Session 3 Applications of Propositional Logic

Introduction

Logic has many important applications to mathematics, computer science, and numerous other disciplines. Statements in mathematics and the sciences and in natural language often are imprecise or ambiguous.

To make such statements precise, they can be translated into the language of logic. For example, logic is used in the specification of software and hardware, because these specifications need to be precise before development begins.

Furthermore, propositional logic and its rules can be used to design computer circuits, to construct computer programs, to verify the correctness of programs, and to build expert systems. Logic can be used to analyze and solve many familiar puzzles. Software systems based on the rules of logic have been developed for constructing some, but not all, types of proofs automatically. We will discuss some of these applications of propositional logic in this section and in later chapters.

1.2.2 Translating English Sentences

There are many reasons to translate English sentences into expressions involving propositional
variables and logical connectives. In particular, English (and every other human language) is often ambiguous. Translating sentences into compound statements (and other types of logical expressions, which we will introduce later in this chapter) removes the ambiguity. Note that this may involve making a set of reasonable assumptions based on the intended meaning of the sentence. Moreover, once we have translated sentences from English into logical expressions,
we can analyze these logical expressions to determine their truth values, we can manipulate them, and

EXAMPLE 1

How can this English sentence be translated into a logical expression?
"You can access the Internet from campus only if you are a computer science major or you are not a freshman."

Solution: There are many ways to translate this sentence into a logical expression. Although it is possible to represent the sentence by a single propositional variable, such as p, this would not be useful when analyzing its meaning or reasoning with it. Instead, we will use propositional variables to represent each sentence part and determine the appropriate logical connectives between them. In particular, we let a, c, and f represent "You can access the Internet from campus," "You are a computer science major," and "You are a freshman," respectively. Noting that "only if" is one way a conditional statement can be expressed, this sentence can be represented as $a \rightarrow (c \vee .f)$. ◂

Exercise

How can this English sentence be translated into a logical expression?
"You cannot ride the roller coaster if you are under 4 feet tall unless you are older than 16 years old."

1Boolean Searches

Logical connectives are used extensively in searches of large collections of information, such as indexes of Web pages. Because these searches employ techniques from propositional logic, they are called **Boolean searches**.
In Boolean searches,

- the connective *AND* is used to match records that contain both of two search terms,
- the connective *OR* is used to match one or both of two search terms, and
- the connective *NOT* (sometimes written as *AND NOT*) is used to exclude a particular search term.

Careful planning of how logical connectives are used is often required when Boolean searches are used to locate information of potential interest.

How Boolean searches are carried out.

EXAMPLE **Web Page Searching** Most Web search engines support Boolean searching techniques, which
is useful for finding Web pages about particular subjects. For instance, using Boolean searching to find Web pages about universities in Kenya, we can look for pages matching KENYA *AND* UNIVERSITIES. The results of this search will include those pages that contain the three words KENYA, and UNIVERSITIES.

Logic Puzzles

Puzzles that can be solved using logical reasoning are known as **logic puzzles**. Solving logic puzzles is an excellent way to practice working with the rules of logic. Also, computer programs designed to carry out logical reasoning often use well-known logic puzzles to illustrate their capabilities. Many people enjoy solving logic puzzles, published in periodicals, books, and on the Web, as a recreational activity.

An example of logic puzzle game of Sudoku.

Example: Solve this puzzle

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Logic Circuits

Propositional logic can be applied to the design of computer hardware. This was first observed in 1938 by Claude Shannon in his MIT master's thesis. In Chapter 12 we will study this topic in depth. (See that chapter for a biography of Shannon.) We give a brief introduction to this application here.

A **logic circuit** (or **digital circuit**) receives input signals $p_1, p_2, \ldots, p_n$, each a bit [either 0 (off) or 1 (on)], and produces output signals $s_1, s_2, \ldots, s_n$, each a bit. In this section we will restrict our attention to logic circuits with a single output signal; in general, digital circuits may have multiple outputs.

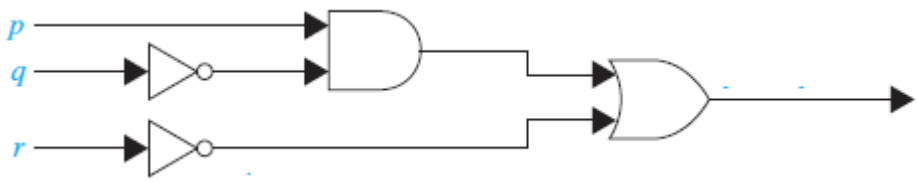
Complicated digital circuits can be constructed from three basic circuits, called **gates**, shown in Figure 1. The **inverter**, or **NOT gate**, takes an input bit p, and produces as output $\neg p$. The **OR gate** takes two input signals p and q, each a bit, and produces as output the signal $p \vee q$. Finally, the **AND gate** takes two input signals p and q, each a bit, and produces as out- put the signal $p \wedge q$. We use combinations of these three basic gates to build more complicated circuits, such as that shown in Figure 2.

Inverter OR gate AND gate

Basic logic gates.

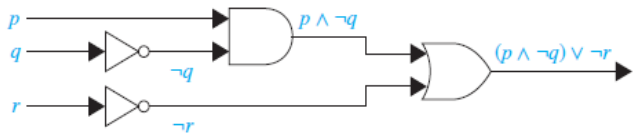
EXAMPLE

Determine the output for the combinatorial circuit in Figure



Solution: In Figure 2 we display the output of each logic gate in the circuit. We see that the AND gate takes input of p and $\neg q$, the output of the inverter with input q, and produces $p \wedge \neg q$. Next, we note that the OR gate takes input $p \wedge \neg q$ and $\neg r$, the output of the inverter with input r, and produces the final output

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



Suppose that we have a formula for the output of a digital circuit in terms of negations, disjunctions, and conjunctions. Then, we can systematically build a digital circuit with the desired output, as illustrated in example that follows.

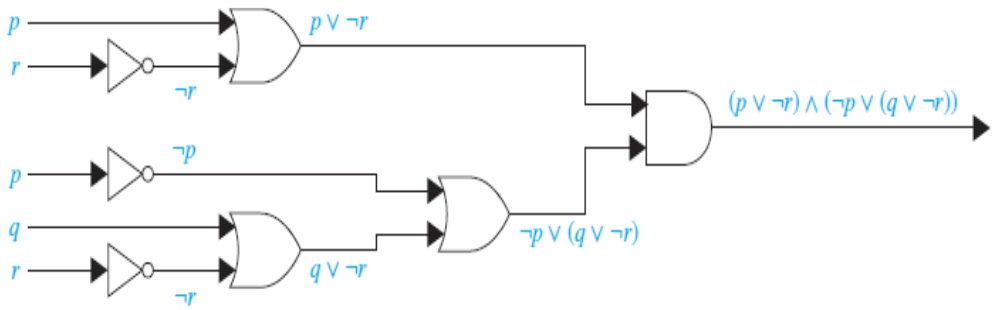
Example

Build a digital circuit that produces the output $(p \vee \neg r) \wedge (\neg p \vee (q \vee \neg r))$ when given input bits p, q, and r.

Solution:

To construct the desired circuit, we build separate circuits for $p \vee \neg r$ and for $\neg p \vee (q \vee \neg r)$ and combine them using an AND gate. To construct a circuit for $p \vee \neg r$, we use an inverter to produce $\neg r$ from the input r. Then, we use an OR gate to combine p and $\neg r$. To build a circuit for $\neg p \vee (q \vee \neg r)$, we first use an inverter to obtain $\neg r$. Then we use an OR gate with inputs q and $\neg r$ to obtain $q \vee \neg r$. Finally, we use another inverter and an OR gate to get $\neg p \vee (q \vee \neg r)$ from the inputs p and $q \vee \neg r$.

To complete the construction, we employ a final AND gate, with inputs $p \vee \neg r$ and $\neg p \vee (q \vee \neg r)$. The resulting circuit is displayed in Figure below.



Exercise

Build a digital circuit that produces the output $(p \vee \neg r) \wedge (\neg p \vee (q \vee \neg r))$ when given input bits

p, q, and r.

Solution: To construct the desired circuit, we build separate circuits for $p \vee \neg r$ and for $\neg p \vee (q \vee \neg r)$ and combine them using an AND gate. To construct a circuit for $p \vee \neg r$, we use an inverter to produce $\neg r$ from the input r. Then, we use an OR gate to combine p and $\neg r$. To build a circuit for $\neg p \vee (q \vee \neg r)$, we first use an inverter to obtain $\neg r$. Then we use an OR gate with inputs q and $\neg r$ to obtain $q \vee \neg r$. Finally, we use another inverter and an OR gate to get $\neg p \vee (q \vee \neg r)$ from the inputs p and $q \vee \neg r$.

To complete the construction, we employ a final AND gate, with inputs $p \vee \neg r$ and $\neg p \vee$

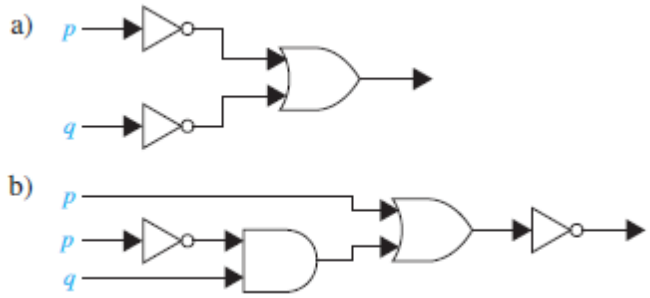
$(q \vee \neg r)$. The resulting circuit is displayed in Figure above. ◂

We will study logic circuits in great detail in **Session 11** in the context of Boolean algebra, and with different notation.

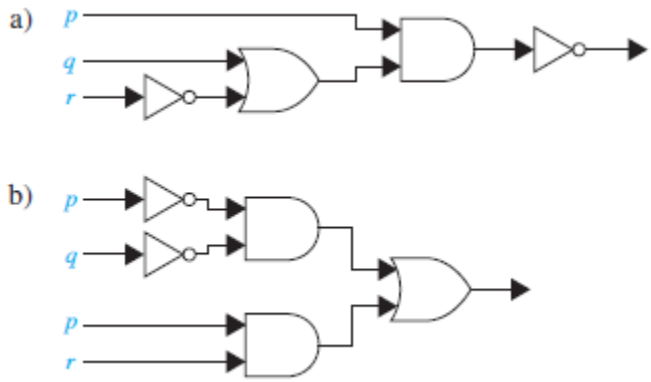
Exercises

. Find the output of each of these combinatorial circuits.

1. Find the output of each of these combinatorial circuits.



2. Find the output of each of these combinatorial circuits.



3. Construct a combinatorial circuit using inverters, OR gates, and AND gates that produces the output $(p \wedge \neg r) \vee (\neg q \wedge r)$ from input bits p, q, and r.

4. Construct a combinatorial circuit using inverters, OR gates, and AND gates that produces the output $((\neg p \vee \neg r) \wedge \neg q) \vee (\neg p \wedge (q \vee r))$ from input bits p, q, and r.