Syntax. Say whether each of the following is a sentence of Propositional Logic.

(a) $p \wedge \neg p$
(b) $\neg p \vee \neg p$
(c) $\neg(q \vee r) \neg\ q \Rightarrow \neg\neg p$
(d) $(p \vee q) \wedge (r \vee q)$
(e) $p \vee \neg q \wedge \neg p \vee \neg q \Rightarrow p \vee q$
(f) $(p \Rightarrow q) \vdash (q \Leftarrow p)$
(g) $(p \Rightarrow q) \models (q \Leftarrow p)$
(h) $((p \Rightarrow q) \Rightarrow s) \Leftrightarrow (r \Leftarrow t)$
(i) $((p \Leftrightarrow q) \Leftrightarrow s) \Leftrightarrow (r \Leftrightarrow t)$
(j) This $\vee$ is $\neg$ correct.

2. *Translation*. Consider a propositional language with three propositional constants – *purple*, *mushroom*, *poisonous* – each indicating the property suggested by its spelling. Using these propositional constants, encode the following English sentences as sentences in Propositional Logic.

(a) If a mushroom is purple, it is poisonous.
(b) A mushroom is poisonous only if it is purple.
(c) A mushroom is not poisonous unless it is purple.
(d) A mushroom is poisonous if and only if it is purple.

3. *Validity, Satisfiability, Unsatisfiability.* For each of the following sentences, indicate whether it is valid, satisfiable, or unsatisfiable.

(a) $(p \Rightarrow q) \vee (q \Rightarrow p)$
(b) $p \wedge (p \Rightarrow \neg q) \wedge q$
(c) $(p \Rightarrow (q \wedge r)) \Leftrightarrow (p \Rightarrow q) \wedge (p \Rightarrow r)$

(d) $(p \Rightarrow (q \Rightarrow r)) \Rightarrow ((p \wedge q) \Rightarrow r)$
(e) $(p \Rightarrow \mathrm{q}) \wedge (p \Rightarrow \neg q)$
(f) $(\neg p \vee \neg q) \Rightarrow \neg(p \wedge q)$
(g) $((\neg p \Rightarrow q) \Rightarrow (\neg q \Rightarrow p)) \wedge (p \vee q)$
(h) $(\neg p \vee q) \Rightarrow (q \wedge (p \Leftrightarrow q))$
(i) $((\neg r \Rightarrow \neg p \wedge \neg q) \vee s) \Leftrightarrow (p \vee q \Rightarrow r \vee s)$
(j) $(p \wedge (\mathrm{q} \Rightarrow r)) \Rightarrow ((\neg p \vee q) \Rightarrow (p \wedge r))$