PERIYAR UNIVERSITY

NAAC 'A++' Grade - State University - NIRF Rank 56–State Public University Rank 25

SALEM - 636 011, Tamil Nadu, India.

CENTRE FOR DISTANCE AND ONLINE EDUCATION (CDOE)

M.Sc. MATHEMATICS

SEMESTER - I

ELECTIVE COURSE: DISCRETE MATHEMATICS

(Candidates admitted from 2024 onwards)

PERIYAR UNIVERSITY

CENTRE FOR DISTANCE AND ONLINE EDUCATION (CDOE)

M.Sc. Mathematics - 2024 admission onwards

ELECTIVE – 2
Discrete Mathematics

Prepared by:

Centre for Distance and Online Education (CDOE)

Periyar University
Salem 636011

ELECTIVE - II - SYLLABUS

GROUP – B

Discrete Mathematics

OBJECTIVE: The objective of this course is to understand the basic ideas of logic, proof methods and strategy, the growth of functions, counting techniques, pigeonhole principle, recurrence relations, solving recurrences using generating functions, Boolean functions, apply Boolean algebra to circuits and gatting networks, use finite state-machines to model computer operations.

UNITI: The Foundation of Logic

Logic – Propositional equivalence – Predicates and quantifiers – Proof Methods and Strategy – The growth of functions.

UNITII: Counting

Basics of counting – The pigeonhole principle – permutations and combinations –Generalized permutations and combinations – Generating permutations and combinations.

UNITIII: Advanced counting techniques

Recurrence relation – Solving recurrence relations – Generating functions.

UNITIV: Boolean Algebra

Boolean functions – Representing Boolean functions – Logic Gates – Minimization of circuits.

UNITV: Modeling Computations

Finite – state machines with output, finite – State machines with no output – Turing machines

***TEXTBOOK**:*

1. **Kenneth H. Rosen,** "Discrete Mathematics and its Applications"**,** 7th Edition, WCB/ McGraw Hill Publications, New Delhi, 2011.

BOOKS FOR SUPPLEMENTAR YREADING AND REFERENCES:

1. Edward A. Bender and S. Gill Williamson, *"A Short Course in Discrete Mathematics",* Dover Publications, 2006.
2. M.O. Albertson and J.P. Hutchinson, *"Discrete Mathematics with Algorithms*", John Wiley & Sons, 2008.
3. Rajendra Akerkar and Rupali Akarkar, *"Discrete Mathematics*", Pearson Education Pvt. Ltd, Singapore,2004.
4. J.P. Trembley and R. Manohar, *"Discrete Mathematical Structures*", Tata McGraw Hill, New Delhi,1997.
5. Martin Aigner, *"A Course in Enumeration"*, Springer-Verlag, Heidelberg, 2007.
6. J.H. Van Lint and R.M. Wilson, *"A Course in Combinatorics"*, 2nd Edition, Cambridge University Press, Cambridge, 2001.

Contents

Unit 1

Logic, Proofs and Algorithms

Objectives

1. To introduce students to the concept such as propositional logic and propositional equivalences.
2. To make precise statements and to design logic circuits.
3. To learn about predicates and quantifiers.
4. To introduce different methods of proof and to apply these methods in proving theorems.
5. To learn about the advantages of big-O notation.

1.1 Propositional Logic

1.1.1 Introduction

The rules of logic give precise meaning to mathematical statements. These rules are used to distinguish between valid and invalid mathematical arguments.

Besides the importance of logic in understanding mathematical reasoning, logic has numerous applications to computer science. These rules are used in the design of computer circuits, the construction of computer programs, the verification of the correctness of programs, and in many other ways.

1.1.2 Propositions

A **proposition** is a declarative sentence (that is, a sentence that declares a fact) that is either true or false, but not both.

Example 1. *All the following declarative sentences are propositions.*

- *Washington, D.C., is the capital of the united States of America.*
- *Tornoto is the capital of Canada.*
- $1 + 1 = 2$.
- $2 + 2 = 3$.

Proposition 1 and 3 are true, whereas 2 and 4 are false.

Example 2. *Consider the following sentences.*

1. *What time is it?*
2. *Read this carefully.*
3. $x + 1 = 2$.
4. $x + y = z$.

Sentences 1 and 2 are not propositions because they are not declarative sentences. Sentences 3 and 4 are not propositions because they are neither true nor false. Note that each of sentences 3 and 4 can be turned into a proposition if we assign values to the variables.

We use letters to denote **propositional variables** (or **sentential variables**), that is, variables that represent propositions, just as letters are used to denote numerical variables. The conventional letters used for propositional variables are $p, q, r, s, \cdots$. The **truth value** of a proposition is true, denoted by T, if it is a true proposition, and the truth value of a proposition is false, denoted by F, if it is a false proposition. Propositions that cannot be expressed in terms of simpler propositions are called **atomic propositions**.

The area of logic that deals with propositions is called the **propositional calculus** or **propositional logic**.

Many mathematical statements are constructed by combining one or more propositions. New propositions, called **compound propositions**, are formed from existing propositions using **logical operators**.

Definition 3. *Let p be a proposition. The negation of p, denoted by $\neg p$ (also denoted by p), is the statement*

"It is not the case that p".

The proposition $\neg p$ is read "not p". The truth value of the negation of p, $\neg p$, is the opposite of the truth value of p.

Example 4. *Find the negation of the proposition*

"Michael's PC runs Linux."

and express this in simple English.

Solution: The negation is

"It is not the case that Michael's PC runs Linux."

This negation can be more simply expressed as

"Michael's PC does not run Linux."

Example 5. *Find the negation of the proposition*

"Vadana's smartphone has at least 32 GB of memory"

and express this in simple English.

Solution: The negation is

"It is not the case that Vandana's smartphone has at leat 32 GB of memory"

or even more simply as

"Vandana's smartphone has less than 32 GB of memory."

Table 1.1 displays the **truth table** for the negation of a proposition p. This table has a row for each of the two possible truth values of a proposition p. Each row shows the truth value of $\neg p$ corresponding to the truth value of p for this row.

Table 1.1: The Truth Table for the Negation of a Proposition

p	$\neg p$
T	F
F	T

The negation of a proposition can also be considered the result of the operation of the **negation operator** on a proposition. The negation operator constructs a new proposition from a single existing proposition. We will now introduce the logical operators that are used to form new propositions from two or more existing propositions. These logical operators are also called **connectives**.

Definition 6. *Let p and q be propositions. The conjunction of p and q, denoted by* $p \wedge q$*, is the proposition " p and q." The conjunction* $p \wedge q$ *is true when both p and q are true and is false otherwise.*

Example 7. *Find the conjunction of the propositions p and q where p is the proposition "Rebeccas PC has more than 16 GB free hard disk space" and q is the proposition "The processor in Rebeccas PC runs faster than 1 GHz."*

Solution: The conjunction of these propositions, $p \wedge q$, is the proposition " Rebeccas PC has more than 16 GB free hard disk space, and the processor in Rebeccas PC runs faster than 1 GHz." This conjunction can be expressed more simply as "Rebeccas PC has more than 16 GB free hard disk space, and its processor runs faster than 1 GHz." For this conjunction to be true, both conditions given must be true. It is false, when one or both of these conditions are false.

Definition 8. *Let p and q be propositions. The disjunction of p and q, denoted by* $p \vee q$*, is the proposition "p or q." The disjunction* $p \vee q$ *is false when both p and q are false and is true otherwise.*

Table 1.3 displays the truth table for $p \vee q$.

Table 1.2: The Truth Table for the Conjunction of two Proposition

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

Table 1.3: The Truth Table for the Disjunction of two Proposition

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

The use of the connective or in a disjunction corresponds to one of the two ways the word or is used in English, namely, as an **inclusive or**. A disjunction is true when at least one of the two propositions is true.

Example 9. *Translate the statement "Students who have taken calculus or introductory computer science can take this class" in a statement in propositional logic using the propositions p: " students who has taken calculus can take this class"*

Solution: We assume that this statement means that students who have taken both calculus and introductory computer science can take the class, as well as the students who have taken only one of the two subjects. Hence, this statement can be expressed as $p \vee q$, the inclusive or, or disjunction, of p and q.

Definition 10. *Let p and q be propositions. The exclusive or of p and q, denoted by $p \oplus q$ (or p XOR q), is the proposition that is true when exactly one of p and q is true and is false otherwise.*

Example 11. *Let p and q be the proposition that state " A student can have a salad with dinner" and "A student can have soup with dinner," respectively. What is $p \oplus q$, the exclusive or of p and q.*

Solution: The exclusive or of p and q is the statement that is true when exactly one of p and q is true. that is, $p \oplus q$ is the statement "A student can have soup or salad, but not both, with dinner." Note that this is often stated as "A student can have soup or a salad with dinner," without explicitly stating that taking both is not permitted.

Table 1.4: The Truth Table for the Exclusive or of Two Proposition

p	q	$p \oplus q$
T	T	F
T	F	T
F	T	T
F	F	F

Example 12. *Express the statement "I will use all my savings to travel to Europe or to buy an electric car" in propositional logic using the statement p: "I will use all my savings to travel to Europe" and the statement q: "I will use all my savings to buy an electric car."*

Solution: To translate this statement, we first note that the or in this statement must be an exclusive or because this person can either use all their savings to travel to Europe or use all these savings to buy an electric car, but cannot both go to Europe and buy an electric car. (This is clear because either option requires all their savings.) Hence, this statement can be expressed as $p \oplus q$.

1.1.3 Conditional Statements

Definition 13. *Let p and q be propositions. The conditional statement* $p \to q$ *is the proposition "if p, then q." The conditional statement* $p \to q$ *is false when p is true and q is false, and true otherwise. In the conditional statement* $p \to q$*, p is called the hypothesis (or antecedent or premise) and q is called the conclusion (or consequence).*

The statement $p \to q$ *is called a conditional statement because* $p \to q$ *asserts that q is true on the condition that p holds. A conditional statement is also called an* **implication**.

The truth table for the conditional statement $p \to q$ *is shown in Table 1.5 .*

Table 1.5: The Truth Table for the Conditional Statement $p \to q$.

p	q	$p \to q$
T	T	T
T	F	F
F	T	T
F	F	F

Example 14. *Let p be the statement "Maria learns discrete mathematics" and q the statement "Maria will find a good job." Express the statement $p \to q$ as a statement in English.*

Solution: From the definition of conditional statements, we see that when p is the statement "Maria learns discrete mathematics and q is the statement Maria will find a good job," $p \to q$ represents the statement

"If Maria learns discrete mathematics, then she will find a good job."

There are many other ways to express this conditional statement in English. Among the most natural of these are:

"Maria will find a good job when she learns discrete mathematics."

"For Maria to get a good job, it is sufficient for her to learn discrete mathematics."

and

"Maria will find a good job unless she does not learn discrete mathematics."

Example 15. *What is the value of the variable x after the statement*

if $2 + 2 = 4$ **then** $x : x + 1$

if $x = 0$ before this statement is encountered? (The symbol:= stands for assignment. The statement $x : x + 1$ means the assignment of the value of $x + 1$ to x.

Solution: Because $2 + 2 = 4$ is true, the assignment statement $x : x + 1$ is executed. Hence, x has the value $0 + 1 = 1$ after this statement is encountered.

CONVERSE, CONTRAPOSITIVE, AND INVERSE:

We can form some new conditional statements starting with a conditional statement $p \to q$. In particular, there are three related conditional statements that occur so often that they have special names. The proposition

$q \to p$ is called the **converse** of $p \to q$. The contrapositive of $p \to q$ is the proposition $\neg q \to \neg p$. The proposition $\neg p \to \neg q$ is called the **inverse** of $p \to q$. We will see that of these three conditional statements formed from $p \to q$, only the contrapositive always has the same truth value as $p \to q$.

We first show that the contrapositive, $\neg q \to \neg p$, of a conditional statement $p \to q$ always has the same truth value as $p \to q$. To see this, note that the contrapositive is false only when $\neg p$ is false and $\neg q$ is true, that is, only when p is true and q is false. We now show that neither the converse, $q \to p$, nor the inverse, $\neg p \to \neg q$, has the same truth value as $p \to q$ for all possible truth values of p and q. Note that when p is true and q is false, the original conditional statement is false, but the converse and the inverse are both true.

When two compound propositions always have the same truth values, regardless of the truth values of its propositional variables, we call them **equivalent**. Hence, a conditional statement and its contrapositive are equivalent. The converse and the inverse of a conditional statement are also equivalent, as the reader can verify, but neither is equivalent to the original conditional statement. Take note that one of the most common logical errors is to assume that the converse or the inverse of a conditional statement is equivalent to this conditional statement.

Example 16. *Find are the contrapositive, the converse, and the inverse of the conditional statement*

"The home team wins whenever it is raining?"

Solution: : Because "q whenever p" is one of the ways to express the conditional statement $p \to q$, the original statement can be rewritten as

"If it is raining, then the home team wins."

Consequently, the contrapositive of this conditional statement is

"If the home team does not win, then it is not raining."

The converse is

"If the home team wins, then it is raining."

The inverse is

"If it is not raining, then the home team does not win."

Only the contrapositive is equivalent to the original statement.

Table 1.6: The Truth Table for the Biconditional Statement $p \leftrightarrow q$.

p	q	$p \leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

BICONDITIONALS: We now introduce another way to combine propositions that expresses that two propositions have the same truth value.

Definition 17. *Let p and q be propositions. The biconditional statement* $p \leftrightarrow q$ *is the proposition "p if and only if q." The biconditional statement* $p \leftrightarrow q$ *is true when p and q have the same truth values, and is false otherwise. Biconditional statements are also called bi-implications.*

The truth table for $p \leftrightarrow q$ *is shown in Table 1.6.*

Example 18. *Let p be the statement "You can take the flight," and let q be the statement "You buy a ticket." Then* $p \leftrightarrow q$ *is the statement*

"You can take the flight if and only if you buy a ticket."

This statement is true if p and q are either both true or both false, that is, if you buy a ticket and can take the flight or if you do not buy a ticket and you cannot take the flight. It is false when p and q have opposite truth values, that is, when you do not buy a ticket, but you can take the flight (such as when you get a free trip) and when you buy a ticket but you cannot take the flight (such as when the airline bumps you).

1.1.4 Truth Tables of Compound Propositions

Example 19. *Construct the truth table of the compound proposition*

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

Solution: Because this truth table involves two propositional variables p and q, there are four rows in this truth table, one for each of the pairs of truth values TT, TF, FT, and FF. The first two columns are used for the truth values of p and q, respectively. In the third column we find the truth value of $\neg q$, needed to find the truth value of $p \vee \neg q$, found in the fourth column. The fifth column gives the truth value of $p \wedge q$. Finally, the truth value of $(p \wedge \neg q) \rightarrow (p \wedge q)$ is found in the last column. The resulting truth table is shown in Table 1.7.

Table 1.7: The Truth Table $(p \vee \neg q) \rightarrow (p \wedge q)$.

p	q	$\neg q$	$p \vee \neg q$	$p \wedge q$	$(p \vee \neg q) \rightarrow (p \wedge q)$
T	T	F	T	T	T
T	F	T	T	F	F
F	T	F	F	F	T
F	F	T	T	F	F

1.1.5 Precedence of Logical Operators

Table 1.8 displays the precedence levels of the logical operators, $\neg$, $\wedge$, $\vee$, $\rightarrow$ and $\leftrightarrow$.

Table 1.8: Precedence of Logical Operators.

Operator	**Precedence**
$\neg$	1
$\wedge$	2
$\vee$	3
$\rightarrow$	4
$\leftrightarrow$	5

1.1.6 Logic and Bit Operations

Computers represent information using bits. A **bit** is a symbol with two possible values, namely, 0 (zero) and 1 (one). This meaning of the word bit comes from binary digit, because zeros and ones are the digits used in binary representations of numbers. The well-known statistician John Tukey introduced this terminology in 1946. A bit can be used to represent a truth value, because there are two truth values, namely, *true and false*. As is customarily done, we will use a 1 bit to represent true and a 0 bit to represent false. That is, 1 represents T (true), 0 represents F (false). A variable is called a Boolean variable if its value is either true or false. Consequently, a **Boolean variable** can be represented using a bit.

Computer **bit operations** correspond to the logical connectives. By replacing true by a one and false by a zero in the truth tables for the

Truth Value	**Bit**
T	1
F	0

operators $\wedge$, $\vee$, and $\oplus$, the tables shown in Table 1.9 for the corresponding bit operations are obtained. We will also use the notation OR, AND, and XOR for the operators $\vee, \wedge$, and $\oplus$, as is done in various programming languages.

Table 1.9: Table for the Bit Operators OR, AND, and XOR.

x	y	$x \vee y$	$x \wedge y$	$x \oplus y$
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0

Definition 20. *A bit string is a sequence of zero or more bits. The length of this string is the number of bits in the string.*

Example 21. *101010011 is a bit string of length nine.*

We can extend bit operations to bit strings. We define the **bitwise OR, bitwise AND***, and* **bitwise XOR** *of two strings of the same length to be the strings that have as their bits the OR, AND, and XOR of the corresponding bits in the two strings, respectively. We use the symbols $\vee, \wedge$, and $\oplus$ to represent the bitwise OR, bitwise AND, and bitwise XOR operations, respectively. We illustrate bitwise operations on bit strings with Example*

22.

Example 22. *Find the bitwise OR, bitwise AND, and bitwise XOR of the bit strings 01 1011 0110 and 11 0001 1101. (Here, and throughout this book, bit strings will be split into blocks of four bits to make them easier to read.)*

Solution: The bitwise OR, bitwise AND, and bitwise XOR of these strings are obtained by taking the OR, AND, and XOR of the corresponding bits, respectively. This gives us

01	1011	0110	
11	0001	1101	
11	1011	1111	bitwise OR
01	0001	0100	bitwise AND
10	1010	1011	bitwise XOR

Let Us Sum Up

In this section, we discussed about

- ∗ Propositions
- ∗ Conditional statements
- ∗ Truth tables of compound propositions
- ∗ Precedence of logical operators
- ∗ Logic and Bit operators.

Check your Progress

1. In the conditional statements $p \to q$, p is called

(a) hypothesis

(b) implication

(c) if then

(d) converse

2. A bit is a

(a) Logical operator

(b) truth value T or F

(c) symbol with two possible values 0 and 1

(d) Boolean variable

3. The length of a bit string is ________ in the string.

(a) number of bit operations

(b) number of truth values

(c) number of 1's

(d) number of bits

1.2 Applications of Propositional Logic

1.2.1 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.

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.

Example 23. *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

$$a \to (c \vee \neg f).$$

Example 24. *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."

Solution: Let q, r, and s represent "You can ride the roller coaster," "You are under 4 feet tall," and "You are older than 16 years old," respectively. Then the sentence can be translated to

$$(r \wedge \neg s) \to \neg q.$$

There are other ways to represent the original sentence as a logical expression, but the one we have used should meet our needs.

1.2.3 System Specifications

Example 25. *Express the specification "The automated reply cannot be sent when the file system is full" using logical connectives.*

Solution: One way to translate this is to let p denote "The automated reply can be sent" and q denote "The file system is full." Then $\neg p$ represents "It is not the case that the automated reply can be sent," which can also be expressed as "The automated reply cannot be sent." Consequently, our specification can be represented by the conditional statement $q \to \neg p$.

System specifications should be consistent, that is, they should not contain conflicting requirements that could be used to derive a contradiction. When specifications are not consistent, there would be no way to develop a system that satisfies all specifications.

Example 26. *Determine whether these system specifications are consistent:*

"The diagnostic message is stored in the buffer or it is retransmitted."

"The diagnostic message is not stored in the buffer."

"If the diagnostic message is stored in the buffer, then it is retransmitted."

Solution: To determine whether these specifications are consistent, we first express them using logical expressions. Let p denote "The diagnostic message is stored in the buffer" and let q denote "The diagnostic message is retransmitted." The specifications can then be written as $p \vee q$, $\neg p$, and $p \to q$. An assignment of truth values that makes all three specifications true must have p false to make $\neg p$ true. Because we want $p \vee q$ to be true but p must be false, q must be true. Because $p \to q$ is true when p is false and q is true, we conclude that these specifications are consistent, because they are all true when p is false and q is true. We could come to the same conclusion by use of a truth table to examine the four possible assignments of truth values to p and q.

Example 27. *Do the system specifications in Example 4 remain consistent if the specification "The diagnostic message is not retransmitted" is added?*

Solution: By the reasoning in Example 4, the three specifications from that example are true only in the case when p is false and q is true. However, this new specification is $\neg q$, which is false when q is true. Consequently, these four specifications are inconsistent.

1.2.4 Boolean 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.

Example 28. Web Page Searching *Most Web search engines support Boolean searching techniques, which usually can help find Web pages about particular subjects. For instance, using Boolean searching to find Web pages about universities in New Mexico, we can look for pages matching NEW AND MEXICO AND UNIVERSITIES. The results of this search will include those pages that contain the three words NEW, MEXICO, and UNIVERSITIES. This will include all of the pages of interest, together with others such as a page about new universities in Mexico. (Note that in Google, and many other search engines, the word "AND" is not needed, although it is understood, because all search terms are included by default. These search engines also support the use of quotation marks to search for specific phrases. So, it may be more effective to search for pages matching "New Mexico" AND UNIVERSITIES.)*

Next, to find pages that deal with universities in New Mexico or Arizona, we can search for pages matching (NEW AND MEXICO OR ARIZONA) AND UNIVERSITIES. (Note: Here the AND operator takes precedence over the OR operator. Also, in Google, the terms used for this search would be NEW MEXICO OR ARIZONA.) The results of this search will include all pages that contain the word UNIVERSITIES and either both the words NEW and MEXICO or the word ARIZONA. Again, pages besides those of interest will be listed. Finally, to find Web pages that deal with universities in Mexico (and not New Mexico), we might first look for pages matching MEXICO AND UNIVERSITIES, but because the results of this search will include pages about universities in New Mexico, as well as universities in Mexico, it might be better to search for pages matching

(MEXICO AND UNIVERSITIES) NOT NEW. The results of this search include pages that contain both the words MEXICO and UNIVERSITIES but do not contain the word NEW. (In Google, and many other search engines, the word "NOT" is replaced by the symbol "-". In Google, the terms used for this last search would be MEXICO UNIVERSITIES -NEW.)

1.2.5 Logic Circuits

A logic circuit (or **digital circuit**) receives input signals $p_1, p_2, \cdots, p_n$, each a bit [either 0 (off) or 1 (on)], and produces output signals $s_1, s_2, \cdots, s_n$, each a bit.

Complicated digital circuits can be constructed from three basic circuits, called gates, shown in Figure 1.1. 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 output the signal $p \wedge q$. We use combinations of these three basic gates to build more complicated circuits, such as that shown in Figure 1.2.

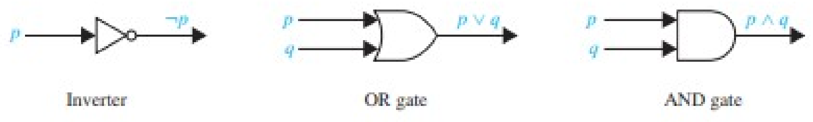


Figure 1.1: Basic logic gates.

Example 29. *Determine the output for the combinatorial circuit in Figure 1.2.*

Solution: In Figure 1.2 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 30.

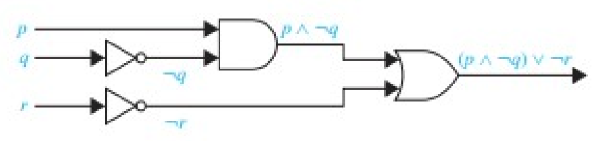


Figure 1.2: A combinatorial circuit.

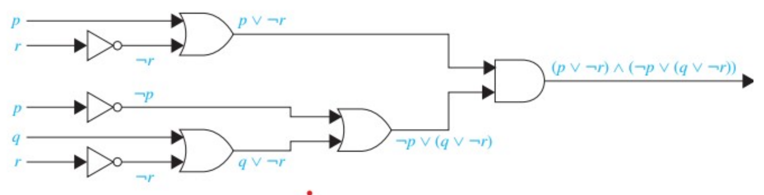

p
r
p
q
r
$p \vee \neg r$
$\neg r$
$\neg p$
$q \vee \neg r$
$\neg r$
$\neg p \vee (q \vee \neg r)$
$(p \vee \neg r) \wedge (\neg p \vee (q \vee \neg r))$

Figure 1.3: The circuit for $(p \vee \neg r) \wedge (\neg p \vee (q \vee \neg r))$.

Example 30. *Build a digital circuit that produces the output* $(p \vee \neg r) \wedge ($ $negp \vee (q \vee \neg r))$ *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 1.3.

Let Us Sum Up

In this section, we discussed about

$*$ Applications of propositional logic.

$*$ Boolean searches.

$*$ Logic circuits

Check your Progress

1. In Boolean searches, the connective AND is used to match records that contain

(a) one or both of the search items

(b) both of two search items

(c) Both (a) and (b)

(d) information excluding a particular search term

2. A compound proposition that is always true is called

(a) tautology

(b) contradiction

(c) contingency

(d) predicate

1.3 Propositional Equivalences

Definition 31. *A compound proposition that is always true, no matter what the truth values of the propositional variables that occur in it, is called a tautology. A compound proposition that is always false is called a contradiction. A compound proposition that is neither a tautology nor a contradiction is called a contingency.*

Example 32. *We can construct examples of tautologies and contradictions using just one propositional variable. Consider the truth tables of* $p \vee \neg p$ *and* $p \wedge \neg p$*, shown in Table 1.10. Because* $p \vee \neg p$ *is always true, it is a tautology. Because* $p \wedge \neg p$ *is always false, it is a contradiction.*

Table 1.10: Examples of a Tautology and a Contradiction.

p	$\neg p$	$p \vee \neg p$	$p \wedge \neg p$
T	F	T	F
F	T	T	F

1.3.1 Logical Equivalences

Definition 33. *The compound propositions p and q are called logically equivalent if* $p \leftrightarrow q$ *is a tautology. The notation* $p \equiv q$ *denotes that p and q are logically equivalent.*

Example 34 illustrates this method to establish an extremely important and useful logical equivalence, namely, that of $\neg(p \vee q)$ with $\neg p \wedge \neg q$. This logical equivalence is one of the two De Morgan laws, shown in Table 1.11, named after the English mathematician Augustus De Morgan, of the mid-nineteenth century.

Table 1.12: Truth Tables for $\neg(p \vee q)$ and $\neg p \wedge \neg q$.

p	q	$p \vee q$	$\neg(p \vee q)$	$\neg p$	$\neg q$	$\neg p \wedge \neg q$
T	T	T	F	F	F	F
T	F	T	F	F	T	F
F	T	T	F	T	F	F
F	F	F	T	T	T	T

Table 1.11: De Morgan's Law

$\neg(p \wedge q) \equiv \neg p \vee \neg q$
$\neg(p \vee q) \equiv \neg p \wedge \neg q$

Example 34. *Show that $\neg(p \vee q)$ and $\neg p \wedge \neg q$ are logically equivalent.*

Solution: The truth tables for these compound propositions are displayed in Table 1.12. Because the truth values of the compound propositions $\neg(p \vee q)$ and $\neg p \wedge \neg q$ agree for all possible combinations of the truth values of p and q, it follows that $\neg(p \vee q) \leftrightarrow (\neg p \wedge \neg q)$ is a tautology and that these compound propositions are logically equivalent.

Example 35. *Show that $p \rightarrow q$ and $\neg p \vee q$ are logically equivalent.*

Solution: We construct the truth table for these compound propositions in Table 1.13. Because the truth values of $\neg p \vee q$ and $p \rightarrow q$ agree, they are logically equivalent.

Example 36. *Show that $p \vee (q \wedge r)$ and $(p \wedge q) \wedge (p \vee r)$ are logically equivalent. This is the distributive law of disjunction over conjunction.*

Solution: We construct the truth table for these compound propositions in Table 1.14. Because the truth values of $p \vee (q \wedge r)$ and $(p \vee q) \wedge (p \vee r)$ agree, these compound propositions are logically equivalent.

Table 1.13: Truth Tables for $\neg p \vee q$ and $p \rightarrow q$.

p	q	$\neg p$	$\neg p \vee q$	$p \rightarrow q$
T	T	F	T	T
T	F	F	F	F
F	T	T	T	T
F	F	T	T	T

Table 1.14: A Demonstration That $p \vee (q \wedge r)$ and $(p \vee q) \wedge (p \vee r)$ are Logically Equivalent.

p	q	r	$q \wedge r$	$p \vee (q \wedge r)$	$p \vee q$	$p \vee r$	$(p \vee q) \wedge (p \vee r)$
T	T	T	T	T	T	T	T
T	T	F	F	T	T	T	T
T	F	T	F	T	T	T	T
T	F	F	F	T	T	T	T
F	T	T	T	T	T	T	T
F	T	F	F	F	T	F	F
F	F	T	F	F	F	T	F
F	F	F	F	F	F	F	F

Table 1.15 contains some important equivalences. In these equivalences, T denotes the compound proposition that is always true and F denotes the compound proposition that is always false. We also display some useful equivalences for compound propositions involving conditional statements and biconditional statements in Tables 1.16 and 1.17, respectively.

1.3.2 Using De Morgans Laws

Example 37. *Use De Morgans laws to express the negations of "Miguel has a cellphone and he has a laptop computer" and "Heather will go to*

Table 1.15: Logically Equivalences.

Equivalences	**Name**
$p \wedge T \equiv p$ $p \vee F \equiv p$	Identity laws
$p \vee T \equiv T$ $p \wedge F \equiv F$	Domination laws
$p \vee p \equiv p$ $p \wedge p \equiv p$	Idempotent laws
$\neg(\neg p) \equiv p$	Double negation law
$p \vee q \equiv q \vee p$ $p \wedge q \equiv q \wedge p$	Commutative laws
$(p \wedge q) \vee r \equiv p \vee (q \vee r)$ $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$	Associative laws
$p \vee (q \wedge r) \equiv (p \vee q) \vee (p \vee r)$ $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$	Distributive laws
$\neg(p \wedge q) \equiv \neg p \vee \neg q$ $\neg(p \vee q) \equiv \neg p \wedge \neg q$	De Morgans laws
$p \vee (p \wedge q) \equiv p$ $p \wedge (p \vee q) \equiv p$	Absorption laws
$p \vee \neg p \equiv T$ $p \wedge \neg p \equiv F$	Negation laws

the concert or Steve will go to the concert."

Solution: Let p be "Miguel has a cellphone" and q be "Miguel has a laptop computer." Then "Miguel has a cellphone and he has a laptop computer" can be represented by $p \wedge q$. By the first of De Morgans laws, $\neg(p \wedge q)$ is equivalent to $\neg p \vee \neg q$. Consequently, we can express the negation of our original statement as "Miguel does not have a cellphone or he does not hav a laptop computer."

Let r be "Heather will go to the concert" and s be "Steve will go to the concert." Then "Heather will go to the concert or Steve will go to the concert" can be represented by $r \vee s$. By the second of De Morgans laws, $\neg(r \vee s)$ is equivalent to $\neg r \wedge \neg s$. Consequently, we can express the negation of our original statement as "Heather will not go to the concert and Steve will not go to the concert."

Table 1.16: Logical Equivalences Involving Conditional Statements.

$p \to q \equiv \neg p \vee q$
$p \to q \equiv \neg q \to \neg p$
$p \vee q \equiv \neg p \to q$
$p \wedge q \equiv \neg(p \to \neg q)$
$\neg(p \to q) \equiv p \wedge \neg q$
$(p \to q) \wedge (p \to r) \equiv p \to (q \wedge r)$
$(p \to r) \wedge (q \to r) \equiv (p \vee q) \to r$
$(p \to q) \vee (p \to r) \equiv p \to (q \vee r)$
$(p \to r) \vee (q \to r) \equiv (p \wedge q) \to r$

1.3.3 Constructing New Logical Equivalences

Example 38. *Show that $\neg(p \to q)$ and $p \wedge \neg q$ are logically equivalent.*

Solution: We could use a truth table to show that these compound propositions are equivalent (similar to what we did in Example 36). Indeed, it

Table 1.17: Logical Equivalences Involving Biconditional Statements.

$p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$
$p \leftrightarrow q \equiv \neg p \leftrightarrow \neg q$
$p \leftrightarrow q \equiv (p \wedge q) \vee (\neg p \wedge \neg q)$
$\neg(p \leftrightarrow q) \equiv p \leftrightarrow \neg q$

would not be hard to do so. However, we want to illustrate how to use logical identities that we already know to establish new logical identities, something that is of practical importance for establishing equivalences of compound propositions with a large number of variables. So, we will establish this equivalence by developing a series of logical equivalences, using one of the equivalences in Table 1.15 at a time, starting with $\neg(p \rightarrow q)$ and ending with $p \wedge \neg q$. We have the following equivalences.

$$\begin{aligned}
\neg(p \rightarrow q) &\equiv \neg(\neg p \vee q) && \text{by the conditional-disjunction} \\
& && \text{equivalence (Example 35)} \\
&\equiv \neg(\neg p) \wedge \neg q && \text{by the second De Morgan law} \\
&\equiv p \wedge \neg q && \text{by the double negation law}
\end{aligned}$$

Example 39. *Show that $\neg(p \vee (\neg p \wedge q))$ and $\neg p \wedge \neg q$ are logically equivalent by developing a series of logical equivalences.*

Solution: We will use one of the equivalences in Table 1.15 at a time, starting with $\neg(p \vee (\neg p \wedge q))$ and ending with $\neg p \wedge \neg q$. (Note: we could also easily establish this equivalence using a truth table.) We have the following equivalences.

$$\begin{aligned}
\neg(p \vee (\neg p \wedge q)) &\equiv \neg p \wedge \neg(\neg p \wedge q) \quad \text{by the second De Morgan law}\\
&\equiv \neg p \wedge [\neg(\neg p) \wedge \neg q] \quad \text{by the first De Morgan law}\\
&\equiv \neg p \wedge (p \vee \neg q) \quad \text{by the double negation law}\\
&\equiv (\neg p \wedge p) \vee (\neg p \wedge \neg q) \quad \text{by the second distributive law}\\
&\equiv (\neg p \wedge \neg q) \vee F \quad \text{by the commutative law for disjunction}
\end{aligned}$$

$$\equiv \neg p \wedge \neg q \quad \text{by the identity law for F}$$

Consequently $\neg(p \vee (\neg p \wedge q))$ and $\neg p \wedge \neg q$ are logically equivalent.

Example 40. *Show that $(p \wedge q) \to (p \vee q)$ is a tautology.*

Solution: To show that this statement is a tautology, we will use logical equivalences to demonstrate that it is logically equivalent to **T**. (Note: This could also be done using a truth table.)

$$\begin{aligned}
(p \wedge q) \to (p \wedge q) &\equiv \neg(p \wedge q) \vee (p \vee q) \quad \text{by Example 35}\\
&\equiv (\neg p \vee \neg q) \vee (p \vee q) \quad \text{by the first De Morgan law}\\
&\equiv (\neg p \vee p) \vee (\neg q \vee q) \quad \text{by the associative and}\\
&\qquad \text{commutative laws for disjunction}\\
&\equiv T \vee T \quad \text{by Example 1 and the}\\
&\qquad \text{commutative law for disjunction}\\
&\equiv T \quad \text{by the domination law}
\end{aligned}$$

1.3.4 Satisfiability

A compound proposition is **satisfiable** if there is an assignment of truth values to its variables that makes it true. When no such assignments exists, that is, when the compound proposition is false for all assignments of truth values to its variables, the compound proposition is **unsatisfiable**.

When we find a particular assignment of truth values that makes a compound proposition true, we have shown that it is satisfiable; such an assignment is called a **solution** of this particular satisfiability problem.

Example 41. *Determine whether each of the compound propositions* $(p \vee \neg q) \wedge (q \vee \neg r) \wedge (r \vee \neg p)$, $(p \vee q \vee r) \wedge (\neg p \vee \neg q \vee \neg r)$, *and* $(p \vee \neg q) \wedge (q \vee \neg r) \wedge (r \vee \neg p) \wedge (p \vee q \vee r) \wedge (\neg p \vee \neg q \vee \neg r)$ *is satisfiable.*

Solution: Instead of using truth table to solve this problem, we will reason about truth values. Note that $(p \vee \neg q) \wedge (q \vee \neg r) \wedge (r \vee \neg p)$ is true when the three variable p, q, and r have the same truth value. Hence, it is satisfiable as there is at least one assignment of truth values for p, q, and r that makes it true. Similarly, note that $(p \vee q \vee r) \wedge (\neg p \vee \neg q \vee \neg r)$ is true when at least one of p, q, and r is true and at least one is false. Hence, $(p \vee q \vee r) \wedge (\neg p \vee \neg q \vee \neg r)$ is satisfiable, as there is at least one assignment of truth values for p, q, and r that makes it true.

Finally, note that for $(p \vee \neg q) \wedge (q \vee \neg \wedge r) \wedge (r \vee \neg p) \wedge (p \vee q \vee r) \wedge (\neg p \vee \neg q \vee \neg r)$ to be true, $(p \vee \neg q) \wedge (q \vee \neg r) \wedge (r \vee \neg p)$ and $(p \vee q \vee r) \wedge (\neg p \vee \neg q \vee \neg r)$ must both be true. For the first to be true, the three variables must have the same truth values, and for the second to be true, at least one of three variables must be true and at least one must be false. However, these conditions are contradictory. From these observations we conclude that no assignment of truth values to p, q, and r makes $(p \vee \neg q) \wedge (q \vee \neg r) \wedge (r \vee \neg p) \wedge (p \vee q \vee r) \wedge (\neg p \vee \neg q \vee \neg r)$ true. Hence, it is unsatisfiable.

Let Us Sum Up

In this section, we discussed about

- $*$ Logical equivalences.
- $*$ Construction of new logical equivalences.
- $*$ Satisfiability.

Check your Progress

1. The statement $p \to q \equiv \neg p \vee q$ is called

   (a) De Morgan's law

   (b) Conditional - disjunction equivalence

   (c) Negation law

   (d) a tautology

2. The statement $P \vee (p \wedge q) \equiv p$ is called

   (a) commutative

   (b) Identity law

   (c) Absorption law

   (d) Associative law

1.4 Predicates and Quantifiers

1.4.1 Predicates

The statement “x is greater than 3” has two parts. The first part, the variable x, is the subject of the statement. The second-part the predicate, “is greater than 3”-refers to a property that the subject of the statement can have. We can denote the statement “x is greater than 3” by $P(x)$, where P denotes the predicate “is greater than 3” and x is the variable. The statement $P(x)$ is also said to be the value of the propositional function P at x. Once a value has been assigned to the variable x, the statement $P(x)$ becomes a proposition and has a truth value. Consider Examples 42 and 43.

Example 42. *Let $P(x)$ denote the statement “x ¿ 3.” What are the truth values of $P(4)$ and $P(2)$?*

Solution: We obtain the statement $P(4)$ by setting $x = 4$ in the statement “$x > 3$.” Hence, $P(4)$, which is the statement “$4 > 3$,” is true. However, $P(2)$, which is the statement “$2 > 3$,” is false.

Example 43. *Let $A(x)$ denote the statement “Computer x is under attack by an intruder.” Suppose that of the computers on campus, only CS2 and MATH1 are currently under attack by intruders. What are truth values of A(CS1), A(CS2), and A(MATH1)?*

Solution: We obtain the statement A(CS1) by setting x = CS1 in the statement “Computer x is under attack by an intruder.” Because CS1 is not on the list of computers currently under attack, we conclude that A(CS1) is false. Similarly, because CS2 and MATH1 are on the list of computers under attack, we know that A(CS2) and A(MATH1) are true.

We can also have statements that involve more than one variable. For instance, consider the statement "$x = y+3$." We can denote this statement by $Q(x, y)$, where x and y are variables and Q is the predicate. When values are assigned to the variables x and y, the statement $Q(x, y)$ has a truth value.

Example 44. *Let $Q(x, y)$ denote the statement "$x = y + 3$." What are the truth values of the propositions $Q(1, 2)$ and $Q(3, 0)$?*

Solution: To obtain $Q(1, 2)$, set $x = 1$ and $y = 2$ in the statement $Q(x, y)$. Hence,$Q(1, 2)$ is the statement "$1 = 2+3$," which is false. The statement $Q(3, 0)$ is the proposition "$3 = 0 + 3$," which is true.

Similarly, we can let $R(x, y, z)$ denote the statement "$x + y = z$." When values are assigned to the variables x, y, and z, this statement has a truth value.

Example 45. *What are the truth values of the propositions $R(1, 2, 3)$ and $R(0, 0, 1)$?*

Solution: The proposition $R(1, 2, 3)$ is obtained by setting $x = 1$, $y = 2$, and $z = 3$ in the statement $R(x, y, z)$. We see that $R(1, 2, 3)$ is the statement "$1 + 2 = 3$," which is true. Also note that $R(0, 0, 1)$, which is the statement "0 + 0 = 1," is false.

Example 46. *Consider the statement*

if $x > 0$ **then** $x := x + 1$.

When this statement is encountered in a program, the value of the variable x at that point in the execution of the program is inserted into $P(x)$, which is "$x > 0$." If $P(x)$ is true for this value of x, the assignment statement $x := x+1$ is executed, so the value of x is increased by 1. If $P(x)$ is *false for this value of x, the assignment statement is not executed, so the value of x is not changed.*

PRECONDITIONS AND POSTCONDITIONS The statements that describe valid input are known as **preconditions** and the conditions that the output should satisfy when the program has run are known as **post-conditions**.

Example 47. *Consider the following program, designed to interchange the values of two variables x and y.*

$$\begin{aligned} temp &:= x \\ x &:= y \\ y &:= temp \end{aligned}$$

Find predicates that we can use as the precondition and the postcondition to verify the correctness of this program. Then explain how to use them to verify that for all valid input the program does what is intended.

Solution: For the precondition, we need to express that x and y have particular values before we run the program. So, for this precondition we can use the predicate $P(x, y)$, where $P(x, y)$ is the statement "$x = a$ and $y = b$," where a and b are the values of x and y before we run the program. Because we want to verify that the program swaps the values of x and y for all input values, for the postcondition we can use $Q(x, y)$, where $Q(x, y)$ is the statement "$x = b$ and $y = a$." To verify that the program always does what it is supposed to do, suppose that the precondition $P(x, y)$ holds. That is, we suppose that the statement "$x = a$ and $y = b$" is true. This means that $x = a$ and $y = b$. The first step of the program, temp $:= x$, assigns the value of x to the variable temp, so after this step we know that $x = a$, temp $= a$, and $y = b$. After the second step of the program, $x := y$, we know that $x = b$, temp $= a$, and $y = b$. Finally, after the third step, we know that $x = b$, temp $= a$, and $y = a$. Consequently, after this program is run, the postcondition $Q(x, y)$ holds, that is, the statement "$x = b$ and $y = a$" is true.

1.4.2 Quantifiers

The area of logic that deals with predicates and quantifiers is called the **predicate calculus**.

The Universal Quantifier: Many mathematical statements assert that a property is true for all values of a variable in a particular domain, called the **domain of discourse** (or the **universe of discourse**), often just referred to as the domain. Such a statement is expressed using universal quantification.

Definition 48. *The universal quantification of $P(x)$ is the statement*

"$P(x)$ for all values of x in the domain."

The notation $\forall\ xP(x)$ denotes the universal quantification of $P(x)$. Here $\forall$ is called the **universal quantifier**. *We read $\forall\ xP(x)$ as "for all $xP(x)$" or "for every $xP(x)$." An element for which $P(x)$ is false is called a* **counter example** *of $\forall\ xP(x)$.*

Example 49. *Let $P(x)$ be the statement "$x + 1 > x$." What is the truth value of the quantification $\forall\ xP(x)$, where the domain consists of all real numbers?*

Solution: : Because $P(x)$ is true for all real numbers x, the quantification

$$\forall\ xP(x)$$

is true.

Example 50. *Let $Q(x)$ be the statement "$x < 2$." What is the truth value of the quantification $\forall\, xQ(x)$, where the domain consists of all real numbers?*

Solution: $Q(x)$ is not true for every real number x, because, for instance, $Q(3)$ is false. That is, $x = 3$ is a counterexample for the statement $\forall\, xQ(x)$. Thus

$$\forall\, xQ(x)$$

is false.

Example 51. *Suppose that $P(x)$ is "$x^2 > 0$." To show that the statement $\forall\, xP(x)$ is false where the universe of discourse consists of all integers, we give a counterexample. We see that $x = 0$ is a counterexample because $x^2 = 0$ when $x = 0$, so that x^2 is not greater than 0 when $x = 0$.*

Example 52. *What does the statement $\forall\, xN(x)$ mean if $N(x)$ is "Computer x is connected to the network" and the domain consists of all computers on campus?*

Solution: The statement $\forall\, xN(x)$ means that for every computer x on campus, that computer x is connected to the network. This statement can be expressed in English as "Every computer on campus is connected to the network."

Example 53. *What is the truth value of $\forall\, x(x^2 \geq x)$ if the domain consists of all real numbers? What is the truth value of this statement if the domain consists of all integers?*

Solution: The universal quantification $\forall\, x(x^2 \geq x)$, where the domain consists of all real numbers, is false. For example, $\left(\frac{1}{2}\right)^2 \not\geq \frac{1}{2}$. Note that

$x^2 \geq x$ if and only if $x^2 - x = x(x-1) \geq 0$. Consequently, $x^2 \geq x$ if and only if $x \leq 0$ or $x \geq 1$. It follows that $\forall\, x(x^2 \geq x)$ is false if the domain consists of all real numbers (because the inequality is false for all real numbers x with $0 < x < 1$). However, if the domain consists of the integers, $\forall\, x(x^2 \geq x)$ is true, because there are no integers x with $0 < x < 1$.

The Existential Quantifier: Many mathematical statements assert that there is an element with a certain property. Such statements are expressed using existential quantification. With existential quantification, we form a proposition that is true if and only if $P(x)$ is true for at least one value of x in the domain

Definition 54. *The existential quantification of $P(x)$ is the proposition*

"There exists an element x in the domain such that $P(x)$."

We use the notation $\exists\, xP(x)$ for the existential quantification of $P(x)$. Here $\exists$ is called the existential quantifier.

Example 55. *Let $P(x)$ denote the statement "$x > 3$." What is the truth value of the quantification $\exists xP(x)$, where the domain consists of all real numbers?*

Solution: Because "$x > 3$" is sometimes true-for instance, when $x = 4$the existential quantification of $P(x)$, which is $\exists xP(x)$, is true.

Example 56. *Let $Q(x)$ denote the statement "$x = x + 1$." What is the truth value of the quantification $\exists\, xQ(x)$, where the domain consists of all real numbers?*

Solution: Because $Q(x)$ is false for every real number x, the existential quantification of $Q(x)$, which is $\exists\, xQ(x)$, is false.

1.4.3 Quantifiers Over Finite Domains

Example 57. *What is the truth value of $\exists\, xP(x)$, where $P(x)$ is the statement "$x^2 < 10$" and the universe of discourse consists of the positive integers not exceeding 4?*

Solution: The statement $\forall\, xP(x)$ is the same as the conjunction $P(1) \wedge P(2) \wedge P(3) \wedge P(4)$, because the domain consists of the integers 1, 2, 3, and 4. Because $P(4)$, which is the statement "42 $<$ 10," is false, it follows that $\forall\, xP(x)$ is false.

Example 58. *What is the truth value of $\exists\, xP(x)$, where $P(x)$ is the statement "$x^2 > 10$" and the universe of discourse consists of the positive integers not exceeding 4?*

Solution: Because the domain is $\{1, 2, 3, 4\}$, the proposition $\exists\, xP(x)$ is the same as the disjunction $P(1) \vee P(2) \vee P(3) \vee P(4)$. Because $P(4)$, which is the statement "42 $>$ 10," is true, it follows that $\exists\, xP(x)$ is true.

1.4.4 Quantifiers with Restricted Domains

Example 59. *What do the statements $\forall\, x < 0(x^2 > 0)$, $\forall y = 0(y^3 = 0)$, and $\exists z > 0(z^2 = 2)$ mean, where the domain in each case consists of the real numbers?*

Solution: The statement $\forall x < 0(x^2 > 0)$ states that for every real number x with $x < 0$, $x^2 > 0$. That is, it states

"The square of a negative real number is positive." This statement is the same as $\forall x(x < 0 \to x^2 > 0)$.

The statement $\forall y = 0(y^3 = 0)$ states that for every real number y with $y = 0$, we have $y^3 = 0$. That is, it states "The cube of every nonzero real number is nonzero." Note that this statement is equivalent to $\forall\, y(y = 0 \to y^3 = 0)$.

Finally, the statement $\exists z > 0(z^2 = 2)$ states that there exists a real number z with $z > 0$ such that $z^2 = 2$. That is, it states "There is a positive square root of 2." This statement is equivalent to $\exists z(z > 0 \wedge z^2 = 2)$.

1.4.5 Precedence of Quantifiers

When a quantifier is used on the variable x, we say that this occurrence of the variable is **bound**. An occurrence of a variable that is not bound by a quantifier or set equal to a particular value is said to be **free**.

The part of a logical expression to which a quantifier is applied is called the **scope** of this quantifier.

Example 60. *In the statement $\exists x(x + y = 1)$, the variable x is bound by the existential quantification $\exists x$, but the variable y is free because it is not bound by a quantifier and no value is assigned to this variable. This illustrates that in the statement $\exists x(x + y = 1)$, x is bound, but y is free.*

In the statement $\exists x(P(x) \wedge Q(x)) \vee \forall x R(x)$, all variables are bound. The scope of the first quantifier, $\exists x$, is the expression $P(x) \wedge Q(x)$ because $\exists x$ is applied only to $P(x) \wedge Q(x)$, and not to the rest of the statement. Similarly, the scope of the second quantifier, $\forall x$, is the expression $R(x)$. That is, the existential quantifier binds the variable x in $P(x) \wedge Q(x)$ and the universal quantifier $\forall x$ binds the variable x in $R(x)$. Observe that we could have written our statement using two different variables x and y, as $\exists x(P(x) \wedge Q(x)) \vee \forall y R(y)$, because the scopes of the two quantifiers do not overlap. The reader should be aware that in common usage, the same letter is often used to represent variables bound by different quantifiers with scopes that do not overlap.

1.4.6 Negating Quantified Expressions

The rules for negations for quantifiers are called **De Morgans laws for quantifiers**. These rules are summarized in Table 1.18.

Table 1.18: De Morgan's Laws for Quantifiers.

Negation	**Equivalent Statement**	**When Is Negation True?**	**When False?**
$\neg\exists x P(x)$	$\forall x \neg P(x)$	For every x, $P(x)$ is false.	There is an x for which $P(x)$ is true.
$\neg\forall x P(x)$	$\exists x \neg P(x)$	There is an x for which $P(x)$ is false.	$P(x)$ is true for every x.

Example 61. *What are the negations of the statements* $\forall x(x^2 > x)$ *and* $\exists x(x^2 = 2)$*?*

Solution: The negation of $\forall x(x^2 > x)$ is the statement $\neg\forall x(x^2 > x)$, which is equivalent to $\exists x \neg(x^2 > x)$. This can be rewritten as $\exists x(x^2 \leq x)$. The negation of $\exists x(x^2 = 2)$ is the statement $\neg\exists x(x^2 = 2)$, which is equivalent to $\forall x \neg(x^2 = 2)$. This can be rewritten as $\forall x(x^2 = 2)$. The truth values of these statements depend on the domain.

Example 62. *Show that* $\neg\forall x(P(x) \to Q(x))$ *and* $\exists x(P(x) \wedge \neg Q(x))$ *are logically equivalent.*

Solution: By De Morgans law for universal quantifiers, we know that $\neg\forall x(P(x) \to Q(x))$ and $\exists x(\neg(P(x) \to Q(x)))$ are logically equivalent. By the fifth logical equivalence, we know that $\neg(P(x) \to Q(x))$ and

$P(x) \wedge \neg Q(x)$ are logically equivalent for every x. Because we can substitute one logically equivalent expression for another in a logical equivalence, it follows that$\neg\forall x(P(x) \rightarrow Q(x))$ and $\exists x(P(x) \wedge \neg Q(x))$ are logically equivalent.

Let Us Sum Up

In this section, we discussed about

- Predicates.
- Quantifiers.
- Quantifiers over finite domains.
- Predence of quantifiers.

Check your Progress

1. Let $P(x)$ denote the statement $x > 3$. What is truth value of $P(4)$?

(a) True

(b) False

(c) neither true nor false

(d) no truth value

2. Let $Q(x)$ denote the statement $x = x + 1$. What is the truth value of $\exists x Q(x)$?

(a) True

(b) False

(c) neither true nor false

(d) no truth value

3. The part of a logical expression to which a quantifier is applied is called

(a) Universal quantifier

(b) Uniqueness quantifier

(c) Scope of a quantifier

(d) a formula

1.5 Proof Methods and Strategy

1.5.1 Exhaustive Proof and Proof by Cases

Exhaustive Proof

Some theorems can be proved by examining a relatively small number of examples. Such proofs are called **exhaustive proofs**, or **proofs by exhaustion** because these proofs proceed by exhausting all possibilities.

Example 63. *Prove that* $(n + 1)^3 \geq 3^n$ *if* n *is a positive integer with* $n \leq 4$.

Solution: We use a proof by exhaustion. We only need verify the inequality $(n+1)^3 \geq 3^n$ when $n = 1, 2, 3$, and 4. For $n = 1$, we have $(n+1)^3 = 2^3 = 8$ and $3^n = 3^1 = 3$; for $n = 2$, we have $(n + 1)^3 = 3^3 = 27$ and $3^n = 3^2 = 9$; for $n = 3$, we have $(n + 1)^3 = 4^3 = 64$ and $3^n = 3^3 = 27$; and for $n = 4$, we have $(n + 1)^3 = 5^3 = 125$ and $3^n = 3^4 = 81$. In each of these four cases, we see that $(n + 1)3 \geq 3^n$. We have used the method of exhaustion to prove that $(n + 1)3 \geq 3^n$ if n is a positive integer with $n \leq 4$.

Example 64. *Prove that the only consecutive positive integers not exceeding 100 that are perfect powers are 8 and 9. (An integer n is a* **perfect power** *if it equals m^a, where m is an integer and a is an integer greater than 1.)*

Solution: We use a proof by exhaustion. In particular, we can prove this fact by examining positive integers n not exceeding 100, first checking whether n is a perfect power, and if it is, checking whether $n + 1$ is also a perfect power. A quicker way to do this is simply to look at all perfect powers not exceeding 100 and checking whether the next largest integer is also a perfect power. The squares of positive integers not exceeding 100 are $1, 4, 9, 16, 25, 36, 49, 64, 81$, and 100. The cubes of positive integers not exceeding 100 are $1, 8, 27$, and 64. The fourth powers of positive integers not exceeding 100 are $1, 16$, and 81. The fifth powers of positive integers not exceeding 100 are 1 and 32. The sixth powers of positive integers not exceeding 100 are 1 and 64. There are no powers of positive integers higher than the sixth power not exceeding 100, other than 1. Looking at this list of perfect powers not exceeding 100, we see that $n = 8$ is the only perfect power n for which $n + 1$ is also a perfect power. That is, $2^3 = 8$ and $3^2 = 9$ are the only two consecutive perfect powers not exceeding 100.

Proof by Cases

A proof by cases must cover all possible cases that arise in a theorem.

Example 65. *Prove that if n is an integer, then $n^2 \geq n$.*

Solution: We can prove that $n^2 \geq n$ for every integer by considering three cases, when $n = 0$, when $n \geq 1$, and when $n \leq -1$. We split the proof into three cases because it is straightforward to prove the result by considering zero, positive integers, and negative integers separately.

Case (i): When $n = 0$, because $0^2 = 0$, we see that $0^2 \geq 0$. It follows that $n^2 \geq n$ is true in this case.
Case (ii): When $n \geq 1$, when we multiply both sides of the inequality $n \geq 1$ by the positive integer n, we obtain $n \cdot n \geq n \cdot 1$. This implies that $n^2 \geq n$ for $n \geq 1$.
Case (iii): In this case $n \leq -1$. However, $n^2 \geq 0$. It follows that $n^2 \geq n$.

Because the inequality $n^2 \geq n$ holds in all three cases, we can conclude that if n is an integer, then $n^2 \geq n$.

Example 66. *Use a proof by cases to show that $|xy| = |x||y|$, where x and y are real numbers. (Recall that $|a|$, the absolute value of a, equals a when $a \geq 0$ and equals $-a$ when $a \leq 0$.)*

Solution: In our proof of this theorem, we remove absolute values using the fact that $|a| = a$ when $a \geq 0$ and $|a| = -a$ when $a < 0$. Because both $|x|$ and $|y|$ occur in our formula, we will need four cases:

(i) x and y both nonnegative, (ii) x nonnegative and y is negative, (iii) x negative and y nonnegative, and (iv) x negative and y negative. We denote by p_1, p_2, p_3, and p_4, the proposition stating the assumption for each of these four cases, respectively.

(Note that we can remove the absolute value signs by making the appropriate choice of signs within each case.)

Case (i): We see that $p_1 \to q$ because $xy \geq 0$ when $x \geq 0$ and $y \geq 0$, so that $|xy| = xy = |x||y|$.

Case (ii): To see that $p_2 \to q$, note that if $x \geq 0$ and $y < 0$, then $xy \leq 0$, so that $|xy| = -xy = x(-y) = |x||y|$. (Here, because $y < 0$, we have $|y| = -y$.)

Case (iii): To see that $p_3 \to q$, we follow the same reasoning as the previous case with the roles of x and y reversed.

Case (iv): To see that $p_4 \to q$, note that when $x < 0$ and $y < 0$, it follows that $xy > 0$. Hence, $|xy| = xy = (-x)(-y) = |x||y|$.
Because $|xy| = |x||y|$ holds in each of the four cases and these cases exhaust all possibilities, we can conclude that $|xy| = |x||y|$, whenever x and y are real numbers. Many theorems are assertions that objects of a particular type exist. A theorem of this type is a proposition of the form $\exists\ xP(x)$, where P is a predicate. A proof of a proposition of the form $\exists xP(x)$ is called an **existence proof**. There are several ways to prove a theorem of this type. Sometimes an existence proof of $\exists xP(x)$ can be given by finding an element a, called a **witness**, such that $P(a)$ is true. This type of existence proof is called **constructive**. It is also possible to give an existence proof that is **nonconstructive**; that is, we do not find an element a such that $P(a)$ is true, but rather prove that $\exists xP(x)$ is true in some other way. One common method of giving a nonconstructive existence proof is to use proof by contradiction and show that the negation of the existential quantification implies a contradiction.

Example 67. A Constructive Existence Proof *Show that there is a positive integer that can be written as the sum of cubes of positive integers in two different ways.*

Solution: After considerable computation (such as a computer search) we find that

$$1729 = 10^3 + 9^3 = 12^3 + 1^3.$$

Because we have displayed a positive integer that can be written as the sum of cubes in two different ways, we are done.

There is an interesting story pertaining to this example. The English mathematician G. H. Hardy, when visiting the ailing Indian prodigy Ramanujan in the hospital, remarked that 1729, the number of the cab he took, was rather dull. Ramanujan replied “No, it is a very interesting number; it is the smallest number expressible as the sum of cubes in two different ways.”

Example 68. A Nonconstructive Existence Proof *Show that there exist irrational numbers x and y such that x^y is rational.*

Solution: we know that $\sqrt{2}$ is irrational. Consider the number $\sqrt{2}^{\sqrt{2}}$. If it is rational, we have two irrational numbers x and y with xy rational, namely, $x = \sqrt{2}$ and $y = \sqrt{2}$. On the other hand if $\sqrt{2}^{\sqrt{2}}$ is irrational, then we can let $x = \sqrt{2}^{\sqrt{2}}$ and $y = \sqrt{2}$ so that $xy = \left(\sqrt{2}^{\sqrt{2}}\right)^{\sqrt{2}} = \sqrt{2}^{(\sqrt{2}\cdot\sqrt{2})} = \sqrt{2}^2 = 2$.

This proof is an example of a nonconstructive existence proof because we have not found irrational numbers x and y such that xy is rational. Rather, we have shown that either the pair $x = \sqrt{2}$, $y = \sqrt{2}$ or the pair $x = \sqrt{2}^{\sqrt{2}}$, $y = \sqrt{2}$ have the desired property, but we do not know which of these two pairs works!

1.5.2 Uniqueness Proofs

Some theorems assert the existence of a unique element with a particular property. In other words, these theorems assert that there is exactly one element with this property. To prove a statement of this type we need to show that an element with this property exists and that no other element has this property. The two parts of a **uniqueness proof** are:

Existence: We show that an element x with the desired property exists.

Uniqueness: We show that if x and y both have the desired property, then $x = y$.

Remark 69. *Showing that there is a unique element x such that $P(x)$ is the same as proving the statement $\exists x(P(x) \land \forall y(y = x \to \neg P(y)))$.*

Example 70. *Show that if a and b are real numbers and $a = 0$, then there is a unique real number r such that $ar + b = 0$.*

Solution: First, note that the real number $r = -b/a$ is a solution of $ar + b = 0$ because $a(-b/a) + b = -b + b = 0$. Consequently, a real number r exists for which $ar + b = 0$. This is the existence part of the proof.

Second, suppose that s is a real number such that $as + b = 0$. Then $ar + b = as + b$, where $r = -b/a$. Subtracting b from both sides, we find that $ar = as$. Dividing both sides of this last equation by a, which is nonzero, we see that $r = s$. This means that if $s = r$, then $as + b = 0$. This establishes the uniqueness part of the proof.

1.5.3 Proof Strategies

Forward and Backward Reasoning

Whichever method you choose, you need a starting point for your proof. To begin a direct proof of a conditional statement, you start with the premises. Using these premises, together with axioms and known theorems, you can construct a proof using a sequence of steps that leads to the conclusion. This type of reasoning, called *forward reasoning*, is the most common type of reasoning used to prove relatively simple results. Similarly, with indirect reasoning you can start with the negation of the conclusion and, using a sequence of steps, obtain the negation of the premises.

Unfortunately, forward reasoning is often difficult to use to prove more complicated results, because the reasoning needed to reach the desired conclusion may be far from obvious. In such cases it may be helpful to use *backward reasoning*.

Example 71. *Given two positive real numbers x and y, their* **arithmetic mean** *is $(x + y)/2$ and their* **geometric mean** *is $\sqrt{xy}$. When we compare the arithmetic and geometric means of pairs of distinct positive real numbers, we find that the arithmetic mean is always greater than the geometric mean. [For example, when $x = 4$ and $y = 6$, we have $5 = (4 + 6)/2 > \sqrt{4 \cdot 6} = \sqrt{24}$.] Can we prove that this inequality is always true?*

Solution: To prove that $(x + y)/2 > \sqrt{xy}$ when x and y are distinct positive real numbers, we can work backward. We construct a sequence of equivalent inequalities. The equivalent inequalities are

$$\begin{aligned}
(x+y)/2 &> \sqrt{xy}, \\
(x+y)^2/4 &> xy, \\
(x+y)^2 &> 4xy, \\
x^2+2xy+y^2 &> 4xy, \\
x^2-2xy+y^2 &> 0, \\
(x-y)^2 &> 0.
\end{aligned}$$

Because $(x - y)^2 > 0$ when $x \neq y$, it follows that the final inequality is true. Because all these inequalities are equivalent, it follows that $(x + y)/2 > \sqrt{xy}$ when $x \neq y$. Once we have carried out this backward reasoning, we can build a proof based on reversing the steps. This produces construct a proof using forward reasoning.

Backward reasoning

Proof: Suppose that x and y are distinct positive real numbers. Then $(x - y)^2 > 0$ because the square of a nonzero real number is positive. Because $(x - y)^2 = x^2 - 2xy + y^2$, this implies that $x^2 - 2xy + y^2 > 0$. Adding $4xy$ to both sides, we obtain $x^2 + 2xy + y^2 > 4xy$. Because $x^2 + 2xy + y^2 = (x + y)^2$, this means that $(x + y)^2 \geq 4xy$. Dividing both sides of this equation by 4, we see that $(x + y)^2/4 > xy$. Finally, taking square roots of both sides (which preserves the inequality because both sides are positive) yields $(x + y)/2 > \sqrt{xy}$. We conclude that if x and y are distinct positive real numbers, then their arithmetic mean $(x + y)/2$ is greater than their geometric mean $\sqrt{xy}$.

1.5.4 Looking for Counterexamples

When confronted with a conjecture, you might first try to prove this conjecture, and if your attempts are unsuccessful, you might try to find a counterexample, first by looking at the simplest, smallest examples. If you cannot find a counterexample, you might again try to prove the statement. In any case, looking for counterexamples is an extremely important pursuit, which often provides insights into problems.

Let Us Sum Up

In this section, we discussed about

- ∗ Exhaustive proofs and proof by cases.
- ∗ Existence proofs.
- ∗ Uniqueness proofs.
- ∗ Proof strategies.
- ∗ Looking for counter examples.

Check your Progress

1. Some theorems can be proved by examining a relatively small number of examples. Such proof are called

(a) Existence proof

(b) Uniqueness proof

(c) Constructive proof

(d) Exhaustive proof

2. A proof of a proposition of the form $\exists x P(x)$ is called

(a) Existence proof

(b) Uniqueness proof

(c) Constructive proof

(d) Exhaustive proof

1.6 The Growth of Functions

1.6.1 Big-O Notation

Definition 72. *Let f and g be functions from the set of integers or the set of real numbers to the set of real numbers. We say that $f(x)$ is $O(g(x))$ if there are constants C and k such that*

$$|f(x)| \leq C|g(x)|$$

whenever $x > k$. [This is read as "$f(x)$ is big-oh of $g(x)$."]

Working with the Definition of Big-O-Notation

A useful approach for finding a pair of witnesses is to first select a value of k for which the size of $|f(x)|$ can be readily estimated when $x > k$ and to see whether we can use this estimate to find a value of C for which $|f(x)| \leq C|g(x)|$ for $x > k$. This approach is illustrated in Example 73.

Example 73. *Show that $f(x) = x^2 + 2x + 1$ is $O(x^2)$.*

Solution: We observe that we can readily estimate the size of $f(x)$ when $x > 1$ because $x < x^2$ and $1 < x^2$ when $x > 1$. It follows that

$$0 \leq x^2 + 2x + 1 \leq x^2 + 2x^2 + x^2 = 4x^2$$

whenever $x > 1$, as shown in Figure 1.4. Consequently, we can take $C = 4$ and $k = 1$ as witnesses to show that $f(x)$ is $O(x^2)$. That is, $f(x) = x^2 + 2x + 1 < 4x^2$ whenever $x > 1$. (Note that it is not necessary to use absolute values here because all functions in these equalities are positive when x is positive.)

Alternatively, we can estimate the size of $f(x)$ when $x > 2$. When $x > 2$, we have $2x \leq x^2$ and $1 \leq x^2$. Consequently, if $x > 2$, we have

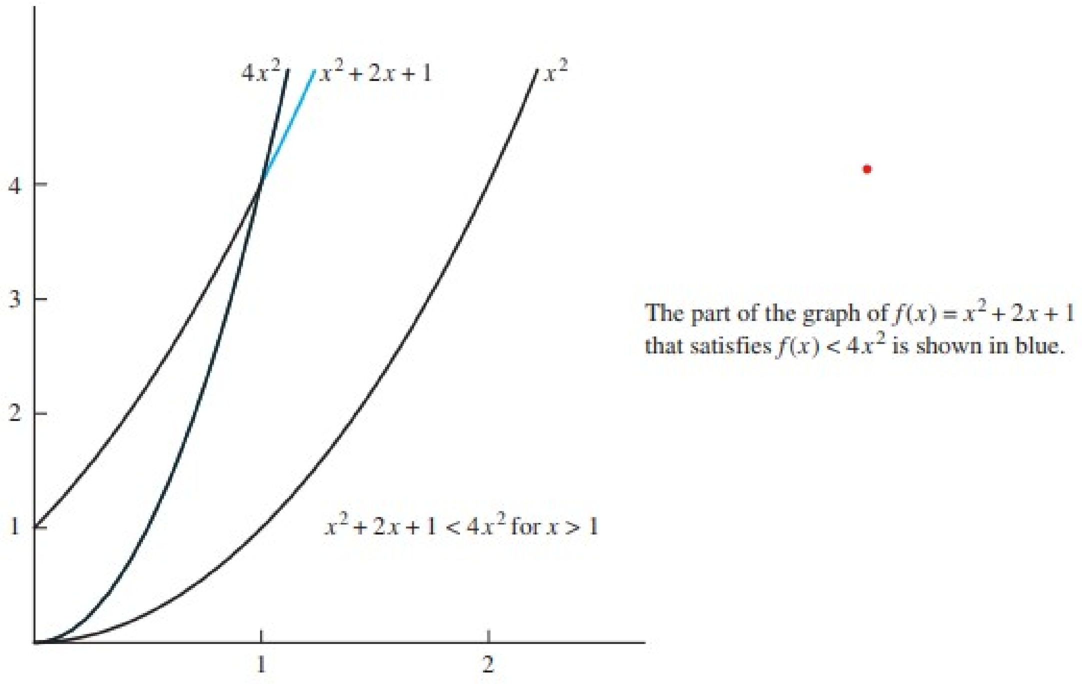


Figure 1.4: The function $x^2 + 2x + 1$ is $O(x^2)$.

$$0 \leq x^2 + 2x + 1 \leq x^2 + x^2 + x^2 = 3x^2.$$

It follows that $C = 3$ and $k = 2$ are also witnesses to the relation $f(x)$ is $O(x^2)$.

Observe that in the relationship "$f(x)$ is $O(x^2)$," x^2 can be replaced by any function with larger values than x^2. For example, $f(x)$ is $O(x^3)$, $f(x)$ is $O(x^2 + x + 7)$, and so on.

It is also true that x^2 is $O(x^2 + 2x + 1)$, because $x^2 < x^2 + 2x + 1$ whenever $x > 1$. This means that $C = 1$ and $k = 1$ are witnesses to the relationship x^2 is $O(x^2 + 2x + 1)$.

Example 74. *Show that* $7x^2$ *is* $O(x^3)$.

Solution: : Note that when $x > 7$, we have $7x^2 < x^3$. (We can obtain this inequality by multiplying both sides of $x > 7$ by x^2.) Consequently, we can take $C = 1$ and $k = 7$ as witnesses to establish the relationship $7x^2$ is $O(x^3)$. Alternatively, when $x > 1$, we have $7x^2 < 7x^3$, so that $C = 7$ and $k = 1$ are also witnesses to the relationship $7x^2$ is $O(x^3)$.

Example 75. *Show that n^2 is not $O(n)$.*

Solution: To show that n^2 is not $O(n)$, we must show that no pair of witnesses C and k exist such that $n^2 \leq Cn$ whenever $n > k$. We will use a proof by contradiction to show this.

Suppose that there are constants C and k for which $n^2 \leq Cn$ whenever $n > k$. Observe that when $n > 0$ we can divide both sides of the inequality $n^2 \leq Cn$ by n to obtain the equivalent inequality $n \leq C$. However, no matter what C and k are, the inequality $n \leq C$ cannot hold for all n with $n > k$. In particular, once we set a value of k, we see that when n is larger than the maximum of k and C, it is not true that $n \leq C$ even though $n > k$. This contradiction shows that n^2 in not $O(n)$.

Example 76. *Whether x^3 is $O(7x^2)$?*

Solution: To determine whether x^3 is $O(7x^2)$, we need to determine whether witnesses C and k exist, so that $x^3 \leq C(7x^2)$ whenever $x > k$. We will show that no such witnesses exist using a proof by contradiction.

If C and k are witnesses, the inequality $x^3 \leq C(7x^2)$ holds for all $x > k$. Observe that the inequality $x^3 \leq C(7x^2)$ is equivalent to the inequality $x \leq 7C$, which follows by dividing both sides by the positive quantity x^2. However, no matter what C is, it is not the case that $x \leq 7C$ for all $x > k$ no matter what k is, because x can be made arbitrarily large. It follows that no witnesses C and k exist for this proposed big-O relationship. Hence, x^3 is not $O(7x^2)$.

1.6.2 Big-O Estimates for Some Important Functions

Theorem 77. *Let $f(x) = a_nx^n + a_{n-1}x^{n-1} + \cdots + a_1x + a_0$, where $a_0, a_1, \cdots, a_{n-1}$, an are real numbers. Then $f(x)$ is $O(x^n)$.*

Proof. Using the triangle inequality, if $x > 1$ we have

$$\begin{aligned}
|f(x)| &= |a_nx^n + a_{n-1}x^{n-1} + \cdots + a_1x + a_0| \\
&\leq |a_n|x^n + |a_{n-1}|x^{n-1} + \cdots + |a_1|x + |a_0| \\
&= x_n(|a_n| + |a_{n-1}|/x + \cdots + |a_1|/x^{n-1} + |a_0|/x^n) \\
&\leq x^n(|a_n| + |a_{n-1}| + \cdots + |a_1| + |a_0|).
\end{aligned}$$

This shows that

$$|f(x)| \leq Cx^n.$$

where $C = |a_n|+|a_{n-1}|+\cdots+|a_0|$ whenever $x > 1$. Hence, the witnesses $C = |a_n| + |a_{n-1}| + \cdots + |a_0|$ and $k = 1$ show that $f(x)$ is $O(x^n)$. □

Example 78. *How can big-O notation be used to estimate the sum of the first n positive integers?*

Solution: Because each of the integers in the sum of the first n positive integers does not exceed n, it follows that

$$1 + 2 + \cdots + n \leq n + n + \cdots + n = n^2.$$

From this inequality it follows that $1 + 2 + 3 + \cdots + n$ is $O(n^2)$, taking $C = 1$ and $k = 1$ as witnesses. (In this example the domains of the functions in the big-O relationship are the set of positive integers.)

In Example 79 big-O estimates will be developed for the factorial function and its logarithm. These estimates will be important in the analysis of the number of steps used in sorting procedures.

Example 79. *Give big-O estimates for the factorial function and the logarithm of the factorial function, where the factorial function* $f(n) = n!$ *is defined by*

$$n! = 1 \cdot 2 \cdot 3 \cdot \cdots \cdot n$$

whenever n *is a positive integer, and* $0! = 1$. *For example,*

$$1! = 1,\ 2! = 1 \cdot 2 = 2,\ 3! = 1 \cdot 2 \cdot 3 = 6,\ 4! = 1 \cdot 2 \cdot 3 \cdot 4 = 24.$$

Note that the function $n!$ *grows rapidly. For instance,*

$$20! = 2,432,902,008,176,640,000.$$

Solution: A big-O estimate for n! can be obtained by noting that each term in the product does not exceed n. Hence,

$$\begin{aligned} n! &= 1 \cdot 2 \cdot 3 \cdot \cdots \cdot n \\ &\leq n \cdot n \cdot n \cdot \cdots \cdot n \\ &= n^n. \end{aligned}$$

This inequality shows that $n!$ is $O(n^n)$, taking $C = 1$ and $k = 1$ as witnesses. Taking logarithms of both sides of the inequality established for $n!$, we obtain

$$log\ n! \leq logn^n = n\ log\ n.$$

This implies that log $n!$ is $O(n\ log\ n)$, again taking $C = 1$ and $k = 1$ as witnesses.

Example 80. *Arrange the functions* $f_1(n) = 8\sqrt{n}$, $f_2(n) = (\log n)^2$, $f_3(n) = 2n\ \log n$, $f_4(n) = n!$, $f_5(n) - (1.1)^n$, *and* $f_6(n) = n^2$ *in a list* *so that each function is big-O of the next function.*

Solution: From the bog-O estimates described in this subsection, we see that $f_2(n) = (\log n)^2$ is the slowest growing of these functions. (This follows because $\log n$ grows slower than any positive power of n.) The next three functions, in order, are $f_1(n) = 8\sqrt{n}$, $f_2(n) = (\log n)^2$, $f_3(n) = 2n \log n$ and $f_6(n) = n^2$. (We know this because $f_1(n) = 8n^{1/2}$, $f_3(n) = 2n \ \log \ n$ is a function that grows faster than n but slower than n^c for every $c > 1$, and $f_6(n) = n^2$ is of the form n^c where $c = 2$.) The next function in the list is $f_5(n) = (.1.1)^n$, because it is an exponential function with base 1.1. Finally, $f_4(n) = n!$ is the faster growing function on the list, because $f(n) = n!$ grows faster than any exponential function of n.

Example 81. *Give a big-O estimate for $f(n) = 3nlog(n!)+(n^2+3)log\, n$, where n is a positive integer.*

Solution: First, the product $3n\ log(n!)$ will be estimated. From Example 73 we know that $log(n!)$ is $O(n\ logn)$. Using this estimate and the fact that $3n$ is $O(n)$, $3nlog(n!)$ is $O(n^2logn)$.

Next, the product $(n^2 + 3)logn$ will be estimated. Because $(n^2 + 3) < 2n^2$ when $n > 2$, it follows that $n^2 + 3$ is $O(n^2)$. Thus, it follows that $(n^2 + 3)logn$ is $O(n^2logn)$. To combine the two big-O estimates for the products shows that $f(n) = 3nlog(n!) + (n^2 + 3)logn$ is $O(n^2logn)$.

Example 82. *Give a big-O estimate for $f(x) = (x+1)log(x^2+1)+3x^2$.*

Solution: First, a big-O estimate for $(x + 1)log(x^2 + 1)$ will be found. Note that $(x + 1)$ is $O(x)$.
Furthermore, $x^2 + 1 \leq 2x^2$ when $x > 1$. Hence,

$$log(x^2 + 1) \leq log(2x^2) = log2 + logx^2 = log\ 2 + 2log\ x \leq 3\ logx,$$

if $x > 2$. This shows that $log(x^2 + 1)$ is $O(logx)$.

It follows that $(x + 1)log(x^2 + 1)$ is $O(x\ logx)$. Because $3x^2$ is $O(x^2)$,we know that $f(x)$ is $O(max(x\ logx, x2))$. Because $x\ logx \leq x^2$, for $x > 1$, it follows that $f(x)$ is $O(x^2)$.

1.6.3 Big-Omega and Big-Theta Notation

Big-O notation is used extensively to describe the growth of functions, but it has limitations. In particular, when $f(x)$ is $O(g(x))$, we have an upper bound, in terms of $g(x)$, for the size of $f(x)$ for large values of x. However, big-O notation does not provide a lower bound for the size of $f(x)$ for large x. For this, we use **big-Omega (big-Ω) notation**. When we want to give both an upper and a lower bound on the size of a function $f(x)$, relative to a reference function $g(x)$, we use **big-Theta (big-Θ) notation**. Both big-Omega and big-Theta notation were introduced by Donald Knuth in the 1970s. His motivation for introducing these notations was the common misuse of big-O notation when both an upper and a lower bound on the size of a function are needed. We now define big-Omega notation and illustrate its use. After doing so, we will do the same for big-Theta notation.

Definition 83. *Let f and g be functions from the set of integers or the set of real numbers to the set of real numbers. We say that $f(x)$ is $\Omega(g(x))$ if there are positive constants C and k such that*

$$|f(x)| \geq C|g(x)|$$

whenever $x > k$. [This is read as "$f(x)$ is big-Omega of $g(x)$."]

Example 84. *The function $f(x) = 8x^3 + 5x^2 + 7$ is $\Omega(g(x))$, where*

$g(x)$ is the function $g(x) = x^3$. This is easy to see because $f(x) = 8x^3 + 5x^2 + 7 \geq 8x^3$ for all positive real numbers x. This is equivalent to saying that $g(x) = x^3$ is $O(8x^3 + 5x^2 + 7)$, which can be established directly by turning the inequality around.

Definition 85. *Let f and g be functions from the set of integers or the set of real numbers to the set of real numbers. We say that $f(x)$ is $\Theta(g(x))$ if $f(x)$ is $O(g(x))$ and $f(x)$ is $\Omega(g(x))$. When $f(x)$ is $\Theta(g(x))$ we say that f is big-Theta of $g(x)$, that $f(x)$ is of order $g(x)$, and that $f(x)$ and $g(x)$ are of the same order.*

Example 86. *Show that $3x^2 + 8x\ logx$ is $\Theta(x^2)$.*

Solution: Because $0 \leq 8x\ log\ x \leq 8x^2$, it follows that $3x^2 + 8x\ log\ x \leq 11x^2$ for $x > 1$. Consequently, $3x^2 + 8x\ log\ x$ is $O(x^2)$. Clearly, x^2 is $O(3x^2 + 8x\ logx)$. Consequently, $3x^2 + 8x\ log\ x$ is $\Theta(x^2)$.

Let us Sum Up

In this section, we discussed about

* Big-O-Notation.
* Big -O-estimates for some important functions.
* Big-Omega and Big-Theta Notation.

Check your Progress

1. An advantage of big-O notation is

(a) to write algorithms

(b) to estimate growth of a function

(c) to find combinations of functions

(d) to find bounds of functions

2. The big-O estimate of $f(x) = (x+1)\log(x^2+1) + 3x^2$ is

(a) $O(\log x)$

(b) $O(x^4)$

(c) $O(x^2 + \log x)$

(d) $O(x^2)$

Summary

In this unit, we discussed about

* Propositional logic.
* Applications of Propositional logic.
* Propositional equivalences.
* Predicates and quantifiers.
* Proof methods and strategy.
* The growth of functions.

Glossary

proposition: a statement that is true or false.

propositional variable: a variable that represents a proposition.

truth value: true or false.

bit: either a 0 or a 1.

bit string: a list of bits.

tautology: a compound proposition that is always true.

contradiction: a compound proposition that is always false.

free variable: a variable not bound in a propositional function.

bound variable: a variable that is quantified.

scope of quantifiers: portion of a statement where the quantifier binds its variable.

argument: a sequence of statements.

Self Assessment Questions

1. Define (using truth tables) the disjunction, conjunction, exclusive or, conditional, and biconditional of the propositions p and q.

2. Describe at least five different ways to write the conditional statement $p \leftrightarrow q$ in English.

3. Describe a way to prove the biconditional $p \leftrightarrow q$.

4. What is the difference between a constructive and non-constructive existence proof? Give an example of each.

5. What are the elements of a proof that there is a unique element x such that $P(x)$, where $P(x)$ is a propositional function?

Exercises

1. Find the truth table of the compound proposition $(p \vee q) \rightarrow (p \wedge \neg r)$.

2. Prove or disprove that if x^2 is irrational, then x^3 is irrational.

3. Prove that given a nonegative integer n, there is a unique nonnegative integer m such that $m^2 \leq n \leq (m+1)^2$.

4. Disprove the statement that every positive integer is the sum of the cubes of eight nonegative integers.

5. Show that $(n \log n + n^2)^3$ is $O(n^6)$.

Answers to Check your Progress

Section 1.1:	1)a	2)c	3) d
Section 1.2:	1)b	2)a	
Section 1.3:	1)b	2)c	
Section 1.4:	1)a	2)b	3)c
Section 1.5:	1)d	2)a	
Section 1.6:	1)b	2)c	

Reference:

1. Kenneth H. Rosen, “Discrete Mathematics and its Applications”, 7th Edition, WCB/ McGraw Hill Publications, New Delhi, 2011.

Suggested Readings:

1. Edward A. Bender and S. Gill Williamson, “A Short Course in Discrete Mathematics”, Dover Publications, 2006.

2. M.O. Albertson and J.P. Hutchinson, “Discrete Mathematics with Algorithms”, John Wiley & Sons, 2008.

3. Rajendra Akerkar and Rupali Akarkar, “Discrete Mathematics”, Pearson Education Pvt. Ltd, Singapore, 2004.

4. J. P. Trembley and R. Manohar, “Discrete Mathematical Structures”, Tata McGraw Hill, New Delhi,1997.

5. Martin Aigner, “A Course in Enumeration”, Springer-Verlag, Heidelberg, 2007.

6. J.H. Van Lint and R.M. Wilson, “A Course in Combinatorics”, 2nd Edition, Cambridge University Press, Cambridge, 2001.

Unit 2

Counting

Objectives

1. To introduce the basic techniques of counting.
2. To illustrate the usefulness of the pigeohole principle.
3. To learn permutations and combinations and to apply them to answer counting problems.
4. To introduce the idea of generalized permutations and combinations to solve problems when repetition of elements is allowed.

2.1 The Basics of Counting

2.1.1 Basic Counting Principles

We first present two basic counting principles, the **product rule** and the **sum rule**. Then we will show how they can be used to solve many different counting problems.

The product rule applies when a procedure is made up of separate tasks.

The Product Rule: Suppose that a procedure can be broken down into a sequence of two tasks. If there are n_1 ways to do the first task and for each of these ways of doing the first task, there are n_2 ways to do the second task, then there are n_1n_2 ways to do the procedure.

Example 87. *A new company with just two employees, Sanchez and Patel, rents a floor of a building with 12 offices. How many ways are there to assign different offices to these two employees?*

Solution: The procedure of assigning offices to these two employees consists of assigning an office to Sanchez, which can be done in 12 ways, then assigning an office to Patel different from the office assigned to Sanchez, which can be done in 11 ways. By the product rule, there are $12 \cdot 11 = 132$ ways to assign offices to these two employees.

Example 88. *The chairs of an auditorium are to be labeled with an uppercase English letter followed by a positive integer not exceeding 100. What is the largest number of chairs that can be labeled differently?*

Solution: The procedure of labeling a chair consists of two tasks, namely, assigning to the seat one of the 26 uppercase English letters, and then assigning to it one of the 100 possible integers. The product rule shows that there are $26 \cdot 100 = 2600$ different ways that a chair can be labeled. Therefore, the largest number of chairs that can be labeled differently is 2600.

Example 89. *There are 32 computers in a data center in the cloud. Each of these computers has 24 ports. How many different computer parts are there in this data center?*

Solution: : The procedure of choosing a port consists of two tasks, first picking a microcomputer and then picking a port on this microcomputer.

Because there are 32 ways to choose the microcomputer and 24 ways to choose the port no matter which microcomputer has been selected, the product rule shows that there are $32 \cdot 24 = 768$ ports.

An extended version of the product rule is often useful. Suppose that a procedure is carried out by performing the tasks $T_1, T_2, \cdots, T_m$ in sequence. If each task $T_i,\ i = 1, 2, \cdots, n$, can be done in ni ways, regardless of how the previous tasks were done, then there are $n_1 \cdot n_2 \cdot \cdots \cdot n_m$ ways to carry out the procedure. This version of the product rule can be proved by mathematical induction from the product rule for two tasks.

Example 90. *How many different bit strings of length seven are there?*

Solution: Each of the seven bits can be chosen in two ways, because each bit is either 0 or 1. Therefore, the product rule shows there are a total of $27 = 128$ different bit strings of length seven.

Example 91. *How many different license plates can be made if each plate contains a sequence of three upper-case English letters followed by three digits (and no sequences of letters are prohibited, even if they are obscene)?*

Solution: There are 26 choices for each of the three upper-case English letters and ten choices for each of the three digits. Hence, by the product rule there are a total of $26 \cdot 26 \cdot 26 \cdot 10 \cdot 10 \cdot 10 = 17,576,000$ possible license plates.

Example 92. Counting Functions *How many functions are there from a set with m elements to a set with n elements?*

Solution: A function corresponds to a choice of one of the n elements in the codomain for each of the m elements in the domain. Hence, by the product rule there are $n \cdot n \cdot \cdots \cdot n = nm$ functions from a set with m elements to one with n elements. For example, there are $53 = 125$ different functions from a set with three elements to a set with five elements.

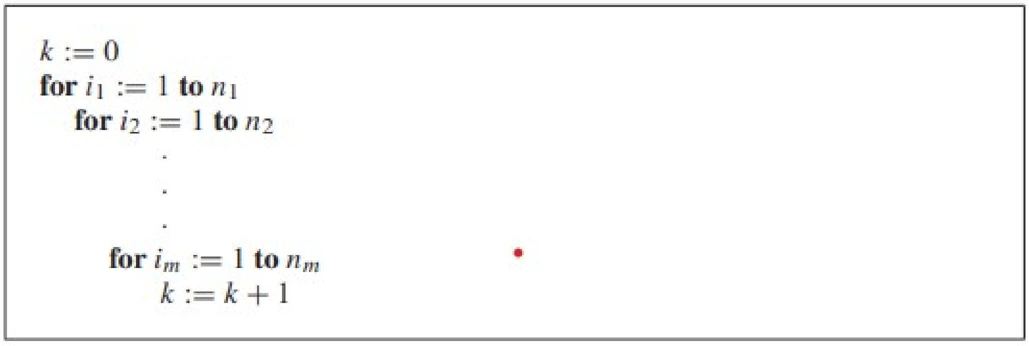
Example 93. Counting One-to-One Functions *How many one-to-one functions are there from a set with m elements to one with n elements?*

Solution: First note that when $m > n$ there are no one-to-one functions from a set with m elements to a set with n elements.

Now let $m \leq n$. Suppose the elements in the domain are $a_1, a_2, \cdots, a_m$. There are n ways to choose the value of the function at a_1. Because the function is one-to-one, the value of the function at a_2 can be picked in $n - 1$ ways (because the value used for a_1 cannot be used again). In general, the value of the function at a_k can be chosen in $n - k + 1$ ways. By the product rule, there are $n(n-1)(n-2)\cdots(n-m+1)$ one-to-one functions from a set with m elements to one with n elements.

For example, there are $5 \cdot 4 \cdot 3 = 60$ one-to-one functions from a set with three elements to a set with five elements.

Example 94. *What is the value of k after the following code, where $n_1, n_2, \cdots, n_m$ are positive integers, has been executed?*



```
k := 0
for i_1 := 1 to n_1
    for i_2 := 1 to n_2
            .
            .
            .
        for i_m := 1 to n_m
            k := k + 1
```

Solution: The initial value of k is zero. Each time the nested loop is traversed, 1 is added to k. Let Ti be the task of traversing the i^{th} loop.

Then the number of times the loop is traversed is the number of ways to do the tasks $T_1, T_2, \cdots, T_m$. The number of ways to carry out the task $T_j,\ \ j = 1, 2, \cdots, m$, is n_j, because the j^{th} loop is traversed once for each integer i_j with $1 \leq i_j \leq n_j$. By the product rule, it follows that the nested loop is traversed $n_1 n_2 \cdots n_m$ times. Hence, the final value of k is $n_1 n_2 \cdots n_m$.

Example 95. Counting Subsets of a Finite Set *Use the product rule to show that the number of different subsets of a finite set S is $2^{|S|}$.*

Solution: Let S be a finite set. List the elements of S in arbitrary order. There is a one-to-one correspondence between subsets of S and bit strings of length $|S|$. Namely, a subset of S is associated with the bit string with a 1 in the i^{th} position if the i^{th} element in the list is in the subset, and a 0 in this position otherwise. By the product rule, there are $2^{|S|}$ bit strings of length $|S|$. Hence, $|P(S)| = 2|S|$.

The product rule is often phrased in terms of sets in this way: If $A_1, A_2, \cdots, A_m$ are finite sets, then the number of elements in the Cartesian product of these sets is the product of the number of elements in each set. To relate this to the product rule, note that the task of choosing an element in the Cartesian product $A_1 \times A_2 \times \cdots \times A_m$ is done by choosing an element in A_1, an element in $A_2, \cdots$, and an element in A_m. By the product rule it follows that

$$|A_1 \times A_2 \times \cdots \times A_m| = |A_1| \cdot |A_2| \cdot \cdots \cdot |A_m|.$$

The Sum Rule: If a task can be done either in one of n_1 ways or in one of n_2 ways, where none of the set of n_1 ways is the same as any of the set of n_2 ways, then there are $n_1 + n_2$ ways to do the task.

Example 96. *Suppose that either a member of the mathematics faculty* *or a student who is a mathematics major is chosen as a representative to a university committee. How many different choices are there for this representative if there are 37 members of the mathematics faculty and 83 mathematics majors and no one is both a faculty member and a student?*

Solution: There are 37 ways to choose a member of the mathematics faculty and there are 83 ways to choose a student who is a mathematics major. Choosing a member of the mathematics faculty is never the same as choosing a student who is a mathematics major because no one is both a faculty member and a student. By the sum rule it follows that there are $37 + 83 = 120$ possible ways to pick this representative.

We can extend the sum rule to more than two tasks. Suppose that a task can be done in one of n_1 ways, in one of n_2 ways, $\cdots$, or in one of nm ways, where none of the set of n_i ways of doing the task is the same as any of the set of n_j ways, for all pairs i and j with $1 \leq i < j \leq m$. Then the number of ways to do the task is $n_1 + n_2 + \cdots + n_m$. This extended version of the sum rule is often useful in counting problems. This version of the sum rule can be proved using mathematical induction from the sum rule for two sets.

Example 97. *A student can choose a computer project from one of three lists. The three lists contain* 23, 15, *and* 19 *possible projects, respectively. No project is on more than one list. How many possible projects are there to choose from?*

Solution: The student can choose a project by selecting a project from the first list, the second list, or the third list. Because no project is on more than one list, by the sum rule there are $23 + 15 + 19 = 57$ ways to choose a project.

Example 98. *What is the value of k after the following code, where $n_1, n_2, \cdots, n_m$ are positive integers, has been executed?*

```
$k := 0$
for $i_1 := 1$ to $n_1$
    $k := k + 1$
for $i_2 := 1$ to $n_2$
    $k := k + 1$
    .
    .
    .
for $i_m := 1$ to $n_m$
    $k := k + 1$
```

Solution: The initial value of k is zero. This block of code is made up of m different loops. Each time a loop is traversed, 1 is added to k. To determine the value of k after this code has been executed, we need to determine how many times we traverse a loop. Note that there are n_i ways to traverse the i^{th} loop. Because we only traverse one loop at a time, the sum rule shows that the final value of k, which is the number of ways to traverse one of the m loops is $n_1 + n_2 + \cdots + n_m$.

The sum rule can be phrased in terms of sets as: If $A_1, A_2, \cdots, A_m$ are pairwise disjoint finite sets, then the number of elements in the union of these sets is the sum of the numbers of elements in the sets. To relate this to our statement of the sum rule, note there are $|A_i|$ ways to choose an element from A_i for $i = 1, 2, \cdots, m$. Because the sets are pairwise disjoint, when we select an element from one of the sets A_i, we do not also select an element from a different set A_j . Consequently, by the sum rule, because we cannot select an element from two of these sets at the same time, the number of ways to choose an element from one of the sets, which is the number of elements in the union, is

$$|A_1 \cup A_2 \cup \cdots \cup A_m| = |A_1| + |A_2| + \cdots + |A_m| \quad \text{when } A_i \cap A_j = \quad \text{for all } i, j.$$

This equality applies only when the sets in question are pairwise disjoint.

2.1.2 The Subtraction Rule (Inclusion-Exclusion for Two Sets)

The Subtraction Rule: If a task can be done in either n_1 ways or n_2 ways, then the number of ways to do the task is $n_1 + n_2$ minus the number of ways to do the task that are common to the two different ways.

The subtraction rule is also known as the principle of inclusion-exclusion, especially when it is used to count the number of elements in the union of two sets. Suppose that A_1 and A_2 are sets. Then, there are $|A_1|$ ways to select an element from A_1 and $|A_2|$ ways to select an element from A_2. The number of ways to select an element from A_1 or from A_2, that is, the number of ways to select an element from their union, is the sum of the number of ways to select an element from A_1 and the number of ways to select an element from A_2, minus the number of ways to select an element that is in both A_1 and A_2. Because there are $|A_1 \cup A_2|$ ways to select an element in either A_1 or in A_2, and $|A_1 \cap A_2|$ ways to select an element common to both sets, we have

$$|A_1 \cup A_2| = |A_1| + |A_2| - |A_1 \cap A_2|.$$

Example 99. *How many bit strings of length eight either start with a 1 bit or end with the two bits 00?*

Solution: Figure 1 illustrates the three counting problems we need to solve before we can apply the principle of inclusion-exclusion.

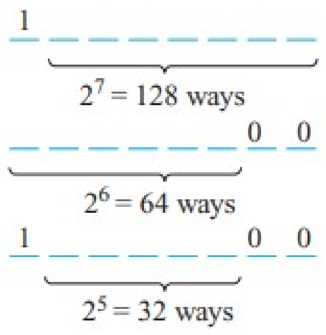


Figure 2.1: Strings starting with 1 or ending with 00

We can construct a bit string of length eight that either starts with a 1 bit or ends with the two bits 00, by constructing a bit string of length eight beginning with a 1 bit or by constructing a bit string of length eight that ends with the two bits 00. We can construct a bit string of length eight that begins with a 1 in $27 = 128$ ways. This follows by the product rule, because the first bit can be chosen in only one way and each of the other seven bits can be chosen in two ways. Similarly, we can construct a bit string of length eight ending with the two bits 00, in $2^6 = 64$ ways. This follows by the product rule, because each of the first six bits can be chosen in two ways and the last two bits can be chosen in only one way.

Some of the ways to construct a bit string of length eight starting with a 1 are the same as the ways to construct a bit string of length eight that ends with the two bits 00. There are $2^5 = 32$ ways to construct such a string. This follows by the product rule, because the first bit can be chosen in only one way, each of the second through the sixth bits can be chosen in two ways, and the last two bits can be chosen in one way. Consequently, the number of bit strings of length eight that begin with a 1 or end with a 00, which equals the number of ways to construct a bit string of length eight that begins with a 1 or that ends with 00, equals $128+64-32 = 160$.

Example 100. *A computer company receives* 350 *applications from computer graduates for a job planning a line of new Web servers. Suppose that* 220 *of these applicants majored in computer science,* 147 *majored in business, and* 51 *majored both in computer science and in business. How many of these applicants majored neither in computer science nor in business?*

Solution: To find the number of these applicants who majored neither in computer science nor in business, we can subtract the number of students who majored either in computer science or in business (or both) from the total number of applicants. Let A_1 be the set of students who majored in computer science and A_2 the set of students who majored in business. Then $A_1 \cup A_2$ is the set of students who majored in computer science or business (or both), and $A_1 \cap A_2$ is the set of students who majored both in computer science and in business. By the subtraction rule the number of students who majored either in computer science or in business (or both) equals

$$|A_1 \cup A_2| = |A_1| + |A_2| - |A_1 \cap A_2| = 220 + 147 - 51 = 316.$$

We conclude that $350 - 316 = 34$ of the applicants majored neither in computer science nor in business. A venn diagram for this example is shown in Figure 2.

2.1.3 The Division Rule

The Division Rule: There are n/d ways to do a task if it can be done using a procedure that can be carried out in n ways, and for every way w,

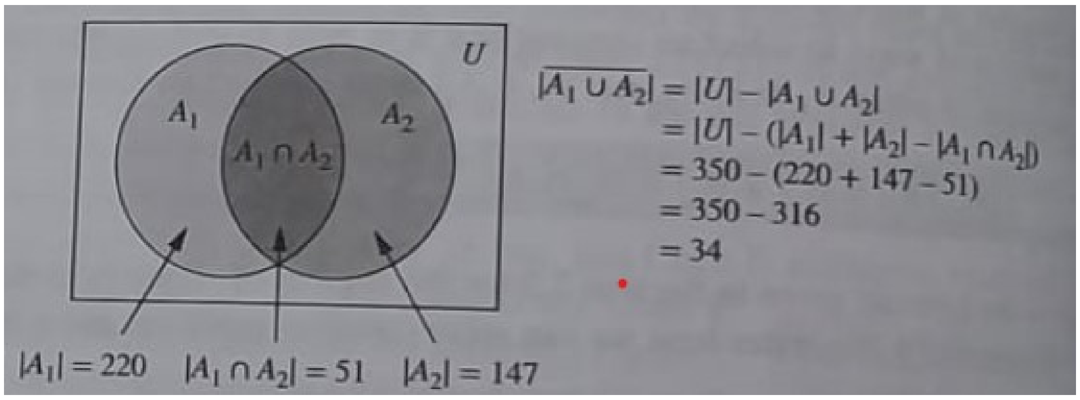


Figure 2.2: Applicants who majored in neither computer science nor business

exactly d of the n ways correspond to way w.

Example 101. *Suppose that an automated system has been developed that counts the legs of cows in a pasture. Suppose that this system has determined that in a farmer's pasture there are exactly 572 legs. How many cows are there is thus pasture, assuming that each cow has four legs and that there are no other animals present?*

Solution: Let n be the number of cow legs counted in a pasture. Because each cow has four legs, by the division rule we know that the pasture contains $n/4$ cows. Consequently, the pasture with 572 cow legs has $572/4 = 143$ cows in it.

Example 102. *How many different ways are there to seat four people around a circular table, where two seatings are considered the same when each person has the same left neighbor and the same right neighbor?*

Solution: We arbitrarily select a seat at the table and label it seat 1. We number the rest of the seats in numerical order, proceeding clockwise around the table. Note that are four ways to select the person for seat 1, three ways to select the person for seat 2, two ways to select the person for seat 3, and one way to select the person for seat 4. Thus, there are $4! = 24$ ways to order the given four people for these seats. However, each of the four choices for seat 1 leads to the same arrangement, as we distinguish two arrangements only when one of the people has a different immediate left or immediate right neighbor. Because there are four ways to choose the person for seat 1, by the division rule there are $24/4 = 6$ different seating arrangements of four people around the circular table.

Example 103. *Suppose that "I Love New Jersey" T-shirts come in five different sizes: S, M, L, XL, and XXL. Further suppose that each size comes in four colors, white, red, green, and black, except for XL, which comes only in red, green, and black, and XXL, which comes only in green and black. How many different shirts does a souvenir shop have to stock to have at least one of each available size and color of the T-shirt?*

Solution: The tree diagram in Figure 3 displays all possible size and color pairs. It follows that the souvenir shop owner needs to stock 17 different T-shirts.

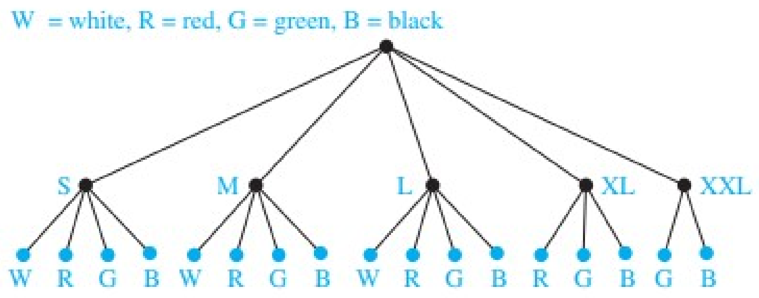

W = white, R = red, G = green, B = black
S
M
L
XL
XXL
W R G B W R G B W R G B R G B G B

Figure 2.3: Counting varieties of T-shirts

Let Us Sum Up

In this section, we discussed about

* Basic counting principles.
* The subtraction rule.
* The division rule.

Check your Progress

1. How many different bit strings of length seven are there?

   (a) 28

   (b) 100

   (c) 120

   (d) 128

2. The subtraction rule is also known as

   (a) principle of inclusion-exclusion

   (b) principle of counting

   (c) pigeonhole principle

   (d) the division rule

2.2 The Pigeonhole Principle

2.2.1 Introduction

Suppose that a flock of 20 pigeons flies into a set of 19 pigeonholes to roost. Because there are 20 pigeons but only 19 pigeonholes, a least one of these 19 pigeonholes must have at least two pigeons in it. To see why this is true, note that if each pigeonhole had at most one pigeon in it, at most 19 pigeons, one per hole, could be accommodated. This illustrates a general principle called the **pigeonhole principle**, which states that if there are more pigeons than pigeonholes, then there must be at least one pigeonhole with at least two pigeons in it.

Theorem 104. [The Pigeonhole Principle] *If k is a positive integer and $k + 1$ or more objects are placed into k boxes, then there is at least one box containing two or more of the objects.*

Proof. We prove the pigeonhole principle using a proof by contraposition. Suppose that none of the k boxes contains more than one object. Then the total number of objects would be at most k. This is a contradiction, because there are at least $k + 1$ objects. □

Corollary 105. *A function f from a set with $k + 1$ or more elements to a set with k elements is not one-to-one.*

Proof. Suppose that for each element y in the codomain of f we have a box that contains all elements x of the domain of f such that $f(x) = y$. Because the domain contains $k + 1$ or more elements and the codomain contains only k elements, the pigeonhole principle tells us that one of these boxes contains two or more elements x of the domain. This means that f cannot be one-to-one. □

Example 106. *How many students must be in a class to guarantee that at least two students receive the same score on the final exam, if the exam is graded on a scale from 0 to 100 points?*

Solution: There are 101 possible scores on the final. The pigeonhole principle shows that among any 102 students there must be at least 2 students with the same score.

Example 107. *Show that for every integer n there is a multiple of n that has only 0s and 1s in its decimal expansion.*

Solution: Let n be a positive integer. Consider the $n + 1$ integers $1, 11, 111, \cdots, 11\cdots1$ (where the last integer in this list is the integer with $n + 1$ 1s in its decimal expansion). Note that there are n possible remainders when an integer is divided by n. Because there are $n + 1$ integers in this list, by the pigeonhole principle there must be two with the same remainder when divided by n. The larger of these integers less the smaller one is a multiple of n, which has a decimal expansion consisting entirely of 0s and 1s.

2.2.2 The Generalized Pigeonhole Principle

Theorem 108. The Generalized Pigeonhole Principle *If N objects are placed into k boxes, then there is at least one box containing at least $\lceil N/k \rceil$ objects.*

Proof. We will use a proof by contraposition. Suppose that none of the boxes contains more than $\lceil N/k \rceil - 1$ objects. Then, the total number of objects is at most

$$k\left(\lceil\frac{N}{k}\rceil - 1\right) < k\left(\left(\frac{N}{k} + 1\right) - 1\right) = N,$$

where the inequality $\lceil N/k \rceil < (N/k) + 1$ has been used. This is a contradiction because there are a total of N objects. $\square$

Example 109. *What is the minimum number of students required in a discrete mathematics class to be sure that at least six will receive the same grade, if there are five possible grades, A, B, C, D, and F?*

Solution: The minimum number of students needed to ensure that at least six students receive the same grade is the smallest integer N such that $\lceil N/5 \rceil = 6$. The smallest such integer is $N = 5 \cdot 5 + 1 = 26$. If you have only 25 students, it is possible for there to be five who have received each grade so that no six students have received the same grade. Thus, 26 is the minimum number of students needed to ensure that at least six students will receive the same grade.

Example 110. *What is the least number of area codes needed to guarantee that the* 25 *million phones in a state can be assigned distinct 10-digit telephone numbers? (Assume that telephone numbers are of the form NXX-NXX-XXXX, where the first three digits form the area code, N represents a digit from* 2 *to* 9 *inclusive, and X represents any digit.)*

Solution: There are eight million different phone numbers of the form NXX-XXXX. Hence, by the generalized pigeonhole principle, among 25 million telephones, at least $\lceil 25,000,000/8,000,000 \rceil = 4$ of them must have identical phone numbers. Hence, at least four area codes are required to ensure that all 10-digit numbers are different.

2.2.3 Some Elegant Applications of the Pigeonhole Principle

Example 111. *During a month with 30 days, a baseball team plays at least one game a day, but no more than 45 games. Show that there must* *be a period of some number of consecutive days during which the team must play exactly 14 games.*

Solution: Let a_j be the number of games played on or before the j^{th} day of the month. Then $a_1, a_2, \cdots, a_{30}$ is an increasing sequence of distinct positive integers, with $1 \leq a_j \leq 45$. Moreover, $a_1+14, a_2+14, \cdots, a_{30}+14$ is also an increasing sequence of distinct positive integers, with $15 \leq a_j + 14 \leq 59$. The 60 positive integers $a_1, a_2, \cdots, a_{30}$, $a_1 + 14, a_2 + 14, \cdots, a_{30}+14$ are all less than or equal to 59. Hence, by the pigeonhole principle two of these integers are equal. Because the integers a_j, $j = 1, 2, \cdots, 30$ are all distinct and the integers $a_j + 14$, $j = 1, 2, \cdots, 30$ are all distinct, there must be indices i and j with $a_i = a_j + 14$. This means that exactly 14 games were played from day $j + 1$ to day i.

Example 112. *Show that among any $n+1$ positive integers not exceeding $2n$ there must be an integer that divides one of the other integers.*

Solution: Write each of the $n+1$ integers $a_1, a_2, \cdots, a_{n+1}$ as a power of 2 times an odd integer. In other words, let $a_j = 2^{k_j} q_j$ for $j = 1, 2, \cdots, n+1$, where k_j is a nonnegative integer and q_j is odd. The integers $q_1, q_2, \cdots, q_{n+1}$ are all odd positive integers less than $2n$. Because there are only n odd positive integers less than $2n$, it follows from the pigeonhole principle that two of the integers $q_1, q_2, \cdots, q_{n+1}$ must be equal. Therefore, there are distinct integers i and j such that $q_i = q_j$. Let q be the common value of q_i and q_j. Then, $a_i = 2^{k_i} q$ and $a_j = 2^{k_j} q$. It follows that if $k_i < k_j$, then a_i divides a_j; while if $k_i > k_j$, then a_j divides a_i.

Suppose that $a_1, a_2, \cdots, a_N$ is a sequence of real numbers. A subsequence of this sequence is a sequence of the form $a_{i_1}, a_{i_2}, \cdots, a_{i_m}$, where $1 \leq i_1 < i_2 < \cdots < i_m \leq N$. Hence, a subsequence is a sequence obtained from the original sequence by including some of the terms of the original sequence in their original order, and perhaps not including other terms. A sequence is called strictly increasing if each term is larger than the one that precedes it, and it is called strictly decreasing if each term is smaller than the one that precedes it.

Theorem 113. *Every sequence of $n^2 + 1$ distinct real numbers contains a subsequence of length $n + 1$ that is either strictly increasing or strictly decreasing.*

Example 114. *The sequence 8, 11, 9, 1, 4, 6, 12, 10, 5, 7 contains 10 terms. Note that $10 = 32 + 1$. There are four strictly increasing subsequences of length four, namely, 1, 4, 6, 12; 1, 4, 6, 7; 1, 4, 6, 10; and 1, 4, 5, 7. There is also a strictly decreasing subsequence of length four, namely, 11, 9, 6, 5.*

Solution: Let $a_1, a_2, \cdots, a_{n^2+1}$ be a sequence of n^2+1 distinct real numbers.Associate an ordered pair with each term of the sequence, namely, associate (i_k, d_k) to the term a_k, where i_k is the length of the longest increasing subsequence starting at a_k, and d_k is the length of the longest decreasing subsequence starting at a_k.

Suppose that there are no increasing or decreasing subsequences of length $n + 1$. Then i_k and d_k are both positive integers less than or equal to n, for $k = 1, 2, \cdots, n^2 + 1$. Hence, by the product rule there are n^2 possible ordered pairs for (i_k, d_k). By the pigeonhole principle, two of these $n^2 + 1$ ordered pairs are equal. In other words, there exist terms a_s and a_t , with $s < t$ such that $i_s = i_t$ and $d_s = d_t$. We will show that this is impossible. Because the terms of the sequence are distinct, either $a_s < a_t$ or $a_s > a_t$. If $a_s < a_t$, then, because $i_s = i_t$, an increasing subsequence of length $i_t + 1$ can be built starting at a_s, by taking a_s followed by an increasing subsequence of length i_t beginning at a_t. This is a contradiction.

Similarly, if $a_s > a_t$, the same reasoning shows that d_s must be greater than d_t, which is a contradiction.

Let Us Sum Up

In this section, we discussed about

$*$ The pigeonhole principle.

$*$ The generalized pigeonhole principle.

$*$ Applications of pigeohole principle.

Check your Progress

1. The pigeonhole principle is known as

(a) Principle of counting

(b) Dirichlet drawer principle

(c) Principle of inclusion-exclusion

(d) Principle of inclusion

2. Among 100 people there are at least ________ who were born in the same month.

(a) 12

(b) 10

(c) 9

(d) 6

2.3 Permutations and Combinations

2.3.1 Permutations

Example 115. *In how many ways can we select three students from a group of five students to stand in line for a picture? In how many ways can we arrange all five of these students in a line for a picture?*

Solution: First, note that the order in which we select the students matters. There are five ways to select the first student to stand at the start of the line. Once this student has been selected, there are four ways to select the second student in the line. After the first and second students have been selected, there are three ways to select the third student in the line. By the product rule, there are $5 \cdot 4 \cdot 3 = 60$ ways to select three students from a group of five students to stand in line for a picture.

To arrange all five students in a line for a picture, we select the first student in five ways, the second in four ways, the third in three ways, the fourth in two ways, and the fifth in one way. Consequently, there are $5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120$ ways to arrange all five students in a line for a picture.

A **permutation** of a set of distinct objects is an ordered arrangement of these objects. We also are interested in ordered arrangements of some of the elements of a set. An ordered arrangement of r elements of a set is called an **r-permutation**.

Example 116. *Let $S = \{1, 2, 3\}$. The ordered arrangement 3, 1, 2 is a permutation of S. The ordered arrangement 3, 2 is a 2-permutation of S.*

The number of r permutations of a set with n elements is denoted by $P(n, r)$. We can find $P(n, r)$ using the product rule.

Example 117. *Let $S = \{a, b, c\}$. The 2-permutations of S are the ordered arrangements a, b; a, c; b, a; b, c; c, a; and c, b. Consequently, there* *are six 2-permutations of this set with three elements. There are always six 2-permutations of a set with three elements. There are three ways to choose the first element of the arrangement. There are two ways to choose the second element of the arrangement, because it must be different from the first element. Hence, by the product rule, we see that* $P(3,2) = 3 \cdot 2 = 6$. *the first element. By the product rule, it follows that* $P(3,2) = 3 \cdot 2 = 6$.

Theorem 118. *If n is a positive integer and r is an integer with $1 \leq r \leq n$, then there are*

$$P(n,r) = n(n-1)(n-2)\cdots(n-r+1)$$

r-permutations of a set with n distinct elements.

Proof. We will use the product rule to prove that this formula is correct. The first element of the permutation can be chosen in n ways because there are n elements in the set. There are $n-1$ ways to choose the second element of the permutation, because there are $n-1$ elements left in the set after using the element picked for the first position. Similarly, there are $n-2$ ways to choose the third element, and so on, until there are exactly $n-(r-1) = n-r+1$ ways to choose the r^{th} element. Consequently, by the product rule, there are

$$n(n-1)(n-2)\cdots(n-r+1)$$

r-permutations of the set. □

Corollary 119. *If n and r are integers with $0 \leq r \leq n$, then*

$$P(n,r) = \frac{n!}{(n-r)!}.$$

Proof. When n and r are integers with $1 \leq r \leq n$, by Theorem 118 we have

$$P(n,r) = n(n-1)(n-2)\cdots(n-r+1) = \frac{n!}{(n-r)!}.$$

Because $\frac{n!}{(n-0)!} = \frac{n!}{n!} = 1$ whenever n is a nonnegative integer, we see that the formula $P(n,r) = \frac{n!}{(n-r)!}$ also holds when $r = 0$. $\square$

Example 120. *How many ways are there to select a first-prize winner, a second-prize winner, and a third-prize winner from 100 different people who have entered a contest?*

Solution: Because it matters which person wins which prize, the number of ways to pick the three prize winners is the number of ordered selections of three elements from a set of 100 elements, that is, the number of 3-permutations of a set of 100 elements. Consequently, the answer is

$$P(100,3) = 100 \cdot 99 \cdot 98 = 970,200.$$

Example 121. *Suppose that there are eight runners in a race. The winner receives a gold medal, the second-place finisher receives a silver medal, and the third-place finisher receives a bronze medal. How many different ways are there to award these medals, if all possible outcomes of the race can occur and there are no ties?*

Solution: The number of different ways to award the medals is the number of 3-permutations of a set with eight elements. Hence, there are $P(8,3) = 8 \cdot 7 \cdot 6 = 336$ possible ways to award the medals.

Example 122. *Suppose that a saleswoman has to visit eight different cities. She must begin her trip in a specified city, but she can visit the* *other seven cities in any order she wishes. How many possible orders can the saleswoman use when visiting these cities?*

Solution: The number of possible paths between the cities is the number of permutations of seven elements, because the first city is determined, but the remaining seven can be ordered arbitrarily. Consequently, there are $7! = 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 5040$ ways for the saleswoman to choose her tour. If, for instance, the saleswoman wishes to find the path between the cities with minimum distance, and she computes the total distance for each possible path, she must consider a total of 5040 paths.

Example 123. *How many permutations of the letters ABCDEFGH contain the string ABC ?*

Solution: Because the letters ABC must occur as a block, we can find the answer by finding the number of permutations of six objects, namely, the block ABC and the individual letters D, E, F, G, and H. Because these six objects can occur in any order, there are $6! = 720$ permutations of the letters ABCDEFGH in which ABC occurs as a block.

2.3.2 Combinations

Example 124. *How many different committees of three students can be formed from a group of four students?*

Solution: To answer this question, we need only find the number of subsets with three elements from the set containing the four students. We see that there are four such subsets, one for each of the four students, because choosing three students is the same as choosing one of the four students to leave out of the group. This means that there are four ways to choose the three students for the committee, where the order in which these students are chosen does not matter. An r**-combination** of elements of a set is an unordered selection of r elements from the set. Thus, an r-combination is simply a subset of the set with r elements.

The number of r-combinations of a set with n distinct elements is denoted by $C(n, r)$. Note that $C(n, r)$ is also denoted by $\binom{n}{k}$ and is called a **binomial coefficient**.

Example 125. *We see that $C(4, 2) = 6$, because the 2-combinations of $\{a, b, c, d\}$ are the six subsets $\{a, b\}$, $\{a, c\}$, $\{a, d\}$, $\{b, c\}$, $\{b, d\}$, and $\{c, d\}$.*

Theorem 126. *The number of r-combinations of a set with n elements, where n is a nonnegative integer and r is an integer with $0 \leq r \leq n$, equals*

$$C(n, r) = \frac{n!}{r!(n-r)!}.$$

Proof. The $P(n, r)$ r-permutations of the set can be obtained by forming the $C(n, r)$ r-combinations of the set, and then ordering the elements in each r-combination, which can be done in $P(r, r)$ ways. Consequently, by the product rule,

$$P(n, r) = C(n, r) \cdot P(r, r).$$

This implies that

$$C(n, r) = \frac{P(n, r)}{P(r, r)} = \frac{n!}{/(n-r)!} r!/(r-r)! = \frac{n!}{r!(n-r)!}.$$

We can also use the division rule for counting to construct a proof of this theorem. Because the order of elements in a combination does not matter and there are $P(r, r)$ ways to order r elements in an r-combination of n elements, each of the $C(n, r)$ r-combinations of a set with n elements corresponds to exactly $P(r,r)$ r-permutations. Hence, by the division rule, $C(n,r) = \frac{P(n,r)}{P(r,r)}$, which implies as before that $C(n,r) = \frac{n!}{r!(n-r)!}$.

□

Example 127. *How many poker hands of five cards can be dealt from a standard deck of 52 cards? Also, how many ways are there to select 47 cards from a standard deck of 52 cards?*

Solution: : Because the order in which the five cards are dealt from a deck of 52 cards does not matter, there are

$$C(52,5) = \frac{52!}{5!47!}$$

different hands of five cards that can be dealt. To compute the value of $C(52,5)$, first divide the numerator and denominator by $47!$ to obtain

$$C(52,5) = \frac{52 \cdot 51 \cdot 50 \cdot 49 \cdot 48}{5 \cdot 4 \cdot 3 \cdot 2 \cdot 1}.$$

This expression can be simplified by first dividing the factor 5 in the denominator into the factor 50 in the numerator to obtain a factor 10 in the numerator, then dividing the factor 4 in the denominator into the factor 48 in the numerator to obtain a factor of 12 in the numerator, then dividing the factor 3 in the denominator into the factor 51 in the numerator to obtain a factor of 17 in the numerator, and finally, dividing the factor 2 in the denominator into the factor 52 in the numerator to obtain a factor of 26 in the numerator. We find that

$$C(52,5) = 26 \cdot 17 \cdot 10 \cdot 49 \cdot 12 = 2,598,960.$$

Consequently, there are 2,598,960 different poker hands of five cards that can be dealt from a standard deck of 52 cards. Note that there are

$$C(52, 47) = \frac{52!}{47!5!}$$

different ways to select 47 cards from a standard deck of 52 cards. We do not need to compute this value because $C(52, 47) = C(52, 5)$. (Only the order of the factors $5!$ and $47!$ is different in the denominators in the formulae for these quantities.) It follows that there are also 2,598,960 different ways to select 47 cards from a standard deck of 52 cards.

Corollary 128. *Let n and r be nonnegative integers with $r \leq n$. Then $C(n, r) = C(n, n-r)$.*

Proof. From Theorem 126 it follows that

$$C(n, r) = \frac{n!}{r!(n-r)!}$$

and

$$C(n, n-r) = \frac{n!}{(n-r)![n-(n-r)]!} = \frac{n!}{(n-r)!r!}.$$

Hence, $C(n, r) = C(n, n-r)$. □

Definition 129. *A combinatorial proof of an identity is a proof that uses counting arguments to prove that both sides of the identity count the same objects but in different ways or a proof that is based on showing that there is a bijection between the sets of objects counted by the two sides of the identity. These two types of proofs are called double counting proofs and bijective proofs, respectively.*

Example 130. *How many ways are there to select five players from a 10-member tennis team to make a trip to a match at another school?*

Solution: The answer is given by the number of 5-combinations of a set with 10 elements. By Theorem 126, the number of such combinations is

$$C(10,5) = \frac{10!}{5!5!} = 252.$$

Example 131. *A group of 30 people have been trained as astronauts to go on the first mission to Mars. How many ways are there to select a crew of six people to go on this mission (assuming that all crew members have the same job)?*

Solution: The number of ways to select a crew of six from the pool of 30 people is the number of 6-combinations of a set with 30 elements, because the order in which these people are chosen does not matter. By Theorem 2, the number of such combinations is

$$C(30,6) = \frac{30!}{6!24!} = \frac{30 \cdot 29 \cdot 28 \cdot 27 \cdot 26 \cdot 25}{6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1} = 593,775.$$

Example 132. *How many bit strings of length n contain exactly r 1s?*

Solution: The positions of r 1s in a bit string of length n form an r-combination of the set $\{1, 2, 3, \cdots, n\}$. Hence, there are $C(n,r)$ bit strings of length n that contain exactly r 1s.

Example 133. *Suppose that there are 9 faculty members in the mathematics department and 11 in the computer science department. How many ways are there to select a committee to develop a discrete mathematics course at a school if the committee is to consist of three faculty members from the mathematics department and four from the computer science department?*

Solution: By the product rule, the answer is the product of the number of 3-combinations of a set with nine elements and the number of

4-combinations of a set with 11 elements. By Theorem 126, the number of ways to select the committee is

$$C(9,3) \cdot C(11,4) = \frac{9!}{3!6!} \cdot \frac{11!}{4!7!} = 84 \cdot 330 = 27,720.$$

Let Us Sum Up

In this section, we discussed about

$*$ Permutations

$*$ Combinations

$*$ Solving counting problems using permutations and combinations.

Check your Progress

1. $P(n,0) =$

(a) n

(b) 1

(c) $n-1$

(d) 0

2. $C(4,2) =$

(a) 4

(b) 2

(c) 8

(d) 6

3. How many bit strings of length n contain exactly r is?

(a) $C(n, r)$

(b) $P(n, r)$

(c) $r!$

(d) $(n - r)!$

2.4 Generalized Permutations and Combinations

2.4.1 Permutations with Repetition

Counting permutations when repetition of elements is allowed can easily be done using the product rule, as Example 134 shows.

Example 134. *How many strings of length r can be formed from the uppercase letters of the English alphabet?*

Solution: By the product rule, because there are 26 uppercase English letters, and because each letter can be used repeatedly, we see that there are 26^r strings of uppercase English letters of length r.

Theorem 135. *The number of r-permutations of a set of n objects with repetition allowed is n^r.*

Proof. There are n ways to select an element of the set for each of the r positions in the r-permutation when repetition is allowed, because for each choice all n objects are available. Hence, by the product rule there are n^r r-permutations when repetition is allowed. □

2.4.2 Combinations with Repetition

Example 136. *How many ways are there to select four pieces of fruit from a bowl containing apples, oranges, and pears if the order in which the pieces are selected does not matter, only the type of fruit and not the individual piece matters, and there are at least four pieces of each type of fruit in the bowl?*

Solution: To solve this problem we list all the ways possible to select the fruit. There are 15 ways:

4 apples	4 oranges	4 pears
3 apples, 1 orange	3 apples, 1 pear	3 oranges, 1 apple
3 oranges, 1 pear	3 pears, 1 apple	3 pears, 1 orange
2 apples, 2 oranges	2 apples, 2 pears	2 oranges, 2 pears
2 apples, 1 orange, 1 pear	2 oranges, 1 apple, 1 pear	2 pears, 1 apple, 1 orange

The solution is the number of 4-combinations with repetition allowed from a three-element set, $\{apple, orange, pear\}$.

Theorem 137. *There are $C(n+r-1, r) = C(n+r-1, n-1)$ r-combinations from a set with n elements when repetition of elements is allowed.*

Proof. Each r-combination of a set with n elements when repetition is allowed can be represented by a list of $n - 1$ bars and r stars. The $n - 1$ bars are used to mark off n different cells, with the i^{th} cell containing a star for each time the i^{th} element of the set occurs in the combination. For instance, a 6-combination of a set with four elements is represented with three bars and six stars. Here

$$**|*||***$$

represents the combination containing exactly two of the first element, one of the second element, none of the third element, and three of the fourth element of the set.

As we have seen, each different list containing $n-1$ bars and r stars corresponds to an r-combination of the set with n elements, when repetition is allowed. The number of such lists is $C(n-1+r, r)$, because each list corresponds to a choice of the r positions to place the r stars from the $n-1+r$ positions that contain r stars and $n-1$ bars. The number of such lists is also equal to $C(n-1+r, n-1)$, because each list corresponds to a choice of the $n-1$ positions to place the $n-1$ bars. $\square$

Example 138. *Suppose that a cookie shop has four different kinds of cookies. How many different ways can six cookies be chosen? Assume that only the type of cookie, and not the individual cookies or the order in which they are chosen, matters.*

Solution: The number of ways to choose six cookies is the number of 6-combinations of a set with four elements. From Theorem 137 this equals $C(4+6-1, 6) = C(9, 6)$. Because

$$C(9,6) = C(9,3) = \frac{9 \cdot 8 \cdot 7}{1 \cdot 2 \cdot 3} = 84,$$

there are 84 different ways to choose the six cookies.

Example 139. *How many solutions does the equation*

$$x_1 + x_2 + x_3 = 11$$

have, where x_1, x_2, and x_3 are nonnegative integers?

Solution: To count the number of solutions, we note that a solution corresponds to a way of selecting 11 items from a set with three elements so that x_1 items of type one, x_2 items of type two, and x_3 items of type three are chosen. Hence, the number of solutions is equal to the number of 11-combinations with repetition allowed from a set with three elements. From Theorem 137 it follows that there are

$$C(3+11-1,11) = C(13,11) = C(13,2) = \frac{13\cdot 12}{1\cdot 2} = 78$$

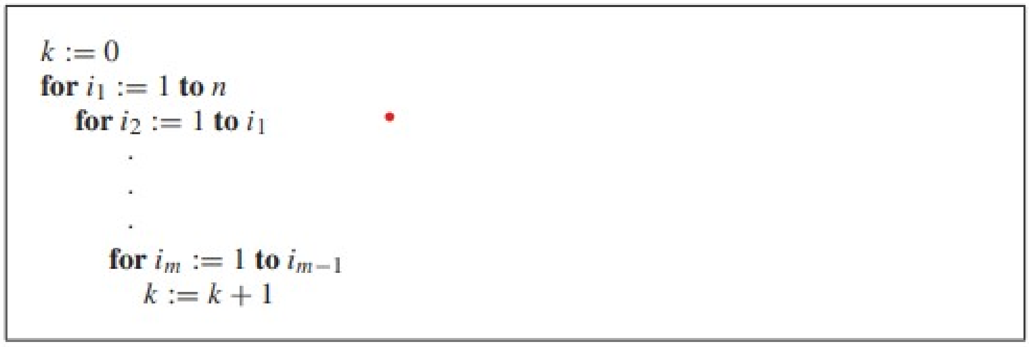
solutions.

The number of solutions of this equation can also be found when the variables are subject to constraints. For instance, we can find the number of solutions where the variables are integers with $x_1 \geq 1$, $x_2 \geq 2$, and $x_3 \geq 3$. A solution to the equation subject to these constraints corresponds to a selection of 11 items with x_1 items of type one, x_2 items of type two, and x_3 items of type three, where, in addition, there is at least one item of type one, two items of type two, and three items of type three. So, a solution corresponds to a choice of one item of type one, two of type two, and three of type three, together with a choice of five additional items of any type. By Theorem 137 this can be done in

$$C(3+5-1,5) = C(7,5) = C(7,2) = \frac{7\cdot 6}{1\cdot 2} = 21$$

ways. Thus, there are 21 solutions of the equation subject to the given constraints.

Example 140. *What is the value of k after the following pseudocode has been executed?*



```
$k := 0$
for $i_1 := 1$ to $n$
    for $i_2 := 1$ to $i_1$
        .
        .
        .
        for $i_m := 1$ to $i_{m-1}$
            $k := k + 1$
```

Solution: Note that the initial value of k is 0 and that 1 is added to k each time the nested loop is traversed with a sequence of integers $i_1, i_2, \cdots, i_m$ such that

$$1 \leq i_m \leq i_{m-1} \leq \cdots \leq i_1 \leq n.$$

The number of such sequences of integers is the number of ways to choose m integers from $\{1, 2, \cdots, n\}$, with repetition allowed. (To see this, note that once such a sequence has been selected, if we order the integers in the sequence in nondecreasing order, this uniquely defines an assignment of $i_m, i_{m-1}, \cdots, i_1$. Conversely, every such assignment corresponds to a unique unordered set.) Hence, from Theorem 126, it follows that $k = C(n + m - 1, m)$ after this code has been executed.

The formulae for the numbers of ordered and unordered selections of r elements, chosen with and without repetition allowed from a set with n elements, are shown in Table 1.1.

Table 2.1: Combinations and Permutations with and without Repetition.

Type	**Repetition Allowed?**	**Formula**
r-permutations	No	$\frac{n!}{(n-r)!}$
r-combinations	No	$\frac{n!}{r!(n-r)!}$
r-permutations	Yes	nr
r-combinations	Yes	$\frac{(n+r-1)!}{r!(n-1)!}$

2.4.3 Permutations with Indistinguishable Objects

Example 141. *How many different strings can be made by reordering the letters of the word SUCCESS?*

Solution: Because some of the letters of SUCCESS are the same, the answer is not given by the number of permutations of seven letters. This word contains three Ss, two Cs, one U, and one E. To determine the number of different strings that can be made by reordering the letters, first note that the three Ss can be placed among the seven positions in $C(7,3)$ different ways, leaving four positions free. Then the two Cs can be placed in $C(4,2)$ ways, leaving two free positions. The U can be placed in $C(2,1)$ ways, leaving just one position free. Hence E can be placed in $C(1,1)$ way. Consequently, from the product rule, the number of different strings that can be made is

$$\begin{aligned} C(7,3)C(4,2)C(2,1)C(1,1) &= \frac{7!}{3!4!}\cdot\frac{4!}{2!2!}\cdot\frac{2!}{1!1!}\cdot\frac{1!}{1!0!} \\ &= \frac{7!}{3!2!1!1!} \\ &= 420. \end{aligned}$$

Theorem 142. *The number of different permutations of n objects, where there are n_1 indistinguishable objects of type 1, n_2 indistinguishable objects of type $2, \cdots$, and n_k indistinguishable objects of type k, is*

$$\frac{n!}{n_1!n_2!\cdots n_k!}.$$

Proof. To determine the number of permutations, first note that the n_1 objects of type one can be placed among the n positions in $C(n, n_1)$ ways, leaving $n - n_1$ positions free. Then the objects of type two can be placed in $C(n - n_1, n_2)$ ways, leaving $n - n_1 - n_2$ positions free. Continue placing the objects of type three,$\cdots$, type $k - 1$, until at the last stage, n_k objects of type k can be placed in $C(n - n_1 - n_2 - \cdots - n_{k-1}, n_k)$ ways. Hence, by the product rule, the total number of different permutations is

$$\begin{aligned} & C(n, n_1)C(n - n_1, n_2)\cdots C(n - n_1 - \cdots - n_{k-1}, n_k) \\ = & \frac{n!}{n_1!(n - n_1)!(n - n_1)!n_2!(n - n_1 - n_2)!\cdots(n - n_1 - \cdots - n_{k-1})!} n_k!0! \\ = & \frac{n!}{n_1!n_2!\cdots n_k!}. \end{aligned}$$

□

2.4.4 Distributing Objects into Boxes

Many counting problems can be solved by enumerating the ways objects can be placed into boxes (where the order these objects are placed into the boxes does not matter). The objects can be either *distinguishable,* that is, different from each other, or *indistinguishable, that is, considered identical. Distinguishable objects are sometimes said to be labeled, whereas indistinguishable objects are said to be unlabeled. Similarly, boxes can be distinguishable, that is, different, or indinguishable, that is, identical. Distinguishable boxes are often said to be labeled, while indistinguish-*

able boxes are said to be unlabeled.

Remark 143. *A* **closed formula** *is an expression that can be evaluated using a finite number of operations and that includes numbers, variables, and values of functions, where the operations and functions belong to a generally accepted set that can depend on the context.*

2.4.5 Distinguishable Objects and Distinguishable Boxes

Example 144. *How many ways are there to distribute hands of 5 cards to each of four players from the standard deck of 52 cards?*

Solution: *We will use the product rule to solve this problem. To begin, note that the first player can be dealt 5 cards in $C(52,5)$ ways. The second player can be dealt 5 cards in $C(47,5)$ ways, because only 47 cards are left. The third player can be dealt 5 cards in $C(42,5)$ ways. Finally, the fourth player can be dealt 5 cards in $C(37,5)$ ways. Hence, the total number of ways to deal four players 5 cards each is*

$$\begin{aligned} C(52,5)C(47,5)C(42,5)C(37,5) &= \frac{52!}{47!5!}\cdot\frac{47!}{42!5!}\cdot\frac{42!}{37!5!}\cdot 37!32!5! \\ &= \frac{52!}{5!5!5!5!32!}. \end{aligned}$$

Theorem 145. *The number of ways to distribute n distinguishable objects into k distinguishable boxes so that n_i objects are placed into box i, $i = 1,2,\cdots,k$, equals*

$$\frac{n!}{n_1!n_2!\cdots n_k!}.$$

2.4.6 Indistinguishable Objects Distinguishable Boxes

Example 146. *How many ways are there to place 10 indistinguishable balls into eight distinguishable bins?*

Solution: *The number of ways to place 10 indistinguishable balls into eight bins equals the number of 10-combinations from a set with eight elements when repetition is allowed. Consequently, there are*

$$C(8+10-1,10) = C(17,10) = \frac{17!}{10!7!} = 19,448.$$

2.4.7 Distinguishable Objects and Indistinguishable Boxes

Example 147. *How many ways are there to put four different employees into three indistinguishable offices, when each office can contain any number of employees?*

Solution: *We will solve this problem by enumerating all the ways these employees can be placed into the offices. We represent the four employees by A, B, C, and D. First, we note that we can distribute employees so that all four are put into one office, three are put into one office and a fourth is put into a second office, two employees are put into one office and two put into a second office, and finally, two are put into one office, and one each put into the other two offices. Each way to distribute these employees to these offices can be represented by a way to partition the elements A, B, C, and D into disjoint subsets.*

We can put all four employees into one office in exactly one way, represented by $\{\{A,B,C,D\}\}$. *We can put three employees into one office and the fourth employee into a different office in exactly four ways, represented by* $\{\{A,B,C\},\{D\}\}$, $\{\{A,B,D\},\{C\}\}$, $\{\{A,C,D\},\{B\}\}$, *and* $\{\{B,C,D\},\{A\}\}$. *We can put two employees into one office and two into a second office in exactly three ways, represented by* $\{\{A,B\},\{C,D\}\}$, $\{\{A,C\},\{B,D\}\}$, *and* $\{\{A,D\},\{B,C\}\}$. *Finally, we can put two employees into one office, and one each into each of the remaining two offices* *in six ways, represented by* $\{\{A, B\}, \{C\}, \{D\}\}$*,*$\{\{A, C\}, \{B\}, \{D\}\}$*,* $\{\{A, D\}, \{B\}, \{C\}\}$*,* $\{\{B, C\}, \{A\}, \{D\}\}$*,* $\{\{B, D\}\}$*,* $\{A\}, \{C\}\}$*, and* $\{\{C, D\}, \{A\}, \{B\}\}$*.*

Counting all the possibilities, we find that there are 14 ways to put four different employees into three indistinguishable offices. Another way to look at this problem is to look at the number of offices into which we put employees. Note that there are six ways to put four different employees into three indistinguishable offices so that no office is empty, seven ways to put four different employees into two indistinguishable offices so that no office is empty, and one way to put four employees into one office so that it is not empty. Let $S(n, j)$ *denote the number of ways to distribute n distinguishable objects into j indistinguishable boxes so that no box is empty. The numbers* $S(n, j)$ *are called* **Stirling numbers of the second kind**.

$$S(n, j) = \frac{1}{j!} \sum_{i=0}^{j-1} (-1)^i \binom{j}{i} (J - i)^n.$$

Consequently, the number of ways to distribute n distinguishable objects into k indistinguishable boxes equals

$$\sum_{j=1}^{k} S(n, j) = \sum_{j=1}^{k} \frac{1}{j!} \sum_{i=0}^{j-1} (-1)^i \binom{j}{i} (j - i)^n.$$

2.4.8 Indistinguishable Objects and Indistinguishable Boxes

Example 148. *How many ways are there to pack six copies of the same book into four identical boxes, where a box can contain as many as six books?*

Solution: *We will enumerate all ways to pack the books. For each way to pack the books, we will list the number of books in the box with the* *largest number of books, followed by the numbers of books in each box containing at least one book, in order of decreasing number of books in a box. The ways we can pack the books are*

6	3, 2, 1
5, 1	3, 1, 1, 1
4, 2	2, 2, 2
4, 1, 1	2, 2, 1, 1.
3, 3	

For example, 4, 1, 1 indicates that one box contains four books, a second box contains a single book, and a third box contains a single book (and the fourth box is empty). We conclude that there are nine allowable ways to pack the books, because we have listed them all.

Let Us Sum Up

In this section, we discussed about

- $*$ *Permutations with repetition.*
- $*$ *Combinations with repetition.*
- $*$ *Permutations with indistinguishable objects.*

Check your Progress

1. How many strings of length r can be formed from uppercase letters of the English alphabet?

(a) $2b!$

(b) $r(2b)$

(c) $2b$

(d) $(2b - r)!$

2. How many different strings can be made by reordering the letters of the word success?

(a) 720

(b) 400

(c) 120

(d) 420

2.5 Generating Permutations and Combinations

2.5.1 Generating Permutations

In **lexiocographic** *(or* **dictionary***)* **ordering** *this ordering, the permutation $a_1a_2\cdots a_n$ precedes the permutation of $b_1b_2\cdots b_n$, if for some k, with $1 \le k \le n$, $a_1 = b_1$, $a_2 = b_2, \cdots, a_{k-1} = b_{k-1}$, and $a_k < b_k$. In other words, a permutation of the set of the n smallest positive integers precedes (in lexicographic order) a second permutation if the number in this permutation in the first position where the two permutations disagree is smaller than the number in that position in the second permutation.*

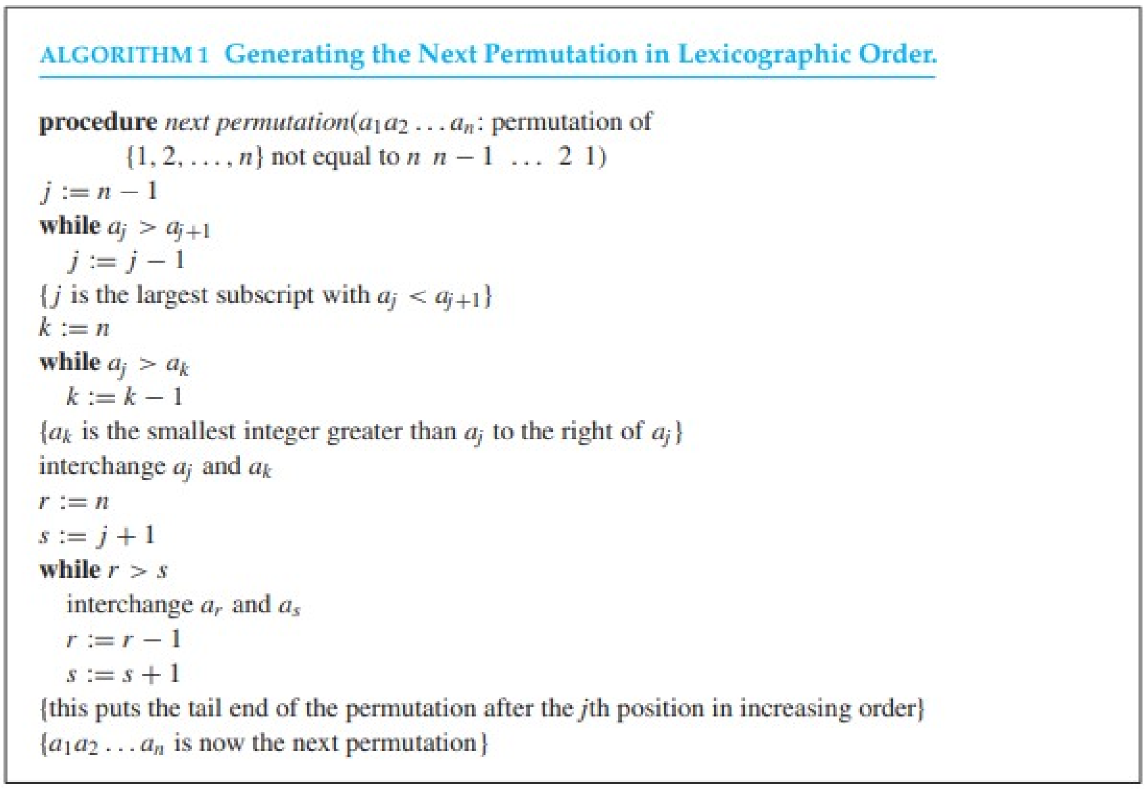
Example 149. *The permutation 23415 of the set $\{1, 2, 3, 4, 5\}$ precedes the permutation 23514, because these permutations agree in the first two positions, but the number in the third position in the first permutation, 4,* *is smaller than the number in the third position in the second permutation, 5. Similarly, the permutation 41532 precedes 52143.*

Example 150. *What is the next permutation in lexicographic order after 362541?*

Solution: *The last pair of integers a_j and a_{j+1} where $a_j < a_{j+1}$ is $a_3 = 2$ and $a_4 = 5$. The least integer to the right of 2 that is greater than 2 in the permutation is $a_5 = 4$. Hence, 4 is placed in the third position. Then the integers 2, 5, and 1 are placed in order in the last three positions, giving 125 as the last three positions of the permutation. Hence, the next permutation is 364125.*

Example 151. *Generate the permutations of the integers 1, 2, 3 in lexicographic order.*

Solution: *Begin with 123. The next permutation is obtained by interchanging 3 and 2 to obtain 132. Next, because $3 > 2$ and $1 < 3$, permute the three integers in 132. Put the smaller of 3 and 2 in the first position, and then put 1 and 3 in increasing order in positions 2 and 3 to obtain 213. This is followed by 231, obtained by interchanging 1 and 3, because $1 < 3$. The next larger permutation has 3 in the first position, followed by 1 and 2 in increasing order, namely, 312. Finally, interchange 1 and 2 to obtain the last permutation, 321. We have generated the permutations of 1, 2, 3 in lexicographic order. They are 123, 132, 213, 231, 312, and 321. Algorithm 1 displays the procedure for finding the next permutation in lexicographic order after a permutation that is not $n\ n-1\ n-2\ \cdots\ 2\ 1$, which is the largest permutation.*



ALGORITHM 1 Generating the Next Permutation in Lexicographic Order.

```
procedure next permutation($a_1 a_2 \ldots a_n$: permutation of
        $\{1, 2, \ldots, n\}$ not equal to $n\ n-1\ \ldots\ 2\ 1$)
$j := n - 1$
while $a_j > a_{j+1}$
    $j := j - 1$
{$j$ is the largest subscript with $a_j < a_{j+1}$}
$k := n$
while $a_j > a_k$
    $k := k - 1$
{$a_k$ is the smallest integer greater than $a_j$ to the right of $a_j$}
interchange $a_j$ and $a_k$
$r := n$
$s := j + 1$
while $r > s$
    interchange $a_r$ and $a_s$
    $r := r - 1$
    $s := s + 1$
{this puts the tail end of the permutation after the $j$th position in increasing order}
{$a_1 a_2 \ldots a_n$ is now the next permutation}
```

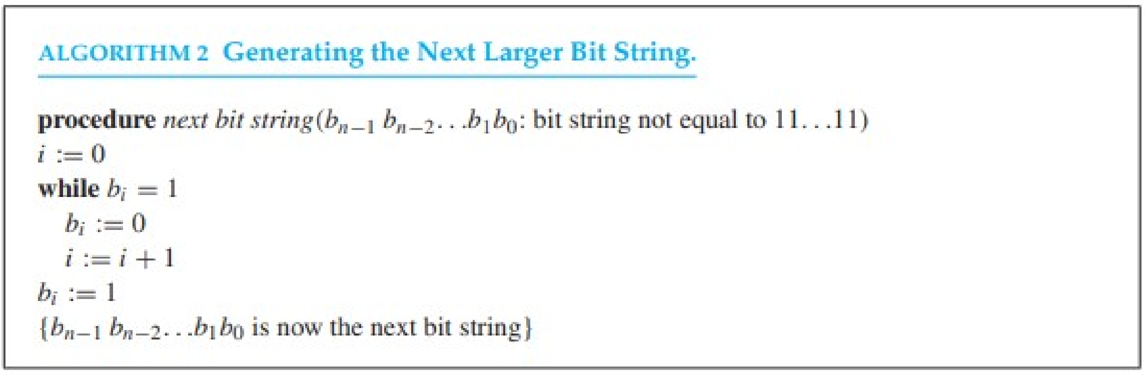
2.5.2 Generating Combinations

Example 152. *Find the next bit string after 10 0010 0111.*

Solution: *The first bit from the right that is not a 1 is the fourth bit from the right. Change this bit to a 1 and change all the following bits to 0s. This produces the next larger bit string, 10 0010 1000. The procedure for producing the next larger bit string after $b_{n-1}\ b_{n-2}\ \cdots\ b_1\ b_0$ is given as Algorithm 2.*

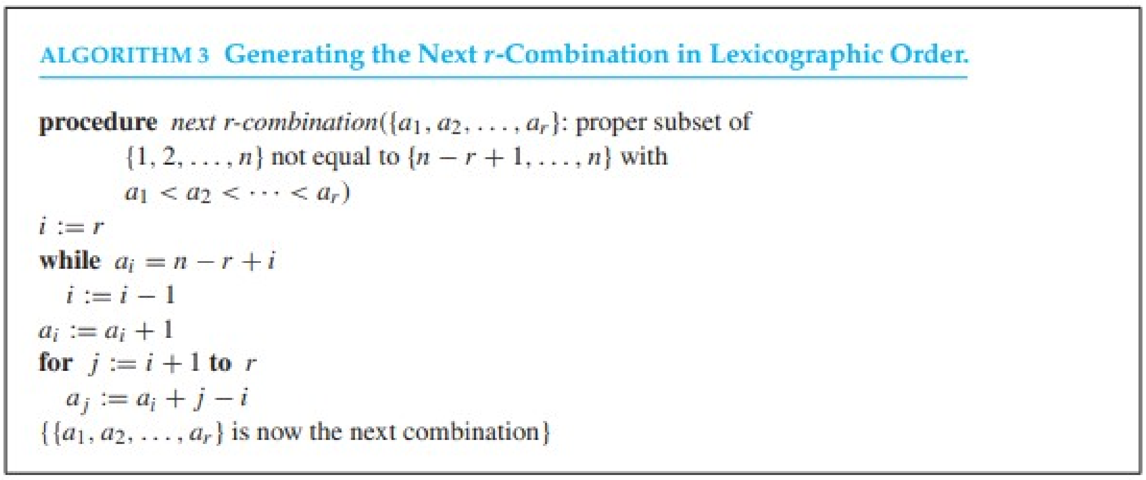
Example 153. *Find the next larger 4-combination of the set $\{1, 2, 3, 4, 5, 6\}$ after $\{1, 2, 5, 6\}$.*

Solution: *The last term among the terms a_i with $a_1 = 1$, $a_2 = 2$, $a_3 = 5$, and $a_4 = 6$ such that $a_i \neq 6 - 4 + i$ is $a_2 = 2$. To obtain the next larger 4-combination, increment a_2 by 1 to obtain $a_2 = 3$. Then set $a_3 = 3+1 =$*


ALGORITHM 2 Generating the Next Larger Bit String.

```
procedure next bit string($b_{n-1}\,b_{n-2}\ldots b_1 b_0$: bit string not equal to 11...11)
$i := 0$
while $b_i = 1$
    $b_i := 0$
    $i := i + 1$
$b_i := 1$
{$b_{n-1}\,b_{n-2}\ldots b_1 b_0$ is now the next bit string}
```

4 and $a_4 = 3+2 = 5$. Hence the next larger 4-combination is $\{1,3,4,5\}$. Algorithm 3 displays pseudocode for this procedure.


ALGORITHM 3 Generating the Next r-Combination in Lexicographic Order.

```
procedure next r-combination({$a_1, a_2, \ldots, a_r$}: proper subset of
        {$1, 2, \ldots, n$} not equal to {$n - r + 1, \ldots, n$} with
        $a_1 < a_2 < \cdots < a_r$)
$i := r$
while $a_i = n - r + i$
    $i := i - 1$
$a_i := a_i + 1$
for $j := i + 1$ to $r$
    $a_j := a_i + j - i$
{{$a_1, a_2, \ldots, a_r$} is now the next combination}
```

Let Us Sum Up

In this section, we discussed about

- *Generating permutations and combinations.*
- *Algorithm for generating permutations and combinations.*

Check your Progress

1. What is the next permutation in lexicographic order after 362541?

(a) 362542

(b) 462541

(c) 362514

(d) 364125

2. Find the next bit string after 10 0010 0111?

(a) 10 0010 1111

(b) 11 0010 0111

(c) 10 0010 1000

(d) 10 0011 1111

Summary

In this unit, we discussed about

∗ *The basics of counting.*

∗ *The subtraction rule.*

∗ *The division rule.*

∗ *The Pigeonhole principle.*

∗ *Generalized pigeonhole principle.*

∗ *Applications of pigeonhole principle.*

∗ *Generalized permutations and combinations.*

∗ *Generating permutations and combinations.*

Glossary

combinatorics: *the study of arrangements of objects.*

enumeration: *the counting of arrangements of objects.*

tree diagram: *a diagram made up of a root, branches leaving the root, and other branches leaving some of the endpoints of branches.*

permutation: *an ordered arrangement of the elements of a set.*

r-permutation: *an ordered arrangement of r elements of a set.*

$P(n, r)$***:*** *the number of r-permutations of a set with n elements.*

r-combination: *an unordered selection of r elements of a set.*

$C(n, r)$***:*** *the number of r-combinations of a set with n elements.*

Self Assessment Questions

1. *Explain how to find the number of bit strings of length not exceeding 10 that have at least one 0 bit.*

2. *Hoe can you find the number of bit strings of length ten that either begin with 101 or end with 010?*

3. *What is meant by a combinatorial proof of an identity? How is such a proof different from an algebraic one?*

4. *Explain how to prove Pascal's identity using a combinatorial argument.*

Exercises

1. *How many strings of length 10 either start with 000 or end with 1111?*

2. *Show that the decimal expansion of a rational number must repeat itself from some point onward.*

3. *Find* n *if*
   a) $P(n,2) = 110$ *b)* $P(n,n) = 5040$ *c)* $P(n,4) = 12P(n,2)$.

4. *Find* n *if*
   a) $C(n,2) = 45$ *b)* $C(n,3) = P(n,2)$ *c)* $C(n,5) = C(n,2)$.

5. *How many ways are there to assign 24 students to five faculty advisors?*

Answers to Check your Progress

Section 2.1: 1)c 2)a

Section 2.2: 1)b 2)c

Section 2.3: 1)b 2)d 3)a

Section 2.4: 1)c 2)d

Section 2.5: 1)d 2)c

Reference:

1. *Kenneth H. Rosen, "Discrete Mathematics and its Applications", 7th Edition, WCB/ McGraw Hill Publications, New Delhi, 2011.*

Suggested Readings:

1. *Edward A. Bender and S. Gill Williamson, "A Short Course in Discrete Mathematics", Dover Publications, 2006.*

2. *M.O. Albertson and J.P. Hutchinson, "Discrete Mathematics with Algorithms", John Wiley & Sons, 2008.*

3. *Rajendra Akerkar and Rupali Akarkar, "Discrete Mathematics", Pearson Education Pvt. Ltd, Singapore, 2004.*

4. *J. P. Trembley and R. Manohar, "Discrete Mathematical Structures", Tata McGraw Hill, New Delhi,1997.*

5. *Martin Aigner, "A Course in Enumeration", Springer-Verlag, Heidelberg, 2007.*

6. *J.H. Van Lint and R.M. Wilson, "A Course in Combinatorics", 2nd Edition, Cambridge University Press, Cambridge, 2001.*

Unit 3

Advanced Counting Techniques

Objectives

1. *To learn techniques which can be used to solve different types of counting problems.*

2. *To discuss a variety of divide-and-conquer algorithms and analyze then using recurrence relations.*

3. *To solve counting problems using formal power series, called generating functions.*

4. *To model counting problems using recurrence relations.*

5. *To study the complexity of algorithms using recurrence relations.*

3.1 Applications of Recurrence Relations

3.1.1 Modeling With Recurrence Relations

We can use recurrence relations to model a wide variety of problems, such as finding compound interest, counting rabbits on an island, deter-

mining the number of moves in the Tower of Hanoi puzzle, and counting bit strings with certain properties.

Example 154 shows how the population of rabbits on an island can be modeled using a recurrence relation.

Example 154. Rabbits and the Fibonacci Numbers *Consider this problem, which was originally posed by Leonardo Pisano, also known as Fibonacci, in the thirteenth century in his book Liber abaci. A young pair of rabbits (one of each sex) is placed on an island. A pair of rabbits does not breed until they are 2 months old. After they are 2 months old, each pair of rabbits produces another pair each month. Find a recurrence relation for the number of pairs of rabbits on the island after n months, assuming that no rabbits ever die.*

Solution: *Denote by f_n the number of pairs of rabbits after n months. We will show that f_n, $n = 1, 2, 3, \cdots$, are the terms of the Fibonacci sequence.*

The rabbit population can be modeled using a recurrence relation. At the end of the first month, the number of pairs of rabbits on the island is $f_1 = 1$. Because this pair does not breed during the second month, $f_2 = 1$ also. To find the number of pairs after n months, add the number on the island the previous month, f_{n-1}, and the number of newborn pairs, which equals f_{n-2}, because each newborn pair comes from a pair at least 2 months old. Consequently, the sequence $\{f_n\}$ satisfies the recurrence relation $f_n = f_{n-1} + f_{n-2}$ for $n \geq 3$ together with the initial conditions $f_1 = 1$ and $f_2 = 1$. Because this recurrence relation and the initial conditions uniquely determine this sequence, the number of pairs of rabbits on the island after n months is given by the n^{th} Fibonacci number.

Example 155. The Tower of Hanoi Puzzle *A popular puzzle of the late* *nineteenth century invented by the French mathematician douard Lucas, called the Tower of Hanoi, consists of three pegs mounted on a board together with disks of different sizes. Initially these disks are placed on the first peg in order of size, with the largest on the bottom (as shown in Figure 1.1). The rules of the puzzle allow disks to be moved one at a time from one peg to another as long as a disk is never placed on top of a smaller disk. The goal of the puzzle is to have all the disks on the second peg in order of size, with the largest on the bottom. Let H_n denote*

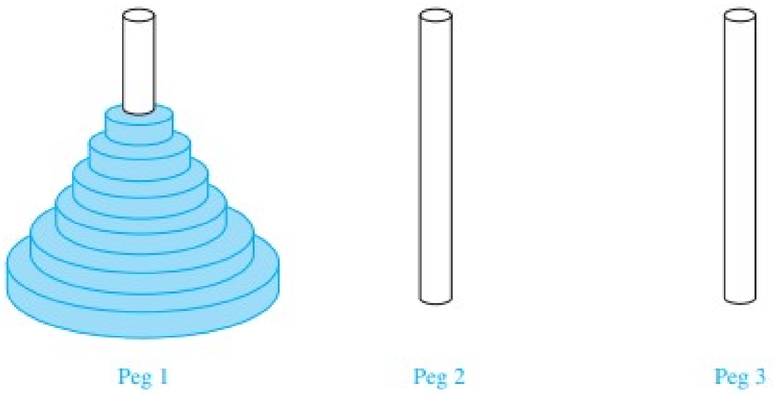


Figure 3.1: The initial position in the Tower of Hanoi.

the number of moves needed to solve the Tower of Hanoi problem with n disks. Set up a recurrence relation for the sequence $\{H_n\}$.

Solution: *Begin with n disks on peg 1. We can transfer the top $n-1$ disks, following the rules of the puzzle, to peg 3 using H_{n-1} moves. We keep the largest disk fixed during these moves. Then, we use one move to transfer the largest disk to the second peg. We can transfer the $n-1$ disks on peg 3 to peg 2 using H_{n-1} additional moves, placing them on top of the largest disk, which always stays fixed on the bottom of peg 2. Moreover, it is easy to see that the puzzle cannot be solved using fewer steps. This* *shows that*

$$H_n = 2H_{n-1} + 1.$$

The initial condition is $H_1 = 1$, because one disk can be transferred from peg 1 to peg 2, according to the rules of the puzzle, in one move.

We can use an iterative approach to solve this recurrence relation. Note that

$$\begin{aligned}
H_n &= 2H_{n-1} + 1 \\
&= 2(2H_{n-2} + 1) + 1 = 22\ H_{n-2} + 2 + 1 \\
&= 2 \cdot 2(2\ H_{n-3} + 1) + 2 + 1 = 23\ H_{n-3} + 22 + 2 + 1 \\
&\ \ \vdots \\
&= 2^{n-1}H_1 + 2^{n-2} + 2^{n-3} + \cdots + 2 + 1 \\
&= 2^{n-1} + 2^{n-2} + \cdots + 2 + 1 \\
&= 2^n - 1.
\end{aligned}$$

We have used the recurrence relation repeatedly to express H_n in terms of previous terms of the sequence. In the next to last equality, the initial condition $H_1 = 1$ has been used. The last equality is based on the formula for the sum of the terms of a geometric series. The iterative approach has produced the solution to the recurrence relation $H_n = 2!H_{n-1}+1$ with the initial condition $H_1 = 1$. This formula can be proved using mathematical induction.

A myth created to accompany the puzzle tells of a tower in Hanoi where monks are transferring 64 gold disks from one peg to another, according to the rules of the puzzle. The myth says that the world will end when they finish the puzzle. How long after the monks started will the world end if the monks take one second to move a disk?

From the explicit formula, the monks require

$$2^{64} - 1 = 18,446,744,073,709,551,615$$

moves to transfer the disks. Making one move per second, it will take them more than 500 billion years to complete the transfer, so the world should survive a while longer than it already has.

Example 156. *Find a recurrence relation and give initial conditions for the number of bit strings of length n that do not have two consecutive 0s. How many such bit strings are there of length five?*

Solution: *Let a_n denote the number of bit strings of length n that do not have two consecutive 0s. We assume that $n \geq 3$, so that the bit string has at least three bits. The bit strings of length n ending with 1 that do not have two consecutive 0s are precisely the bit strings of length $n-1$ with no two consecutive 0s with a 1 added at the end. Consequently, there are a_{n-1} such bit strings.*

Bit strings of length n ending with a 0 that do not have two consecutive 0s must have 1 as their $(n-1)$st bit; otherwise they would end with a pair of 0s. It follows that the bit strings of length n ending with a 0 that have no two consecutive 0s are precisely the bit strings of length $n-2$ with no two consecutive 0s with 10 added at the end. Consequently, there are a_{n-2} such bit strings. We conclude, as illustrated in Figure 1.2, that

$$a_n = a_{n-1} + a_{n-2}$$

for $n \geq 3$.

The initial conditions are $a_1 = 2$, because both bit strings of length one, 0 and 1 do not have consecutive 0s, and $a_2 = 3$, because the valid bit strings of length two are 01, 10, and 11. To obtain a_5, we use the

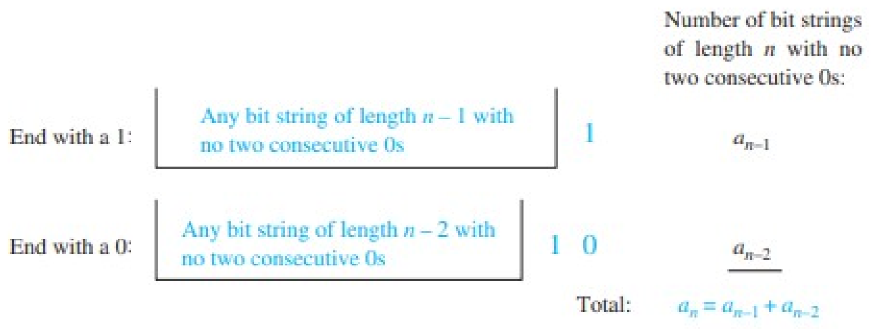

Number of bit strings of length n with no two consecutive 0s:
End with a 1:
Any bit string of length $n-1$ with no two consecutive 0s
1
a_{n-1}
End with a 0:
Any bit string of length $n-2$ with no two consecutive 0s
1 0
a_{n-2}
Total:
$a_n = a_{n-1} + a_{n-2}$

Figure 3.2: Counting bit strings of length n with no two consecutive 0s.

recurrence relation three times to find that

$$\begin{aligned} a_3 &= a_2 + a_1 = 3 + 2 = 5, \\ a_4 &= a_3 + a_2 = 5 + 3 = 8, \\ a_5 &= a_4 + a_3 = 8 + 5 = 13. \end{aligned}$$

Example 157. Codeword Enumeration *A computer system considers a string of decimal digits a valid codeword if it contains an even number of 0 digits. For instance, 1230407869 is valid, whereas 120987045608 is not valid. Let a_n be the number of valid n-digit codewords. Find a recurrence relation for a_n.*

Solution: *Note that $a_1 = 9$ because there are 10 one-digit strings, and only one, namely, the string 0, is not valid. A recurrence relation can be derived for this sequence by considering how a valid n-digit string can be obtained from strings of $n-1$ digits. There are two ways to form a valid string with n digits from a string with one fewer digit.*

First, a valid string of n digits can be obtained by appending a valid string of $n-1$ digits with a digit other than 0. This appending can be done in nine ways. Hence, a valid string with n digits can be formed in this manner in $9\,a_{n-1}$ ways.

Second, a valid string of n digits can be obtained by appending a 0 *to a string of length* $n-1$ *that is not valid. (This produces a string with an even number of 0 digits because the invalid string of length* $n-1$ *has an odd number of 0 digits.) The number of ways that this can be done equals the number of invalid* $(n-1)$*-digit strings. Because there are* 10^{n-1} *strings of length* $n-1$*, and* a_{n-1} *are valid, there are* $10^{n-1} - a_{n-1}$ *valid* n*-digit strings obtained by appending an invalid string of length* $n-1$ *with a* 0*.*

Because all valid strings of length n *are produced in one of these two ways, it follows that there are*

$$\begin{aligned} a_n &= 9\,a_{n-1} + (10^{n-1} - a_{n-1}) \\ &= 8\,a_{n-1} + 10^{n-1} \end{aligned}$$

valid strings of length n*.*

Example 158. *Find a recurrence relation for* C_n*, the number of ways to parenthesize the product of* $n+1$ *numbers,* $x_0 \cdot x_1 \cdot x_2 \cdot \cdots \cdot x_n$*, to specify the order of multiplication. For example,* $C_3 = 5$ *because there are five ways to parenthesize* $x_0 \cdot x_1 \cdot x_2 \cdot x_3$ *to determine the order of multiplication:*

$$((x_0 \cdot x_1) \cdot x_2 \cdot x_3 \quad (x_0 \cdot (x_1 \cdot x_2)) \cdot x_3 \quad (x_0 \cdot x_1) \cdot (x_2 \cdot x_3)$$
$$x_0 \cdot ((x_1 \cdot x_2) \cdot x_3) \quad (x_0 \cdot (x_1 \cdot (x_2 \cdot x_3)).$$

Solution: *To develop a recurrence relation for* C_n*, we note that however we insert parentheses in the product* $x_0 \cdot x_1 \cdot x_2 \cdot \cdots \cdot x_n$*, one "" operator remains outside all parentheses, namely, the operator for the final multiplication to be performed. [For example, in (*$x_0 \cdot (x_1 \cdot x_2)) \cdot x_3$*, it is the final "", while in* $(x_0 \cdot x_1) \cdot (x_2 \cdot x_3)$ *it is the second "".] This final operator appears between two of the* $n+1$ *numbers, say,* x_k *and* x_{k+1}*. There* *are* C_kC_{n-k-1} *ways to insert parentheses to determine the order of the* $n+1$ *numbers to be multiplied when the final operator appears between* x_k *and* x_{k+1}*, because there are* C_k *ways to insert parentheses in the product* $x_0 \cdot x_1 \cdot \cdots \cdot x_k$ *to determine the order in which these* $k+1$ *numbers are to be multiplied and* C_{n-k-1} *ways to insert parentheses in the product* $x_{k+1} \cdot x_{k+2} \cdot \cdots \cdot x_n$ *to determine the order in which these* $n-k$ *numbers are to be multiplied. Because this final operator can appear between any two of the* $n+1$ *numbers, it follows that*

$$\begin{aligned} C_n &= C_0C_{n-1} + C_1C_{n-2} + \cdots + C_{n-2}C_1 + C_{n-1}C_0 \\ &= \sum_{k=0}^{n-1} C_kC_{n-k-1}. \end{aligned}$$

Note that the initial conditions are $C_0 = 1$ *and* $C_1 = 1$*. The recurrence relation in Example 158 can be solved using the method of generating functions. It can be shown that* $C_n = C(2n, n)/(n+1)$ *and that* $C_n \sim \frac{4n}{n^{3/2}\sqrt{\pi}}$*. The sequence* $\{C_n\}$ *is the sequence of* **Catalan numbers***, named after Eugene Charles Catalan.*

3.1.2 Algorithms and Recurrence Relations

Example 159. *Consider seven talks with these start times and end times, as illustrated in Figure 1.3.*

Talk 1: start 8 a.m., end 10 a.m.

Talk 2: start 9 a.m., end 11 a.m.

Talk 3: start 10:30 a.m., end 12 noon

Talk 4: start 9:30 a.m., end 1 p.m.

Talk 5: start 8:30 a.m., end 2 p.m.

Talk 6: start 11 a.m., end 2 p.m.

Talk 7: start 1 p.m., end 2 p.m.

Find $p(j)$ *for* $j = 1, 2, \cdots, 7$.

Solution: *We have* $p(1) = 0$ *and* $p(2) = 0$, *because no talks end before either of the first two talks begin. We have* $p(3) = 1$ *because talk 3 and talk 1 are compatible, but talk 3 and talk 2 are not compatible;* $p(4) = 0$ *because talk 4 is not compatible with any of talks 1, 2, and 3;* $p(5) = 0$ *because talk 5 is not compatible with any of talks 1, 2, 3, and 4; and* $p(6) = 2$ *because talk 6 and talk 2 are compatible, but talk 6 is not compatible with any of talks 3, 4, and 5. Finally,* $p(7) = 4$, *because talk 7 and talk 4 are compatible, but talk 7 is not compatible with either of talks 5 or 6. To develop a dynamic programming algorithm for this problem, we first*

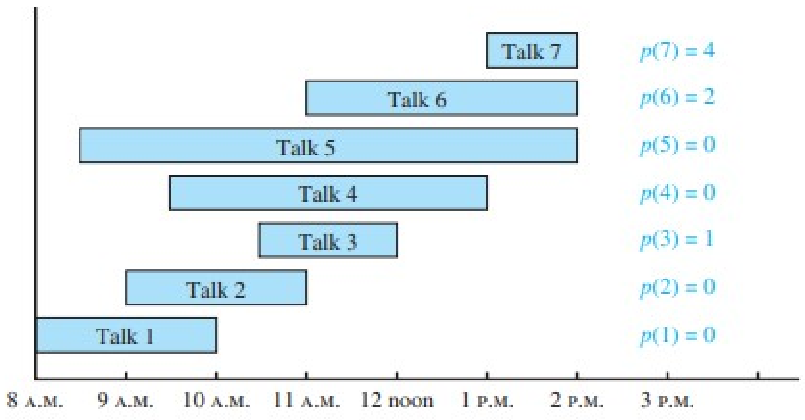


Figure 3.3: A schedule of lectures with the values of $p(n)$ shown.

develop a key recurrence relation. To do this, first note that if $j \leq n$, there are two possibilities for an optimal schedule of the first j talks (recall that we are assuming that the n talks are ordered by increasing end time): (i) talk j belongs to the optimal schedule or (ii) it does not.

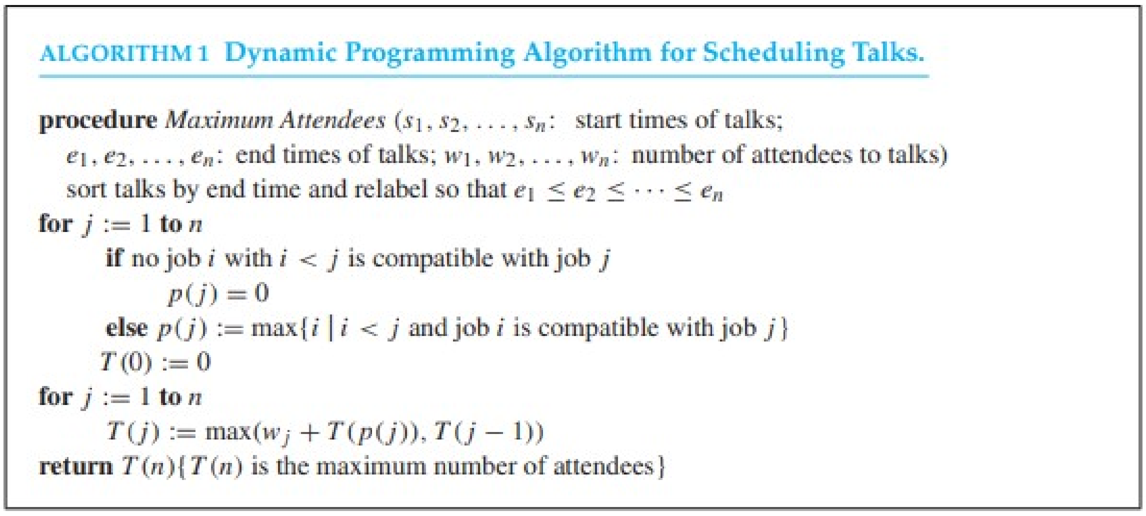
Case (i): We know that talks $p(j)+1, \cdots, j-1$ do not belong to this schedule, for none of these other talks are compatible with talk j . Furthermore, the other talks in this optimal schedule must comprise an optimal schedule for talks $1, 2, \cdots, p(j)$. For if there were a better schedule for talks $1, 2, \cdots, p(j)$, by adding talk j , we will have a schedule better than the overall optimal schedule. Consequently, in case (i), we have

$$T(j) = w_j + T(p(j)).$$

Case (ii): When talk j does not belong to an optimal schedule, it follows that an optimal schedule from talks $1, 2, \cdots, j$ is the same as an optimal schedule from talks $1, 2, \cdots, j-1$. Consequently, in case (ii), we have $T(j) = T(j-1)$. Combining cases (i) and (ii) leads us to the recurrence relation

$$T(j) = max(w_j + T(p(j)),\ T(j-1)).$$

Now that we have developed this recurrence relation, we can construct an efficient algorithm, Algorithm 1, for computing the maximum total number of attendees.We ensure that the algorithm is efficient by storing the value of each $T(j)$ after we compute it. This allows us to compute $T(j)$ only once. If we did not do this, the algorithm would have exponential worst-case complexity. The process of storing the values as each is computed is known as **memoization** *and is an important technique for making recursive algorithms efficient.*



ALGORITHM 1 Dynamic Programming Algorithm for Scheduling Talks.

```
procedure Maximum Attendees ($s_1, s_2, \ldots, s_n$: start times of talks;
  $e_1, e_2, \ldots, e_n$: end times of talks; $w_1, w_2, \ldots, w_n$: number of attendees to talks)
  sort talks by end time and relabel so that $e_1 \leq e_2 \leq \cdots \leq e_n$
for $j := 1$ to $n$
      if no job $i$ with $i < j$ is compatible with job $j$
            $p(j) = 0$
      else $p(j) := \max\{i \mid i < j$ and job $i$ is compatible with job $j\}$
      $T(0) := 0$
for $j := 1$ to $n$
      $T(j) := \max(w_j + T(p(j)), T(j-1))$
return $T(n)\{T(n)$ is the maximum number of attendees$\}$
```

Let Us Sum Up

In this section, we discussed about

∗ *Modeling with recurrence relations.*

∗ *Algorithms and recurrence relations.*

∗ *Dynamic programming.*

Check your Progress

1. the Fibonacci numbers satisfy the recurrence relation

(a) $f_n = f_{n-1} - f_{n-2}$

(b) $f_n = f_{n-1} + f_{n-2}$

(c) $f_n = 2f_{n-1} - f_{n-2}$

(d) $f_n = f_{n-1} + 2f_{n-2}$

2. The initial condition $H_1 =$ ________ *is the Tower of Hanoi puzzle.*

(a) 0

(b) 1

(c) 2

(d) 3

3.2 Solving Linear Recurrence Relations

Definition 160. *A linear homogeneous recurrence relation of degree k with constant coefficients is a recurrence relation of the form*

$$a_n = c_1a_{n-1} + c_2a_{n-2} + \cdots + c_ka_{n-k},$$

where $c_1, c_2, \cdots, c_k$ are real numbers, and $c_k \neq 0$.

The recurrence relation in the definition is **linear** *because the right-hand side is a sum of previous terms of the sequence each multiplied by a function of n. The recurrence relation is* **homogeneous** *because no terms occur that are not multiples of the a_js. The coefficients of the terms of the sequence are all* **constants***, rather than functions that depend on n. The* **degree** *is k because an is expressed in terms of the previous k terms of the sequence.*

A consequence of the second principle of mathematical induction is that a sequence satisfying the recurrence relation in the definition is uniquely determined by this recurrence relation and the k initial conditions

$$a_0 = C_0,\ a_1 = C_1, \cdots, a_{k-1} = C_{k-1}.$$

Example 161. *The recurrence relation $P_n = (1.11)P_{n-1}$ is a linear homogeneous recurrence relation of degree one. The recurrence relation*

$f_n = f_{n-1} + f_{n-2}$ is a linear homogeneous recurrence relation of degree two. The recurrence relation $a_n = a_{n-5}$ is a linear homogeneous recurrence relation of degree five.

Example 162. *The recurrence relation $a_n = a_{n-1} + a^2$ is not linear. The recurrence relation $H_n = 2H_{n-1} + 1$ is not homogeneous. The recurrence relation $B_n = nB_{n-1}$ does not have constant coefficients.*

3.2.1 Solving Linear Homogeneous Recurrence Relations with Constant Coefficients

Recurrence relations may be difficult to solve, but fortunately this is not the case for linear homogenous recurrence relations with constant coefficients. We can use two key ideas to find all their solutions. First these recurrence relations have solutions of the form $a_n = r^n$, where r is a constant. To see this, observe that $a_n = r^n$ is a solution of the recurrence relation $a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k\, a_{n-k}$ if and only if

$$r_n = c_1\, r^{n-1} + c_2\, r^{n-2} + \cdots + c_k\, r^{n-k}.$$

When both sides of this equation are divided by r^{n-k} and the right-hand side is subtracted from the left, we obtain the equation

$$r^k - c_1\, r^{k-1} - c_2\, r^{k-2} - \cdots - c_{k-1} r - c_k = 0.$$

Consequently, the sequence $\{a_n\}$ with $a_n = r^n$ is a solution if and only if r is a solution of this last equation. We call this the **characteristic equation** *of the recurrence relation. The solutions of this equation are called the* **characteristic roots** *of the recurrence relation. As we will see, these characteristic roots can be used to give an explicit formula for all* *the solutions of the recurrence relation.*

The other key observation is that a linear combination of two solutions of a linear homogeneous recurrence relation is also a solution. To see this, suppose that s_n *and* t_n *are both solutions of* $a_n = c_1a_{n-1} + c_2a_{n-2} + \cdots + c_ka_{n-k}$. *Then*

$$s_n = c_1s_{n-1} + c_2s_{n-2} + \cdots + c_ks_{n-k}$$

and

$$t_n = c_1t_{n-1} + c_2t_{n-2} + \cdots + c_kt_{n-k}.$$

Now suppose that b_1 *and* b_2 *are real numbers. Then*

$$\begin{aligned} b_1s_n + b_2t_n &= b_1(c_1s_{n-1} + c_2s_{n-2} + \\ &\quad \cdots + c_ks_{n-k}) + b_2(c_1t_{n-1} + c_2t_{n-2} + \cdots + c_kt_{n-k}) \\ &= c_1(b_1s_{n-1} + b_2t_{n-1}) + c_2(b_1s_{n-2} + b_2t_{n-2}) + \\ &\quad \cdots + c_k(b_1s_{n-k} + b_kt_{n-k}). \end{aligned}$$

This means that $b_1s_n + b_2t_n$ *is also a solution of the same linear homogeneous recurrence relation.*

Using these key observations, we will show how to solve linear homogeneous recurrence relations with constant coefficients.

The Degree Two Case *We now turn our attention to linear homogeneous recurrence relations of degree two. First, consider the case when there are two distinct characteristic roots.*

Theorem 163. *Let* c_1 *and* c_2 *be real numbers. Suppose that* $r^2 - c_1r - c_2 = 0$ *has two distinct roots* r_1 *and* r_2. *Then the sequence* $\{a_n\}$ *is a solution of the recurrence relation* $a_n = c_1\ a_{n-1} + c_2a_{n-2}$ *if and only if* $a_n = \alpha_1r_1^n + \alpha_2^n$ *for* $n = 0, 1, 2, \cdots$, *where* α_1 *and* α_2 *are constants.*

Proof. We must do two things to prove the theorem. First, it must be shown that if r_1 and r_2 are the roots of the characteristic equation, and α_1 and α_2 are constants, then the sequence $\{a_n\}$ with $a_n = \alpha_1 r_1^n + \alpha_2 r_2^n$ is a solution of the recurrence relation. Second, it must be shown that if the sequence $\{a_n\}$ is a solution, then $a_n = \alpha_1 r_1^n + \alpha_2 r_2^n$ for some constants α_1 and α_2.

Now we will show that if $a_n = \alpha_1 r_1^n + \alpha_2 r_2^n$, then the sequence $\{a_n\}$ is a solution of the recurrence relation. Because r_1 and r_2 are roots of $r^2 - c_1 r - c_2 = 0$, it follows that $r_1^2 = c_1 r_1 + c_2$ and $r_2^2 = c_1 r_2 + c_2$.

From these equations, we see that

$$\begin{aligned}
c_1 a_{n-1} + c_2 a_{n-2} &= c_1(\alpha_1 r_1^{n-1} + \alpha_2 r_2^{n-1}) + c_2(\alpha_1 r_1^{n-2} + a_2 r_2^{n-2}) \\
&= \alpha_1 r_1^{n-2}(c_1 r_1 + c_2) + a_2 r_2^{n-2}(c_1 r_2 + c_2) \\
&= \alpha_1 r_1^{n-2} r_1^2 + \alpha_2 r_2^{n-2} r_2^2 \\
&= \alpha_1 r_1^n + \alpha_2 r_2^n \\
&= a_n.
\end{aligned}$$

This shows that the sequence $\{a_n\}$ with $a_n = \alpha_1 r_1^n + \alpha_2 r_2^n$ is a solution of the recurrence relation.

To show that every solution $\{a_n\}$ of the recurrence relation $a_n = c_1\ a_{n-1} + c_2 a_{n-2}$ has $a_n = \alpha_1 r_1^n + \alpha_2 r_2^n$ for $n = 0, 1, 2, \cdots$, for some constants α_1 and α_2, suppose that $\{a_n\}$ is a solution of the recurrence relation, and the initial conditions $a_0 = C_0$ and $a_1 = C_1$ hold. It will be shown that there are constants α_1 and α_2 such that the sequence $\{a_n\}$ with $a_n = \alpha_1 r_1^n + \alpha_2 r_2^n$ satisfies these same initial conditions. This requires that

$$\begin{aligned} a_0 &= C_0 = \alpha_1 + \alpha_2, \\ a_1 &= C_1 = \alpha_1 r_1 + \alpha_2 r_2. \end{aligned}$$

We can solve these two equations for α_1 and α_2. From the first equation it follows that $\alpha_2 = C_0 - \alpha_1$. Inserting this expression into the second equation gives

$$C_1 = \alpha_1 r_1 + (C_0 - \alpha_1) r_2.$$

Hence,

$$C_1 = \alpha_1 (r_1 - r_2) + C_0 r_2.$$

This shows that

$$\alpha_1 = \frac{C_1 - C_0 r_2}{r_1 - r_2}$$

and

$$\alpha_2 = C_0 - \alpha_1 = C_0 - \frac{C_1 - C_0 r_2}{r_1 - r_2} = \frac{C_0 r_1 - C_1}{r - 1 - r - 2},$$

where these expressions for α_1 and α_2 depend on the fact that $r_1 \neq r_2$. (When $r_1 = r_2$, this theorem is not true.) Hence, with these values for α_1 and α_2, the sequence $\{a_n\}$ with $\alpha_1 r_1^n + \alpha_2 r_2^n$ satisfies the two initial conditions.

We know that $\{a_n\}$ and $\{\alpha_1 r_1^n + \alpha_2 r_2^n\}$ are both solutions of the recurrence relation $a_n = c_1 a_{n-1} + c_2 a_{n-2}$ and both satisfy the initial conditions when $n = 0$ and $n = 1$. Because there is a unique solution of a linear homogeneous recurrence relation of degree two with two initial conditions, it follows that the two solutions are the same, that is, $a_n = \alpha_1 r_1^n + \alpha_2 r_2^n$ for all nonnegative integers n. We have completed the proof by showing that a solution of the linear homogeneous recurrence relation with constant coefficients of degree two must be of the form $a_n = \alpha_1 r_1^n + \alpha_2 r_2^n$, where α_1 and α_2 are constants. □

Example 164. *What is the solution of the recurrence relation*

$$a_n = a_{n-1} + 2a_{n-2}$$

with $a_0 = 2$ and $a_1 = 7$?

Solution: *Theorem 163 can be used to solve this problem. The characteristic equation of the recurrence relation is $r_2 - r - 2 = 0$. Its roots are $r = 2$ and $r = -1$. Hence, the sequence $\{a_n\}$ is a solution to the recurrence relation if and only if*

$$a_n = \alpha_1 2^n + \alpha_2(-1)^n,$$

for some constants α_1 and α_2. From the initial conditions, it follows that

$$\begin{aligned} a_0 &= 2 = \alpha_1 + \alpha_2, \\ a_1 &= 7 = \alpha_1 \cdot 2 + \alpha_2 \cdot (-1). \end{aligned}$$

Solving these two equations shows that $\alpha_1 = 3$ and $\alpha_2 = -1$. Hence, the solution to the recurrence relation and initial conditions is the sequence $\{a_n\}$ with

$$a_n = 3 \cdot 2^n - (-1)^n.$$

Example 165. *Find an explicit formula for the Fibonacci numbers.*

Solution: *Recall that the sequence of Fibonacci numbers satisfies the recurrence relation $f_n = f_{n-1} + f_{n-2}$ and also satisfies the initial conditions $f_0 = 0$ and $f_1 = 1$. The roots of the characteristic equation $r^2 - r - 1 = 0$ are $r_1 = (1 + \sqrt{5})/2$ and $r_2 = (1 - \sqrt{5})/2$. Therefore, from Theorem 163* *it follows that the Fibonacci numbers are given by*

$$f_n = \alpha_1 \left(\frac{1+\sqrt{5}}{2}\right)^n + \alpha_2 \left(\frac{1-\sqrt{5}}{2}\right)^n,$$

for some constants α_1 and α_2. The initial conditions $f_0 = 0$ and $f_1 = 1$ can be used to find these constants. We have

$$\begin{aligned} f_0 &= \alpha_1 + \alpha_2 = 0, \\ f_1 &= \alpha_1 \left(\frac{1+\sqrt{5}}{2}\right) + \alpha_2 \left(\frac{1-\sqrt{5}}{2}\right) = 1. \end{aligned}$$

The solution to these simultaneous equations for α_1 and α_2 is

$$\alpha_1 = 1/\sqrt{5}, \qquad \alpha_2 = -1/\sqrt{5}.$$

Consequently, the Fibonacci numbers are given by

$$f_n = \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}} \left(\frac{1-\sqrt{5}}{2}\right)^n.$$

Theorem 166. *Let c_1 and $_2$ be real numbers with $c_2 \neq 0$. Suppose that $r^2 - c_1 r - c_2 = 0$ has only one root r_0. A sequence $\{a_n\}$ is a solution of the recurrence relation $a_n = c_1 a_{n-1} + c_2 a_{n-2}$ if and only if $a_n = \alpha_1 r_0^n + \alpha_2 n r_0^n$, for $n = 0, 1, 2, \cdots$, where α_1 and α_2 are constants.*

Example 167. *What is the solution of the recurrence relation*

$$a_n = 6a_{n-1} - 9a_{n-2}$$

with initial conditions $a_0 = 1$ and $a_1 = 6$?

Solution: *The only root of $r^2 - 6r + 9 = 0$ is $r = 3$. Hence, the solution* *to this recurrence relation is*

$$a_n = \alpha_1 3^n + \alpha_2 n 3^n$$

for some constants α_1 *and* α_2*. Using the initial conditions, it follows that*

$$\begin{aligned} a_0 &= 1 = \alpha_1, \\ a_1 &= 6 = \alpha_1 \cdot 3 + \alpha_2 \cdot 3. \end{aligned}$$

Solving these two equations shows that $\alpha_1 = 1$ *and* $\alpha_2 = 1$*. Consequently, the solution to this recurrence relation and the initial conditions is*

$$a_n = 3^n + n3^n.$$

The General Case: *We will now state the general result about the solution of linear homogeneous recurrence relations with constant coefficients, where the degree may be greater than two, under the assumption that the characteristic equation has distinct roots.*

Theorem 168. *Let* $c_1, c_2, \cdots, c_k$ *be real numbers. Suppose that the characteristic equation*

$$r^k - c_1\, r^{k-1} - \cdots - c_k = 0$$

has k *distinct roots* $r_1, r_2, \cdots, r_k$*. Then a sequence* $\{a_n\}$*is a solution of the recurrence relation*

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}$$

if and only if

$$a_n = \alpha_1 r_1^n + \alpha_2 r_2^n + \cdots + \alpha_k r_k^n$$

for $n = 0, 1, 2, \cdots$, *where* $\alpha_1, \alpha_2, \cdots, \alpha_k$ *are constants.*

Example 169. *Find the solution to the recurrence relation*

$$a_n = 6a_{n-1} - 11a_{n-2} + 6a_{n-3}$$

with the initial conditions $a_0 = 2$, $a_1 = 5$, *and* $a_2 = 15$.

Solution: *The characteristic polynomial of this recurrence relation is*

$$r^3 - 6r^2 + 11r - 6.$$

The characteristic roots are $r = 1$, $r = 2$, *and* $r = 3$, *because* $r^3 - 6r^2 + 11r - 6 = (r-1)(r-2)(r-3)$. *Hence, the solutions to this recurrence relation are of the form*

$$a_n = \alpha_1 \cdot 1^n + \alpha_2 \cdot 2^n + \alpha_3 \cdot 3^n.$$

To find the constants α_1, α_2, *and* α_3, *use the initial conditions. This gives*

$$\begin{aligned} a_0 &= 2 = \alpha_1 + \alpha_2 + \alpha_3, \\ a_1 &= 5 = \alpha_1 + \alpha_2 \cdot 2 + \alpha_3 \cdot 3, \\ a_2 &= 15 = \alpha_1 + \alpha_2 \cdot 4 + \alpha_3 \cdot 9. \end{aligned}$$

When these three simultaneous equations are solved for α_1, α_2, *and* α_3, *we find that* $\alpha_1 = 1$, $\alpha_2 = -1$, *and* $\alpha_3 = 2$. *Hence, the unique solution to this recurrence relation and the given initial conditions is the sequence* $\{a_n\}$ *with*

$$a_n = 1 - 2^n + 2 \cdot 3^n.$$

Theorem 170. *Let* $c_1, c_2, \cdots, c_k$ *be real numbers. Suppose that the char-*

acteristic equation

$$r^k - c_1r^{k-1} - \cdots - c_k = 0$$

has t *distinct roots* $r_1, r_2, \cdots, r_t$ *with multiplicities* $m_1, m_2, \cdots, m_t$*, respectively, so that* $m_i \geq 1$ *for* $i = 1, 2, \cdots, t$ *and* $m_1+m_2+\cdots+m_t = k$*. Then a sequence* $\{a_n\}$ *is a solution of the recurrence relation*

$$a_n = c_1a_{n-1} + c_2a_{n-2} + \cdots + c_ka_{n-k}$$

if and only if

$$\begin{aligned} a_n &= (\alpha_{1,0} + \alpha_{1,1}n + \cdots + \alpha_{1,m_1-1}n^{m_1-1}r_1^n \\ &\quad +(\alpha_{2,0} + \alpha_{2,1}n + \cdots + a_{2,m_2-1}n^{m_2-1})r_2^n \\ &\quad +\cdots + (\alpha_{t,0} + \alpha_{t,1}n + \cdots \alpha_{t,m_t-1}n^{m_t-1})r_t^n \end{aligned}$$

for $n = 0, 1, 2, \cdots$ *, where* $\alpha_{i,j}$ *are constants for* $1 \leq i \leq t$ *and* $0 \leq j \leq m_{i-1}$.

Example 171. *Suppose that the roots of the characteristic equation of a linear homogeneous recurrence relation are 2, 2, 2, 5, 5, and 9 (that is, there are three roots, the root 2 with multiplicity three, the root 5 with multiplicity two, and the root 9 with multiplicity one). What is the form of the general solution?*

Solution: *By Theorem 170, the general form of the solution is*

$$(\alpha_{1,0} + \alpha_{1,1}n + \alpha_{1,2}n^2)2^n + (\alpha_{2,0} + \alpha_{2,1}n)5^n + \alpha_{3,0}9^n.$$

Example 172. *Find the solution to the recurrence relation*

$$a_n = -3a_{n-1} - 3a_{n-2} - a_{n-3}$$

with initial conditions $a_0 = 1$, $a_1 = -2$, *and* $a_2 = -1$.

Solution: *The characteristic equation of this recurrence relation is*

$$r^3 + 3r^2 + 3r + 1 = 0.$$

Because $r^3 + 3r^2 + 3r + 1 = (r+1)^3$, *there is a single root* $r = -1$ *of multiplicity three of the characteristic equation. By Theorem 170 the solutions of this recurrence relation are of the form*

$$a_n = \alpha_{1,0}(-1)^n + \alpha_{1,1}n(-1)^n + \alpha_{1,2}n^2(-1)^n.$$

To find the constants $\alpha_{1,0}, \alpha_{1,1}$ *and* $\alpha_{1,2}$, *use the initial conditions. This gives*

$$\begin{aligned} a_0 &= 1 = \alpha_{1,0}, \\ a_1 &= -2 = -\alpha_{1,0} - \alpha_{1,1} - \alpha_{1,2}, \\ a_2 &= -1 = \alpha_{1,0} + 2\alpha_{1,1} + 4\alpha_{1,2}. \end{aligned}$$

The simultaneous solution of these three equations is $\alpha_{1,0} = 1$, $\alpha_{1,1} = 3$, *and* $\alpha_{1,2} = -2$. *Hence, the unique solution to this recurrence relation and the given initial conditions is the sequence* $\{a_n\}$ *with*

$$a_n = (1 + 3n - 2n^2)(-1)^n.$$

3.2.2 Linear Nonhomogeneous Recurrence Relations with Constant Coefficients

We have seen how to solve linear homogeneous recurrence relations with constant coefficients. Is there a relatively simple technique for solving a linear, but not homogeneous, recurrence relation with constant coeffi-

cients, such as $a_n = 3a_{n-1} + 2n$? We will see that the answer is yes for certain families of such recurrence relations.

The recurrence relation $a_n = 3a_{n-1} + 2n$ is an example of a **linear nonhomogeneous recurrence relation with constant coefficients***, that is, a recurrence relation of the form*

$$a_n = c_1a_{n-1} + c_2a_{n-2} + \cdots + c_ka_{n-k} + F(n),$$

where $c_1, c_2, \cdots, c_k$ are real numbers and $F(n)$ is a function not identically zero depending only on n. The recurrence relation

$$a_n = c_1a_{n-1} + c_2a_{n-2} + \cdots + c_ka_{n-k}$$

is called the **associated homogeneous recurrence relation***. It plays an important role in the solution of the nonhomogeneous recurrence relation.*

Example 173. *Each of the recurrence relations $a_n = a_{n-1} + 2^n$, $a_n = a_{n-1} + a_{n-2} + n^2 + n + 1$, $a_n = 3a_{n-1} + n3^n$, and $a_n = a_{n-1} + a_{n-2} + a_{n-3} + n!$ is a linear nonhomogeneous recurrence relation with constant coefficients. The associated linear homogeneous recurrence relations are $a_n = a_{n-1}$, $a_n = a_{n-1}+a_{n-2}$, $a_n = 3a_{n-1}$, and $a_n = a_{n-1}+a_{n-2}+a_{n-3}$, respectively.*

Theorem 174. *If $\{a_n^{(p)}\}$ is a particular solution of the nonhomogeneous linear recurrence relation with constant coefficients*

$$a_n = c_1a_{n-1} + c_2a_{n-2} + \cdots + c_ka_{n-k} + F(n),$$

then every solution is of the form $\{a_n^{(p)} + a_n^{(h)}\}$, where $\{a_n^{(h)}\}$ is a solution of the associated homogeneous recurrence relation

$$a_n = c_1a_{n-1} + c_2a_{n-2} + \cdots + c_ka_{n-k}.$$

Proof. Because $\{a_n^{(p)}\}$ is a particular solution of the nonhomogeneous recurrence relation, we know that

$$a_n^{(p)} = c_1 a_{n-1}^{(p)} + c_2 a_{n-2}^{(p)} + \cdots + c_k a_{n-k}^{(p)} + F(n).$$

Now suppose that $\{b_n\}$ is a second solution of the nonhomogeneous recurrence relation, so that

$$b_n = c_1 b_{n-1} + c_2 b_{n-2} + \cdots + c_k b_{n-k} + F(n).$$

Subtracting the first of these two equations from the second shows that

$$b_n - a_n^{(p)} = c_1(b_{n-1} - a_{n-1}^{(p)} + c_2(b_{n-2} - a_{n-2}^{(p)}) + \cdots + c_k(b_{n-k} - a_{n-k}^{(p)}).$$

It follows that $\{b_n - a_n^p\}$ is a solution of the associated homogeneous linear recurrence, say, $\{a_n^{a(h)}\}$. Consequently, $b_n = a_n^{(p)} + a_n^{(h)}$ for all n. □

Example 175. *Find all solutions of the recurrence relation $a_n = 3a_{n-1} + 2n$. What is the solution with $a_1 = 3$?*

Solution: *To solve this linear nonhomogeneous recurrence relation with constant coefficients, we need to solve its associated linear homogeneous equation and to find a particular solution for the given nonhomogeneous equation. The associated linear homogeneous equation is $a_n = 3a_{n-1}$. Its solutions are $a_n^{(h)} = \alpha 3^n$, where α is a constant.*

We now find a particular solution. Because $F(n) = 2n$ is a polynomial in n of degree one, a reasonable trial solution is a linear function in n, say, $p_n = cn + d$, where c and d are constants. To determine whether there are any solutions of this form, suppose that $p_n = cn + d$ is such a solution. Then the equation $a_n = 3a_{n-1} + 2n$ becomes $cn + d = 3(c(n-1) + d) + 2n$. Simplifying and combining like terms gives $(2 + 2c)n + (2d - 3c) = 0$. It *follows that* $cn + d$ *is a solution if and only if* $2 + 2c = 0$ *and* $2d - 3c = 0$. *This shows that* $cn + d$ *is a solution if and only if* $c = -1$ *and* $d = -3/2$. *Consequently,* $a_n^{(p)} = -n - 3/2$ *is a particular solution.*

By Theorem 174 all solutions are of the form

$$a_n = a_n^{(p)} + a_n^{(h)} = -n - \frac{3}{2} + \alpha \cdot 3^n,$$

where α *is a constant.*

To find the solution with $a_1 = 3$, *let* $n = 1$ *in the formula we obtained for the general solution. We find that* $3 = -1 - 3/2 + 3\alpha$, *which implies that* $\alpha = 11/6$. *The solution we seek is* $a_n = -n - 3/2 + (11/6)3^n$.

Example 176. *Find all solutions of the recurrence relation*

$$a_n = 5a_{n-1} - 6a_{n-2} + 7^n.$$

Solution: *This is a linear nonhomogeneous recurrence relation. The solutions of its associated homogeneous recurrence relation*

$$a_n = 5a_{n-1} - 6a_{n-2}$$

are $a_n^{(h)} = \alpha_1 \cdot 3^n + \alpha_2 \cdot 2^n$, *where* α_1 *and* α_2 *are constants. Because* $F(n) = 7^n$, *a reasonable trial solution is* $a_n^{(p)} = C \cdot 7^n$, *where* C *is a constant. Substituting the terms of this sequence into the recurrence relation implies that* $C \cdot 7^n = 5C \cdot 7^{n-1} - 6C \cdot 7^{n-2} + 7n$. *Factoring out* 7^{n-2}, *this equation becomes* $49C = 35C - 6C + 49$, *which implies that* $20C = 49$, *or that* $C = 49/20$. *Hence,* $a_n^{(p)} = (49/20)7^n$ *is a particular solution. By Theorem 174, all solutions are of the form*

$$a_n = \alpha_1 \cdot 3^n + \alpha_2 \cdot 2^n + (49/20)7^n.$$

Theorem 177. *Suppose that $\{a_n\}$ satisfies the linear nonhomogeneous recurrence relation*

$$a_n = c_1a_{n-1} + c_2a_{n-2} + \cdots + c_ka_{n-k} + F(n),$$

where $c_1, c_2, \cdots, c_k$ are real numbers, and

$$F(n) = (b_tn^t + b_{t-1}n^{t-1} + \cdots + b_{1n} + b_0)s^n,$$

where $b_0, b_1, \cdots, b_t$ and s are real numbers. When s is not a root of the characteristic equation of the associated linear homogeneous recurrence relation, there is a particular solution of the form

$$(p_tn^t + p_{t-1}n^{t-1} + \cdots + p1n + p_0)s^n.$$

When s is a root of this characteristic equation and its multiplicity is m, there is a particular solution of the form

$$n^m(p_tn^t + p_{t-1}n^{t-1} + \cdots + p_1n + p_0)s^n.$$

Example 178. *What form does a particular solution of the linear nonhomogeneous recurrence relation $a_n = 6a_{n-1} - 9a_{n-2} + F(n)$ have when $F(n) = 3n$, $F(n) = n3^n$, $F(n) = n^22^n$, and $F(n) = (n^2 + 1)3^n$?*

Solution: *The associated linear homogeneous recurrence relation is $a_n = 6a_{n-1} - 9a_{n-2}$. Its characteristic equation, $r^2 - 6r + 9 = (r - 3)2 = 0$, has a single root, 3, of multiplicity two. To apply Theorem 177, with $F(n)$ of the form $P(n)s^n$, where $P(n)$ is a polynomial and s is a constant, we need to ask whether s is a root of this characteristic equation.*

Because $s = 3$ is a root with multiplicity $m = 2$ but $s = 2$ is not a root, Theorem 177 tells us that a particular solution has the form $p_0n^23^n$

if $F(n) = 3^n$, the form $n^2(p_1n + p0)3^n$ if $F(n) = n3^n$, the form $(p_2n^2 + p_1n + p_0)2^n$ if $F(n) = n^22^n$, and the form $n^2(p_2n^2 + p_1n + p_0)3^n$ if $F(n) = (n^2 + 1)3^n$.

Example 179. *Let a_n be the sum of the first n positive integers, so that*

$$a_n = \sum_{k=1}^{n} k.$$

Note that a_n satisfies the linear nonhomogeneous recurrence relation

$$a_n = a_{n-1} + n.$$

(To obtain a_n, the sum of the first n positive integers, from a_{n-1}, the sum of the first $n-1$ positive integers, we add n.) Note that the initial condition is $a_1 = 1$.

The associated linear homogeneous recurrence relation for a_n is

$$a_n = a_{n-1}.$$

The solutions of this homogeneous recurrence relation are given by $a_n^{(h)} = c(1)^n = c$, where c is a constant. To find all solutions of $a_n = a_{n-1} + n$, we need find only a single particular solution. By Theorem 177, because $F(n) = n = n\cdot(1)^n$ and $s = 1$ is a root of degree one of the characteristic equation of the associated linear homogeneous recurrence relation, there is a particular solution of the form $n(p_1n + p0) = p_1n^2 + p_0n$.

Inserting this into the recurrence relation gives $p_1n^2 + p_0n = p_1(n-1)^2 + p_0(n-1) + n$. Simplifying, we see that $n(2p_1 - 1) + (p_0 - p_1) = 0$, which means that $2p_1 - 1 = 0$ and $p_0 - p_1 = 0$, so $p_0 = p_1 = 1/2$. Hence,

$$a_n^{(p)} = \frac{n^2}{2} + \frac{n}{2} = \frac{n(n+1)}{2}$$

is a particular solution. Hence, all solutions of the original recurrence relation $a_n = a_{n-1} + n$ *are given by* $a_n = a_n^{(h)} + a_n^{(p)} = c + n(n+1)/2$. *Because* $a_1 = 1$, *we have* $1 = a_1 = c + 1 \cdot 2/2 = c + 1$, *so* $c = 0$. *It follows that* $a_n = n(n+1)/2$.

Let Us Sum Up

In this section, we discussed about

- *Solving linear homogeneous recurrence relations.*
- *Linear non homogeneous recurrence relations.*
- *Solving recurrence relations using substitutions.*

Check your Progress

1. What is the solution of the recurrence relation $a_n = 6a_{n-1} - 9a_{n-2}$ *with initial conditions* $a_0 = 1$ *and* $a_1 = 6$*?*

(a) $3^n + n3^{n-1}$

(b) 3^n

(c) 3^{n-1}

(d) $3^n + n3^n$

2. The solution of the recurrence relation $a_n = 3a_{n-1} + 2n$ *when* $a_1 = 3$ *is*

(a) $-n - \frac{3}{2} + \left(\frac{11}{6}\right)3^n$

(b) $\left(\frac{11}{6}\right)3^n$

(c) $\frac{3}{2} + \left(\frac{11}{6}\right)^n$

(d) $n + \frac{3}{2} + \left(\frac{11}{6}\right)^n$

3.3 Generating Functions

Definition 180. *The generating function for the sequence* $a_0, a_1, \cdots, a_k, \cdots$ *of real numbers is the infinite series*

$$G(x) = a_0 + a_{1x} + \cdots + a_k x^k + \cdots = \sum_{k=0}^{\infty} a_k x^k.$$

Remark 181. *The generating function forak given in Definition 180 is sometimes called the* **ordinary generating function** *of* $\{a_k\}$ *to distinguish it from other types of generating functions for this sequence.*

Example 182. *The generating functions for the sequences* $\{a_k\}$ *with* $a_k = 3$, $a_k = k+1$, *and* $a_k = 2^k$ *are* $\sum_{k=0}^{\infty} 3x^k$, $\sum_{k=0}^{\infty}(k+1)x^k$, *and* $\sum_{k=0}^{\infty} 2^k x^k$, *respectively.*

Definition 183. *Let* u *be a real number and* k *a nonnegative integer. Then the extended binomial coefficient* $\binom{u}{k}$ *is defined by*

$$\binom{u}{k} = \begin{cases} u(u-1)\cdots(u-k+1)/k! & \textit{if } k > 0 \\ 1 & \textit{if } k = 0. \end{cases}$$

Example 184. *Find the values of the extended binomial coefficient* $\binom{-2}{3}$ *and* $\binom{1/2}{3}$.

Solution: *Taking* $u = -2$ *and* $k = 3$ *in Definition 183 gives us*

$$\binom{-2}{3} = \frac{(-2)(-3)(-4)}{3!} = -4.$$

Similarly, taking $u = 1/2$ and $k = 3$ gives us

$$\begin{aligned}\binom{1/2}{3} &= \frac{(1/2)(1/2-1)(1/2-2)}{3!} \\ &= (1/2)(-1/2)(-3/2)/6 \\ &= 1/16.\end{aligned}$$

Example 185. *When the top parameter is a negative integer, the extended binomial coefficient can be expressed in terms of an ordinary binomial coefficient. To see that this is the case, note that*

$$\begin{aligned}\binom{-n}{r} &= \frac{(-n)(-n-1)\cdots(-n-r+1)}{r!} \\ &= \frac{(-1)^n n(n+1)\cdots(n+r-1)}{r!} \\ &= \frac{(-1)^r(n+r-1)(n+r-2)\cdots n)}{r!} \\ &= (-1)^r\frac{(n+r-1)!}{r!(n-1)!} \\ &= (-1)^r\binom{n+r-1}{r} \\ &= (-1)^r C(n+r-1, r).\end{aligned}$$

Theorem 186. THE EXTENDED BINOMIAL THEOREM *Let x be a real number with $|x| < 1$ and let u be a real number. Then*

$$(1+x)^u = \sum_{k=0}^{\infty}\binom{u}{k}x^k.$$

Example 187. *Find the generating functions for $(1+x)^{-n}$ and $(1-x)^{-n}$, where n is a positive integer, using the extended binomial theorem.*

Solution: *By the extended binomial theorem, it follows that*

$$(1+x)^{-n} = \sum_{k=0}^{\infty}\binom{-n}{k}x^k.$$

Using Example 185, which provides a simple formula for $\binom{-n}{k}$, we obtain

$$(1+x)^{-n} = \sum_{k=0}^{\infty}(-1)^k C(n+k-1,k)x^k.$$

Replacing x by $-x$, we find that

$$(1-x)^{-n} = \sum_{k=0}^{\infty} C(n+k-1,k)x^k.$$

3.3.1 Counting Problems and Generating Functions

Example 188. *Find the number of solutions of*

$$e_1 + e_2 + e_3 = 17,$$

where e_1, e_2, and e_3 are nonnegative integers with $2 \le e_1 \le 5$, $3 \le e_2 \le 6$, and $4 \le e_3 \le 7$.

Solution: *The number of solutions with the indicated constraints is the coefficient of x^{17} in the expansion of*

$$(x^2+x^3+x^4+x^5)(x^3+x^4+x^5+x^6)(x^4+x^5+x^6+x^7).$$

This follows because we obtain a term equal to x^{17} in the product by picking a term in the first sum x^{e_1} , a term in the second sum x^{e_2}, and a term in the third sum x^{e_3}, where the exponents e_1, e_2, and e_3 satisfy the equation $e_1 + e_2 + e_3 = 17$ and the given constraints.

It is not hard to see that the coefficient of x^{17} in this product is 3. Hence, there are three solutions.

Table 3.1: Useful Generating Functions.

$G(x)$	a_k
$(1+x)^n = \sum_{k=0}^{n} C(n,k)x^k$ $= 1 + C(n,1)x + C(n,2)x^2 + \cdots + x^n$	$C(n,k)a^k$
$(1+ax)^n = \sum_{k=0}^{n} C(n,k)a^k x^k$ $= 1 + C(n,1)ax + C(n,2)a^2x^2 + \cdots + a^n x^n$	$C(n,k)a^k$
$(1+x^r)^n = \sum_{k=0}^{n} C(n,k)x^{rk}$ $= 1 + C(n,1)x^r + C(n,2)x^{2r} + \cdots + x^{rn}$	$C(n,k/r)$ if $r \mid k$; 0 otherwise
$\frac{1-x^{n+1}}{1-x} = \sum_{k=0}^{n} x^k = 1 + x + x^2 + \cdots + x^n$	1 if $k \leq n$; 0 otherwise
$\frac{1}{1-x} = \sum_{k=0}^{\infty} x^k = 1 + x + x^2 + \cdots$	1
$\frac{1}{1-ax} = \sum_{k=0}^{\infty} a^k x^k = 1 + ax + a^2x^2 + \cdots$	a^k
$\frac{1}{1-x^r} = \sum_{k=0}^{\infty} x^{rk} = 1 + x^r + x^{2r} + \cdots$	1 if $r \mid k$; 0 otherwise
$\frac{1}{1-x)^2} = \sum_{k=0}^{\infty}(k+1)x^k = 1 + 2x + 3x^2 + \cdots$	$k+1$
$\frac{1}{(1-x)^n} = \sum_{k=0}^{\infty} C(n+k-1,k)x^k$ $= 1 + C(n,1)x + C(n+1,2)x^2 + \cdots$	$C(n+k-1,k)$ $= C(n+k-1,n-1)$
$\frac{1}{(1+x)^n} = \sum_{k=0}^{\infty} C(n+k-1,k)(-1)^k x^k$ $= 1 - C(n,1)x + C(n+1,2)x^2 + \cdots$	$(-1)^k C(n+k-1,k)$ $= (-1)^k C(n+k-1,n-1)$
$\frac{1}{(1-ax)^n} = \sum_{k=0}^{\infty} C(n+k-1,k)a^k x^k$ $= 1 + C(n,1)ax + C(n+1,2)a^2x^2 + \cdots$	$C(n+k-1,k)a^k$ $= C(n+k-1,n-1)a^k$
$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$	$1/k!$
$In(1+x) = \sum_{k=1}^{\infty} \frac{(-1)^{k+1)}}{k} x^k = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \cdots$	$(-1)^{k+1}/k$

Example 189. *In how many different ways can eight identical cookies be distributed among three distinct children if each child receives at least*

two cookies and no more than four cookies?

Solution: *Because each child receives at least two but no more than four cookies, for each child there is a factor equal to*

$$(x^2 + x^3 + x^4)$$

in the generating function for the sequence $\{c_n\}$, where c_n is the number of ways to distribute n cookies. Because there are three children, this generating function is

$$(x^2 + x^3 + x^4)^3.$$

We need the coefficient of x^8 in this product. The reason is that the x^8 terms in the expansion correspond to the ways that three terms can be selected, with one from each factor, that have exponents adding up to 8. Furthermore, the exponents of the term from the first, second, and third factors are the numbers of cookies the first, second, and third children receive, respectively. Computation shows that this coefficient equals 6. Hence, there are six ways to distribute the cookies so that each child receives at least two, but no more than four, cookies.

Example 190. *Use generating functions to find the number of k-combinations of a set with n elements. Assume that the binomial theorem has already been established.*

Solution: *: Each of the n elements in the set contributes the term $(1 + x)$ to the generating function $f(x) = \sum_{k=0}^{n} a_k x^k$. Here $f(x)$ is the generating function for $\{a_k\}$, where a_k represents the number of k-combinations of a set with n elements. Hence,*

$$f(x) = (1 + x)^n.$$

But by the binomial theorem, we have

$$f(x) = \sum_{k=0}^{\infty} \binom{n}{k} x^k,$$

where

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}.$$

Hence, $C(n,k)$, the number of k-combinations of a set with n elements, is

$$\frac{n!}{k!(n-k)!}.$$

Example 191. *Use generating functions to find the number of r-combinations from a set with n elements when repetition of elements is allowed.*

Solution: *Let $G(x)$ be the generating function for the sequence $\{a_r\}$, where a_r equals the number of r-combinations of a set with n elements with repetitions allowed. That is, $G(x) = \sum_{r=0}^{\infty} a_r x^r$. Because we can select any number of a particular member of the set with n elements when we form an r-combination with repetition allowed, each of the n elements contributes $(1+x+x^2+x^3+\cdots)$ to a product expansion for $G(x)$. Each element contributes this factor because it may be selected zero times, one time, two times, three times, and so on, when an r-combination is formed (with a total of r elements selected). Because there are n elements in the set and each contributes this same factor to $G(x)$, we have*

$$G(x) = (1 + x + x^2 + \cdots)^n.$$

As long as $|x| < 1$, we have $1 + x + x^2 + \cdots = 1/(1-x)$, so

$$G(x) = 1/(1-x)^n = (1-x)^{-n}.$$

Applying the extended binomial theorem (Theorem 166), it follows that

$$(1-x)^{-n} = (1+(-x))^{-n} = \sum_{k=0}^{\infty} \binom{-n}{r} (-x)^r.$$

The number of r-combinations of a set with n elements with repetitions allowed, when r is a positive integer, is the coefficient a_r of x_r in this sum. Consequently, using above Example we find that a_r equals

$$\begin{aligned}\binom{-n}{r}(-1)^r &= (-1)^r C(n+r-1,r)\cdot(-1)^r \\ &= C(n+r-1,r).\end{aligned}$$

Example 192. *Use generating functions to find the number of ways to select r objects of n different kinds if we must select at least one object of each kind.*

Solution: *: Because we need to select at least one object of each kind, each of the n kinds of objects contributes the factor $(x+x^2+x^3+\cdots)$to the generating function $G(x)$for the sequence $\{a_r\}$, where a_r is the number of ways to select r objects of n different kinds if we need at least one object of each kind. Hence*

$$G(x) = (x+x^2+x^3+\cdots)^n = x^n(1+x+x^2+\cdots)^n = x^n/(1-x)^n.$$

Using the extended binomial theorem and above Example, we have

$$\begin{aligned}G(x) &= x^n/(1-x)^n \\ &= x^n\cdot(1-x)^{-n} \\ &= x^n \sum_{r=0}^{\infty} \binom{-n}{r}(-x)^r \\ &= x^n \sum_{r=0}^{\infty} (-1)^r C(n+r-1,r)(-1)^r x^r\end{aligned}$$

$$\begin{aligned}
&= \sum_{r=0}^{\infty} C(n+r-1,r)x^{n+r}\\
&= \sum_{t=n}^{\infty} C(t-1,t-n)x^{t}\\
&= \sum_{r=n}^{\infty} C(r-1,r-n)x^{r}.
\end{aligned}$$

We have shifted the summation in the next-to-last equality by setting $t = n + r$ *so that* $t = n$ *when* $r = 0$ *and* $n + r - 1 = t - 1$*, and then we replaced* t *by* r *as the index of summation in the last equality to return to our original notation. Hence, there are* $C(r - 1, r - n)$ *ways to select* r *objects of* n *different kinds if we must select at least one object of each kind.*

3.3.2 Exponential Generating Functions

Earlier, we have seen that generating functions can be used for enumerating combinations. It is natural to see whether the idea can be excluded to permutations. But the first obstracle we face is that in permutations abc is different from cba whereas selecting a,b,c from a set is the same in whatever order your select. trying to use a power series for permutation of three elements a, b, c, we must get

$$1+(a+b+c)x+(ab+ba+ac+ca+cb)x^2+(abc+acb+bac+bca+cab+cba)x^3.$$

But is polynomial is equivalent to $1 + (a + b + c)x + (ab + ba + ac + ca + cb)x^2 + 6(abc)x^3$*. We cannot distinguish between abc and bca. Since, we do not want to discard commutative property in power series, the following idea is used for enumerating permutations.*

A direct extension of the notion of the enumerators for combinations *indicates that an enumerator for the permutation of n distinct objects would have the form*

$$\begin{aligned}
F(x) &= P(n,0)x^0 + P(n,1)x + P(n,2)x^2 + P(n,3)x^3 + \cdots + P(n,r)x^r \\
&\quad + \cdots + P(n,n)x^n \\
&= 1 + \frac{n!}{(n-1)!}x + \frac{n!}{(n-2)!}x^2 + \frac{n!}{(n-3)!}x^3 + \\
&\quad \cdots + \frac{n!}{(n-r)!}x^r + \cdots + n^1x^n.
\end{aligned}$$

Unfortunately, there is no simple closed form expression for the above function. But we know

$$\begin{aligned}
(1+x)^n &= 1 + \binom{n}{1}x + \binom{n}{2}x^2 + \cdots + \binom{n}{n}x^n \\
&= 1 + \frac{P(n,1)}{1!}x + \frac{P(n,2)}{2!}x^2 + \cdots + \frac{P(n,r)}{r!}x^r + \cdots + \frac{P(n,n)}{n!}x^n.
\end{aligned}$$

This we call as an exponential generating function. Let $(a_0, a_1, a_2, \cdots ,_r, \cdots)$ *be a sequence. the function*

$$F(x) = a_0 + \frac{a_1}{1!}x + \frac{a_2}{2!}x^2 + \frac{a_3}{3!}x^3 + \cdots + \frac{a_r}{r!}x^r + \cdots$$

is called the exponential generating function of the sequence $(a_0, a_1, a_2, \cdots a_r, \cdots)$.

Thus $(1+x)^n$ *is the exponential generating function of the* $P(n,r)$*s, i.e., permutations of* r *objects out of* n *objects.*

Example 193. *The exponential enumerator for the permutation of all* p *of* p *identical objects is* $\frac{x^p}{p!}$ *as there is only one way of doing so. Thus the exponential enumerator for the permutation of none, one, two, ... , p of p identical object is*

$$1 + \frac{1}{1!}x + \frac{1}{2!}x^2 + \cdots + \frac{1}{p!}x^p.$$

The exponential enumerator for the permutations of one, two, ... , p+q of p+q objects where p of them are of one kind and q of them are of another kind is

$$\left(1+\frac{1}{1!}x+\frac{1}{2!}x^2+\cdots+\frac{1}{p!}x^p\right)\left(1+\frac{1}{1!}x+\frac{1}{2!}x^2+\cdots+\frac{1}{q!}x^q\right).$$

To get the permutations of $p+q$ objects, where p of them are of one kind and q of them are of another kind, we see the factor $x^p x^q$, which is

$$\frac{x^p}{p!}\frac{x^q}{q!}=\frac{x^{p+q}}{p!q!}.$$

In the expression the answer we expect is given by

$$\frac{a}{(p+q)!}x^{p+q}.$$

Hence we find $a=\frac{(p+q)!}{p!q!}$ which we know is correct.

Example 194. *Let the alphabet consist of $\{0,1,2\}$. Find the number of r-digit binary sequences that contain an even number of 0's.*

The exponential enumerators for the permutation of digit 0 is

$$\left(1+\frac{x^2}{2!}+\frac{x^4}{4!}+\frac{x^6}{6!}+\cdots\right)=\frac{1}{2}(e^x+e^{-x}).$$

The exponential enumerator for the permutations of each of the digit 1 and 2 is

$$\left(1+\frac{x}{1!}+\frac{x^2}{2!}+\cdots\right)=e^x.$$

It follows that the exponential enumerator for the number of binary sequences containing an even number of 0's is

$$\frac{1}{2}(e^x+e^{-x})e^x x^x=\frac{1}{2}(e^{3x}+e^x=1+\sum_{r=1}^{\infty}\frac{1}{2}\left(\frac{3^r+1^r}{r!}\right)x^r.$$

Hence the number of r-digit ternary sequences that contain an even number of 0's is $\frac{3^r+1}{2}$.

For example when $r = 1$, *we have the possibilities 1 and 2 and* $\frac{3+1}{2} = 2$.

When $r = 2$ *we have possibilities 00, 12, 21,11, 22 and* $\frac{3^2+1}{2} = 5$.

3.3.3 Using Generating Functions to Solve Recurrence Relations

Example 195. *Solve the recurrence relation* $a_k = 3a_{k-1}$*for* $k = 1, 2, 3, \cdots$ *and initial condition* $a_0 = 2$.

Solution: *Let* $G(x)$ *be the generating function for the sequence* $\{a_k\}$, *that is,* $G(x) = \sum_{k=0}^{\infty} a_k x^k$. *First note that*

$$xG(x) = \sum_{k=0}^{\infty} a_k x^{k+1} = \sum_{k=1}^{\infty} a_{k-1} x^k.$$

Using the recurrence relation, we see that

$$\begin{aligned} G(x) - 3xG(x) &= \sum_{k=0}^{\infty} a_k x^k - 3\sum_{k=1}^{\infty} a_{k-1} x^k \\ &= a_0 + \sum_{k=1}^{\infty} (a_k - 3a_{k-1}) x^k \\ &= 2, \end{aligned}$$

because $a_0 = 2$ *and* $a_k - 3a_{k-1}$. *Thus,*

$$G(x) - 3xG(x) = (1 - 3x)G(x) = 2.$$

Solving for $G(x)$ *shows that* $G9x) = 2/(1 - 3x)$. *Using the identity*

$1/(1-ax) = \sum_{k=0}^{\infty} a^k x^k$, *from Table 1.2, we have*

$$G(x) = 2\sum_{k=0}^{\infty} 3^k x^k = \sum_{k=0}^{\infty} 2 \cdot 3^k x^k.$$

Consequently, $a_k = 2 \cdot 3^k$.

Example 196. *Suppose that a valid codeword is an n-digit number in decimal notation containing an even number of 0s. Let an denote the number of valid codewords of length* n. *We showed that the sequence* $\{a_n\}$ *satisfies the recurrence relation*

$$a_n = 8a_{n-1} + 10^{n-1}$$

and the initial condition $a_1 = 9$. *Use generating functions to find an explicit formula for* a_n.

Solution: *To make our work with generating functions simpler, we extend this sequence by setting* $a_0 = 1$; *when we assign this value to* a_0 *and use the recurrence relation, we have* $a_1 = 8a_0 + 10^0 = 8 + 1 = 9$, *which is consistent with our original initial condition. (It also makes sense because there is one code word of length 0the empty string.)*

We multiply both sides of the recurrence relation by x^n *to obtain*

$$a_n x^n = 8a_{n-1}x^n + 10^{n-1}x^n.$$

Let $G(x) = \sum_{n=0}^{\infty} a_n x^n$ *be the generating function of the sequence* $a_0, a_1, a_2, \cdots$. *We sum both sides of the last equation starting with* $n = 1$, *to find that*

$$\begin{aligned} G(x) - 1 &= \sum_{n=1}^{\infty} a_n x^n = \sum_{n=1}^{\infty} (8a_{n-1}x^n + 10^{n-1}x^n) \\ &= 8\sum_{n=1}^{\infty} a_{n-1}x^n + \sum_{n=1}^{\infty} 10^{n-1}x^n \end{aligned}$$

$$\begin{aligned}
&= 8x\sum_{n=1}^{\infty} a_{n-1}x^{n-1} + x\sum_{n=1}^{\infty} 10^{n-1}x^{n-1}\\
&= 8x\sum_{n=0}^{\infty} a_n x^n + x\sum_{n=0}^{\infty} 10^n x^n\\
&= 8xG(x) + x/(1-10x).
\end{aligned}$$

where we have used previous results to evaluate the second summation. Therefore, we have

$$G(x) - 1 = 8xG(x) + x/(1-10x).$$

Solving for $G(x)$ shows that

$$G(x) = \frac{1-9x}{(1-8x)(1-10x)}.$$

Expanding the right-hand side of this equation into partial fractions (as is done in the integration of rational functions studied in calculus) gives

$$G(x) = \frac{1}{2}\left(\frac{1}{1-8x} + \frac{1}{1-10x}\right).$$

Using above Examples (once with a = 8 and once with a = 10) gives

$$\begin{aligned}
G(x) &= \frac{1}{2}\left(\sum_{n=0}^{\infty} 8^n x^n + \sum_{k=0}^{\infty} 10^n x^n\right)\\
&= \sum_{n=0}^{\infty} \frac{1}{2}(8^n + 10^n)x^n.
\end{aligned}$$

Consequently, we have shown that

$$a_n = \frac{1}{2}(8^n + 10^n).$$

3.3.4 Proving Identities via Generating Functions

Example 197. *Use generating functions to show that*

$$\sum_{k=0}^{n} C(n,k)^2 = C(2n,n)$$

whenever n is a positive integer.

Solution: *: First note that by the binomial theorem $C(2n,n)$ is the coefficient of x^n in $(1+x)^{2n}$. However, we also have*

$$\begin{aligned}(1+x)^{2n} &= [(1+x)^n]^2 \\ &= [C(n,0)+C(n,1)x+C(n,2)x^2+\cdots+C(n,n)x^n]^2.\end{aligned}$$

The coefficient of x^n in this expression is

$$C(n,0)C(n,n)+C(n,1)C(n,n-1)+C(n,2)C(n,n-2)+\cdots+C(n,n)C(n,0).$$

This equals $\sum_{k=0}^{n} C(n,k)^2$, because $C(n,n-k)=C(n,k)$. Because both $C(2n,n)$ and $\sum_{k=0}^{n} C(n,k)^2$ represent the coefficient of x^n in $(1+x)^{2n}$, they must be equal.

Let Us Sum Up

In this section, we discussed about

∗ *Formal power series.*

∗ *The extended binomial theorem.*

∗ *Counting problems and generating functions.*

∗ *Exponential generating functions.*

- $*$ *Proving identities through generating functions.*

Check your Progress

1. The generating function of the sequence $1, 1, 1, 1, \cdots$ *is*

(a) $1 + x$

(b) $1 - x$

(c) $\frac{1}{1+x}$

(d) $\frac{1}{1-x}$

2. The generating function of the sequence $\{\frac{1}{k!}\}$ *is*

(a) e^x

(b) $\log x$

(c) $k + 1$

(d) $\frac{1}{1+x}$

Summary

In this unit, we discussed about

- $*$ *Applications of recurrence relations.*
- $*$ *Dynamic programming.*
- $*$ *Algorithms and recurrence relations.*
- $*$ *Solving linear recurrence relations.*

∗ *Linear non-homogeneous recurrence relations.*

∗ *Generating functions.*

∗ *Solving recurrence relations using generating functions.*

Glossary

recurrence relation: *a formula expressing terms of a sequence, except for some initial terms, as a function of one or more previous terms of the sequence.*
initial conditions for a recurrence relation: *the values of the terms of a sequence satisfying the recurrence relation before this relation takes effect.*
dynamic programming: *an algorithmic paradigm that finds the solution to an optimization problem by recursively breaking down the problem into overlapping subproblems and combining their solutions with the help of a recurrence relation.*
Eratosthenes: *a procedure for finding the primes less than a specified positive integer.*
derangement: *a permutation of objects such that no object is in its original place.*

Self Assessment Questions

1. *Explain how the Fibonacci numbers are used to solve Fibonacci's problem about rabbits.*

2. *Explain how dynamic programming can be used to schedule talks in*

a lecture hall from a set of possible talks to maximize overall attendance.

3. *Define a linear homogeneous recurrence relation of degree k.*

4. *State the principle of inclusion-exclusion.*

5. *Explain how the inclusion-exclusion principle can be used to count the number of primes not exceeding the positive integer n.*

Exercises

1. *Find the solutions of the simultaneous system of recurrence relations*
   $a_n = a_{n-1} + b_{n-1}$
   $b_n = a_{n-1} - b_{n-1}$
   with $a_0 = 1$ and $b_0 = 2$.

2. *Find the solution of the recurrence relation $a_n = 3a_{n-1} - 3a_{n-2} + a_{n-3} + 1$ if $a_0 = 2$, $a_1 = 4$, and $a_2 = 8$.*

3. *Find the solution to the recurrence relation $f(n) = f(n/2) + n^2$ for $n = 2^k$ where k is a positive integer and $f(1) = 1$.*

4. *How many bit strings of length six do not contain four consecutive 1s?*

5. *What is the probability that a bit string of length six chosen at random contains at least four 1s?*

Answers to Check your Progress

Section 3.1: 1)b 2)b

Section 3.2: 1)d 2)a

Section 3.3: 1)d 2)a

Reference:

1. *Kenneth H. Rosen, "Discrete Mathematics and its Applications", 7th Edition, WCB/ McGraw Hill Publications, New Delhi, 2011.*

Suggested Readings:

1. *Edward A. Bender and S. Gill Williamson, "A Short Course in Discrete Mathematics", Dover Publications, 2006.*
2. *M.O. Albertson and J.P. Hutchinson, "Discrete Mathematics with Algorithms", John Wiley & Sons, 2008.*
3. *Rajendra Akerkar and Rupali Akarkar, "Discrete Mathematics", Pearson Education Pvt. Ltd, Singapore, 2004.*
4. *J. P. Trembley and R. Manohar, "Discrete Mathematical Structures", Tata McGraw Hill, New Delhi,1997.*
5. *Martin Aigner, "A Course in Enumeration", Springer-Verlag, Heidelberg, 2007.*
6. *J.H. Van Lint and R.M. Wilson, "A Course in Combinatorics", 2nd Edition, Cambridge University Press, Cambridge, 2001.*

Unit 4

Boolean Algebra

Objectives

1. *To learn Boolean expressions an Boolean functions.*
2. *To find Boolean expressions that represents a Boolean function.*
3. *To show that all Boolean functions can be represented by using only one operator.*
4. *To model the circuitry of electronic devices using Boolean Algebra.*
5. *To learn Karnaugh maps and the Quine-Mc Cluskey Method to minimize Boolean functions.*

4.1 Boolean Functions

4.1.1 Introduction

Boolean algebra provides the operations and the rules for working with the set $\{0, 1\}$. *Electronic and optical switches can be studied using this set and the rules of Boolean algebra. The three operations in Boolean algebra that we will use most are complementation, the Boolean sum, and* *the Boolean product. The complement of an element, denoted with a bar, is defined by* $\bar{0} = 1$ *and* $\bar{1} = 0$*. The Boolean sum, denoted by* $+$ *or by OR, has the following values:*

$1 + 1 = 1,$ $1 + 0 = 1,$ $0 + 1 = 1,$ $0 + 0 = 0.$

The Boolean product, denoted by or by AND, has the following values:

$1 \cdot 1 = 1,$ $1 \cdot 0 = 0,$ $0 \cdot 1 = 0,$ $0 \cdot 0 = 0.$

Example 198. *Find the value of* $1 \cdot 0 + \overline{(0 + 1)}$.

Solution: Using the definitions of complementation, the Boolean sum, and the Boolean product, it follows that

$$\begin{aligned} 1 \cdot 0 + \overline{(0+1)} &= 0 + \bar{1} \\ &= 0 + 0 \\ &= 0. \end{aligned}$$

The complement, Boolean sum, and Boolean product correspond to the logical operators, $\neg$, $\vee$ and $\wedge$, respectively, where 0 corresponds to F (false) and 1 corresponds to T (true). □

Example 199. *Translate* $1 \cdot 0 + \overline{(0 + 1)} = 0$*, the equality found in Example* 1*, into a logical equivalence.*

Solution: We obtain a logical equivalence when we translate each 1 into a T, each 0 into an F, each Boolean sum into a disjunction, each Boolean product into a conjunction, and each complementation into a negation. We obtain

$$(T \wedge F) \vee \neg (T \vee F) \equiv F.$$

□

Example 200. *Translate the logical equivalence $(T \wedge T) \vee \neg F \equiv T$ into an identity in Boolean algebra.*

Solution: We obtain an identity in Boolean algebra when we translate each T into a 1, each F into a 0, each disjunction into a Boolean sum, each conjunction into a Boolean product, and each negation into a complementation. We obtain
$(1 \cdot 1) + \bar{0} = 1.$ □

4.1.2 Boolean Expressions and Boolean Functions

Let $B = \{0, 1\}$. Then $B^n = \{(x_1, x_2, \cdots, x_n) | x_i \in B$ for $1 \leq i \leq n\}$ is the set of all possible n-tuples of 0s and 1s. The variable x is called a **Boolean variable** *if it assumes values only from B, that is, if its only possible values are 0 and 1. A function from B^n to B is called a* **Boolean function of degree** *n.*

Example 201. *The function $F(x, y) = \overline{xy}$ from the set of ordered pairs of Boolean variables to the set $\{0, 1\}$ is a Boolean function of degree 2 with $F(1, 1) = 0$, $F(1, 0) = 1$, $F(0, 1) = 0$, and $F(0, 0) = 0$. We display these values of F in Table 4.1.*

Boolean functions can be represented using expressions made up from variables and Boolean operations. The **Boolean expressions** *in the variables $x_1, x_2, \cdots, x_n$ are defined recursively as $0, 1, x_1, x_2, \cdots, x_n$ are Boolean expressions; if E_1 and E_2 are Boolean expressions, then $\bar{E}_1$, $(E_1 E_2)$, and $(E_1 + E_2)$ are Boolean expressions.*

Each Boolean expression represents a Boolean function. The values of this function are obtained by substituting 0 and 1 for the variables in the expression.

Table 4.1: For Example 201

x	y	F(x,y)
1	1	0
1	0	1
0	1	0
0	0	0

Example 202. *Find the values of the Boolean function represented by* $F(x, y, z) = xy + \bar{z}$.

Solution: The values of this function are displayed in Table 4.2. □

Table 4.2: For Example 202

x	y	z	xy	$\bar{z}$	$F(x, y, z) = xy + \bar{z}$
1	1	1	1	0	1
1	1	0	1	1	1
1	0	1	0	0	0
1	0	0	0	1	1
0	1	1	0	0	0
0	1	0	0	1	1
0	0	1	0	0	0
0	0	0	0	1	1

Example 203. *The function* $F(x, y, z) = xy + \bar{z}$ *from* B^3 *to* B *from Example 202 can be represented by distinguishing the vertices that correspond to the five* 3*-tuples* $(1, 1, 1)$, $(1, 1, 0)$, $(1, 0, 0)$, $(0, 1, 0)$, *and* $(0, 0, 0)$, *where* $F(x, y, z) = 1$, *as shown in Figure 4.1. These vertices are displayed using solid black circles.*

Boolean functions F *and* G *of* n *variables are equal if and only if* $F(b_1, b_2, \cdots, b_n) = G(b_1, b_2, \cdots, b_n)$ *whenever* $b_1, b_2, \cdots, b_n$ *belong to* B. *Two different Boolean expressions that represent the same function are called equivalent. For*

Table 4.3: The 16 Boolean Functions of Degree Two.

x	y	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}
1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
1	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
0	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

instance, the Boolean expressions xy, $xy + 0$, and $xy \cdot 1$ are equivalent. The complement of the Boolean function F is the function $\bar{F}$, where $\bar{F}(x_1, \cdots, x_n) = F(x_1, \cdots, x_n)$. Let F and G be Boolean functions of degree n. The Boolean sum $F + G$ and the Boolean product FG are defined by

$$(F + G)(x_1, \cdots, x_n) = F(x_1, \cdots, x_n) + G(x_1, \cdots, x_n),$$

$$(FG)(x_1, \cdots, x_n) = F(x_1, \cdots, x_n)G(x_1, \cdots, x_n),$$

A Boolean function of degree two is a function from a set with four elements, namely, pairs of elements from $B = \{0, 1\}$, to B, a set with two elements. Hence, there are 16 different Boolean functions of degree two. In Table 4.3 we display the values of the 16 different Boolean functions of degree two, labeled F_1, $F_2, \cdots, F_{16}$.

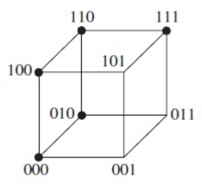


Figure 4.1

Example 204. *How many different Boolean functions of degree n are there?*

Solution: From the product rule for counting, it follows that there are 2^n different n-tuples of 0s and 1s. Because a Boolean function is an assignment of 0 or 1 to each of these 2^n different n-tuples, the product rule shows that there are 2^{2^n} different Boolean functions of degree n. □

4.1.3 Identities of Boolean Algebra

There are many identities in Boolean algebra. The most important of these are displayed in Table 4.4. These identities are particularly useful in simplifying the design of circuits.

Table 4.4: Boolean Identities

Identity	**Name**
$\bar{\bar{x}} = x$	Law of the double complement
$x + x = x$ $x \cdot x = x$	Idempotent laws
$x + 0 = x$ $x \cdot 1 = x$	Identity laws
$x + 1 = 1$ $x \cdot 0 = 0$	Domination laws
$x + y = y + x$ $xy = yx$	Commutative laws
$x + (y + z) = (x + y) + z$ $x(yz) = (xy)z$	Associative laws
$x + yz = (x + y)(x + z)$ $x(y + z) = xy + xz$	Distributive laws
$\overline{(xy)} = \bar{x} + \bar{y}$ $\overline{(x + y)} = \bar{x}\bar{y}$	De Morgan's laws
$x + xy = x$ $x(x + y) = x$	Absorption laws
$x + \bar{x} = 1$	Unit property
$x\bar{x} = 0$	Zero property

Example 205. *Show that the distributive law $x(y + z) = xy + xz$ is valid.*

Solution: The verification of this identity is shown in Table 4.5. The identity holds because the last two columns of the table agree. □

Example 206. *Translate the distributive law $x + yz = (x + y)(x + z)$ in Table 4.4 into a logical equivalence.*

Solution: To translate a Boolean identity into a logical equivalence, we change each Boolean variable into a propositional variable. Here we will change the Boolean variables x, y, and z into the propositional variables p, q, and r. Next, we change each Boolean sum into a disjunction and each Boolean product into a conjunction. (Note that 0 and 1 do not appear in this identity and complementation also does not appear.) This transforms the Boolean identity into the logical equivalence
$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$. □

Table 4.5: Verifying One of the Distribution Laws

x	y	z	y + z	xy	xz	x(y + z)	xy + xz
1	1	1	1	1	1	1	1
1	1	0	1	1	0	1	1
1	0	1	1	0	1	1	1
1	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0

Example 207. *Prove the absorption law x(x + y) = x using the other identities of Boolean algebra.*

Solution: We display steps used to derive this identity and the law used in each step:

$x(x+y)$	$= (x+0)(x+y)$	Identity law for the Boolean sum
	$= x + 0 \cdot y$	Distributive law of the Boolean sum over the Boolean product
	$= x + y \cdot 0$	Commutative law for the Boolean product
	$= x + 0$	Domination law for the Boolean product
	$= x$	Identity law for the Boolean sum.

□

4.1.4 Duality

The dual of a Boolean expression is obtained by interchanging Boolean sums and Boolean products and interchanging 0*s and* 1*s.*

Example 208. *Find the duals of* $x(y+0)$ *and* $\bar{x} \cdot 1 + (\bar{y} + z)$.

Solution: Interchanging $\cdot$ signs and $+$ signs and interchanging 0s and 1s in these expressions produces their duals. The duals are $x + (y \cdot 1)$ and $(x+0)(yz)$, respectively. □

The dual of a Boolean function F *represented by a Boolean expression is the function represented by the dual of this expression. This dual function, denoted by* F^d *, does not depend on the particular Boolean expression used to represent* F*. An identity between functions represented by Boolean expressions remains valid when the duals of both sides of the identity are taken. This result, called the duality principle, is useful for obtaining new identities.*

Example 209. *Construct an identity from the absorption law* $x(x+y) = x$ *by taking duals.*

Solution: Taking the duals of both sides of this identity produces the identity $x + xy = x$, which is also called an absorption law. □

4.1.5 The Abstract Definition of a Boolean Algebra

In this section we have focused on Boolean functions and expressions. However, the results we have established can be translated into results about propositions or results about sets.

Definition 210. *A Boolean algebra is a set B with two binary operations $\vee$ and $\wedge$, elements 0 and 1, and a unary operation $^-$ such that these properties hold for all x, y, and z in B:*

$$\left.\begin{aligned} x \vee 0 &= x \\ x \wedge 1 &= x \end{aligned}\right\} \quad \textit{Identity laws}$$

$$\left.\begin{aligned} x \vee \bar{x} &= 1 \\ x \wedge \bar{x} &= 0 \end{aligned}\right\} \quad \textit{Complement laws}$$

$$\left.\begin{aligned} (x \vee y) \vee z &= x \vee (y \vee z) \\ (x \wedge y) \wedge z &= x \wedge (y \wedge z) \end{aligned}\right\} \quad \textit{Associative laws}$$

$$\left.\begin{aligned} x \vee y &= y \vee x \\ x \wedge y &= y \wedge x \end{aligned}\right\} \quad \textit{Commutative laws}$$

$$\left.\begin{aligned} x \vee (y \wedge z) &= (x \vee y) \wedge (x \vee z) \\ x \wedge (y \vee z) &= (x \wedge y) \vee (x \wedge z) \end{aligned}\right\} \quad \textit{Distributive laws}$$

Boolean algebras may also be defined using the notion of a lattice. A lattice L is a partially ordered set in which every pair of elements x, y has a least upper bound, denoted by $lub(x, y)$ and a greatest lower bound denoted by $glb(x, y)$. Given two elements x and y of L, we can define two operations $\vee$ and $\wedge$ on pairs of elements of L by $x \vee y = lub(x, y)$ and $x \wedge y = glb(x, y)$.

For a lattice L to be a Boolean algebra as specified in Definition 210, *it must have two properties. First, it must be complemented. For a lattice to be complemented it must have a least element* 0 *and a greatest element* 1 *and for every element* x *of the lattice there must exist an element* x *such that* $x \vee x = 1$ *and* $x \wedge x = 0$. *Second, it must be distributive. This means that for every* $x, y,$ *and* z *in L,* $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$ *and* $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$.

Let Us Sum Up

In this section, we discussed about

- *Boolean expressions and Boolean functions.*
- *Identities of Boolean Algebra.*
- *The abstract definition of Boolean Algebra.*

Check your Progress

1. The value of $1.0 + (0 \bar{+} 1)$

(a) 1

(b) 0

(c) 2

(d) 3

2. The property $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$ *called*

(a) Commutative law

(b) Identity law

(c) Associative law

(d) Distributive law

4.2 Representing Boolean Functions

4.2.1 Sum-of-Products Expansions

Example 211. *Find Boolean expressions that represent the functions $F(x, y, z)$ and $G(x, y, z)$, which are given in Table 4.6.*

Solution: An expression that has the value 1 when $x = z = 1$ and $y = 0$, and the value 0 otherwise, is needed to represent F. Such an expression can be formed by taking the Boolean product of $x, \bar{y}$, and z. This product, $x\bar{y}z$, has the value 1 if and only if $x = \bar{y} = z = 1$, which holds if and only if $x = z = 1$ and $y = 0$.

To represent G, we need an expression that equals 1 when $x = y = 1$ and $z = 0$, or $x = z = 0$ and $y = 1$. We can form an expression with these values by taking the Boolean sum of two different Boolean products. The Boolean product $xy\bar{z}$ has the value 1 if and only if $x = y = 1$ and $z = 0$. Similarly, the product $\bar{x}y\bar{z}$ has the value 1 if and only if $x = z = 0$ and $y = 1$. The Boolean sum of these two products, $xy\bar{z} + \bar{x}y\bar{z}$, represents G, because it has the value 1 if and only if $x = y = 1$ and $z = 0$, or $x = z = 0$ and $y = 1$. □

Definition 212. *A literal is a Boolean variable or its complement. A minterm of the Boolean variables $x_1, x_2, \cdots, x_n$ is a Boolean product $y_1 y_2 \cdots y_n$, where $y_i = x_i$ or $y_i = \bar{x}_i$. Hence, a minterm is a product of n literals, with one literal for each variable*

Table 4.6: For Example 211

x	y	z	F	G
1	1	1	0	0
1	1	0	0	1
1	0	1	1	0
1	0	0	0	0
0	1	1	0	0
0	1	0	0	1
0	0	1	0	0
0	0	0	0	0

Example 213. *Find a minterm that equals* 1 *if* $x_1 = x_3 = 0$ *and* $x_2 = x_4 = x_5 = 1$*, and equals* 0 *otherwise.*

Solution: The minterm $\bar{x_1}x_2\bar{x_3}x_4x_5$ has the correct set of values. □

The sum of minterms that represents the function is called the **sum-of-products expansion** *or the* **disjunctive normal form** *of the Boolean function.*

Example 214. *Find the sum-of-products expansion for the function* $F(x, y, z) = (x + y)\bar{z}$.

Solution: We will find the sum-of-products expansion of $F(x, y, z)$ in two ways. First, we will use Boolean identities to expand the product and simplify. We find that

$$\begin{aligned}
F(x,y,z) &= (x+y)\bar{z} \\
&= x\bar{z} + y\bar{z} \quad \text{(Distributive lam)} \\
&= x1\bar{z} + 1y\bar{z} \quad \text{(Identity law)} \\
&= x(y+\bar{y})\bar{z} + (x+\bar{x})y\bar{z} \quad \text{(Unit property)} \\
&= xy\bar{z} + x\bar{y}z + xy\bar{z} + \bar{x}y\bar{z} \quad \text{(Distributive law)} \\
&= xy\bar{z} + x\bar{y}\bar{z} + \bar{x}y\bar{z} \quad \text{(Idempotent law)}
\end{aligned}$$

Second, we can construct the sum-of-products expansion by determining the values of F for all possible values of the variables x, y, and z. These values are found in Table 4.7. The sum-of products expansion of F is the Boolean sum of three minterms corresponding to the three rows of this table that give the value 1 for the function. This gives

$$F(x, y, z) = xy\bar{z} + x\bar{y}\bar{z} + \bar{x}y\bar{z}.$$

It is also possible to find a Boolean expression that represents a Boolean

Table 4.7: For Example 214

x	y	z	x+yx	$\bar{z}$	$(x + y)\bar{z}$
1	1	1	1	0	0
1	1	0	1	1	1
1	0	1	1	0	0
1	0	0	1	1	1
0	1	1	1	0	0
0	1	0	1	1	1
0	0	1	0	0	0
0	0	0	0	1	0

function by taking a Boolean product of Boolean sums. The resulting expansion is called the **conjunctive normal form or product-of-sums expansion** of the function. These expansions can be found from sum-of-products expansions by taking duals.

□

4.2.2 Functional Completeness

Every Boolean function can be expressed as a Boolean sum of minterms. Each minterm is the Boolean product of Boolean variables or their complements. This shows that every Boolean function can be represented using the Boolean operators $\cdot$, $+$, *and* $-$. *Because every Boolean function* *can be represented using these operators we say that the set* $\{\cdot, +, -\}$ *is* **functionally complete**. *Can we find a smaller set of functionally complete operators? We can do so if one of the three operators of this set can be expressed in terms of the other two. This can be done using one of De Morgans laws. We can eliminate all Boolean sums using the identity*

$$x + y = \overline{\bar{x}\bar{y}}$$

which is obtained by taking complements of both sides in the second De Morgan law, and then applying the double complementation law. This means that the set $\{\cdot, -\}$ *is functionally complete. Similarly, we could eliminate all Boolean products using the identity*

$$xy = \overline{\bar{x} + \bar{y}}$$

which is obtained by taking complements of both sides in the first De Morgan law, and then applying the double complementation law. Consequently $\{+, -\}$ *is functionally complete. Note that the set* $\{+, \cdot\}$ *is not functionally complete, because it is impossible to express the Boolean function* $F(x) = x$ *using these operators.*

We have found sets containing two operators that are functionally complete. Can we find a smaller set of functionally complete operators, namely, a set containing just one operator? Such sets exist. Define two operators, the — or **NAND** *operator, defined by* $1|1 = 0$ *and* $1|0 = 0|1 = 0|0 = 1$*; and the* $\downarrow$ *or* **NOR** *operator, defined by* $1 \downarrow 1 = 1 \downarrow 0 = 0 \downarrow 1 = 0$ *and* $0 \downarrow 0 = 1$. *Both of the sets* $\{|\}$ *and* $\{\downarrow\}$ *are functionally complete. To see that* $\{|\}$ *is functionally complete, because* $\{\cdot, -\}$ *is functionally complete, all that we have to do is show that both of the operators* $\cdot$ *and* $-$ *can be expressed using just the* $|$ *operator. This can* *be done as*

$$\begin{aligned} x &= x|x, \\ xy &= (x|y)|(x|y). \end{aligned}$$

Let Us Sum Up

In this section, we discussed about

∗ *Sum-of-products expansion.*

∗ *Functional completeness.*

∗ *Simple problems in product -of-sums.*

Check your Progress

1. The minterm that equals 1 if $x_1 = x_3 = 0$ *and* $x_2 = x_4 = x_5 = 1$ *and equals* 0 *otherwise is*

(a) $x_1x_2x_3x_4x_5$

(b) $x_1\bar{x_2}x_3\bar{x_4}\bar{x_5}$

(c) $\bar{x_1}x_2x_3x_4x_5$

(d) $x_1x_2x_3x_4\bar{x_5}$

2. $xy =$ ________

(a) $\bar{x} + \bar{y}$

(b) $\bar{x}y$

(c) $\bar{x} \bar{+} \bar{y}$

(d) $x \bar{+} y$

4.3 Logic Gates

4.3.1 Introduction

Boolean algebra is used to model the circuitry of electronic devices. Each input and each output of such a device can be thought of as a member of the set $\{0, 1\}$*. A computer, or other electronic device, is made up of a number of circuits. Each circuit can be designed using the rules of Boolean algebra. The basic elements of circuits are called* **gates***. Each type of gate implements a Boolean operation. In this section we define several types of gates. Using these gates, we will apply the rules of Boolean algebra to design circuits that perform a variety of tasks. The circuits that we will study in this chapter give output that depends only on the input, and not on the current state of the circuit. In other words, these circuits have no memory capabilities. Such circuits are called* **combinational circuits** *or* **gating networks***. We will construct combinational circuits using three*

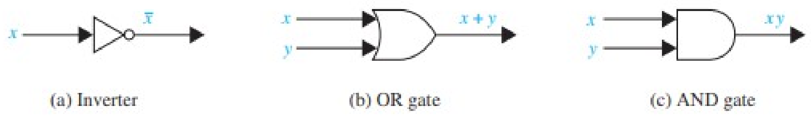


Figure 4.2: Basic types of gates.

types of elements. The first is an **inverter***, which accepts the value of one Boolean variable as input and produces the complement of this value as its output. The symbol used for an inverter is shown in Figure 4.2(a). The input to the inverter is shown on the left side entering the element, and the output is shown on the right side leaving the element.*

The next type of element we will use is the **OR gate***. The inputs to this gate are the values of two or more Boolean variables. The output is the Boolean sum of their values. The symbol used for an OR gate is shown* *in Figure 4.2(b). The inputs to the OR gate are shown on the left side entering the element, and the output is shown on the right side leaving the element.*

The third type of element we will use is the **AND gate***. The inputs to this gate are the values of two or more Boolean variables. The output is the Boolean product of their values. The symbol used for an AND gate is shown in Figure 4.2(c). The inputs to the AND gate are shown on the left side entering the element, and the output is shown on the right side leaving the element.*

We will permit multiple inputs to AND and OR gates. The inputs to each of these gates are shown on the left side entering the element, and the output is shown on the right side. Examples of AND and OR gates with n inputs are shown in Figure 4.3.

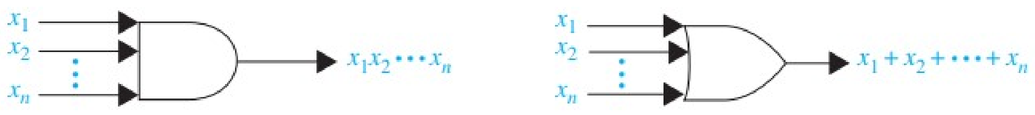


Figure 4.3: Gates with n Inputs.

4.3.2 Combinations of Gates

Example 215. *Construct circuits that produce the following outputs: (a) $(x + y)\bar{x}$, (b) $\bar{x}(y \mp z)$, and (c) $(x + y + z)(\bar{x}\bar{y}\bar{z})$.*

Solution: Circuits that produce these outputs are shown in Figure 4.4. □

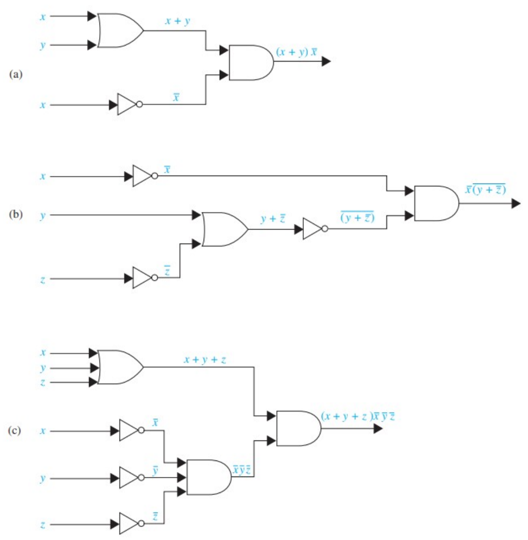


Figure 4.4: Circuits that Produce the Outputs Specified in Example 215.

4.3.3 Examples of Circuits

Example 216. *A committee of three individuals decides issues for an organization. Each individual votes either yes or no for each proposal that arises. A proposal is passed if it receives at least two yes votes. Design a circuit that determines whether a proposal passes.*

Solution: Let $x = 1$ if the first individual votes yes, and $x = 0$ if this individual votes no; let $y = 1$ if the second individual votes yes, and $y = 0$ if this individual votes no; let $z = 1$ if the third individual votes yes, and $z = 0$ if this individual votes no. Then a circuit must be designed that produces the output 1 from the inputs x, y, and z when two or more of x, y, and z are 1. One representation of the Boolean function that has these output values is $xy + xz + yz$. The circuit that implements this function is shown in Figure 4.5. □

Example 217. *Sometimes light fixtures are controlled by more than one switch. Circuits need to be designed so that flipping any one of the switches for the fixture turns the light on when it is off and turns the light off when*

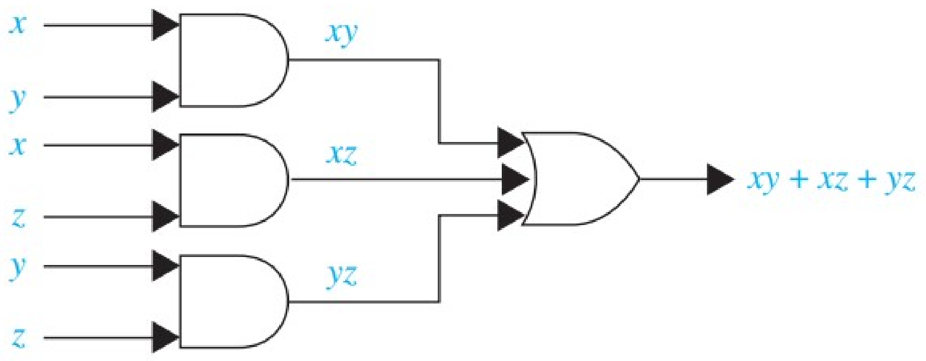


Figure 4.5: A Circuit for Majority Voting.

it is on. Design circuits that accomplish this when there are two switches and when there are three switches.

Solution: We will begin by designing the circuit that controls the light fixture when two different switches are used. Let $x = 1$ when the first switch is closed and $x = 0$ when it is open, and let $y = 1$ when the second switch is closed and $y = 0$ when it is open. Let $F(x, y) = 1$ when the light is on and $F(x, y) = 0$ when it is off. We can arbitrarily decide that the light will be on when both switches are closed, so that $F(1, 1) = 1$. This determines all the other values of F. When one of the two switches is opened, the light goes off, so $F(1, 0) = F(0, 1) = 0$. When the other switch is also opened, the light goes on, so $F(0, 0) = 1$. Table 4.8 displays these values. Note that $F(x, y) = xy + \bar{x}\bar{y}$. This function is implemented by the circuit shown in Figure 4.6.

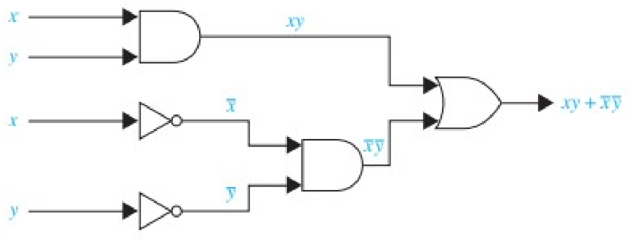


Figure 4.6: A Circuit for a Light Controlled by Two Switches.

Table 4.8: For Example 217

x	y	$F(x, y)$
1	1	1
1	0	0
0	1	0
0	0	1

We will now design a circuit for three switches. Let x, y, and z be the Boolean variables that indicate whether each of the three switches is closed. We let $x = 1$ when the first switch is closed, and $x = 0$ when it is open; $y = 1$ when the second switch is closed, and $y = 0$ when it is open; and $z = 1$ when the third switch is closed, and $z = 0$ when it is open. Let $F(x, y, z) = 1$ when the light is on and $F(x, y, z) = 0$ when the light is off. We can arbitrarily specify that the light be on when all three switches are closed, so that $F(1, 1, 1) = 1$. This determines all other values of F. When one switch is opened, the light goes off, so $F(1, 1, 0) = F(1, 0, 1) = F(0, 1, 1) = 0$. When a second switch is opened, the light goes on, so $F(1, 0, 0) = F(0, 1, 0) = F(0, 0, 1) = 1$. Finally, when the third switch is opened, the light goes off again, so $F(0, 0, 0) = 0$. Table 4.9 shows the values of this function.

The function F can be represented by its sum-of-products expansion as $F(x, y, z) = xyz + x\bar{y}\bar{z} + \bar{x}y\bar{z} + \bar{x}\bar{y}z$. The circuit shown in Figure 4.7 implements this function.

□

4.3.4 Minimization of Circuits

The goal is to produce Boolean sums of Boolean products that represent a Boolean function with the fewest products of literals such that these products contain the fewest literals possible among all sums of products

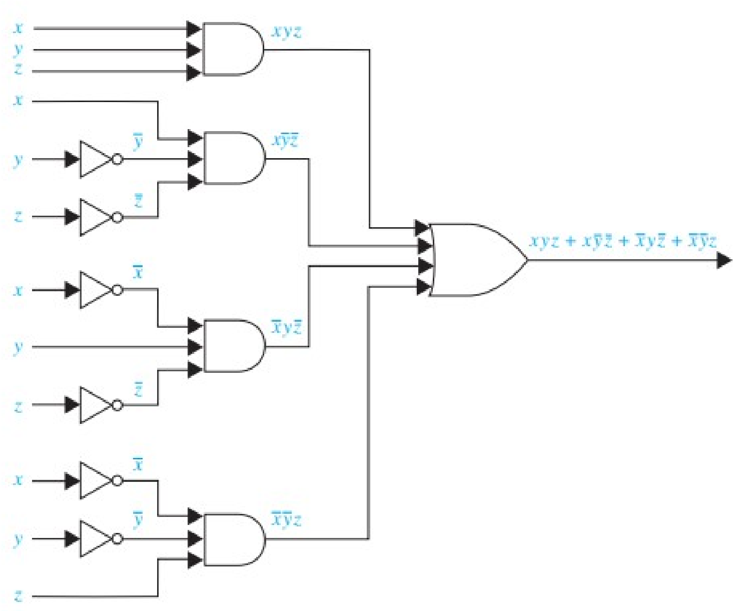


Figure 4.7: A Circuit for a Light Controlled by Two Switches.

Table 4.9: A Circuit for a Fixture Controlled by Three Switches.

x	y	z	$F(x, y, z)$
1	1	1	1
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	1
0	0	1	1
0	0	0	0

that represent a Boolean function. Finding such a sum of products is called **minimization of the Boolean function***. Minimizing a Boolean function makes it possible to construct a circuit for this function that uses the fewest gates and fewest inputs to the AND gates and OR gates in the circuit, among all circuits for the Boolean expression we are minimizing.*

4.3.5 Karnaugh Maps

To reduce the number of terms in a Boolean expression representing a circuit, it is necessary to find terms to combine. There is a graphical method, called a **Karnaugh map** *or* **K-map**, *for finding terms to combine for Boolean functions involving a relatively small number of variables.*

There are four possible minterms in the sum-of-products expansion of a Boolean function in the two variables x and y. A K-map for a Boolean function in these two variables consists of four cells, where a 1 is placed in the cell representing a minterm if this minterm is present in the expansion. Cells are said to be **adjacent** *if the minterms that they represent differ in exactly one literal. For instance, the cell representing xy is adjacent to the cells representing xy and $\bar{x}\bar{y}$. The four cells and the terms that they represent are shown in Figure 4.8.*

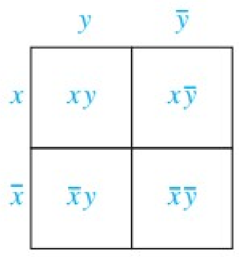


Figure 4.8: K-maps in Two Variables.

Example 218. *Find the K-maps for (a) $xy + \bar{x}y$, (b) $x\bar{y} + \bar{x}y$, and (c) $x\bar{y} + \bar{x}y + \bar{x}\bar{y}$.*

Solution: We include a 1 in a cell when the minterm represented by this cell is present in the sum-of-products expansion. The three K-maps are shown in Figure 4.9. □

Example 219. *Simplify the sum-of-products expansions given in Example 218.*

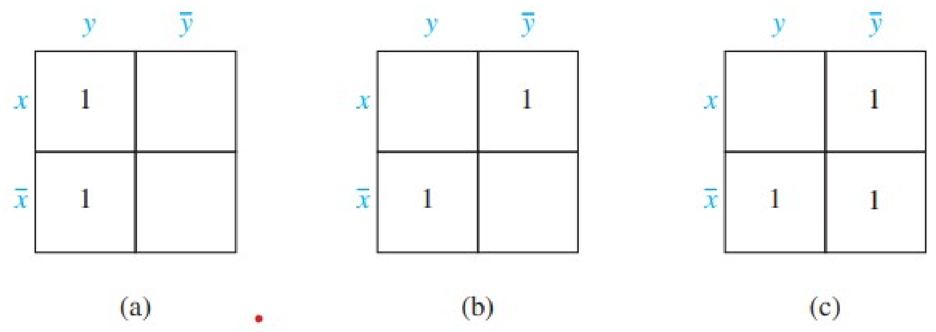


Figure 4.9: K-maps for the Sum-of-Products Expansions in Example 218.

Solution: The grouping of minterms is shown in Figure 4.10 using the K-maps for these expansions. Minimal expansions for these sums-of-products are (a) y, (b) $x\bar{y} + \bar{x}y$, and (c) $\bar{x} + \bar{y}$.

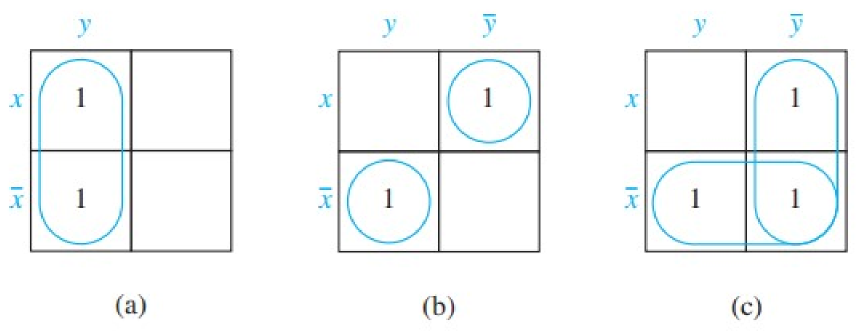


Figure 4.10: Simplifying the Sum-of-Products Expansions from Example 219.

To simplify a sum-of-products expansion in three variables, we use the K-map to identify blocks of minterms that can be combined. Blocks of two adjacent cells represent pairs of minterms that can be combined into a product of two literals; 2×2 and 4×1 blocks of cells represent minterms that can be combined into a single literal; and the block of all eight cells represents a product of no literals, namely, the function 1. In Figure 4.11, 1×2, 2×1, 2×2, 4×1, and 4×2 blocks and the products they represent are shown.

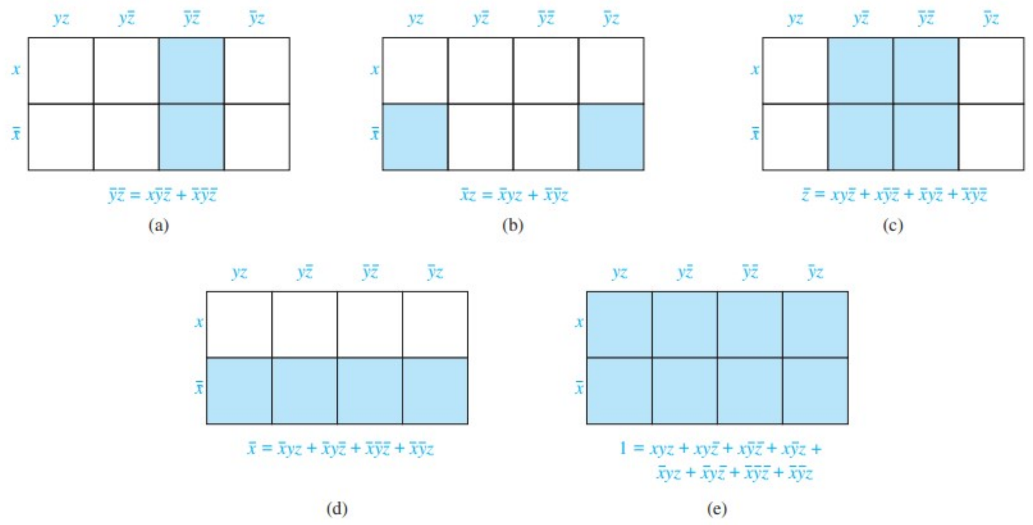

yz $y\bar{z}$ $\bar{y}\bar{z}$ $\bar{y}z$
x
$\bar{x}$
$\bar{y}\bar{z} = x\bar{y}\bar{z} + \bar{x}\bar{y}\bar{z}$
(a)
$\bar{x}z = \bar{x}yz + \bar{x}\bar{y}z$
(b)
$\bar{z} = xy\bar{z} + x\bar{y}\bar{z} + \bar{x}y\bar{z} + \bar{x}\bar{y}\bar{z}$
(c)
$\bar{x} = \bar{x}yz + \bar{x}y\bar{z} + \bar{x}\bar{y}\bar{z} + \bar{x}\bar{y}z$
(d)
$1 = xyz + xy\bar{z} + x\bar{y}\bar{z} + x\bar{y}z + \bar{x}yz + \bar{x}y\bar{z} + \bar{x}\bar{y}\bar{z} + \bar{x}\bar{y}z$
(e)

Figure 4.11: Blocks in K-maps in Three Variables.

The product of literals corresponding to a block of all 1s in the K-map is called an **implicant** of the function being minimized. It is called a **prime implicant** if this block of 1s is not contained in a larger block of 1s representing the product of fewer literals than in this product.

The goal is to identify the largest possible blocks in the map and cover all the 1s in the map with the least number of blocks, using the largest blocks first. The largest possible blocks are always chosen, but we must always choose a block if it is the only block of 1s covering a 1 in the K-map. Such a block represents an **essential prime implicant**. By covering all the 1s in the map with blocks corresponding to prime implicants we can express the sum of products as a sum of prime implicants. Note that there may be more than one way to cover all the 1s using the least number of blocks.

□

Example 220 illustrates how K-maps in three variables are used.

Example 220. *Use K-maps to minimize these sum-of-products expan-*

sions.

(a) $xy\bar{z} + x\bar{y}\bar{z} + \bar{x}yz + \bar{x}\bar{y}\bar{z}$.

(b) $x\bar{y}z + x\bar{y}\bar{z} + \bar{x}\bar{y}z + \bar{x}\bar{y}\bar{z}$

(c) $xyx + xy\bar{z} + x\bar{y}z + x\bar{y}\bar{z} + \bar{x}yz + \bar{x}\bar{y}z + \bar{x}\bar{y}\bar{z}$

(d) $xy\bar{z} + x\bar{y}\bar{z} + \bar{x}\bar{y}z + \bar{x}\bar{y}\bar{z}$

Solution: The K-maps for these sum-of-products expansions are shown in Figure 4.12. The grouping of blocks shows that minimal expansions into Boolean sums of Boolean products are (a) $x\bar{z} + \bar{y}\bar{z} + \bar{x}yz$, (b) $\bar{y} + \bar{x}z$, (c) $x + \bar{y} + z$, and (d) $x\bar{z} + \bar{x}\bar{y}$. In part (d) note that the prime implicants $x\bar{z}$ and $\bar{x}\bar{y}$ are essential prime implicants, but the prime implicant $\bar{y}\bar{z}$ is a prime implicant that is not essential, because the cells it covers are covered by the other two prime implicants.

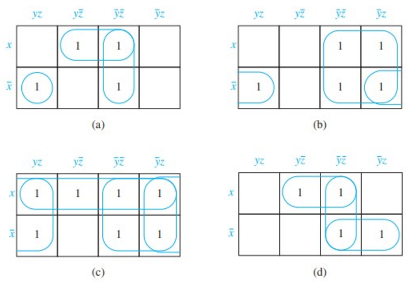


Figure 4.12: Using K-maps in Three Variables.

A K-map in four variables is a square that is divided into 16 cells. The cells represent the 16 possible minterms in four variables. One of the ways to form a K-map in four variables is shown in Figure 4.13.

□

Example 221. *Use K-maps to simplify these sum-of-products expansions.*

(a) $wxyz + wxy\bar{z} + wx\bar{y}\bar{z} + w\bar{x}yz + w\bar{x}\bar{y}z + w\bar{x}\bar{y}\bar{z} + \bar{w}x\bar{y}z + \bar{w}\bar{x}yz + \bar{w}\bar{x}y\bar{z}$

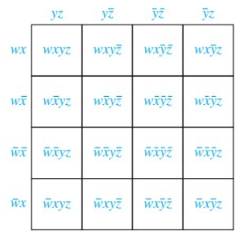

yz
$y\bar{z}$
$\bar{y}\bar{z}$
$\bar{y}z$
wx
$w\bar{x}$
$\bar{w}\bar{x}$
$\bar{w}x$
$wxyz$ $wxy\bar{z}$ $wx\bar{y}\bar{z}$ $wx\bar{y}z$
$w\bar{x}yz$ $w\bar{x}y\bar{z}$ $w\bar{x}\bar{y}\bar{z}$ $w\bar{x}\bar{y}z$
$\bar{w}\bar{x}yz$ $\bar{w}\bar{x}y\bar{z}$ $\bar{w}\bar{x}\bar{y}\bar{z}$ $\bar{w}\bar{x}\bar{y}z$
$\bar{w}xyz$ $\bar{w}xy\bar{z}$ $\bar{w}x\bar{y}\bar{z}$ $\bar{w}x\bar{y}z$

Figure 4.13: K-maps in Four Variables.

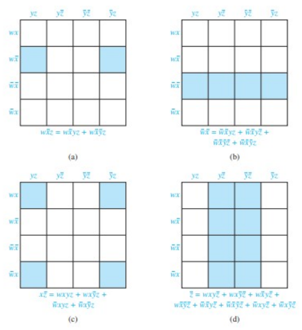


Figure 4.14: Blocks in K-maps in Four Variables.

(b) $wx\bar{y}\bar{z} + w\bar{x}yz + w\bar{x}y\bar{z} + w\bar{x}\bar{y}\bar{z} + \bar{w}x\bar{y}\bar{z} + \bar{w}\bar{x}y\bar{z} + \bar{w}\bar{x}\bar{y}\bar{z}$

(c) $wxy\bar{z} + wx\bar{y}\bar{z} + w\bar{x}yz + w\bar{x}y\bar{z} + w\bar{x}\bar{y}\bar{z} + \bar{w}xyz + \bar{w}xy\bar{z} + \bar{w}x\bar{y}\bar{z} + \bar{w}x\bar{y}z + \bar{w}\bar{x}y\bar{z} + \bar{w}\bar{x}\bar{y}\bar{z}$

Solution: The K-maps for these expansions are shown in Figure 4.15. Using the blocks shown leads to the sum of products (a) $wyz + wx\bar{z} + w\bar{x}\bar{y} + \bar{w}\bar{x}y + \bar{w}x\bar{y}$z, (b) $\bar{y}\bar{z} + w\bar{x}y + \bar{x}\bar{z}$, and (c) $\bar{z} + \bar{w}x + w\bar{x}y$. The reader should determine whether there are other choices of blocks in each part that lead to different sums of products representing these Boolean functions.

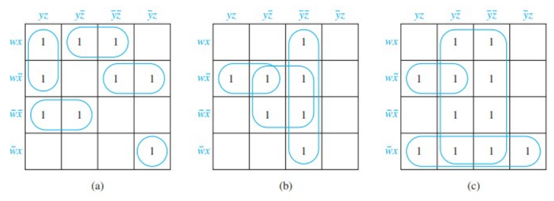


Figure 4.15: Using K-maps in Four Variables.

□

4.3.6 The Quine-Mc Cluskey Method

We have seen that K-maps can be used to produce minimal expansions of Boolean functions as Boolean sums of Boolean products. However, K-maps are awkward to use when there are more than four variables. Furthermore, the use of K-maps relies on visual inspection to identify terms to group. For these reasons there is a need for a procedure for simplifying sum-of-products expansions that can be mechanized. The Quine-McCluskey method is such a procedure. It can be used for Boolean functions in any number of variables. It was developed in the 1950s by W. V. Quine and E. J. McCluskey, Jr. Basically, the Quine-McCluskey method consists of two parts. The first part finds those terms that are candidates for inclusion in a minimal expansion as a Boolean sum of Boolean products. The second part determines which of these terms to actually use. We will use Example 222 to illustrate how, by successively combining implicants into implicants with one fewer literal, this procedure works.

Example 222. *We will show how the Quine-McCluskey method can be used to find a minimal expansion equivalent to*

$$xyz + x\bar{y}z + \bar{x}yz + \bar{x}\bar{y}z + \bar{x}\bar{y}\bar{z}.$$

We will represent the minterms in this expansion by bit strings. The first bit will be 1 if x occurs and 0 if $\bar{x}$ occurs. The second bit will be 1 if y occurs and 0 if $\bar{y}$ occurs. The third bit will be 1 if z occurs and 0 if $\bar{z}$ occurs. We then group these terms according to the number of 1s in the corresponding bit strings. This information is shown in Table 4.10.

Minterms that can be combined are those that differ in exactly one

Table 4.10: Minterms and the corresponding Bit String for Example 222

Minterm	Bit String	Number of 1s
xyz	111	3
$x\bar{y}z$	101	2
$\bar{x}yz$	011	2
$\bar{x}\bar{y}z$	001	1
$\bar{x}\bar{y}\bar{z}$	000	0

literal. Hence, two terms that can be combined differ by exactly one in the number of 1s in the bit strings that represent them. When two minterms are combined into a product, this product contains two literals. A product in two literals is represented using a dash to denote the variable that does not occur. For instance, the minterms $x\bar{y}z$ and $\bar{x}\bar{y}z$, represented by bit strings 101 and 001, can be combined into yz, represented by the string -01. All pairs of minterms that can be combined and the product formed from these combinations are shown in Table 4.11.

Table 4.11: Procedure for Simplification for Example 222

Next, all pairs of products of two literals that can be combined are combined into one literal. Two such products can be combined if they contain literals for the same two variables, and literals for only one of the two variables differ. In terms of the strings representing the products, these strings must have a dash in the same position and must differ in exactly one of the other two slots. We can combine the products yz and yz, represented by the strings -11 and -01, into z, represented by the string$--1$. We show all the combinations of terms that can be formed in this way in Table 4.11.

In Table 4.11 we also indicate which terms have been used to form products with fewer literals; these terms will not be needed in a minimal *expansion. The next step is to identify a minimal set of products needed to represent the Boolean function. We begin with all those products that were not used to construct products with fewer literals. Next, we form Table 4.12, which has a row for each candidate product formed by combining original terms, and a column for each original term; and we put an* X *in a position if the original term in the sum-of-products expansion was used to form this candidate product. In this case, we say that the candidate product covers the original minterm. We need to include at least one product that covers each of the original minterms. Consequently, whenever there is only one* X *in a column in the table, the product corresponding to the row this* X *is in must be used. From Table 4.12 we see that both* z *and* $\bar{x}\bar{y}$ *are needed. Hence, the final answer is* $z + \bar{x}\bar{y}$.

Table 4.12: Terms used to form Products with LowerLiterals for Example 222

	xyz	$\bar{x}yz$	$\bar{x}yz$	$\bar{x}\bar{y}z$	$\bar{x}\bar{y}\bar{z}$
z	X	X	X	X	
$\bar{x}\bar{y}$				X	X

As was illustrated in Example 222, the QuineMcCluskey method uses this sequence of steps to simplify a sum-of-products expression.

1. *Express each minterm in* n *variables by a bit string of length* n *with a 1 in the* i^{th} *position if* x_i *occurs and a 0 in this position if* $\bar{x}_i$ *occurs.*

2. *Group the bit strings according to the number of 1s in them.*

3. *Determine all products in* $n - 1$ *variables that can be formed by taking the Boolean sum of minterms in the expansion. Minterms that can be combined are represented by bit strings that differ in exactly one position. Represent these products in* $n-1$ *variables with strings*

that have a 1 in the i^{th} position if x_i occurs in the product, a 0 in this position if $\bar{x}_i$ occurs, and a dash in this position if there is no literal involving x_i in the product.

4. *Determine all products in $n-2$ variables that can be formed by taking the Boolean sum of the products in $n-1$ variables found in the previous step. Products in $n-1$ variables that can be combined are represented by bit strings that have a dash in the same position and differ in exactly one position.*

5. *Continue combining Boolean products into products in fewer variables as long as possible.*

6. *Find all the Boolean products that arose that were not used to form a Boolean product in one fewer literal.*

7. *Find the smallest set of these Boolean products such that the sum of these products represents the Boolean function. This is done by forming a table showing which minterms are covered by which products. Every minterm must be covered by at least one product. The first step in using this table is to find all essential prime implicants. Each essential prime implicant must be included because it is the only prime implicant that covers one of the minterms. Once we have found essential prime implicants, we can simplify the table by eliminating the rows for minterms covered by these prime implicants. Furthermore, we can eliminate any prime implicants that cover a subset of minterms covered by another prime implicant (as the reader should verify). Moreover, we can eliminate from the table the row for a minterm if there is another minterm that is covered by a subset of the prime implicants that cover this minterm. This pro-*

cess of identifying essential prime implicants that must be included, followed by eliminating redundant prime implicants and identifying minterms that can be ignored, is iterated until the table does not change. At this point we use a backtracking procedure to find the optimal solution where we add prime implicants to the cover to find possible solutions, which we compare to the best solution found so far at each step.

Example 223. *Use the Quine-McCluskey method to simplify the sum-of-products expansion* $wxy\bar{z}+w\bar{x}yz+w\bar{x}y\bar{z}+\bar{w}xyz+\bar{w}x\bar{y}z+\bar{w}\bar{x}yz+\bar{w}\bar{x}\bar{y}z$.

Solution: We first represent the minterms by bit strings and then group these terms together according to the number of 1s in the bit strings. This is shown in Table 4.13. All the Boolean products that can be formed by taking Boolean sums of these products are shown in Table 4.14.

The only products that were not used to form products in fewer variables are $\bar{w}z$, $wy\bar{z}$, $w\bar{x}y$, and $\bar{x}yz$. In Table 4.15 we show the minterms covered by each of these products. To cover these minterms we must include $\bar{w}z$ and $wy\bar{z}$, because these products are the only products that cover $\bar{w}xyz$ and $wxy\bar{z}$, respectively. Once these two products are included, we see that only one of the two products left is needed. Consequently, we can take either $\bar{w}z + wy\bar{z} + w\bar{x}y$ or $\bar{w}z + wy\bar{z} + \bar{x}yz$ as the final answer.

□

Table 4.13: Minterms and Bit Strings for Example 223

Term	Bit String	Number of 1s
$wxy\bar{z}$	1110	3
$w\bar{x}yz$	1011	3
$\bar{w}xyz$	0111	3
$w\bar{x}y\bar{z}$	1010	2
$\bar{w}x\bar{y}z$	0101	2
$\bar{w}\bar{x}yz$	0011	2
$\bar{w}\bar{x}\bar{y}z$	0001	1

Table 4.14: Procedure for Simplification for Example 223

Table 4.15: Terms used to Form Products with Fewer Literals for Example 223

	$wxy\bar{z}$	$w\bar{x}yz$	$\bar{w}xyz$	$w\bar{x}y\bar{z}$	$\bar{w}x\bar{y}z$	$\bar{w}\bar{x}yz$	$\bar{w}\bar{x}\bar{y}z$
$\bar{w}z$			X		X	X	X
$wy\bar{z}$	X			X			
$w\bar{x}y$		X		X			
$\bar{x}yz$		X				X	

Let Us Sum Up

In this section, we discussed about

- *Combination of Gates.*
- *Examples of circuits.*
- *Minimization of circuits.*
- *Karnaugh Maps.*
- *The Quine-Mc Cluskey method.*

Check your Progress

1. The circuit which accepts the value of the Boolean variable as input and produces the complement of this value as output is called

(a) Combinatorial circuit

(b) AND gate

(c) OR gate

(d) an inverter

2. To find the minimal expansions of Boolean functions in any number of variables, we use

(a) Logic gates

(b) Quine-Mc Cluskey method

(c) Karnaugh maps

(d) Combinatorial circuits

Summary

In this unit, we discussed about

- *Boolean functions.*
- *Identities of Boolean Algebra.*
- *Sum-of-product expansion.*
- *Functional completeness.*
- *Logic gates.*

$*$ *Examples of circuits.*

$*$ *Examples of circuits.*

$*$ *Minimization of circuits.*

$*$ *Karnaugh Maps.*

$*$ *Quine-Mc Cluskey method.*

Glossary

Boolean variable: *a variable that assumes only the values* 0 *and* 1.

dual of a Boolean expression: *the expression obtained by interchanging* $+$ *signs and* $\cdot$ *signs and interchanging* $0s$ *and* 1*'s.*

Boolean function of degree n***:*** *a function from* B^n *to* B *where* $B = \{0, 1\}$.

Boolean algebra: *a set* B *with two binary operations* $\vee$ *and* $\wedge$*, elements* 0 *and* 1*, and a complementation operator*$^-$*that satisfies the identity, complement, associative, commutative, and distributive laws.*

literal of the Boolean variable x***:*** *either* x *or* $\bar{x}$.

minterm of $x_1, x_2, \cdots, x$***:*** *a Boolean product* $y_1y_2\cdots y_n$*, where each* y_i *is either* x_i *or* $\bar{x}_i$.

sum-of-products expansion (or disjuctive normal form: *the representation of a Boolean function as a disjunction of minterms.*

functionally complete: *a set of Boolean operators is called functionally complete if every Boolean function can be represented using these operators.*

half adder: *a circuit that adds two bits, producing a sum bit.*

full adder: *a circuit that adds two bits and a carry, producing a sum bit and a carry bit.*

Self Assessment Questions

1. *How many Boolean functions of degree two are there?*
2. *Give a recursive definition of the set of Boolean expressions.*
3. *Explain how to construct the sum-of-products expansion of a Boolean function.*
4. *What does it mean for a set of operations to be functionally complete?*
5. *Explain how to build a circuit for a light controlled by two switches using OR gates, AND gates, and inverters.*

Exercises

1. *Given an example of a self-dual Boolean function of three variables.*
2. *Show that if F and G are Boolean functions of degree n, then*
   a) $F \leq F + G$ b) $FG \leq F$.
3. *Show that if F, G, and H are Boolean functions of degree n, then $F + G \leq H$ if and only if $F \leq H$ and $G \leq H$.*
4. *Show that $x \odot y = xy + \bar{x}\bar{y}$.*
5. *Is it always true that $(x \odot y) \odot z = x \odot (y \odot z)$?*

Answers to Check your Progress

Section 4.1: 1)b 2)d

Section 4.2: 1)c 2)c

Section 4.3: 1)d 2)b

Reference:

1. *Kenneth H. Rosen, "Discrete Mathematics and its Applications", 7th Edition, WCB/ McGraw Hill Publications, New Delhi, 2011.*

Suggested Readings:

1. *Edward A. Bender and S. Gill Williamson, "A Short Course in Discrete Mathematics", Dover Publications, 2006.*
2. *M.O. Albertson and J.P. Hutchinson, "Discrete Mathematics with Algorithms", John Wiley & Sons, 2008.*
3. *Rajendra Akerkar and Rupali Akarkar, "Discrete Mathematics", Pearson Education Pvt. Ltd, Singapore, 2004.*
4. *J. P. Trembley and R. Manohar, "Discrete Mathematical Structures", Tata McGraw Hill, New Delhi,1997.*
5. *Martin Aigner, "A Course in Enumeration", Springer-Verlag, Heidelberg, 2007.*
6. *J.H. Van Lint and R.M. Wilson, "A Course in Combinatorics", 2nd Edition, Cambridge University Press, Cambridge, 2001.*

Unit 5

Modelling Computation

Objectives

1. *To introduce finite-state machines that produce output.*
2. *To apply finite-state machines in language recognition.*
3. *To learn Turing machines and its importance.*
4. *To discuss computational complexcity, computability and decidability.*
5. *To measure the complexcity of algorithms.*

5.1 Finite-State Machines with Output

Definition 224. *A finite-state machine M = (S, I, O, f, g, s_0) consists of a finite set S of states, a finite input alphabet I, a finite output alphabet O, a transition function f that assigns to each state and input pair a new state, an output function g that assigns to each state, and input pair an output, and an initial state s_0.*

Let M = (S, I, O, f, g, s_0) be a finite-state machine. We can use a **state table** *to represent the values of the transition function f and the output function g for all pairs of states and input.*

Example 225. *The state table shown in Table 1.1 describes a finite-state machine with $S = \{s_0, s_1, s_2, s_3\}$, $I = \{0, 1\}$, and $O = \{0, 1\}$. The values of the transition function f are displayed in the first two columns, and the values of the output function g are displayed in the last two columns.*

Another way to represent a finite-state machine is to use a **state diagram**, *which is a directed graph with labeled edges. In this diagram, each state is represented by a circle. Arrows labeled with the input and output pair are shown for each transition.*

Example 226. *Construct the state diagram for the finite-state machine with the state table shown in Table 1.1. The state diagram for this machine is shown in Figure 1.1.*

Solution: *The state diagram for this machine is shown in Figure 1.1.*

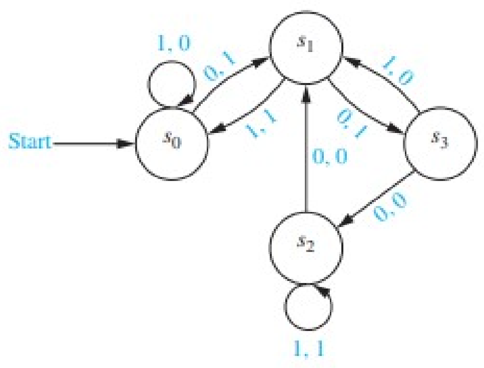


Figure 5.1: The State Diagram for the Finite-State Machine Shown in Table 1.1.

Table 5.1

State	f		g	
	Input		Input	
	0	1	0	1
s_0	s_1	s_0	1	0
s_1	s_3	s_0	1	1
s_2	s_1	s_2	0	1
s_3	s_2	s_1	0	0

Example 227. *Construct the state table for the finite-state machine with the state diagram shown in Figure 1.2.*

Solution: *The state table for this machine is shown in Table 1.2.*

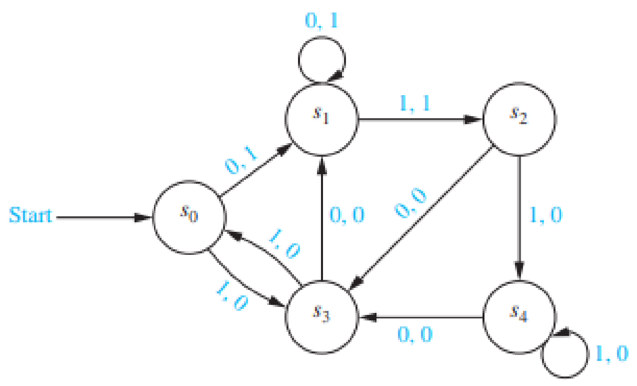


Figure 5.2: A finite-state machine.

Table 5.2

State	f		g	
	Input		Input	
	0	1	0	1
s_0	s_1	s_3	1	0
s_1	s_1	s_2	1	1
s_2	s_3	s_4	0	0
s_3	s_1	s_0	0	0
s_4	s_3	s_4	0	0

Example 228. *Find the output string generated by the finite-state machine in Figure 1.2 if the input string is 101011.*

solution: *The output obtained is 001000. The successive states and outputs are shown in Table 1.3.*

Table 5.3

Input	1	0	1	0	1	1	-
State	s_0	s_3	s_1	s_2	s_3	s_0	s_3
Output	0	0	1	0	0	0	-

Example 229. *An important element in many electronic devices is a unit-delay machine, which produces as output the input string delayed by a specified amount of time. How can a finite-state machine be constructed that delays an input string by one unit of time, that is, produces as output the bit string* $0x_1x_2\cdots x_{k-1}$ *given the input bit string* $x_1x_2\cdots x_k$*?*

solution: *A delay machine can be constructed that has two possible in-*

puts, namely, 0 and 1. The machine must have a start state s_0. Because the machine has to remember whether the previous input was a 0 or a 1, two other states s_1 and s_2 are needed, where the machine is in state s_1 if the previous input was 1 and in state s_2 if the previous input was 0. An output of 0 is produced for the initial transition from s_0. Each transition from s_1 gives an output of 1, and each transition from s_2 gives an output of 0. The output corresponding to the input of a string $x_1 \cdots x_k$ is the string that begins with 0, followed by x_1, followed by $x_2, \cdots$ ending with $x_k - 1$. The state diagram for this machine is shown in Figure 1.3.

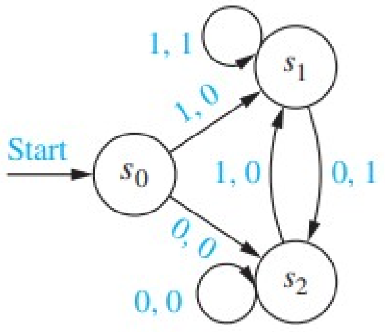


Figure 5.3: A unit-delay machine

Example 230. *Produce a finite-state machine that adds two positive integers using their binary expansions.*

solution: *When $(x_n \cdots x_1x_0)_2$ and $(y_n \cdots y_1y_0)_2$ are added, the following procedure is followed. First, the bits x_0 and y_0 are added, producing a sum bit z_0 and a carry bit c_0. This carry bit is either 0 or 1. Then, the bits x_1 and y_1 are added, together with the carry c_0. This gives a sum bit z_1 and a carry bit c_1. This procedure is continued until the nth stage, where x_n, y_n, and the previous carry c_{n-1} are added to produce the sum bit z_n and the carry bit c_n, which is equal to the sum bit z_{n+1}.*

A finite-state machine to carry out this addition can be constructed using just two states. For simplicity we assume that both the initial bits x_n *and* y_n *are 0 (otherwise we have to make special arrangements concerning the sum bit* z_{n+1}*). The start state* s_0 *is used to remember that the previous carry is 0 (or for the addition of the rightmost bits). The other state,* s_1*, is used to remember that the previous carry is 1.*

Because the inputs to the machine are pairs of bits, there are four possible inputs. We represent these possibilities by 00 (when both bits are 0), 01 (when the first bit is 0 and the second is 1), 10 (when the first bit is 1 and the second is 0), and 11 (when both bits are 1). The transitions and the outputs are constructed from the sum of the two bits represented by the input and the carry represented by the state. For instance, when the machine is in state s_1 *and receives 01 as input, the next state is* s_1 *and the output is 0, because the sum that arises is* $0 + 1 + 1 = (10)_2$*. The state diagram for this machine is shown in Figure 1.4.*

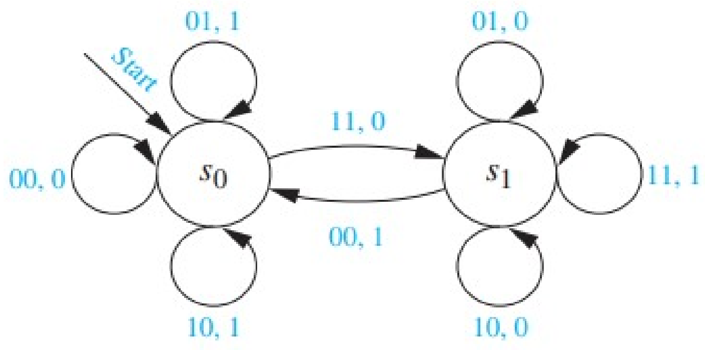


Figure 5.4: A Finite-State Machine for Addition.

Example 231. *In a certain coding scheme, when three consecutive 1s appear in a message, the receiver of the message knows that there has been a transmission error. Construct a finite-state machine that gives a 1* *as its current output bit if and only if the last three bits received are all 1s.*

solution:*Three states are needed in this machine. The start state s_0 remembers that the previous input value, if it exists, was not a 1. The state s_1 remembers that the previous input was a 1, but the input before the previous input, if it exists, was not a 1. The state s_2 remembers that the previous two inputs were 1s.*

An input of 1 takes s_0 to s_1, because now a 1, and not two consecutive 1s, has been read; it takes s_1 to s_2, because now two consecutive 1s have been read; and it takes s_2 to itself, because at least two consecutive 1s have been read. An input of 0 takes every state to s_0, because this breaks up any string of consecutive 1s. The output for the transition from s_2 to itself when a 1 is read is 1, because this combination of input and state shows that three consecutive 1s have been read. All other outputs are 0. The state diagram of this machine is shown in Figure 1.5.

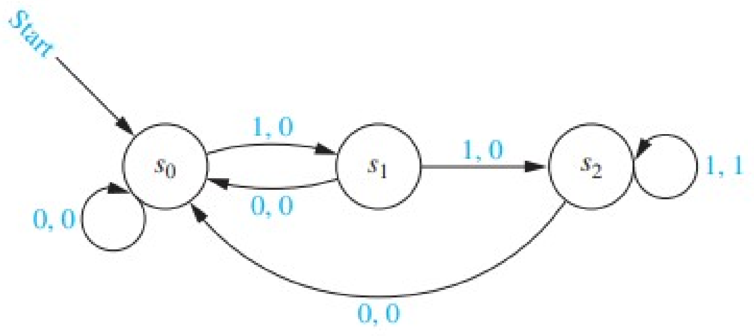


Figure 5.5: A Finite-State Machine That Gives an Output of 1 If and Only If the Input String Read So Far Ends with 111.

Definition 232. *Let $M = (S, I, O, f, g, s_0)$ be a finite-state machine and $L \subseteq I^*$. We say that M recognizes (or accepts) L if an input string x belongs to L if and only if the last output bit produced by M when given x as input is a 1.*

Let Us Sum Up

In this section, we discussed about

- ∗ *Finite-state machines with output.*
- ∗ *Finite-state machine that adds two positive integers.*
- ∗ *Types of finite-state machines.*

Check your Progress

1. In state diagram, each state is represented by

(a) Arrows

(b) Circle

(c) 1

(d) 0

2. The components of computers can be modeled using

(a) Boolean Algebra

(b) Electronic circuits

(c) Finite-State machines

(d) Logic gates

5.2 Finite-State Machines with No Output

5.2.1 Set of Strings

Definition 233. *Suppose that A and B are subsets of V^*, where V is a vocabulary. The concatenation of A and B, denoted by AB, is the set of all strings of the form xy, where x is a string in A and y is a string in B.*

Example 234. *Let $A = \{0, 11\}$ and $B = \{1, 10, 110\}$. Find AB and BA.*

Solution: *The set AB contains every concatenation of a string in A and a string in B. Hence, $AB = \{01, 010, 0110, 111, 1110, 11110\}$. The set BA contains every concatenation of a string in B and a string in A. Hence, $BA = \{10, 111, 100, 1011, 1100, 11011\}$.*

Example 235. *Let $A = \{1, 00\}$. Find A^n for $n = 0, 1, 2,$ and 3.*

Solution: *We have $A^0 = \{\lambda\}$ and $A^1 = A^0A = \{\lambda\}A = \{1, 00\}$. To find A^2 we take concatenations of pairs of elements of A. This gives $A^2 = \{11, 100, 001, 0000\}$. To find A^3 we take concatenations of elements in A^2 and A; this gives $A^3 = \{111, 1100, 1001, 10000, 0011, 00100, 00001, 000000\}$.*

Definition 236. *Suppose that A is a subset of V^*. Then the Kleene closure of A, denoted by A^*, is the set consisting of concatenations of arbitrarily many strings from A. That is, $A^* = \bigcup_{k=0}^{\infty} A^k$.*

Example 237. *What are the Kleene closures of the sets $A = \{0\}$, $B = \{0, 1\}$, and $C = \{11\}$?*

Solution: *The Kleene closure of A is the concatenation of the string 0 with itself an arbitrary finite number of times. Hence, $A^* = \{0^n | n = 0, 1, 2, \cdots\}$. The Kleene closure of B is the concatenation of an arbitrary number of strings, where each string is either 0 or 1. This is the set of* *all strings over the alphabet $V = \{0, 1\}$. That is, $B^* = V^*$. Finally, the Kleene closure break of C is the concatenation of the string 11 with itself an arbitrary number of times. Hence, C^* is the set of strings consisting of an even number of 1s. That is, $C^* = \{1^{2n} | n = 0, 1, 2, \cdots\}$.*

5.2.2 Finite-State Automata

We will now give a definition of a finite-state machine with no output. Such machines are also called **finite-state automata***.*

Definition 238. *A finite-state automaton $M = (S, I, f, s_0, F)$ consists of a finite set S of states, a finite input alphabet I, a transition function f that assigns a next state to every pair of state and input (so that $f : S \times I \to S$), an initial or start state s_0, and a subset F of S consisting of final (or accepting states).*

Example 239. *Construct the state diagram for the finite-state automaton $M = (S, I, f, s_0, F)$, where $S = \{s_0, s_1, s_2, s_3\}$, $I = \{0, 1\}$, $F = \{s_0, s_3\}$, and the transition function f is given in Table 1.4.*

Solution: *The state diagram is shown in Figure 1.6. Note that because both the inputs 0 and 1 take s_2 to s_0, we write 0,1 over the edge from s_2 to s_0.*

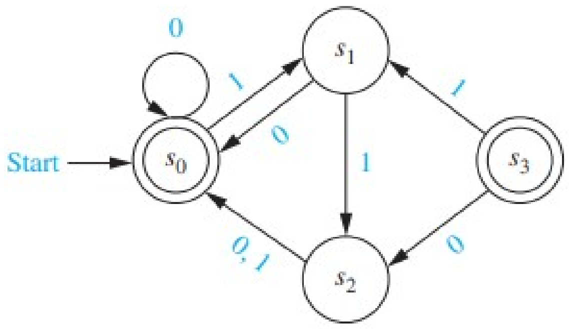


Figure 5.6: The State Diagram for a Finite-State Automaton.

Table 5.4

State	f Input 0	f Input 1
s_0	s_0	s_1
s_1	s_0	s_2
s_2	s_0	s_0
s_3	s_2	s_1

5.2.3 Language Recognition by Finite-State Machines

Definition 240. *A string x is said to be recognized or accepted by the machine $M = (S, I, f, s_0, F)$ if it takes the initial state s_0 to a final state, that is, $f(s_0, x)$ is a state in F. The language recognized or accepted by the machine M, denoted by $L(M)$, is the set of all strings that are recognized by M. Two finite-state automata are called equivalent if they recognize the same language.*

Designing Finite-State Automata

Example 241. *Construct deterministic finite-state automata that recognize each of these languages.*

(a) the set of bit strings that begin with two 0s

(b) the set of bit strings that contain two consecutive 0s

(c) the set of bit strings that do not contain two consecutive 0s

(d) the set of bit strings that end with two 0s

(e) the set of bit strings that contain at least two 0s

Solution: *(a) Our goal is to construct a deterministic finite-state automaton that recognizes the set of bit strings that begin with two 0s. Besides the start state s_0, we include a non final state s_1; we move to s_1 from s_0 if the first bit is a 0. Next, we add a final state s_2, which we move to from s_1 if the second bit is a 0. When we have reached s_2 we know that the first two input bits are both 0s, so we stay in the state s_2 no matter what the succeeding bits (if any) are. We move to a nonfinal state s_3 from s_0 if the first bit is a 1 and from s_1 if the second bit is a 1. The reader should verify that the finite-state automaton in Figure 1.7(a) recognizes the set of bit strings that begin with two 0s.*
(b) Our goal is to construct a deterministic finite-state automaton that recognizes the set of bit strings that contain two consecutive 0s. Besides the start state s_0, we include a nonfinal state s_1, which tells us that the last input bit seen is a 0, but either the bit before it was a 1, or this bit was the initial bit of the string. We include a final state s_2 that we move to from s_1 when the next input bit after a 0 is also a 0. If a 1 follows a 0 in the string (before we encounter two consecutive 0s), we return to s_0

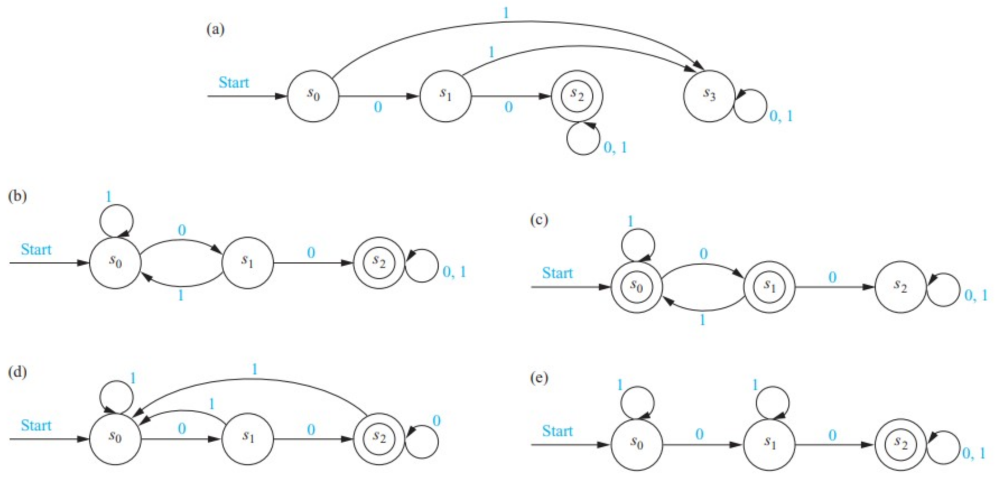


Figure 5.7: Deterministic Finite-State Automata Recognizing the Languages

and begin looking for consecutive 0s all over again. The reader should verify that the finite-state automaton in Figure 1.7(b) recognizes the set of bit strings that contain two consecutive 0s.

(c) Our goal is to construct a deterministic finite-state automaton that recognizes the set of bit strings that do not contain two consecutive 0s. Besides the start state s_0*, which should be a final state, we include a final state* s_1*, which we move to from* s_0 *when 0 is the first input bit. When an input bit is a 1, we return to, or stay in, state* s_0*. We add a state* s_2*, which we move to from* s_1 *when the input bit is a 0. Reaching* s_2 *tells us that we have seen two consecutive 0s as input bits. We stay in state* s_2 *once we have reached it; this state is not final. The reader should verify that the finite-state automaton in Figure 1.7(c) recognizes the set of bit strings that do not contain two consecutive 0s.*

(d) Our goal is to construct a deterministic finite-state automaton that recognizes the set of bit strings that end with two 0s. Besides the start state s_0*, we include a nonfinal state* s_1*, which we move to if the first bit is*

0. We include a final state s_2, which we move to from s_1 if the next input bit after a 0 is also a 0. If an input of 0 follows a previous 0, we stay in state s_2 because the last two input bits are still 0s. Once we are in state s_2, an input bit of 1 sends us back to s_0, and we begin looking for consecutive 0s all over again. We also return to s_0 if the next input is a 1 when we are in state s_1. The reader should verify that the finite-state automaton in Figure 1.7 (d) recognizes the set of bit strings that end with two 0s.

(e) Our goal is to construct a deterministic finite-state automaton that recognizes the set of bit strings that contain two 0s. Besides the start state, we include a state s_1, which is not final; we stay in s_0 until an input bit is a 0 and we move to s_1 when we encounter the first 0 bit in the input. We add a final state s_2, which we move to from s_1 once we encounter a second 0 bit. Whenever we encounter a 1 as input, we stay in the current state. Once we have reached s_2, we remain there. Here, s_1 and s_2 are used to tell us that we have already seen one or two 0s in the input string so far, respectively. The reader should verify that the finite-state automaton in Figure 1.7 (e) recognizes the set of bit strings that contain two 0s.

Example 242. *Construct a deterministic finite-state automaton that recognizes the set of bit strings that contain an odd number of 1s and that end with at least two consecutive 0s.*

Solution: *We can build a deterministic finite-state automaton that recognizes the specified set by including states that keep track of both the parity of the number of 1 bits and whether we have seen no, one, or at least two 0s at the end of the input string.*

The start state s_0 can be used to tell us that the input read so far contains an even number of 1s and ends with no 0s (that is, is empty or ends with a 1). Besides the start state, we include five more states. We move to *states $s_1, s_2, s_3, s_4,$ and s_5, respectively, when the input string read so far contains an even number of 1s and ends with one 0; when it contains an even number of 1s and ends with at least two 0s; when it contains an odd number of 1s and ends with no 0s; when it contains an odd number of 1s and ends with one 0; and when it contains an odd number of 1s and ends with two 0s. The state s_5 is a final state.*

The reader should verify that the finite-state automaton in Figure 1.8 recognizes the set of bit strings that contain an odd number of 1s and end with at least two consecutive 0s

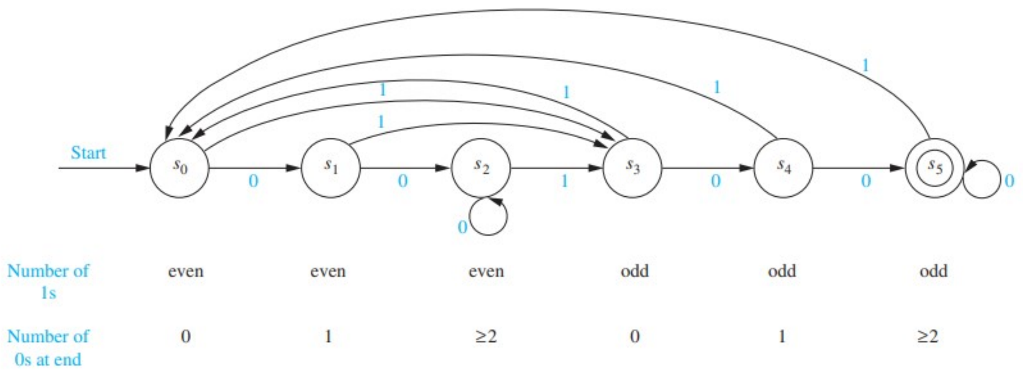


Figure 5.8: A Deterministic Finite-State Automaton Recognizing the Set of Bit Strings Containing an Odd Number of 1s and Ending with at Least Two 0s.

Equivalent Finite-State Automata

Two finite state automata are equivalent if they recognize the same language. Example 243 provides an example of two equivalent deterministic finite-state machines.

Example 243. *Show that the two finite-state automata M_0 and M_1 shown in Figure 1.9 are equivalent.*

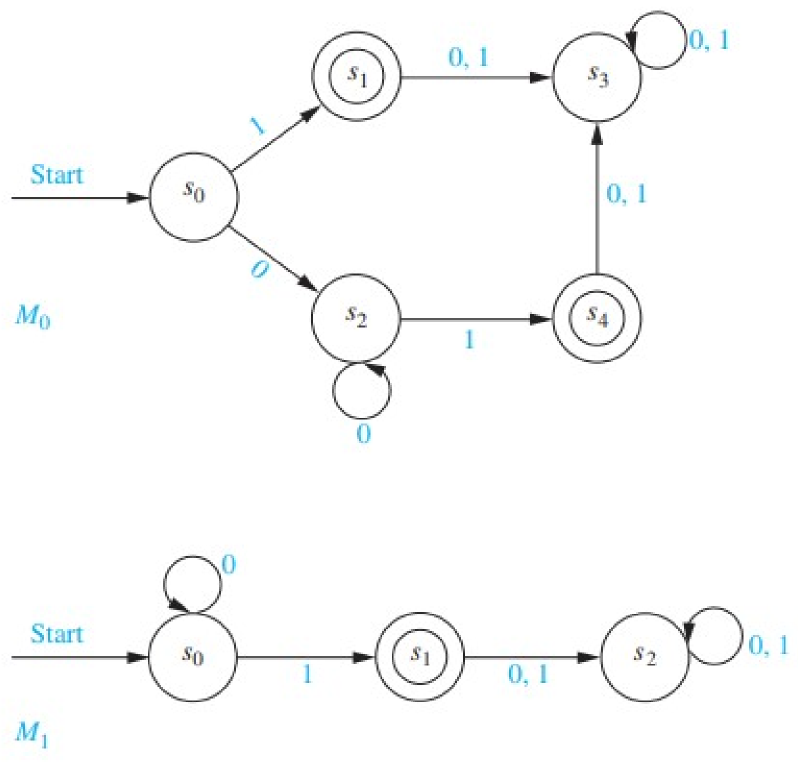


Figure 5.9: M_0 and M_1 Are Equivalent Finite-State Automata

Solution: *For a string x to be recognized by M_0, x must take us from s_0 to the final state s_1 or the final state s_4. The only string that takes us from s_0 to s_1 is the string 1. The strings that take us from s_0 to s_4 are those strings that begin with a 0, which takes us from s_0 to s_2, followed by zero or more additional 0s, which keep the machine in state s_2, followed by a 1, which takes us from state s_2 to the final state s_4. All other strings take us from s_0 to a state that is not final. We conclude that $L(M_0)$ is the set of strings of zero or more 0 bits followed by a final 1.*

For a string x to be recognized by M_1, x must take us from s_0 to the final state s_1. So, for x to be recognized, it must begin with some number of 0s, which leave us in state s_0, followed by a 1, which takes us to the final state s_1. A string of all zeros is not recognized because it leaves us in *state s_0, which is not final. All strings that contain a 0 after 1 are not recognized because they take us to state s_2, which is not final. It follows that $L(M_1)$ is the same as $L(M_0)$. We conclude that M_0 and M_1 are equivalent. Note that the finite-state machine M_1 only has three states. No finite state machine with fewer than three states can be used to recognize the set of all strings of zero or more 0 bits followed by a 1. The construction of a finite-state automaton with the fewest states possible among all finite-state automata equivalent to a given finite-state automaton, is known as* **machine minimization**.

5.2.4 Theorem (Myhill-Nerode Theorem)

The following three statement one equivalent.

1. *$L \subseteq \Sigma^*$ is accepted by a DFSA.*
2. *L is the union of some of the equivalance classes of a right irrvariant equivalance relation of finite index on Σ^*.*
3. *Let equivalance relation R' be defined over Σ^* as follows: $xR'y$ if and only if, for all z in Σ^*, xz is in L exactly when yz is in L. Then R' is of finite index.*

Proof. We shall prove $1 \to 2$, $2 \to 3$, $3 \to 1$.

$1 \to 2$. Let L be accepted by a FSA $M = (S, \sum, \delta, s_0, F)$. Define a relation R on Σ^* such that xRy if $\delta(s_0, x) = \delta(s_0, y)$. R is an equivalence relation as equality relation is reflexive, symmetric and transitive.

So R divide Σ^* into equivalence classes. The set of strings which take the machine from s_0 to a particular state q are in one equivalence class. The number of equivalence classes is therefore equivalent to the number of states of M, assuming every state is reachable from $_0$. It can be seen that this equivalence relation is right invariant.

$$\text{i.e., } xRy \to xzRyz \text{ for all } z \in \Sigma^*$$

$$\begin{aligned}\delta(s_0, x) &= \delta(s_0, y) \text{ if } xRY \\ \delta(s_0, xz) &= \delta(\delta(s_0, x), z) \\ &= \delta(\delta(s_0, y), z) \\ &= \delta(s_0, yz)\end{aligned}$$

$\therefore\ xzRyz.$

L is the union of those equivalence classes of R which correspond to final states of M.

$2 \to 3$ Assume 2 of the theorem. Let E be the equivalence relation considered. Let $R^{'}$ be defined as given in the statement of the theorem. We can prove that $xEy \to xR^{'}y$.

Let xEy. Then $xz\ Eyz$ for all $x \in \Sigma^* \cdot xz$ and yz are in the same equivalence class of E. Hence xz and yz are both in L or both are not in L as L is the union of some of the equivalence classes of E. Hence $xR^{'}y$.

Thus we note that any equivalence class of E is completely contained in one equivalence class of $R^{'}$. Therefore, since E is of finite index, $R^{'}$ is of finite index too.

$3 \to 1$

We first show that $R^{'}$ is right invariant., $xR^{'}y$ if for all z in Σ^*, xz is in L exactly when yz is in L. We can also write it as follows:

$xR^{'}y$ if for all w, z in $\Sigma^* xwz$ is in L exactly when ywz is in L.

Hence it follows $xwR^{'}yw$.

Hence $R^{'}$ is right invariant.

Let $\{x\}$ denote the equivalence class of $R^{'}$ to which x belongs.

Construct a DFSA $M_L = (S^{'}\Sigma, \delta^{'}, s^{'}_0, F^{'})$. $S^{'}$ contains one state corresponding to each equivalence class of $R^{'} \cdot [\epsilon]$ corresponds to $s^{'}_0$. $F^{'}$ corresponds to those states $[x],\ x \in L$. $\delta^{'}$ is defined as follows: $\delta^{'}([x], a) = [xa]$.

This definition is consistent as $R^{'}$ is right invariant. Suppose x and y belong to the same equivalence class of $R^{'}$. Then xa and ya will belong to the same equivalence class of $R^{'}$. If $x \in L$, $[x]$ is a final state of M_L. the automaton M_L accepts L.

□

Example 244. *Consider the FSA M given in Figure 1.10. The set of strings having at least one a will be accepted $\{a, b, c\}^*$ is divide into three equivalence classes.*

C_1, the set of strings which take the machine from s_0 to $s_0 - \{b, c\}^*$

C_2, the set of strings which take the machine from s_0 to s_1-set of strings having odd number of a's.

C_3 the set of strings which take the machine from s_0 to s_2-set of strings having even number of a's

It is see that $L(M) = C_2 \cup C_3$. We note that

If$x \in C_1$ and $y \in C_2$, then $xb \in C_1$ and $yb \in C_2$, $xb \notin L$, $yb \in L$.

$\therefore\ \ x/R^{'}y$.

Similarly it can be seen that if $x \in C_1$ and $y \in C_3$

$x \notin R^{'}y$.

Butif $x \in C_2$ and $y \in C_3$, $xR^{'}y$.

So in $R^{'}C_2$ and C_3 are merged.

If we construct the automaton as given in the theorem we get, the automaton in figure 1.11.

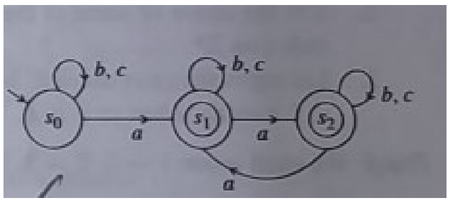


Figure 5.10: Figure for Example 244

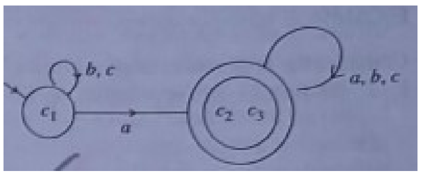


Figure 5.11: Figure for Example 244

Theorem 245. *The minimum state automaton accepting a regular set L is unique upto an isomorphism and is given by M_L of the proof of Myhill-Nerode Theorem.*

Proof. In the proof of Myhill-Nerode theorem, we started with M and found equivalence relation R, moved to $R^{'}$ and constructed M_L. The index of $R^{'}$ is less than or equal to the index of R. The number of states of M is the index of R and the number of states of M_L is the index of $R^{'}$. If M and M_L have the same number of states, then we can find a mapping $\theta : S \to S^{'}$ such that if

$$\theta(q) = q^{'} \quad \theta(\delta(q, a)) = \delta^{'}(q^{'}, a).$$

This is achieved by mapping the initial state of M into initial state of $M_L \cdot \theta(s_0) = s_0^{'}$. If q is in s, then $\exists x$ in Σ^* such that $\delta(s_0, x) = q \cdot q$ i mapped on to $q^{'}$ where $\delta^{'}(s_0^{'}, x) = q^{'}$. This is a consistent definition. For if $\delta(q, a) = p$ and $\delta^{'}(q^{'}, a) = p^{'}$, then $\delta(s_0, xa) =^{'}$ and $\theta(p) = p^{'}$. $\square$

5.2.5 Minimization of DFSA

Let $M = (S, \Sigma, \delta, s_0, F)$ be a DFSA. Let $\backsim$ be an equivalence relation on S such that $p \backsim q$, if and only if for each input string x, $\delta(p, x) \in F$ if and only if $\delta(q, x) \in F$. The meaning of this is that for any $x, \delta(p, x)$ and $\delta(q, x)$ are both in F or both are not in F. If for a string $x, \delta(p, x)$ is in $\delta(q, x)$ is not in F or the other way round, we say p and q are distinguishable by x.

If $p \backsim q$ then $\delta(p, a) \backsim \delta(q, a)$.

If $\delta(p, a) = r$ and $\delta(q, a) = s$ and r and s are distinguishable by x, then p and q are distinguishable by ax.

Algorithm to find minimum DFSA for $M = (\delta, \Sigma, \delta, s_0, F)$.

We partition the set of states in M as follows:

1. Divide S into non final states. F and $K - F$. Any state in F is distinguishable from a state is $K - F$ by λ.

2. Consider the set of states in a block. Consider the successors of them for each a in Σ. If they belong to different blocks, split this block into two or more blocks, depending on the successors of the states.

3. Repeat step 2 till no more splitting is possible.

4. For each block B_i, consider a state p_i. Construct $M' = (S, \Sigma, \delta', s_0', F')$ where $S' = \{p_i | B_i$ is a block of the partition obtained in step 2 $\}$, s_0' corresponds to the block containing s_0. $\delta(p_i, a) = p_j$, if there exists $q_i \epsilon B_i$ and $q_j \epsilon B_j$ such that $\delta(q_i, a) = q_j$. F' consists of states corresponding to the blocks containing states in F.

Example 246. *Consider the previous example splitting $\bar{s_0}, \bar{s_1}, \bar{s_2}$ in to final and non final states we get $\bar{s_0}$ and $\bar{s_1}$, $\bar{s_2}$. No further split is possible, so we get the minimum automaton.*

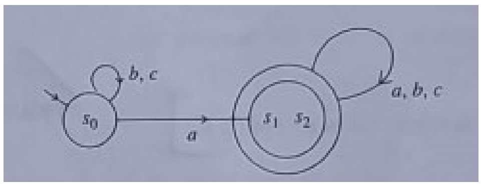


Figure 5.12: Figure for Example 246

5.2.6 Nondeterministic Finite-State Automata

A finite-state automata is deterministic, if for each pair of state and input value there is a unique next state given by the transition function. A finite-state automaton in which there may be several possible next states for each pair of input value and state, machines are called **nondeterministic**.

Definition 247. *A nondeterministic finite-state automaton $M = (S, I, f, s_0, F)$ consists of a set S of states, an input alphabet I, a transition function f that assigns a set of states to each pair of state and input (so that $f : S \times I \to P(S)$), a starting state s_0, and a subset F of S consisting of the final states.*

Example 248. *Find the state diagram for the nondeterministic finite-state automaton with the state table shown in Table 1.5. The final states are s_2 and s_3.*

Solution: *The state diagram for this automaton is shown in Figure 1.10.*

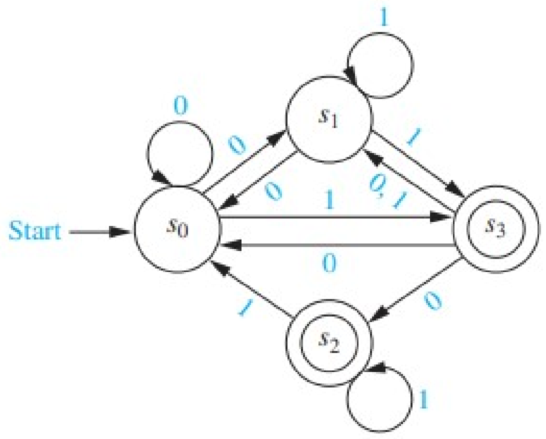


Figure 5.13: The Nondeterministic Finite-State Automaton with State Table 1.5.

Table 5.5

State	f	
	Input	
	0	1
s_0	s_0, s_1	s_3
s_1	s_0	s_1, s_3
s_2		s_0, s_2
s_3	s_0, s_1, s_2	s_1

Example 249. *Find the state table for the nondeterministic finite-state automaton with the state diagram shown in Figure 1.14.*

Solution: *The state table is given as Table 1.6.*

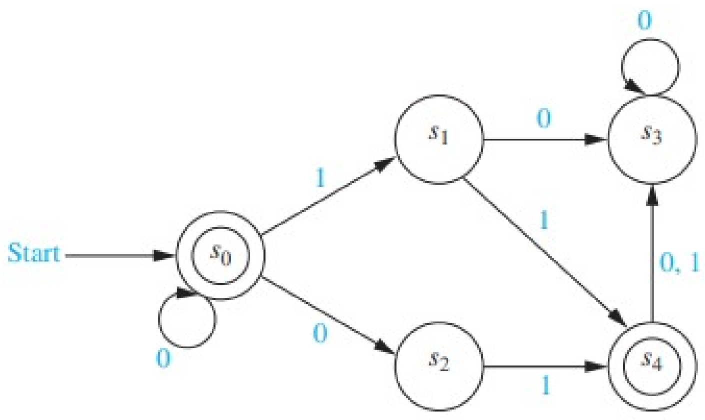


Figure 5.14: A Nondeterministic Finite-State Automaton.

Table 5.6

State	f	
	Input	
	0	1
s_0	s_0, s_2	s_1
s_1	s_3	s_4
s_2		s_4
s_3	s_3	s_3
s_4	s_3	s_3

The **language recognized** *by a nondeterministic finite-state automaton is the set of all strings recognized by this automaton.*

Example 250. *Find the language recognized by the nondeterministic finite-state automaton shown in Figure 1.14.*

Solution: *Because s_0 is a final state, and there is a transition from s_0 to itself when 0 is the input, the machine recognizes all strings consisting of zero or more consecutive 0s. Furthermore, because s_4 is a final state, any* *string that has s_4 in the set of states that can be reached from s0 with this input string is recognized. The only such strings are strings consisting of zero or more consecutive 0s followed by 01 or 11. Because s_0 and s_4 are the only final states, the language recognized by the machine is $\{0^n, 0^n01, 0^n11 | n \geq 0\}$.*

Theorem 251. *If the language L is recognized by a nondeterministic finite-state automaton M_0, then L is also recognized by a deterministic finite-state automaton M_1.*

Proof. We will describe how to construct the deterministic finite-state automaton M_1 that recognizes L from M_0, the nondeterministic finite-state automaton that recognizes this language. Each state in M_1 will be made up of a set of states in M_0. The start symbol of M_1 is $\{s_0\}$, which is the set containing the start state of M_0. The input set of M_1 is the same as the input set of M_0.

Given a state $\{s_{i_1}, s_{i_2}, \cdots, s_{i_k}\}$ of M_1, the input symbol x takes this state to the union of the sets of next states for the elements of this set, that is, the union of the sets $f(s_{i_1}, x)$, $f(s_{i_2}, x), \cdots, f(s_{i_k}, x)$. The states of M_1 are all the subsets of S, the set of states of M_0, that are obtained in this way starting with s_0. (There are as many as 2^n states in the deterministic machine, where n is the number of states in the nondeterministic machine, because all subsets may occur as states, including the empty set, although usually far fewer states occur.) The final states of M_1 are those sets that contain a final state of M_0.

Suppose that an input string is recognized by M_0. Then one of the states that can be reached from s_0 using this input string is a final state (the reader should provide an inductive proof of this). This means that in M_1, this input string leads from $\{s_0\}$ to a set of states of M_0 that contains a final state. This subset is a final state of M_1, so this string is also recognized by M_1. Also, an input string not recognized by M_0 does not lead to any final states in M_0. (The reader should provide the details that prove this statement.) Consequently, this input string does not lead from $\{s_0\}$ to a final state in M_1.

□

Example 252. *Find a deterministic finite-state automaton that recognizes the same language as the nondeterministic finite-state automaton in Example 249.*

Solution: *The deterministic automaton shown in Figure 1.15 is constructed from the nondeterministic automaton in Example 249. The states of this deterministic automaton are subsets of the set of all states of the nondeterministic machine. The next state of a subset under an input symbol is the subset containing the next states in the nondeterministic machine of all elements in this subset. For instance, on input of 0, $\{s_0\}$ goes to $\{s_0, s_2\}$, because s_0 has transitions to itself and to s_2 in the nondeterministic machine; the set $\{s_0, s_2\}$ goes to $\{s_1, s_4\}$ on input of 1, because s_0 goes just to $_1$ and s_2 goes just to s_4 on input of 1 in the nondeterministic machine; and the set $\{s_1, s_4\}$ goes to $\{s_3\}$ on input of 0, because s_1 and s_4 both go to just s_3 on input of 0 in the deterministic machine. All subsets that are obtained in this way are included in the deterministic finite-state machine. Note that the empty set is one of the states of this machine, because it is the subset containing all the next states of $\{s_3\}$ on input of 1. The start state is $\{s_0\}$, and the set of final states are all those that include s_0 or s_4.*

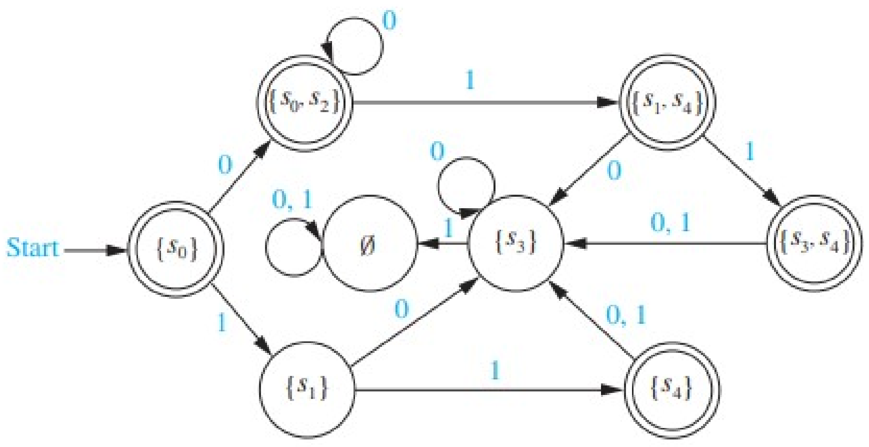


Figure 5.15: A Deterministic Automaton Equivalent to the Nondeterministic Automaton

Let Us Sum Up

In this section, we discussed about

- *Finite-state Automata.*
- *Language recognition by finite-state machines.*
- *Designing finite-state automata.*
- *Minimization of DFSA.*

Check your Progress

1. Let $A = \{1, 00\}$. *Then* $A^2 =$

(a) $\{11, 0000\}$

(b) $\{11, 10, 01, 00\}$

(c) $\{11, 100, 001, 0000\}$

(d) $\{11, 00, 1000\}$

2. *Two finite-state automata are called equivalent if*

(a) they recognize the same language

(b) they have the same design

(c) they are deterministic

(d) they satisfy Myhill-Nerode theorem.

5.3 Turing Machines

5.3.1 Introduction

The finite-state automata studied earlier in this chapter cannot be used as general models of computation. They are limited in what they can do. For example, finite-state automata are able to recognize regular sets, but are not able to recognize many easy-to-describe sets, including $\{0^n 1^n | n \geq 0\}$*, which computers recognize using memory. We can use finite-state automata to compute relatively simple functions such as the sum of two numbers, but we cannot use them to compute functions that computers can, such as the product of two numbers. To overcome these deficiencies we can use a more powerful type of machine known as a Turing machine, after Alan Turing, the famous mathematician and computer scientist who invented them in the 1930s.*

5.3.2 Definition of Turing Machines

Definition 253. *A Turing machine* $T = (S, I, f, s_0)$ *consists of a finite set* S *of states, an alphabet* I *containing the blank symbol* B*, a partial* *function f from $S \times I$ to $S \times I \times \{R, L\}$, and a starting state s_0.*

To interpret this definition in terms of a machine, consider a control unit and a tape divided into cells, infinite in both directions, having only a finite number of non blank symbols on it at any given time, as pictured in Figure 1.16. The action of the Turing machine at each step of its operation depends on the value of the partial function f for the current state and tape symbol.

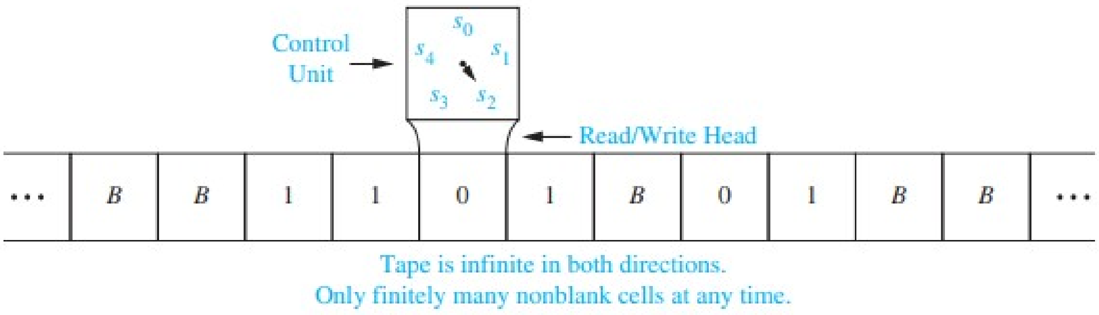

Control Unit
s_0
s_1
s_2
s_3
s_4
Read/Write Head
...
B
B
1
1
0
1
B
0
1
B
B
...
Tape is infinite in both directions.
Only finitely many nonblank cells at any time.

Figure 5.16: A Representation of a Turing Machine.

At each step, the control unit reads the current tape symbol x. If the control unit is in state s and if the partial function f is defined for the pair (s, x) with $f(s, x) = (s^{'}, x^{'}, d)$, the control unit

1. *enters the state $s^{'}$,*

2. *writes the symbol $x^{'}$ in the current cell, erasing x, and*

3. *moves right one cell if $d = R$ or moves left one cell if $d = L$.*

We write this step as the five-tuple $(s, x, s^{'}, x^{'}, d)$. If the partial function f is undefined for the pair (s, x), then the Turing machine T will halt.

A common way to define a Turing machine is to specify a set of five-tuples of the form $(s, x, s^{'}, x^{'}, d)$. The set of states and input alphabet is implicitly defined when such a definition is used.

At the beginning of its operation a Turing machine is assumed to be in the initial state s_0 and to be positioned over the leftmost nonblank symbol on the tape. If the tape is all blank, the control head can be positioned over any cell. We will call the positioning of the control head over the leftmost nonblank tape symbol the initial position of the machine. Example 254 illustrates how a Turing machine works.

Example 254. *What is the final tape when the Turing machine T defined by the seven five-tuples $(s_0, 0, s_0, 0, R)$, $(s_0, 1, s_1, 1, R)$, (s_0, B, s_3, B, R), $(s_1, 0, s_0, 0, R)$, $(s_1, 1, s_2, 0, L)(s_1, B, s_3, B, R)$, and $(s_2, 1, s_3, 0, R)$ is run on the tape shown in Figure 1.17(a)?*

Solution: *We start the operation with T in state s_0 and with T positioned over the leftmost non blank symbol on the tape. The first step, using the five-tuple $(s_0, 0, s_0, 0, R)$, reads the 0 in the leftmost non blank cell, stays in state s_0, writes a 0 in this cell, and moves one cell right. The second step, using the five-tuple $(s_0, 1, s_1, 1, R)$, reads the 1 in the current cell, enters state s_1, writes a 1 in this cell, and moves one cell right. The third step, using the five-tuple $(_s1, 0, s_0, 0, R)$, reads the 0 in the current cell, enters state s_0, writes a 0 in this cell, and moves one cell right. The fourth step, using the five-tuple $(s_0, 1, s_1, 1, R)$, reads the 1 in the current cell, enters state s_1, writes a 1 in this cell, and moves right one cell. The fifth step, using the five-tuple $(s_1, 1, s_2, 0, L)$, reads the 1 in the current cell, enters state s_2, writes a 0 in this cell, and moves left one cell. The sixth step, using the five-tuple $(s_2, 1, s_3, 0, R)$, reads the 1 in the current cell, enters the state s_3, writes a 0 in this cell, and moves right one cell. Finally, in the seventh step, the machine halts because there is no five-tuple beginning with the pair $(s_3, 0)$ in the description of the machine. The steps are shown in Figure 1.17.*

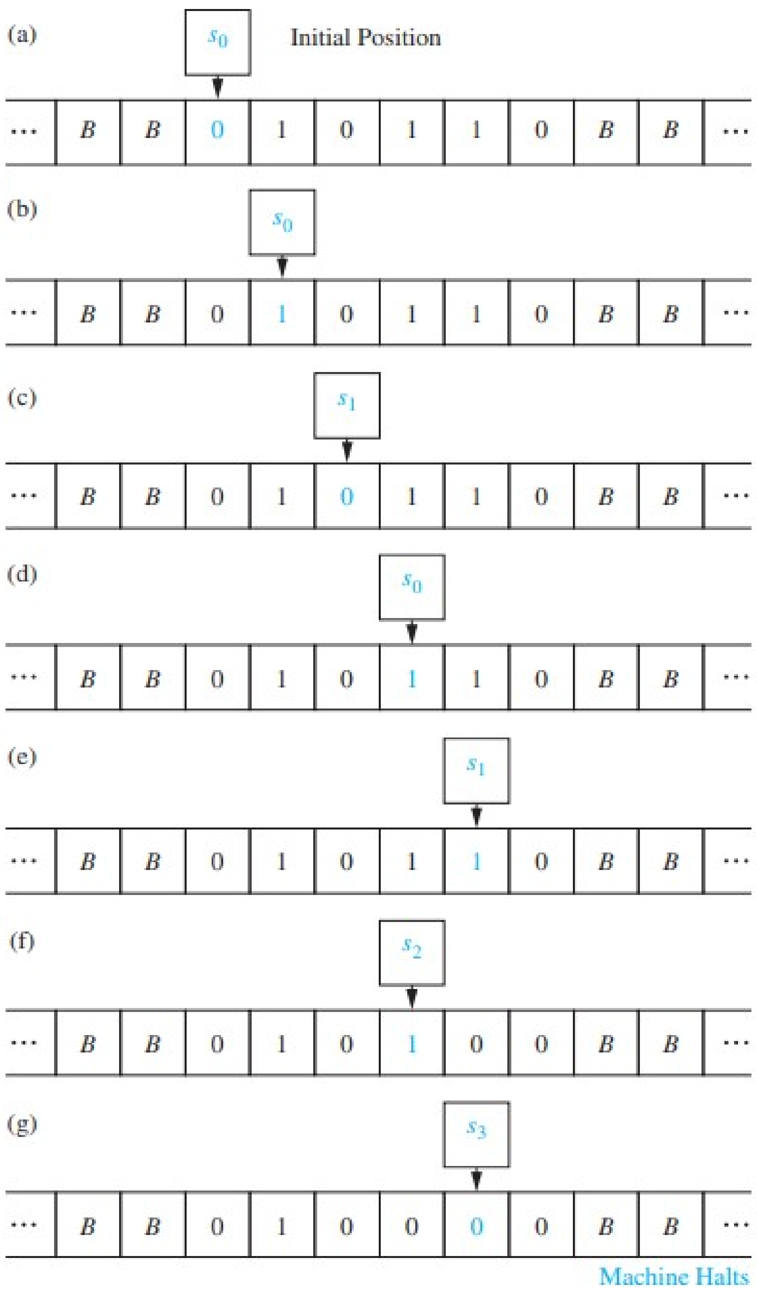


Figure 5.17: The Steps Produced by Running T on the Tape.

Note that T changes the first pair of consecutive 1s on the tape to 0s and then halts.

5.3.3 Using Turing Machines to Recognize Sets

Turing machines can be used to recognize sets. To do so requires that we define the concept of a final state as follows. A final state of a Turing machine T is a state that is not the first state in any five-tuple in the description of T using five-tuples.

We can now define what it means for a Turing machine to recognize a string. Given a string, we write consecutive symbols in this string in consecutive cells.

Definition 255. *Let V be a subset of an alphabet I. A Turing machine $T = (S, I, f, s_0)$ recognizes a string x in V^* if and only if T, starting in the initial position when x is written on the tape, halts in a final state. T is said to recognize a subset A of V^* if x is recognized by T if and only if x belongs to A.*

Example 256. *Find a Turing machine that recognizes the set of bit strings that have a 1 as their second bit, that is, the regular set* $(0 \cup 1)1(0 \cup 1)^*$.

Solution: *We want a Turing machine that, starting at the leftmost non blank tape cell, moves right, and determines whether the second symbol is a 1. If the second symbol is 1, the machine should move into a final state. If the second symbol is not a 1, the machine should not halt or it should halt in a nonfinal state.*

To construct such a machine, we include the five-tuples $(s_0, 0, s_1, 0, R)$ *and* $(s_0, 1, s_1, 1, R)$ *to read in the first symbol and put the Turing machine in state s1. Next, we include the five-tuples* $(s_1, 0, s_2, 0, R)$ *and* $(s_1, 1, s_3, 1, R)$ *to read in the second symbol and either move to state* s_2 *if this symbol is a 0, or to state* s_3 *if this symbol is a 1. We do not want to recognize strings that have a 0 as their second bit, so* s_2 *should not be a* *final state. We want s_3 to be a final state. So, we can include the five-tuple $(s_2, 0, s_2, 0, R)$. Because we do not want to recognize the empty string or a string with one bit, we also include the five-tuples $(s_0, B, s_2, 0, R)$ and $(s_1, B, s_2, 0, R)$.*

The Turing machine T consisting of the seven five-tuples listed here will terminate in the final state s_3 if and only if the bit string has at least two bits and the second bit of the input string is a 1. If the bit string contains fewer than two bits or if the second bit is not a 1, the machine will terminate in the nonfinal state s_2.

Example 257. *Find a Turing machine that recognizes the set $\{0^n1^n | n \geq 1\}$.*

Solution: *To build such a machine, we will use an auxiliary tape symbol M as a marker. We have $V = \{0, 1\}$ and $I = \{0, 1, M\}$. We wish to recognize only a subset of strings in V^*. We will have one final state, s_6. The Turing machine successively replaces a 0 at the leftmost position of the string with an M and a 1 at the rightmost position of the string with an M, sweeping back and forth, terminating in a final state if and only if the string consists of a block of 0s followed by a block of the same number of 1s.*

Although this is easy to describe and is easily carried out by a Turing machine, the machine we need to use is somewhat complicated. We use the marker M to keep track of the leftmost and rightmost symbols we have already examined. The five-tuples we use are $(s_0, 0, s_1, M, R)$, $(s_1, 0, s_1, 0, R)$, $(s_1, 1, s_1, 1, R)$, (s_1, M, s_2, M, L), (s_1, B, s_2, B, L), $(s_2, 1, s_3, M, L)$, $(s_3, 1, s_3, 1, L)$, $(s_3, 0, s_4, 0, L)$, (s_3, M, s_5, M, R), $(s_4, 0, s_4, 0, L)$, (s_4, M, s_0, M, R), and (s_5, M, s_6, M, R). For example, the string 000111 would successively become M00111, M0011M, MM011M, MM01MM, MMM1MM, MMM-

MMM as the machine operates until it halts. Only the changes are shown, as most steps leave the string unaltered.

We leave it to the reader to explain the actions of this Turing machine and to explain why it recognizes the set $\{0^n1^n | n \geq 1\}$.

5.3.4 Computing Functions with Turing Machines

Example 258. *Construct a Turing machine for adding two nonnegative integers.*

Solution: *We need to build a Turing machine T that computes the function* $f(n_1, n_2) = n_1 + n_2$*. The pair* (n_1, n_2) *is represented by a string of* $n_1 + 1$ *1s followed by an asterisk followed by* $n_2 + 1$ *1s. The machine* T *should take this as input and produce as output a tape with* $n_1 + n_2 + 1$ *1s. One way to do this is as follows. The machine starts at the leftmost 1 of the input string, and carries out steps to erase this 1, halting if* $n_1 = 0$ *so that there are no more 1s before the asterisk, replaces the asterisk with the leftmost remaining 1, and then halts. We can use these five-tuples to do this:* $(s_0, 1, s_1, B, R)$, $(s_1, *, s_3, B, R)$, $(s_1, 1, s_2, B, R)$, $(s_2, 1, s_2, 1, R)$, *and* $(s_2, *, s_3, 1, R)$.

5.3.5 Computational Complexity, Computability, and Decidability

Definition 259. *A decision problem asks whether statements from a particular class of statements are true. Decision problems are also known as yes-or-no problems.*

Definition 260. *The halting problem is the decision problem that asks whether a Turing machine* T *eventually halts when given an input string*

x.

Theorem 261. *The halting problem is an unsolvable decision problem. That is, no Turing machine exists that, when given an encoding of a Turing machine T and its input string x as input, can determine whether T eventually halts when started with x written on its tape.*

Other examples of unsolvable problems include:

(i) the problem of determining whether two context-free grammars generate the same set of strings;

(ii) the problem of determining whether a given set of tiles can be used with repetition allowed to cover the entire plane without overlap; and

(iii) Hilberts Tenth Problem, which asks whether there are integer solutions to a given polynomial equation with integer coefficients.

COMPUTABILITY *A function that can be computed by a Turing machine is called* **computable** *and a function that cannot be computed by a Turing machine is called uncomputable.*

Definition 262. *A decision problem is in P, the class of polynomial-time problems, if it can be solved by a deterministic Turing machine in polynomial time in terms of the size of its input. That is, a decision problem is in P if there is a deterministic Turing machine T that solves the decision problem and a polynomial $p(n)$ such that for all integers n, T halts in a final state after no more than $p(n)$ transitions whenever the input to T is a string of length n. A decision problem is in NP, the class of nondeterministic polynomial-time problems, if it can be solved by a nondeterministic Turing machine in polynomial time in terms of the size of its input. That* *is, a decision problem is in NP if there is a nondeterministic Turing machine T that solves the problem and a polynomial $p(n)$ such that for all integers n, T halts for every choice of transitions after no more than $p(n)$ transitions whenever the input to T is a string of length n.*

Problems in P are called **tractable**, *whereas problems not in P are called* **intractable**.

Let Us Sum Up

In this section, we discussed about

∗ *Definition of Turing Machine.*

∗ *Using Turing machines to recognize sets.*

∗ *Computing functions with Turing machines.*

∗ *Computational complexity.*

Check your Progress

1. A function is computable if it is

(a) a busy bearer function

(b) solvable by Boolean algebra

(c) computable by a Turing machine

(d) a decision problem

2. Problems not in p are called

(a) intractable

(b) tractable

(c) polynomial-time problems

(d) decision problems

Summary

In this unit, we discussed about

- *Finite-state machines with output.*
- *Types of finite-state machines.*
- *Finite-state automata.*
- *Myhill-Nerode Theorem.*
- *Minimization of DFSA.*
- *Non-deterministic finite-state automata.*
- *Turing machines.*
- *Computing functions with Turing machine.*
- *Computational complexity, computability and Decidability.*

Glossary

AB (concatenation of A and B: *the set of all strings formed by concatenating a string in A and a string in B in that order.*

A^* ***(Kleene closure of A):*** *the set of all strings made up by concatenating arbitrarily many strings from A.*

deterministic finite-state automaton (S, I, f, s_0, F)***:*** *a five-tuple containing a set S of states, an input letter I, a transition function f that assigns a next state to every pair of a state and an input, a starting state s_0, and a set of final states F.*

non deterministic finite-state automaton (S, I, f, s_0, F)***:*** *a five-tuple containing a set S of states, an input letter I, a transition function f that assigns a set of possible next states to every pair of state and an input, a starting state s_0, and a set of final states F.*

language recognized by an automaton: *the set of input strings that take the start state to a final state of the automaton.*

regular expression: *an expression defined recursively by specifying that $\varnothing, \lambda$, and x, for all x in the input alphabet, are regular expressions, and that (AB), $(A \cup B)$, and A^* re regular expressions when A and B are regular expressions.*

regular set: *a set defined by a regular expression.*

decision problem: *a problem that asks whether statements from a particular class of statements are true.*

solvable problem: *a problem with the property that there is an effective* *algorithm that can solve all instance of the problem.*

unsolvable problem: *a problem with the property that no effective algorithm exists that can solve all instance of the problem.*

computable function: *a function whose values can be computed using a Turing machine.*

uncomputable function: *a function whose values cannot be computed using a Turing machine.*

Self Assessment Questions

1. *What is a finite-state machine?*
2. *Construct a deterministic finite-state automaton that recognizes the set of bit strings that start with 1 and end with 1.*
3. *Find the Kleene closure of the set* $\{11, 0\}$.
4. *What does it mean for a string to be recognized by a finite-state automaton?*
5. *Show that given a nondeterministic finite-state automaton, there is a deterministic finite-state automaton that recognizes the same language.*

Exercises

1. *Construct a finite-state machine with output that produces an output of 1 if the bit string read so far as input contains four or more 1s. Then construct a deterministic finite-state automaton that recognizes this set.*

2. *Construct finite-state automata that recognize the sets* $0^*(10)^*$.

3. *Find regular expressions that represent the set of all strings of 0s and 1s, made up of blocks of even numbers of 1s interspersed with odd numbers of 0s.*

4. *Show that if* A *is regular set, then so is* $\bar{A}$.

5. *Find finite-state automata that recognize the set of all strings that start with no more than three consecutive 0s and contain at least two consecutive 1s.*

Answers to Check your Progress

Section 5.1:	*1)b*	*2)c*
Section 5.2:	*1)c*	*2)a*
Section 5.3:	*1)c*	*2)a*

Reference:

1. *Kenneth H. Rosen, "Discrete Mathematics and its Applications", 7th Edition, WCB/ McGraw Hill Publications, New Delhi, 2011.*

Suggested Readings:

1. *Edward A. Bender and S. Gill Williamson, "A Short Course in Discrete Mathematics", Dover Publications, 2006.*

2. *M.O. Albertson and J.P. Hutchinson, "Discrete Mathematics with Algorithms", John Wiley & Sons, 2008.*

3. *Rajendra Akerkar and Rupali Akarkar, "Discrete Mathematics", Pearson Education Pvt. Ltd, Singapore, 2004.*

4. *J. P. Trembley and R. Manohar, "Discrete Mathematical Structures", Tata McGraw Hill, New Delhi,1997.*

5. *Martin Aigner, "A Course in Enumeration", Springer-Verlag, Heidelberg, 2007.*

6. *J.H. Van Lint and R.M. Wilson, "A Course in Combinatorics", 2nd Edition, Cambridge University Press, Cambridge, 2001.*