The Language of Mathematics

Utilizing Math in Practice

Robert Laurence Baber

WILEY

THE LANGUAGE OF MATHEMATICS

THE LANGUAGE OF MATHEMATICS

Utilizing Math in Practice

Robert Laurence Baber
Bad Homburg vor der Höhe, Germany

WILEY

A JOHN WILEY & SONS, INC., PUBLICATION

Library of Congress Cataloging-in-Publication Data:

Baber, Robert Laurence.
The language of mathematics : utilizing math in practice / Robert L. Baber.
p. cm.
Includes index.
ISBN 978-0-470-87889-7 (hardback)
1. Mathematical notation. 2. Translating English to a mathematical model. I. Title.
QA41.B235 2011
510.1′4–dc22

2010054063

Printed in the United States of America

oBook ISBN: 9781118061770
ePDF ISBN: 9781118061718
ePub ISBN: 9781118061763

10 9 8 7 6 5 4 3 2 1

This book is dedicated

to those who would like to improve their ability
to apply mathematics effectively to practical problems,

to teachers of mathematics who would like to improve their ability
to convey a better understanding and appreciation of mathematics
to their students,

and

to those who are curious about
the linguistic nature and aspects of mathematics and its notation.

CONTENTS

LIST OF TABLES

PREFACE

We live today in a highly technological world built upon science and engineering. These, in turn, are based extensively on mathematics. It is not an exaggeration to state that mathematics is the language of engineering. Thus, to be able to understand science and engineering—and hence, the physical world in which we live—one must have at least a basic understanding of mathematics. This need will increase in time as the world in which we live becomes ever more technological in nature.

Unfortunately, too few people today have a sufficient understanding of mathematics to enable them to understand important technological topics. They are inadequately prepared to contribute substantially to resolving related issues, such as the safe employment of nuclear systems in our society, avoiding or resolving environmental problems, or structuring transportation systems (including vehicles, their energy, roadways, terminal facilities), and especially, to making trade-off decisions among the many aspects of such issues.

An important reason for this widespread lack of familiarity with mathematics and the disciplines based on mathematics is the way in which mathematics is typically introduced and taught. Many people are turned off mathematics early in their school experience. Although current teaching approaches are effective for a relatively small group of pupils already oriented to technical, mathematical, and scientific subjects, they fail to motivate the majority. They do not build on the prior knowledge and interests of the target group. They are typically too late in addressing the ultimate and nontechnical advantages of applying mathematics, doing so only after many students have already lost interest and have turned off their minds to mathematics. A primary goal of this book is to present a view of mathematics that can overcome these shortcomings.

In this book I present a new and unique way of looking at mathematics. In it, mathematics is viewed through the specialized language and notation that mathematicians have developed for communicating among themselves, for recording the results of their work, and perhaps most important, for reasoning and conducting the various analyses involved in their investigations. This view of mathematics differs significantly from that presented in the traditional works on mathematics available in the extensive mathematical literature. It also differs significantly from the ways in which mathematics is taught today. This book will improve and increase the reader's insight into mathematics and how to utilize it in practice.

No particular previous knowledge of mathematics by the reader is required. All readers will, of course, have encountered arithmetic and some mathematics in school, and whatever they remember correctly will make it easier for them to read and understand some of the consequences of the material and concepts presented in this book. Readers with a more extensive prior knowledge of mathematics will be able to read the book faster, but they will still find many ideas to be new and different from their previous views of mathematics and its language. They will find that the material in the book will help them to apply mathematics to practical problems more easily, efficiently, and effectively than they could have earlier.

The book is an introduction to how to apply mathematics to practical problems by translating English statements of a problem to be solved into the Language of Mathematics. We also study some fundamental aspects of mathematics via the language used in mathematics, but that is only a by-product of investigating the Language of Mathematics.

The first step in solving a problem stated in English with the help of mathematics is to reformulate the English text into appropriate mathematical expressions reflecting the essential aspects of the problem and the requirements that its solution must satisfy. Reformulating the English text into such mathematical expressions is often the hardest part of solving a problem. It is often presumed to be part of the mathematical task, but actually, it is a *translation* problem—a *language* problem. Omissions and errors in this step will often be discovered only later, when the final mathematical solution is found to be wrong or inadequate—or found to be a solution to a different problem. Only after a suitable mathematical formulation of the problem and its solution has been completed can one begin to apply mathematics itself to find the desired solution.

As with most large and complex bodies of knowledge made up of a number of different subdisciplines, mathematics can be viewed from many different standpoints and in many different ways. None of these views exclude the validity of the others; rather, they complement each other. Each view typically offers something that the others do not offer. The most appropriate view depends on the viewer's goals, interests, particular purpose at the time, background knowledge, experience, and many other factors. Any person will find it useful to view mathematics from a different viewpoint—and the more, the better. The more able one is to take advantage of many different views, the better one will understand a subject and be able to apply mathematics to it efficiently, effectively, and productively. The approach taken in this book consolidates many of these different viewpoints within a unifying umbrella of language. It builds a bridge between natural languages such as English and mathematics.

My own experience learning, utilizing, and teaching mathematics has led me to the conclusion that mathematics should be introduced by examining the basics of the Language of Mathematics. I believe that learning mathematics in this way will help—even enable—many people to understand mathematics who would otherwise be turned off the subject by the current and traditional approaches to learning mathematics. Unfortunately, there are many such people in today's world whose work would benefit through simple applications of mathematics. This conclusion is based on my experience learning mathematics, learning how to apply it to a variety of technical, business, and economic problems, utilizing it extensively in practice in these areas, as well as teaching certain areas of mathematics and how to apply them both to university students and to people working in various technical, business, and management positions.

This language-oriented approach will make mathematics more accessible to those who like language and languages, but who have until now avoided—even disliked—mathematics. In my experience as a pupil in primary school through to teaching university courses involving applying mathematics to various types of problems, I have repeatedly observed that students at all levels and people on the job have considerable trouble solving word problems using mathematics. They have as much, usually more trouble coping with translating the English statement of the problem into mathematical notation as they do with solving the resulting mathematical expressions for the answers desired—if they ever get that far. It is my considered opinion that this difficulty is due to an inappropriate approach to teaching this material. The normal teaching approach presents word problems within the context of mathematics and as mathematics problems. In reality, they are, as mentioned above, translation and language problems, not mathematics problems. The mathematics comes later, after the word problem has been translated into the Language of Mathematics.

I believe that presenting word problems as language problems will draw students' conscious attention to the real issues involved in applying mathematics and will make learning this material easier. It will give them a broader and deeper basic understanding of mathematics, link mathematics with their previous knowledge of language, and provide them with a better foundation upon which specific skills in applying mathematics can then be developed. Instead of learning mathematics as something different, new, and unrelated to their previous experience and knowledge, they will learn mathematics as an extension of their already accumulated experience with and knowledge of language.

Viewing mathematics, mathematical models and mathematical expressions from a language standpoint can, in my experience, facilitate communication between people with different areas of expertise working on specific problems to which mathematics is applied. A language viewpoint diverts attention away from explaining mathematics to the less mathematically literate experts working on a problem. Instead, it directs attention to the real need to translate between the language of the application domain and the mathematical model and expressions representing an application problem and its solution. Secondarily, it can help those working on and affected by the application to improve their ability to read, at least passively, the mathematical model and expressions.

I also believe that many people who already understand mathematics well will find the new view presented in this book beneficial and that conscious awareness of and familiarity with it will help them when applying mathematics to practical problems and when explaining mathematics to others. At least that was my experience after I began to consider, first subconsciously, then consciously, the linguistic aspects of mathematics and to view mathematics from the standpoint of the Language of Mathematics as presented in this book.

While the explicitly language-oriented view of mathematics presented in this book is atypical and new, the mathematical material itself is old, having been developed over five or more millenia. This development has been uneven and sporadic, with flurries of creative phases interspersed between longer intervals of slow or no improvement. In the last few centuries, the development of mathematics has tended to become somewhat more regular, continuous, and productive. Whereas some aspects of mathematics are millenia old (e.g., numbers and arithmetic operations on numbers), other important features have been introduced comparatively recently: variable names to represent numbers or other values, functions and functional notation, compact standard forms for writing mathematical expressions, and symbolic logic.

The idea of viewing mathematics (or a part thereof) as a language is not at all widespread, nor is it completely new. To the best of my knowledge, however, the particular approach taken in this book is new. Whereas other works nominally dealing with linguistic aspects of mathematics tend to view the topic from the standpoint of *mathematics,* this book quite intentionally views the Language of Mathematics from the opposite side: from the standpoint of *language*. Whereas other works tend to concentrate on defining and understanding mathematical concepts and terms in English, this book deals explicitly and extensively with translating English statements into the Language of Mathematics, pointing out grammatical clues useful as guidelines. Ways of modeling dynamic, temporal processes described in English using the static, tenseless Language of Mathematics are dealt with in this book. Also new in this book is the observation that all verbs implicit in expressions in the Language of Mathematics are stative in nature (timeless, tenseless verbs of state or being), a characteristic that has significant implications for translating from English to the Language of Mathematics. In particular, many sentences in English cannot be translated directly into the Language of Mathematics, but must first be substantially reformulated.

In composing the presentation of the Language of Mathematics in this book, I have followed an old, common, and very successful strategy for formulating a mathematical model to be used as the basis for solving a given problem:

- Generalize.
- Identify the essential aspects of the problem and the corresponding mathematical model.
- Simplify, retaining the essentials but eliminating nonessentials where helpful.

Nonessential details often confuse both a model's developers and its readers by distracting their attention from the essentials. Nonessential details also make a mathematical model larger, more complex, and therefore more complicated. The resulting structure is more difficult to understand and use than one including only the essential details would be.

In introductory articles, lectures, and so on, one often encounters an apology for mathematical formulas and a statement that the reader or listener does not really have to understand the formulas in detail, only generally what they are about, and even that not really seriously or deeply. In this book, the reader will find no such apology or excuse. Such false rationalization is like telling the audience attending a play by Shakespeare that they need listen only to the poetic, musical flow of the voices—that the actual meaning of the words is unimportant. In this book, the meaning of each mathematical expression (formula) is important; the meaning, not the poetic style, is the message. The style can help or hinder the reader to understand the meaning, but appreciating the style is not enough; the meaning must be understood. If you read a play by Shakespeare but do not understand the language used, you will not get the message. The same applies to expressions in the Language of Mathematics.

The Language of Mathematics has evolved to facilitate reasoning logically about things. It has been developed to make it easy to make exact, precise, unambiguous logical statements and to make it difficult—even impossible—to make vague, ambiguous statements. One should take advantage of these characteristics of the Language of Mathematics and use it, not English, when reasoning about things. Therefore, convert from English to the Language of Mathematics at as early a stage in the reasoning process as possible.

When asked about their motivation for writing a book, authors often state that they wrote the book that they would have liked to have read earlier themselves. That was definitely an important reason for my writing this book. I would have liked very much to have had a copy of it when I was in high school and during my early undergraduate years. It would have given me a view of mathematics and mathematical notation that would have helped me to learn mathematics better and faster and to understand it more thoroughly and deeply. It would not have replaced any of the other books from which I learned mathematics, but it would have been a very helpful adjunct and introduction to them. I hope that you find reading this book as useful and as enjoyable as I would have so many years ago, and as I did conceiving and writing it.

Acknowledgments

I would like to thank the many people who have contributed directly and indirectly to this book and to my ability to write it. They begin with my teachers in primary, secondary, and tertiary schools, especially those who taught me mathematics, languages, and related courses. Authors of the books and other works I have read over the years have also contributed unknowingly, indirectly, but surely. Many of my work colleagues, consulting clients, and other friends contributed by posing questions and helping me to answer them. Others, by sharing their ideas on many different subjects with me over the years, also contributed indirectly but importantly to the book. Most recently, the reviewers of partial drafts of this book contributed valuable suggestions and advice.

Over the last five or more millenia, numerous mathematicians have made many significant contributions, both large and small, to the development of mathematics, its notation, its language, and its practical application. Through their efforts they have guided the evolution of mathematics from very basic beginnings to its present state. That evolution started with counting, went through stages of arithmetic and reasoning about numerical quantities and measures, and led to reasoning about nonnumerical entities, qualities, properties, and attributes. This book is built upon the cumulative results of their work. As the author of this book, I owe a great debt to all of them.

Last but not least, I thank my many students. The best students, who easily grasp the concepts presented to them and then build upon them with only a little help, are always a joy to have. But the most valuable ones to any teacher are those students who ask at first seemingly simple questions and who have difficulty coming to grips with the material. Instructors who dismiss their questions and difficulties lightly not only fail to rise to the challenges of ~~teaching~~ helping them to learn, but also pass up many opportunities to develop the material further and to consolidate and structure it better. They pass up opportunities for a research paper or even a book such as this one.

ROBERT LAURENCE BABER

Bad Homburg, Germany
http://Language-of-Mathematics.eu
November 2010

PART A
Introductory Overview

1 Introduction

Welcome to mathematics and particularly to its language. You will find it to be a simple language, with only a little grammar and a limited vocabulary, but quite different from the other languages you know. Unlike natural languages such as English, its semantics are precisely defined and unambiguous. In particular, its complete lack of ambiguity enables exact reasoning, probably its greatest advantage. On the negative side, one cannot express such a wide variety of things in the Language of Mathematics as in English, and intentional vagueness, so important in English poetry and much prose, cannot be expressed directly. Nonetheless, it is often surprising what one can express in and with the help of the Language of Mathematics, especially when combined appropriately with English or some other natural language.

Vagueness cannot be expressed directly in the Language of Mathematics, but it can be modeled—precisely and unambiguously—with mathematics. Expressed differently, the Language of Mathematics enables one to make precise and unambiguous statements about vaguely determined things. Probability theory, statistics, and, more recently, fuzzy theory are the mathematical subdisciplines that enable one to talk and write about uncertainty and vagueness—but with precision and without ambiguity.

Although the Language of Mathematics is quite limited in the range of things that can be expressed in it directly, many things outside the Language of Mathematics can be related to mathematical objects as needed for specific applications. Thus, the Language of Mathematics is, in effect, a template language for such applications. Adapting it to the needs of a particular application extends its usefulness greatly and poses the main challenge in its application. This challenge is primarily linguistic, not mathematical, in nature. Helping the reader to meet this challenge is an important goal of this book and underlies essentially all of the material in it.

One of the limitations in the Language of Mathematics is the fact that the notion of time is absent from it completely. This fact is mentioned here, at the very beginning, because the lack of conscious awareness of it has led to many students becoming (and sometimes remaining) very confused without realizing this source of confusion. Time and dynamic processes can easily be and often are modeled mathematically, but this is part of the adaptation of the template Language of Mathematics to the particular application in question. How this can be done is covered in several places in the book, in particular in Section 7.5.

1.1 WHAT IS LANGUAGE?

A *language* is a medium for:

- Expressing or communicating:
  - Verbally or visually (e.g., in written form)
  - Facts, opinions, thoughts, ideas, feelings, desires, or commands
  - At one time or from one time to another
  - Between different people or from one person to her/himself at a different time
- Thinking
- Analyzing or reasoning

Every language employs abstract symbols—verbal, visual, and sometimes using other senses, such as touch—to represent things. In many natural languages, the visual form was developed to represent the verbal form, so that there is a close relationship between the spoken and written forms. Other languages, however, have developed spoken and written forms which are not directly related. Originally, their symbols were often pictorial in nature, albeit often rather abstractly. One can think of such a language as two distinct languages, a spoken language and a written language. In the case of Sumerian (the earliest known written language), written symbols (in cuneiform) represented what we think of today as words, so that there was no direct connection between the written and spoken forms of the language. Later, the cuneiform symbols were taken over by other languages (e.g., Akkadian) to represent syllables in the spoken language, establishing a direct connection between the spoken and written forms of the language. Still later, other languages introduced symbols for parts of a syllable, leading to the abstract symbols that we now call letters.

Mathematics exhibits the characteristics of a language described above. The range and distribution of topics communicated in natural languages such as English and those communicated in the Language of Mathematics differ in some significant ways, however. Feelings and emotions are rarely expressed in mathematical terms. Vague (i.e., imprecisely defined) terms are not permitted in the Language of Mathematics. Otherwise, all of the characteristics of a language mentioned above are found in the Language of Mathematics, albeit with different emphasis and importance.

Scientists and historians believe that language began by our distant ancestors communicating with one another via sounds made by using the vocal chords, the mouth, the lips, and the tongue (hence our term *language*, from *lingua*, Latin for "tongue"). This form of language was useful for communicating between individuals at one time and when they were physically close to one another. Sounds made in other ways (e.g., by drums) were used for communication over greater distances, but still between people at essentially one time. Visual signals of various kinds were also employed in much the same way.

Marks on bones apparently representing numbers are believed to be an early (perhaps the earliest) form of record keeping: communicating from one time to another. Gradually, this idea was extended to symbols for various things, ideas, concepts, and so on, leading in a long sequence of developmental steps to language as we know it today. It is noteworthy that even precursors to writing apparently included numbers, the basic objects of arithmetic, and hence of mathematics. The earliest forms of writing known to us today certainly included numbers. Thus, the development of languages included elements of mathematics from very early times.

Symbols and signs recorded physically in visually observable form constitute records that store information for later use. They are a major source of our knowledge of early civilizations. Their durability seriously limits our knowledge of those early civilizations. The most durable records known to date are clay tablets inscribed with cuneiform characters and inscriptions on stone monuments. Records of old societies that used less durable forms of writing have decomposed in the meantime and are no longer available. Those potentially interesting historical records are lost forever.

Recently, humans have begun to communicate with symbols they cannot observe visually but only with the help of technical equipment. The symbols are in the form of electrically and magnetically recorded analog signals and, still more recently, digital symbols. In some cases, these signals and symbols are direct representations of previous forms of human language. In other cases, they are not; rather, they represent new linguistic structures and forms.

Natural languages such as English, Chinese, and Arabic have evolved to enable people to communicate about all the kinds of things they encounter in everyday life. Therefore, the universes of discourse of natural languages overlap considerably. The Language of Mathematics has, however, evolved to fulfill quite different, very specific, and comparatively quite limited goals. It is a language dealing only with abstract things and concepts and having only a rather limited scope. Therefore, for the purposes of applications to real-world situations, the Language of Mathematics is not a finished language but, instead, a *template language*. When applying mathematics, the Language of Mathematics must be adapted for each application. This is done by specifying how the elements of the mathematical description are to be *interpreted* in the terminology of the application area. A new *interpretation* must normally be given for each application, or at least for each group of closely related applications.

1.2 WHAT IS MATHEMATICS?

The archaeological record suggests that mathematics probably originated with counting and measuring things and recording those quantities. Soon, however, people began to pose and answer questions about the quantities of the things counted or needed for some purpose; that is, they began to reason about quantities and to solve related problems. As early as about 4000 years ago, mathematics included the study of geometrical figures: in particular, of relationships between their parts and between numerical measures of their parts. Later, mathematicians turned their attention to ever more abstract things and concepts, including ones not necessarily of a numerical or geometrical nature.

The description of language at the beginning of Section 1.1. also applies to mathematics. The relative emphasis on communication on the one hand and on reasoning and analysis on the other hand is perhaps different, but there is much common ground. Some would say that mathematics itself, in the narrow sense, concentrates on concepts and techniques for reasoning and analyzing, and that mathematics is therefore not itself a language. However, mathematics does use extensively a particular language that has evolved to facilitate reasoning and analyzing. Such reasoning and analyzing is performed primarily by manipulating the symbols of mathematical language mechanistically, according to precise rules. The Language of Mathematics is also used extensively for expressing and communicating both over time and between people. It is also used for thinking.

What is mathematics today? Someone once answered that question with "What mathematicians do." That, of course, begs the question "What do mathematicians do?" Answered most succinctly, they reason logically about things—artificial, abstract things—not just about quantities, numbers, or numerical properties of various objects.

Many of those things, although artificial and abstract, are useful in modeling actual things in the real world; for example:

- Structures of buildings, dams, bridges, and other engineering artifacts
- Materials of all kinds and their properties
- Mechanical devices and equipment
- Machines, engines, and all kinds of energy conversion devices and systems
- Vehicles of all types: land, underwater, water surface, air, space
- Electrical circuits and systems composed of them
- Communication systems—wired and wireless—and their components
- Systems for cryptography
- Molecules, atoms, nuclei, and subatomic particles
- Chemical reactions and chemical reactors
- Systems for generating and distributing electrical power
- Nuclear decay and interaction processes and nuclear reactors
- Heating and cooling systems
- Computer software
- Prices in financial markets
- Sales and markets
- Order processing and billing systems
- Inventory control systems
- Various business assets and liabilities
- Data and information of all types, including names and addresses
- Social, economic, business and technical systems
- Relationships among objects, properties, and values of all (not just numerical) types
- Structural aspects of languages, natural and artificial

Such models better enable us to describe, to understand, and to predict things in the real world—to our considerable benefit.

It is important that the reader always be consciously aware that mathematics today consists of much more than numbers and arithmetic. Important as these are, they constitute only a part of mathematics. The main goal of mathematics is not to work with numbers but to reason about objects, properties, values, and so on, of all types. Mathematicians work mostly with relationships between these things. Mathematicians actually spend very little of their time calculating with numbers. They spend most of their time reasoning about abstract things. Logic is an important part of that work.

Different subdisciplines of mathematics have been created in the course of time. Numbers, counting, geometric figures, and quantitative analyses constituted the first subdisciplines. Among the more recent is logic. Unfortunately, especially for the novice learning mathematics, logic used its own terminology and symbols, and this distinction is still evident in the ways in which mathematical logic is often taught today. This leads many beginners to believe that logic is somehow fundamentally different from the other subdisciplines and that a different notation and point of view must be learned. The linguistic approach presented in this book integrates these views and notational schemes, so that the beginner need learn only one mathematical language. Although this integrating view is already present in some mathematical work, it is not really widespread yet, especially not in teaching mathematics.

1.3 WHY USE MATHEMATICS?

Among the several reasons for using mathematics in practice, the two most important are:

- To find a solution to a problem. The statement of a problem or the requirements that a solution must fulfill can often be transformed into the solution itself.
- To understand something better and more thoroughly: for example, to identify all possibilities that must be considered when defining a problem and solving it.

Examples are given in Chapter 2, in Section 6.13.2, and in Chapter 8.

The author and many others have found in the course of their work that mathematics frequently enables them to think effectively about and solve problems they could not have come to grips with in any other way. As long as a problem is expressed in English, one can reason about the problem and deduce its solution only when one constantly keeps the precise meaning of the words, phrases, and sentences consciously in mind. If the text is at all long, this becomes unworkable and very subject to error. It is likely that some important detail will be overlooked. After formulating the problem in mathematical language, the expressions can be transformed in ways reflecting and representing the reasoning about the corresponding English sentences. However, the expressions can be transformed mechanistically according to generally applicable rules without regard to the meaning of the expressions. This effectively reduces reasoning to transformations independent of the interpretation of the expressions being transformed, simplifying the process considerably and enabling much more complicated problems to be considered and solved. People who are specialists in transforming mathematical expressions but who are not specialists in the application area can find solutions. In this way, the mental work of reasoning can be largely reduced to the mechanistic manipulation of symbols. In the words of Edsger W. Dijkstra, a well-known computer scientist whose areas of special interest included mathematics and logic, one can and should "let the symbols do the work."

For the reasons cited above, one should convert from English to the Language of Mathematics at as early a stage as possible when reasoning about anything. Transforming the mathematically formulated statement of a problem into its solution sounds easy. Although it is, in principle, straightforward, it can be computationally intensive and tedious to do manually. For a great many applications, algorithms for solving the problem and computer programs for calculating the numerical solutions exist. Where such solutions do not already exist, mathematicians can often develop them.

The usefulness of the Language of Mathematics for the purposes listed above derives from its precision, the absence of ambiguity, and rules for transforming mathematical expressions into various equivalent forms while preserving meaning. These characteristics are unique to the Language of Mathematics. Natural languages, lacking these characteristics, are much less satisfactory and useful for the purposes noted above.

1.4 MATHEMATICS AND ITS LANGUAGE

In order to reason logically about things, mathematicians have developed a particular language with particular characteristics. That language—the Language of Mathematics—and other languages developed by human societies—such as English—are similar in some respects and different in some ways.

Distinguishing characteristics of the Language of Mathematics are its precision of expression and total lack of ambiguity. These characteristics make it particularly useful for exact reasoning. They also make it useful for specifying technical things. The Language of Mathematics is a language of *uninterpreted* expressions, which are described in Section 3.4. This does not imply that mathematical expressions are uninterpretable. They can be and often are interpreted when applying mathematics in the real world: when associating mathematical values, variables, and expressions with entities in the application area (see Chapters 6 and 7, especially Section 6.13).

Within the Language of Mathematics, however, expressions are never interpreted. When transforming mathematical expressions in order to reason or analyze, one should be very careful not to interpret them, as doing so takes one out of the Language of Mathematics and into English. This can result in the loss of precision and the introduction of ambiguity—the loss of the very reasons for using mathematics—without one being aware that the loss is occurring. Reasoning must be conducted only and strictly within the abstract world of uninterpreted expressions, applying only mathematically valid transformations to the expressions without interpreting them. In this way, mathematical expressions represent the ultimate form of abstraction of the logically essential aspects of a practical problem, containing only the logical relationships between its various aspects and without any inherent reference to the real world represented. The final results of the transformations representing reasoning are mathematical expressions representing solutions. The latter expressions are, of course, interpreted in order to implement the solution in the application domain.

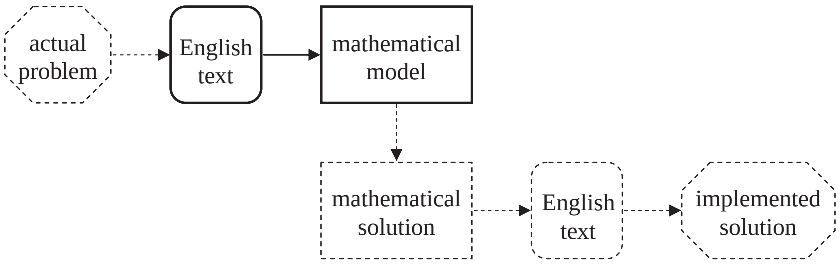
Mathematics and the Language of Mathematics are not the same thing. Facility with the language is a prerequisite for understanding and applying mathematics effectively. Unfortunately, mathematics is usually taught without explicitly introducing the language used. The student of mathematics is left to discover the language unassisted. Although this is possible for some people, it makes learning mathematics unnecessarily difficult and time consuming for many. For others, it constitutes the difference between learning mathematics and giving up before getting very far.

My experience learning, using, and teaching others mathematics and how to use it in practice has convinced me that looking at mathematics consciously as a language can facilitate the learning process, understanding, and the ability to apply mathematics in practice. I believe that it can even enable some people to learn how to use mathematics effectively who would otherwise be turned off mathematics completely by their early exposure to it—and unfortunately, there are many such people in today's world.

One must distinguish between the Language of Mathematics on the one hand and that part of English that is used to talk and write about mathematics on the other hand. The Language of Mathematics itself builds expressions upon values, variables, functions, and structures of these components. To communicate with other people about mathematics, one typically uses a combination of normal English and specialized mathematical terminology and jargon, just as is done in other specialized disciplines, such as the several scientific and engineering fields, medicine, and law. This distinction is discussed in greater depth in Section 6.10.

1.5 THE ROLE OF TRANSLATING ENGLISH TO MATHEMATICS IN APPLYING MATHEMATICS

The steps in the overall process of applying mathematics to a problem are illustrated in the following diagram, in which translating English to mathematics is highlighted.



Translating from an English description of a problem to the Language of Mathematics is the second step in the process of applying mathematics to a problem. The mathematical model is needed in order to reason logically about the problem, to analyze it systematically, accurately, and precisely, and to find a solution.

The mathematical model itself represents an interface between:

- The English language–oriented analysis and identification of the problem and the requirements for its solution, and
- The purely mathematical analysis and determination of one or more solutions

The mathematical model, being written in the Language of Mathematics, is an unambiguous statement of the problem and the requirements that any solution must satisfy. Its meaning in terms of the application is defined by the interpretation of the values, variables, and functions in the mathematical model, but its meaning in terms of the subsequent mathematical analysis is independent of that interpretation and the application. Thus, the mathematical model represents a boundary between the English language view of the application and the mathematical view of the application. The mathematical model connects, couples, the application and the mathematical worlds with each other, and at the same time it separates, insulates, isolates each from the other.

This, in turn, means that a solution can be determined in the mathematical world without regard to the application world, and correspondingly, any solution that satisfies the mathematical model will be applicable to the application world, without regard to how that solution was found. In the extreme, the specialists who find the mathematical solution do not really need to know anything about the application world to the left of the mathematical model in the diagram above. Correspondingly, the specialists in the application domain do not need to know or understand how the solution was found in the mathematical world below the mathematical model in the diagram at the beginning of this section.

Both specialist groups must, however, be able to read and understand the mathematical model (the interface specification) itself. Two factors are critical:

- That the application specialists agree that the mathematical model is an appropriate statement of the application problem and the requirements any solution must satisfy
- That the mathematical specialists agree that the mathematical model is a syntactically correct and semantically meaningful mathematical expression in the Language of Mathematics

That is the extent of the need for communication between the two groups of specialists. Lest the reader think that this is an unrealistic, utopian view, it must be pointed out that exactly this type of interface specification underlies all engineering work. Such interface specifications enable—and are prerequisites for—the division of labor required for the efficient and effective realization of any large-scale task, such as the design of a system for generating and distributing electricity regionally, nationally, or internationally; the design of a vehicle of any type (ship, automobile, truck, airplane, etc.); the design of a building; or the design of international telephone and communication systems.

Expressed differently but equivalently, communication and mutual understanding among the people involved is the key in such efforts. It is not necessary that every team member have the mathematical ability to solve all aspects of a problem or that every team member be an expert in all aspects of the application area. What is important is that they all understand what the problem is: what problem is being solved. The ability to read the expressions in a mathematical model and understand their meaning is sufficient; actually finding a solution can be left to specialists. That is one of the advantages of a mathematical formulation of the problem: Finding a solution depends only on the unambiguous mathematical expressions, not on what they are interpreted to mean in the application domain.

Although problems regarding accuracy, discrepancies, and errors can have their origins in any and all steps shown in the earlier diagram, particularly severe consequences arise from inaccuracies in translating the English text into the mathematical model. The especially important step of translating from English into the Language of Mathematics is the subject of this book.

1.6 THE LANGUAGE OF MATHEMATICS VS. MATHEMATICS VS. MATHEMATICAL MODELS

The Language of Mathematics, mathematical models, and mathematics are three different but closely related entities. The *Language of Mathematics* is the language of the notational forms used in mathematics. *Mathematical models* express relationships among the various variables, values, and functions that describe some part of the world to which mathematics is being applied. *Mathematics*, what one does in and with the Language of Mathematics, includes the notational forms, that is, the Language of Mathematics, and definitions of many different mathematical objects, techniques for transforming mathematical expressions and the proofs of their general validity, the theory underlying such techniques, and proofs of characteristics of the various mathematical objects.

A rough comparison will perhaps help to make these distinctions clearer. Corresponding to the Language of Mathematics, the English language can be thought of as the definitions of English words together with the grammar and accepted conventions for forming variations of the words (e.g., conjugating verbs, forming plurals and participles) and for combining words into sentences. Corresponding to mathematics, English in general can be considered to be the collection of the language itself together with what one does in and with the language, that is, the literature written (and spoken) in English and the associated culture. Corresponding to a mathematical model is an individual piece of English literature.

The distinction between the English language, English in general (i.e., together with its literature and culture), and particular pieces of English literature is commonly made in teaching and learning. The goals and contents of a course in English grammar are different from the goals and contents of a course in English literature and literary culture. The goals and contents of a course in an individual piece of literature, or in a collection of closely related literature (such as by one author), are different again. The approaches employed in such different types of courses are, correspondingly, different.

Unfortunately, the corresponding distinction between the Language of Mathematics, mathematical models, and mathematics is usually not made in teaching or learning any of these topics. The Language of Mathematics is, for the most part, treated implicitly and the student is, also implicitly, expected to absorb intuitively the linguistic aspects of the Language of Mathematics on his or her own. The nominal topics of courses are either mathematics or particular application areas. Courses on mathematics deal with specific subdisciplines of mathematics, such as differential calculus, integral calculus, linear algebra, real analysis, analytical geometry, and number theory. Definitions of mathematical objects relevant to the subdiscipline and techniques for manipulating expressions typically arising in the subdiscipline make up the content of those courses. Courses on particular application areas deal with phenomena in the application area in question and present the relevant mathematical models together with relevant aspects of mathematics. The mathematical models are presented as the relevant mathematics, not explicitly as models as such. Some examples of such application-oriented courses are physics, atomic physics, nuclear reactor physics, chemistry, mechanics (statics and dynamics), operations research, inventory control, electrical circuit theory, switching circuits, control theory, thermodynamics, heat transfer, and fluid mechanics. The mathematical flavor varies among such courses, but all emphasize the mathematical models relevant to the particular application area.

Notable in both the mathematics courses and the application-oriented courses are (1) that linguistic aspects of notation—the Language of Mathematics—are either absent or only implicit; and (2) that formulating new mathematical models (e.g., translating an English text into a mathematical model) is not dealt with.

The material in this book distinguishes consciously between these three topics: the Language of Mathematics, mathematical models, and mathematics. The reader should pay conscious attention to the distinction between them and to each individually. A major goal of the material in this book is to provide explicit guidelines for formulating new mathematical models based on descriptive English text.

1.7 GOALS AND INTENDED READERSHIP

By presenting the Language of Mathematics explicitly and systematically, this book is intended to help its readers to improve their ability to apply mathematics beneficially in their own work: in particular, by improving their ability to translate English descriptions into the Language of Mathematics. This book is not intended as a textbook on mathematics itself or on any subdiscipline of mathematics.

In summary, this book is written for the following people:

- Those who would like to improve their ability to apply mathematics effectively, systematically, and efficiently to practical problems
- Teachers of mathematics who would like to improve their ability to convey to their students a better understanding and appreciation of mathematics and how to apply it in practice
- Those who are curious about the linguistic nature and aspects of mathematical notation

More specifically, the intended readership includes:

- Engineers, consultants, managers, scientists, technicians, and others who could benefit vocationally and professionally by a greater ability to use and apply mathematics in their work
- Students in tertiary educational institutions
- Students in secondary schools especially interested in mathematics, science, or languages
- Educators designing mathematics curricula, course content, and teaching materials for students at all levels
- Teachers of mathematics, science, or languages in tertiary educational institutions (universities, polytechnics, and vocational and technical schools)
- Teachers of mathematics, science, or languages in secondary schools
- Teachers in primary schools who introduce pupils to mathematics and especially to word problems
- Persons with a general or an intellectual interest in mathematics, science, or language

The prerequisites for reading this book are a recognition and conscious awareness that mathematics might be useful in your work or other activities and a desire to realize its potential benefits. Basic knowledge of English grammar is also necessary; the essentials needed are summarized in Section 6.2. This book is self-contained in the sense that no particular mathematical background is assumed or needed.

Readers with an extensive mathematical background will find much of the mathematical notation presented in this book familiar. Their earlier mathematical courses will have given them the mathematical models needed for classical professional practice, but will not have taught them how to formulate mathematical models for new or significantly different types of problems themselves. Some, but not all, readers will have developed this ability intuitively and implicitly. This book will show all of them explicitly how to formulate new mathematical models based on English descriptions of problems to be analyzed and solved. Logical mathematical expressions will also be new to some readers with a mathematical background, especially to those whose mathematics concentrated on differential and integral calculus.

Readers with limited or no prior mathematical knowledge will find both the mathematical notation and the mathematics presented in this book largely new. Of particular importance to this group of readers is this book's goal of helping them to develop their ability to contribute actively to the translation of English descriptions of application requirements into the Language of Mathematics: that is, to formulate mathematical models. Part of this is helping them to become familiar with mathematical notation—with the Language of Mathematics itself.

This book is not about mathematics as a subject and is not intended to help you learn mathematics itself or any particular subdiscipline of mathematics. The book does not deal with the various topics typically covered in texts on mathematics. If you encounter mathematical topics in this book that you want to know more about, refer to an appropriate book on the relevant area of mathematics.

1.8 STRUCTURE OF THE BOOK

The **Preface** outlines the societal background and environment in which mathematics and its application are relevant and useful. It also describes the author's experience, observations, and thoughts leading to the decision to write the book and to the selection of its contents.

Part A, ***Introductory Overview*** (Chapters 1 and 2), deals with the subject of the book. Chapter 1 introduces the topics covered in the book: language, mathematics, reasons for applying mathematics to practical problems, the distinction between mathematics, its language (notational forms), and mathematical models. Chapter 1 also states the goals and outlines the intended readership. Guidelines for the reader are presented. Chapter 2 gives examples of the application of mathematics and mathematical models.

Part B, *Mathematics and Its Language* (Chapters 3, 4, and 5): An important purpose and goal of mathematical notation—the Language of Mathematics—is to enable ideas and concepts to be expressed unambiguously and to enable and encourage a corresponding way of thinking. In addition to mathematical notation, Part B presents a number of concepts that have been found in the course of time to be beneficial and important for many applications of mathematics and that have therefore become fundamental parts of mathematics. They are presented here because they are some of the reasons for the form and structure that the Language of Mathematics has acquired. Some acquaintance with these mathematical concepts is a prerequisite for understanding the nature of the Language of Mathematics and for acquiring even a passive knowledge of it.

In short, Part B is an introductory overview of those things that mathematicians and nonmathematicians who apply mathematics in their work often use, think, talk, and write about.

Part C*, English, the Language of Mathematics, and Translating Between Them* (Chapters 6, 7, and 8):

- Compares important characteristics of English and the Language of Mathematics
- Identifies their similarities and differences and the implications of their differences for translating
- Describes how to translate between English and the Language of Mathematics, giving extensive guidelines and illustrating this process with extensive examples.

Part D, *Conclusion* (Chapter 9), summarizes the main points developed in the book.

The **appendices** present various aspects of mathematics that some readers will find interesting and useful as additional background. **Appendix B** comprises a list of mathematical symbols used in the book, gives their meanings, and refers the reader to sections of the book describing them in detail. **Appendix G** is a **glossary** of English terms and their usual translations into the Language of Mathematics. The other appendices give additional information on numbers, selected structures in mathematics, mathematical logic, the mathematical treatment of waves, and programming languages in contrast with the Language of Mathematics. Finally, recommendations are given to the reader for finding works on selected subtopics among the vast literature on mathematics.

1.9 GUIDELINES FOR THE READER

Everything in this book is simple, and some of it is trivial. While reading, look for generality and simplicity—the simple things, structures, concepts—not complexity. Don't look for complicated things because you will not find them. If you expect them, you will be confused by their absence. If something looks complicated, you are reading complexity into the material where there is none. Read it again, looking for the simplicity. What you encounter may seem strange, unfamiliar, and lengthy, sometimes tedious, but it is not complicated.

Although every step in this book's development of the description of the Language of Mathematics is simple, there are many such steps. Especially in the mathematically oriented Chapters 3, 4, and 5 of Part B, try to understand the material in each step before proceeding to the next. Only partial understanding at one stage will usually be followed by a weaker understanding at the next stage, and your degree of understanding will lessen progressively. The material will then seem to be complicated.

On the other hand, it is sometimes useful to skip over material you do not at first understand, read other parts of the book, and return to that material later. This strategy is particularly useful to newcomers to material in any book who are looking for challenging material to stretch themselves and to widen their horizons. It is also useful to newcomers who seek only selected subtopics on their first reading.

Some readers will find some, perhaps much, of the material in the book to be intuitive. That intuitive, unconscious knowledge will be transferred into conscious, explicit knowledge. Experience shows that people can apply knowledge more effectively, more extensively, and to more complicated problems when they are explicitly and consciously aware of that knowledge than when that knowledge is only intuitive.

While and after reading the book, you will find it helpful to refer to standard books on mathematics and particular subdisciplines for more specific information on particular areas of mathematics. Select books and articles on those mathematical topics of relevance to the applications in which you are interested.

The reader already familiar with mathematics and its application will find some material in the book to be old and familiar, especially the contents of Chapter 3. These readers should scan those parts of the book, however, as some of the material is presented in new, different, and unconventional ways.

Readers will find that the topic of the book is presented in ways quite different from traditional mathematics teaching, and some might therefore question its validity. The very point of the book is that mathematics can be viewed from different standpoints and in different ways and that some of these approaches are absent from traditional mathematics teaching and learning. Those missing approaches can be useful, even critical, for some people. Starting by viewing the linguistic aspects of mathematics is the most important of these approaches.

Nothing in the book is mathematically incorrect, at least not intentionally so. If you do find an error, please let me know so that I can correct it.

Mathematics and the Language of Mathematics are like literature and the language in which that literature is written. If you don't understand the language, you will not understand the literature written in that language. Similarly, if you are not familiar with the Language of Mathematics, you will not get very far with your study of mathematics. If you try to study the literature (e.g., mathematics) anyway, without first learning its language—as newcomers and students of mathematics are too often forced today to do—your progress will at best be unnecessarily slow and frustrating. You will have to learn the language implicitly as you try to study the subject. The result will be that it will take you unnecessarily long to learn either the language or the subject, your knowledge of both will be incomplete, and the level of knowledge of both that you will attain will be unnecessarily limited. So begin your study of mathematics by examining explicitly its language, and you will find that you will be able to proceed faster and go farther in mathematics than you would otherwise be able to do.

At this point it is worthwhile to consider how people learn their first language (i.e., their native language) and how they learn subsequent languages. "Native" ability in a person's first language is acquired initially from the parents, especially the mother, who does not try so much to teach the child the language but simply to use it to communicate with the child. Later, in play and in school, this native ability is developed further by communicating with other speakers of the language, especially peers. Later, formal instruction complements and refines the native ability already acquired. When learning a second language, this learning process is usually reversed: First, basic aspects of the new language are learned in formal instruction. Then the learners develop their capability and fluency further by communicating in the language: the more, the better. This suggests that the best way to develop a thorough knowledge of and ability in the Language of Mathematics is to learn the basics and then, as soon as possible, to use it to communicate, both with others and with yourself, as much as possible.

Given serious motivation and a reasonable amount of time to spend on learning, most people find that they can learn a new language faster than they learned their first language. (A question for the reader to ponder: How many years elapsed before you were really fluent in your native language?) However, developing a truly native ability in a subsequent language can take much longer than developing fluency. In fact, many people who develop complete fluency in a foreign language never develop truly native ability, that intuitive feeling for colloquial, formal, and common usage patterns and many other aspects of the language: for those aspects of the language never covered in school. Knowing a language and living with it, living in it, embracing it, are not the same things.

Whenever one learns a new language (e.g., French), one must read, write, and talk about something. That something is typically taken from the daily lives of the speakers of the language in question. So, when learning French, one will read about France, typical cultural aspects of France and its people, its politics, life in France, situations arising there, its history, its literature, and so on. When learning the Language of Mathematics, one similarly must read, write, and talk about things about which mathematicians and people who apply mathematics in practice typically read, write, and talk: the culture of the field of mathematics, objects defined and used in mathematics and in its typical application areas, and so on. Therefore, this book presents both the language and some aspects of the subject of mathematics, hand in hand. The language is the initial and guiding factor. Mathematics itself is presented here only in conjunction with linguistic aspects of the Language of Mathematics and insofar as is necessary or helpful to proceed with an examination of the language. After completing the book, the reader will be in a much better position to read traditional mathematical books and articles and thereby learn more mathematics.

The reader who wishes to learn this material in order to be able to apply it actively and effectively to practical problems should examine the passages in the book thoroughly, filling in the sketches of proofs presented here. On the other hand, the reader who wishes only to become passively aware of the ideas and concepts presented may believe them uncritically, reading them somewhat cursorily and noting only the main points developed. Practice in reading mathematical expressions and models is still required.

The book may be read in various ways. Depending on the interests and prior knowledge of the reader, different balances between scanning cursorily, reading, and studying the several parts, chapters, and sections are appropriate.

It is recommended that the reader begin by scanning or reading the Preface and Part A, Introductory Overview, consisting of Chapters 1 and 2. While perusing the examples in Chapter 2, the reader should keep in mind the questions posed at the beginning of the Chapter.

Chapters 3 through 7 provide the theoretical basis for the Language of Mathematics and for translating from English to the Language of Mathematics. Sections 3.1, 3.2, and 3.3 present the foundations of the Language of Mathematics and are therefore required reading. At least one subsection of Section 3.4 on expressions should be read and understood and the others at least scanned.

Chapter 8, with its many examples, can be viewed as “practice.” When reading each of the longer examples, scan it first, noting only a few details of each kind, to gain an overview of the overall structure of the problem and of its solution. Then reread the example in more detail. If you do not first gain an overview of a lengthy example, there is a danger that you will become bogged down in the middle of a long series of details and lose sight of the path through which you are being led. This can lead to confusion and frustration. First acquiring and then maintaining a good overview of the structure of the example will enable you to maintain your orientation on your way from the beginning through to the end of the example.

Whether one reads the theory or the practice first, or skips between them while reading, is a matter of personal choice; pick the sequence that you feel enables you to make the best progress. Ultimately, both are necessary to acquire full understanding and satisfactory application skills, so your choice is not theory or practice, it is the sequence in which you cover both. It has long been recognized that in the practically oriented professions such as medicine, law, and engineering, a good balance between both theory and practice, with a thorough, well-developed knowledge of both, is necessary for success. (The Roman architect and engineer Vitruvius recognized the need for a balanced and thorough knowledge of both theory and practice over 2000 years ago.) If either theory or practice is lacking, the results will be unsatisfactory. Furthermore, a little knowledge and skill can get one into trouble, while more extensive knowledge and ability are required to get one out. Worse yet, with inadequate knowledge, one will often not even recognize that one is in trouble and how and why one got there.

Finally, Chapter 9, the summary, should be read—again if the reader read it earlier.

The appendices are included for reference and, if desired, additional insight into certain selected topics.

2 Preview: Some Statements in English and the Language of Mathematics

In each section of this chapter an English statement of a problem or a description of a situation is given, followed by a translation into the Language of Mathematics. No explanation of how the translation was made is given; in most cases, the statements and mathematical expressions are sufficiently simple that the correspondence between the English statements and the mathematical expressions is clear. Translating longer and more complex English text is covered in other chapters of the book.

This chapter serves only as a brief introduction to the Language of Mathematics and a summary of selected, simple parts of it. Characteristics of the Language of Mathematics, its notational forms, its grammar, manipulation permitted, and transformation of mathematical expressions are covered in detail in subsequent chapters.

When reading the examples in this chapter, the reader should determine the meaning of the English text and notice how that meaning is expressed in the mathematical expressions. While doing so, try to answer the following questions:

- Which parts of the English text correspond to which parts of the mathematical expressions?
- What are the grammatical forms of those corresponding parts of the English text and of the mathematical expressions: for example, nouns, verbs, noun phrases, adjectives, adverbs, prepositional phrases, and clauses in the English text, and variables, values, and functions (e.g., $+$, $-$, $*$, $/$, $\wedge$, $\Rightarrow$, $<$) in the mathematical expressions?
- What information is contained in the mathematical model but not in the English text? Where did it come from?
- What information is contained in the English text but not in the mathematical model? Why is it missing from the mathematical model?
- How are the meanings of the nouns and noun phrases in the English statements expressed in the mathematical expressions?
- How are the meanings of the verbs in the English statements expressed in the mathematical expressions?

- Are different types of verbs in the English text handled differently in the mathematical expressions? If so, what are those different categories of verbs in English, and how is each type expressed in the Language of Mathematics?
- Can you identify a relationship between the grammatical categories of the words and phrases in the English text and the corresponding parts of the mathematical expressions?
- How are tenses of verbs in the English text expressed in the mathematical expressions? How is time handled in general in the mathematical expressions?

In summary, consider the grammatical structure of the English text and notice the ways in which the meaning implied by that grammatical structure is written in the mathematical expressions. Read the mathematical expressions both in the context of the preceding English text and separately, without any context whatsoever. Note how your interpretation of the mathematical expressions is influenced by the context or a complete absence thereof.

More generally, note how the meaning of English texts is written in the Language of Mathematics.

Note also that some statements in the English texts are translated into mathematical expressions and some are not. Identify which are and which are not translated, and why.

Identify also what, if any, "common knowledge" is expressed in the mathematical model but not stated explicitly in the English text.

Note also that in examples below the dimensions of certain quantities (e.g., watts, watt-hours, barrels per day) are converted. Although such conversions are absolutely critical for accuracy in applying mathematics to practical problems, they are of secondary importance from a language standpoint. This topic is covered in more detail in Section 6.13.1.

Finally, the reader should describe the meaning of every variable in the mathematical models below in terms of the application area in question (e.g., "the length of the canal in meters"). Then, after reading the rest of the book, review those descriptions and revise them appropriately.

The following examples have been devised to illustrate various types of applications of mathematics. Although the numbers in them have been selected to be realistic and within the ranges in the relevant published research literature, any reader needing corresponding numbers for serious analyses or research should refer to the published research literature. This comment applies especially to the examples in Sections 2.2, 2.7, and 2.13.

2.1 AN ANCIENT PROBLEM: PLANNING THE DIGGING OF A CANAL

As long as several thousand years ago it was necessary to plan construction projects of various kinds in order that adequate resources could be made available and supported logistically when needed and for the length of time required. One simple example involves determining the number of workers needed and the number of days they were needed to dig a canal of given dimensions.

This arithmetic calculation can be expressed in today's mathematical notation as follows:

$$\text{NWorkers} * \text{NDays} = (\text{Depth} * \text{Width} * \text{Length})/\text{VolumePerWorkerDay} \quad [2.1\text{-}1]$$

If only a certain number of workers are available, the number of days needed to complete the task can be calculated using the formula

$$\text{NDays} = (\text{Depth} * \text{Width} * \text{Length})/(\text{VolumePerWorkerDay} * \text{NWorkers}) \quad [2.1\text{-}2]$$

If the task must be completed within a certain number of days, the number of workers needed to complete the task can be calculated using the formula

$$\text{NWorkers} = (\text{Depth} * \text{Width} * \text{Length})/(\text{VolumePerWorkerDay} * \text{NDays}) \quad [2.1\text{-}3]$$

Similar calculations were required to calculate the storage capacity of grain silos, the number of clay bricks needed to build a wall, the labor needed to produce the bricks, the amount of material needed to build a pyramid, and so on. Records of such calculations are contained on ancient Mesopotamian clay tablets in cuneiform script and in ancient Egyptian records. In those records, the calculations were described in words and numbers, not with variables and mathematical expressions as we do today, but the substance of the descriptions was the same. The computational steps in the calculations were described and listed in a form comparable to short computer programs of today.

2.2 THE WALL AROUND THE ANCIENT CITY OF URUK

According to the Epic of Gilgamesh, probably the world's oldest known written literary work, the wall around the ancient Sumerian city of Uruk was built by Gilgamesh, one of its rulers. Archaeologists have dated the wall to the third millenium B.C.E. Its size is not known accurately, and estimates vary considerably. It was, apparently, between 6 and 11 km long, between 5 and 15 m high, and between 3.5 and 6 m wide.

We want to estimate the manual effort required to mold the bricks for this wall. The bricks were made of mud or clay and measured about $10 \times 15 \times 40$ cm. We will assume that the actual average dimensions of the bricks were within 20% of these estimates. The bricks were made by laying a mold on the ground, filling the mold with muddy material, leveling the top by hand, and then lifting off the mold to let the brick dry in the sun. This process was then repeated for the next brick. We will assume that one worker could mold one brick in this way in between 0.5 and 1.5 minutes and that, allowing for breaks for meals and rest, a worker actually worked between 7 and 9 hours a day. What was the manual effort in worker-days required to mold the bricks for the wall of Uruk?

All of the quantities mentioned above may be real numbers, that is, have whole and/or fractional parts, except the number of bricks, which must be an integer.

Because of the uncertainty of each of the several factors in this problem, one cannot calculate precisely the effort required. One can, however, calculate the range of effort required given the ranges of uncertainty of the several factors involved. One can often draw interesting and useful conclusions from the lower and upper bounds calculated for the values of the quantities in question.

Many variables (LWm, HWm, WWm, LBcm, HBcm, WBcm, VolWcubm, VolBcubm, WMperB, HperD, NB, WD) are used in the mathematical model below. To what in the English text does each refer or correspond? In the English text, which references are explicit and which are implicit? Why were those variables with only implicit references to the statement of the problem introduced?

Translated into the Language of Mathematics, the statements above become

$$\begin{array}{l}
\mathrm{LWm}{\in}\mathbb{R} \wedge \mathrm{HWm}{\in}\mathbb{R} \wedge \mathrm{WWm}{\in}\mathbb{R} \wedge \mathrm{HBcm}{\in}\mathbb{R} \wedge \mathrm{WBcm}{\in}\mathbb{R} \wedge \\
\quad \mathrm{LBcm}{\in}\mathbb{R} \wedge \\
\mathrm{VolWcubm}{\in}\mathbb{R} \wedge \mathrm{VolBcubm}{\in}\mathbb{R} \wedge \mathrm{WMperB}{\in}\mathbb{R} \wedge \mathrm{HperD}{\in}\mathbb{R} \wedge \\
\quad \mathrm{NB}{\in}\mathbb{Z} \wedge \mathrm{WD}{\in}\mathbb{R} \wedge \\
6000 \le \mathrm{LWm} \le 11000 \wedge 5 \le \mathrm{HWm} \le 15 \wedge 3.5 \le \mathrm{WWm} \le 6 \wedge \\
10*0.8 \le \mathrm{HBcm} \le 10*1.2 \wedge 15*0.8 \le \mathrm{WBcm} \le 15*1.2 \wedge \\
\quad 40*0.8 \le \mathrm{LBcm} \le 40*1.2 \wedge \\
\mathrm{VolWcubm} = \mathrm{LWm}*\mathrm{HWm}*\mathrm{WWm} \wedge \\
\mathrm{VolBcubm} = (\mathrm{HBcm}*\mathrm{WBcm}*\mathrm{LBcm})/1{,}000{,}000 \wedge \\
0.5 \le \mathrm{WMperB} \le 1.5 \wedge 7 \le \mathrm{HperD} \le 9 \wedge \\
\mathrm{NB} = \mathrm{VolWcubm}/\mathrm{VolBcubm} \wedge \\
\mathrm{WD} = \mathrm{NB}*\mathrm{WMperB}/(60*\mathrm{HperD})
\end{array} \qquad [2.2\text{-}1]$$

The expression above implies that

$$\begin{array}{l}
6000*5*3.5 \le \mathrm{VolWcubm} \le 11{,}000*15*6 \wedge \\
8*12*32/1{,}000{,}000 \le \mathrm{VolBcubm} \le 12*18*48/1{,}000{,}000 \wedge \\
6000*5*3.5/(12*18*48/1{,}000{,}000) \le \mathrm{NB} \le \\
\quad 11{,}000*15*6/(8*12*32/1{,}000{,}000) \wedge \\
10{,}127{,}315 \le \mathrm{NB} \le 322{,}265{,}625 \wedge \\
10{,}127{,}315*0.5/(60*9) \le \mathrm{WD} \le 322{,}265{,}625*1.5/(60*7) \wedge \\
9377.14 \le \mathrm{WD} \le 1{,}150{,}948.66
\end{array} \qquad [2.2\text{-}2]$$

The analysis above brackets the number of worker-days required between about 9000 and 1,000,000, a very wide range. This example illustrates how the uncertainties of several factors become compounded, leading to a greater uncertainty of derived quantities: in this case, the number of bricks required and the manual effort needed to mold them. The lower and upper bounds can still be useful and of interest, even when their difference is very large. Perhaps more importantly, their difference clearly indicates the limits of our knowledge based on the estimates available. The numerical analysis also indicates where better estimates of the input factors are needed if we want to calculate more precise estimates of the quantities of interest.

If, later, additional information is obtained, the analysis above can be extended and the range of the number of worker-days needed reduced. For example, the discovery that between 500 and 550 bricks could be or were produced in one day by one worker would enable us to extend the mathematical model above by the following:

$$\begin{aligned}&500 \le \text{BperWD} \le 550 \wedge\\ &\text{BperWD} = \text{NB/WD} \wedge\\ &500 \le \text{NB/WD} \le 550 \wedge\\ &10{,}127{,}315/550 \le \text{WD} \le 322{,}265{,}625/500\end{aligned} \qquad [2.2\text{-}3]$$

Thus, the additional information about the productivity in bricks per worker-day has enabled us to reduce the range of worker-days needed to between 18,413.3 and 644,531.25, still a wide range but much narrower than the first estimate above.

This additional information might enable us to reduce the uncertainty of our knowledge of the length of the working day in hours (the variable HperD) and the number of minutes that a worker needed to produce each brick (the variable WMperB). The relevant parts of the mathematical model above and the new information lead to the following additional extension of the mathematical model:

$$\begin{aligned}&0.5 \le \text{WMperB} \le 1.5 \wedge 7 \le \text{HperD} \le 9 \wedge\\ &500 \le \text{BperWD} \le 550 \wedge\\ &\text{BperWD} = (60{*}\text{HperD})/\text{WMperB} \wedge\\ &500/60 \le \text{HperD/WMperB} \le 550/60 \wedge\\ &500/120 \le \text{HperD} \le 550/40 \wedge\\ &420/550 \le \text{WMperB} \le 540/500 \wedge\\ &7 \le \text{HperD} \le 9 \wedge\\ &0.76 \le \text{WMperB} \le 1.08\end{aligned} \qquad [2.2\text{-}4]$$

Thus, the additional information has also enabled us to reduce the range of worker-minutes per brick produced. It has not enabled us to reduce the range of working hours per day; that is, it has not given us any additional knowledge of the working hours in a day.

2.3 A NUMERICAL THOUGHT PUZZLE

Once upon a time, a baseball bat cost \$1 more than a ball. A bat and a ball together cost \$1.10. How much did each cost?

Translated into the Language of Mathematics, these statements become

$$\text{PrBat} = 1 + \text{PrBall} \wedge \text{PrBat} + \text{PrBall} = 1.1 \qquad [2.3\text{-}1]$$

The reader should compare her/his immediate intuitive answer to the question above with the correct answer derived by manipulating the mathematical expression above into the form

$$\text{PrBall} = \text{a number} \wedge \text{PrBat} = \text{a number}$$

Substituting the expression $1 + \text{PrBall}$ from the first equation in expression 2.3-1 for PrBat in the second equation in expression 2.3-1 leads to

$$1 + \text{PrBall} + \text{PrBall} = 1.1 \qquad [2.3\text{-}2]$$

which is the same as

$$1 + 2{*}\text{PrBall} = 1.1 \qquad [2.3\text{-}3]$$

which is equivalent to

$$\text{PrBall} = (1.1-1)/2 \qquad [2.3\text{-}4]$$

which is equivalent to

$$\text{PrBall} = 0.05 \qquad [2.3\text{-}5]$$

Substituting 0.05 for PrBall in the first equation in expression 2.3-1 gives the rest of the solution:

$$\text{PrBat} = 1.05 \qquad [2.3\text{-}6]$$

2.4 A NURSERY RHYME

An old nursery rhyme poses a problem most systematically and reliably solved by first translating it into the Language of Mathematics and then manipulating the resulting expression appropriately. The well-known nursery rhyme is:

As I was going to St. Ives,
I met a man with seven wives;
Every wife had seven sacks,
Every sack had seven cats,
Every cat had seven kits.
Kits, cats, sacks, and wives,
How many were going to St. Ives?

Translated into the Language of Mathematics, the relevant lines of the rhyme become

$$\begin{aligned}
&\text{NWives} = 7 \wedge \\
&\text{NSacks} = 7{*}\text{NWives} \wedge \\
&\text{NCats} = 7{*}\text{NSacks} \wedge \\
&\text{NKits} = 7{*}\text{NCats} \wedge \\
&\text{NGoingToStIves} = \text{NKits}+\text{NCats}+\text{NSacks}+\text{NWives} \qquad [2.4\text{-}1]
\end{aligned}$$

The expression above can be transformed into the equivalent expression

$$\begin{aligned}&\text{NWives} = 7 \wedge\\ &\text{NSacks} = 49 \wedge\\ &\text{NCats} = 343 \wedge\\ &\text{NKits} = 2401 \wedge\\ &\text{NGoingToStIves} = 2800\end{aligned} \qquad [2.4\text{-}2]$$

the last line of which gives the desired answer to one of several different interpretations of the question.

One should note that the question refers to English language text that is ambiguous in several ways. Major ambiguity starts in the second line with the second word, “met.” Did the storyteller (“I”) meet the man with seven wives as a fellow traveler, as a traveler going in some other way, or as someone staying at a stopping point? Did the storyteller also meet the seven wives? Were the man’s seven wives with him or not? Thus, it is not clearly stated whether the man mentioned in the second line was or was not going to St. Ives (i.e., is or is not to be counted). The next-to-last line of the rhyme suggests that the man is not to be included in the count, but the first two lines could be interpreted to imply that the man was also going to St. Ives. The same comment applies to the “I” mentioned in the first line. Increasing the count of people and things going to St. Ives to include the man or the storyteller would increase the answer to 2801 or 2802.

Furthermore, the statements in the first five lines of the rhyme do not state explicitly that the wives, sacks, cats, and kits are actually going to St. Ives, only that the storyteller met the man with the wives, and so on, while he (the storyteller) was going to St. Ives. This interpretation leads to the answer 1 (the storyteller only). Probably many people will interpret the next-to-last line to imply that the wives, and the others, are going to St. Ives, but this is not, in fact, clearly stated.

Thus, the process of translating the English rhyme into the Language of Mathematics can help in identifying ambiguities in the original English text. In practical problems, such ambiguities arise often, more often than is sometimes realized. They are most reliably identified by reading and analyzing the English text repeatedly, pedantically, and critically. Then, they must be resolved explicitly before completing the translation into the Language of Mathematics and then proceeding to calculate the answer desired.

The main message in this example is important: Before translating an English text into the Language of Mathematics, identify the ambiguities in the English text. If you do not find any, look again and again until you do find some. Then look for still more. Some ambiguities are present in almost every English text.

2.5 MAKING A POT OF TEA

To make a pot of tea, an old rule of thumb says to put between 1 and 2 teaspoons of tea for each cup to be made, plus “one for the pot,” into a pot and add hot water for the number of cups of tea desired. The exact number of teaspoons of tea per cup depends on the particular kind of tea and the desired strength of the final product. Beforehand, one should warm the pot by putting hot water into it and let it stand for some time.

The quantity of tea and water required can be expressed by the mathematical expression

$$\begin{aligned}&1\leq\text{TspTeaPerCup}\leq 2 \wedge \text{NTspTea} = \text{TspTeaPerCup}*\text{NCups} + 1 \wedge\\ &\quad\text{NCupsWater} = \text{NCups}\end{aligned} \qquad [2.5\text{-}1]$$

This example illustrates a simple translation of an English statement into the Language of Mathematics. It does not represent a problem to be solved or a question to be answered. It could be part of a larger example containing a question to be answered, but it need not be. Like text in any other language, mathematical expressions in the Language of Mathematics can represent anything: a simple statement, a question to be answered, a problem to be solved, a specification of a mechanism or system to be designed, or something else.

2.6 COMBINING DATA FILES

Jane and George, two teenage friends in secondary school, helped older people in their town to use and maintain their various computer systems. Jane and George each kept a file of data on his/her own customers. They did not compete with each other, so had no customer in common. When Jane left to attend a university, she turned over her business to George, who merged the two files and sorted the combined file, which became his new customer file.

As a brief homework problem, George wrote the following mathematical expression to describe the relationship between the two original files and his new, combined file. In it, he modeled each file as a sequence of data items, one item for each customer. He modeled the combined file as the concatenation of the two files. His new file was a sorted permutation of the combined file.

$$\begin{array}{ll}
\text{NGCustNew} = \text{NGCustOrg} + \text{NJCustOrg} \ \wedge & [2.6\text{-}1]\\
(\ \ ([\&\ i : i{\in}\mathbb{Z} \wedge 1 \leq i \leq \text{NGCustOrg} : [\text{GCustOrg}(i)]] & [2.6\text{-}2]\\
\quad \&[\&\ i : i{\in}\mathbb{Z} \wedge 1 \leq i \leq \text{NJCustOrg} : [\text{JCustOrg}(i)]]) & [2.6\text{-}3]\\
\quad \text{Perm} & [2.6\text{-}4]\\
\quad ([\&\ i : i{\in}\mathbb{Z} \wedge 1 \leq i \leq \text{NGCustNew} : [\text{GCustNew}(i)]])\ \ )\ \wedge & [2.6\text{-}5]\\
[\wedge\ i : i{\in}\mathbb{Z} \wedge 1 \leq i < \text{NGCustNew} : \text{GCustNew}(i) < \text{GCustNew}(i{+}1)] & \\
 & [2.6\text{-}6]
\end{array}$$

The reader should interpret each numbered line in the mathematical expression above in terms of the English description. To which part of the English text does each line or group of lines in the mathematical expression correspond?

2.7 SELECTING A TELEPHONE TARIFF

Two telephone tariffs are offered to a subscriber. One couples a low monthly fixed cost with a high usage cost. The other is just the opposite: a high monthly fixed cost with a low usage cost. Depending on the subscriber's monthly usage, one will be cheaper than the other. What criteria should the subscriber use in selecting the less expensive tariff?

We define the several variables as follows:

t:	an identification of the tariff, either 1 or 2
MFC(t):	the monthly fixed cost for tariff t
VUC(t):	the variable usage cost per minute for tariff t
MU:	the subscriber's average monthly usage in minutes
Opt:	the identification number of the optimum tariff or 0 if the costs are the same

One of the several equivalent mathematical expressions defining the value of Opt given values for the other variables is as follows:

$$\begin{aligned}&[(\mathrm{MFC}(1)+\mathrm{MU}*\mathrm{VUC}(1)) < (\mathrm{MFC}(2)+\mathrm{MU}*\mathrm{VUC}(2)) \Rightarrow \mathrm{Opt}=1] \wedge\\ &[(\mathrm{MFC}(1)+\mathrm{MU}*\mathrm{VUC}(1)) > (\mathrm{MFC}(2)+\mathrm{MU}*\mathrm{VUC}(2)) \Rightarrow \mathrm{Opt}=2] \wedge\\ &[(\mathrm{MFC}(1)+\mathrm{MU}*\mathrm{VUC}(1)) = (\mathrm{MFC}(2)+\mathrm{MU}*\mathrm{VUC}(2)) \Rightarrow \mathrm{Opt}=0] \qquad [2.7\text{-}1]\end{aligned}$$

Another equivalent expression is

$$\begin{aligned}&[(\mathrm{MFC}(1)+\mathrm{MU}*\mathrm{VUC}(1)) < (\mathrm{MFC}(2)+\mathrm{MU}*\mathrm{VUC}(2)) \wedge \mathrm{Opt}=1] \vee\\ &[(\mathrm{MFC}(1)+\mathrm{MU}*\mathrm{VUC}(1)) > (\mathrm{MFC}(2)+\mathrm{MU}*\mathrm{VUC}(2)) \wedge \mathrm{Opt}=2] \vee\\ &[(\mathrm{MFC}(1)+\mathrm{MU}*\mathrm{VUC}(1)) = (\mathrm{MFC}(2)+\mathrm{MU}*\mathrm{VUC}(2)) \wedge \mathrm{Opt}=0] \qquad [2.7\text{-}2]\end{aligned}$$

A shorter and more easily readable expression can be written if one defines the function MCost(t) (monthly cost for tariff t) to be the value of the expression MFC(t)+MU∗VUC(t). The first expression above can then be rewritten as

$$\begin{aligned}&[\mathrm{MCost}(1) < \mathrm{MCost}(2) \Rightarrow \mathrm{Opt}=1] \wedge\\ &[\mathrm{MCost}(1) > \mathrm{MCost}(2) \Rightarrow \mathrm{Opt}=2] \wedge\\ &[\mathrm{MCost}(1) = \mathrm{MCost}(2) \Rightarrow \mathrm{Opt}=0] \qquad [2.7\text{-}3]\end{aligned}$$

Any of the three expressions above serves to define a value of the variable Opt. If that value is 0, the two tariffs result in the same monthly cost and it does not matter which tariff the subscriber selects. Otherwise, the value of Opt is the identification number of the cheaper tariff.

To determine the value of the variable Opt, substitute the values of the variables MFC(1), MFC(2), VUC(1), VUC(2), and MU in any of the three expressions above and simplify the expression as much as possible. The result will be an equation of the form Opt=…, which gives the value of the variable Opt and, hence, the optimum tariff.

2.8 INTEREST ON SAVINGS ACCOUNTS, BONDS, ETC.

An investor deposits \$1000 at 5% interest annually. What is the value of this investment after 1 year? 2 years? n years?

The value in dollars of the investment after 1 year is $1000+1000*5/100$, which is equal to $1000*(1+5/100)$, or 1050.

The value in dollars of the investment after 2 years is:

- $1000*(1+2*5/100)$ (i.e., 1100) if the interest is “simple” interest
- $1000*(1+5/100)*(1+5/100)$, which is equal to $1000*(1+5/100)^2$ (i.e., 1102.5) if the interest is compounded annually

More generally, the value of an initial investment of V after n years at an interest rate of r% per year is:

- $V*(1+n*r/100)$ if the interest is “simple” interest
- $V*(1+r/100)^n$ if the interest is compounded annually

After how many years will the value of the investment double at an interest rate of r% per year compounded annually? The final value will be double the initial value if

$$2*V = V*(1+r/100)^n \qquad [2.8\text{-}1]$$

that is, if

$$2 = (1+r/100)^n \qquad [2.8\text{-}2]$$

or, equivalently, if

$$\log(2) = n*\log(1+r/100) \qquad [2.8\text{-}3]$$

which, in turn, will be the case if

$$n = \log(2)/\log(1+r/100) \qquad [2.8\text{-}4]$$

If $r = 100\%$ per year, the initial investment will, of course, double after 1 year. If $r = 15\%$ per year, the initial investment will double after 4.96 years. If $r = 5\%$ per year, the initial investment will double after 14.21 years.

2.9 SALES AND VALUE-ADDED TAX ON SALES OF GOODS AND SERVICES

In many places a tax is imposed on sales of goods and services. It is sometimes called a *sales tax* and sometimes, a *value-added tax*. (The economic difference between these two types of taxes is not relevant for the purposes of this example.) In both cases, the amount of the tax is a fraction of the base price of the goods or services being sold and purchased. The tax rate is usually expressed as a percentage. The price to be paid by the purchaser to the seller is the sum of the base price and the tax.

In the description of the tax above, the tax rate and three monetary amounts are mentioned: the base price, the tax, and the total price to be paid. Therefore, the variables in our mathematical model are:

TaxRate:	the percentage of the base price imposed as the tax
BasePrice:	the price of the goods and services, exclusive of the tax
Tax:	the monetary amount of the tax
TotalPrice:	the price of the goods and services, including the tax

Mathematical formulas are required that give each monetary amount in terms of the tax rate and any other single monetary amount. Also needed are formulas giving the tax rate in terms of any two of the monetary amounts.

From the description in the first paragraph in this section, the mathematical model can be written

$$\text{Tax} = \text{BasePrice}*\text{TaxRate}/100 \wedge \text{TotalPrice} = \text{BasePrice} + \text{Tax} \qquad [2.9\text{-}1]$$

The formulas requested in the statement of the problem but not in the mathematical model above follow from it:

$$\text{TotalPrice} = \text{BasePrice}*(1 + \text{TaxRate}/100) \qquad [2.9\text{-}2]$$

$$\text{BasePrice} = \text{Tax}*100/\text{TaxRate} \qquad [2.9\text{-}3]$$

$$\text{BasePrice} = \text{TotalPrice}/(1 + \text{TaxRate}/100) \qquad [2.9\text{-}4]$$

$$\text{TotalPrice} = \text{Tax}*(1 + 100/\text{TaxRate}) \qquad [2.9\text{-}5]$$

$$\text{Tax} = \text{TotalPrice}/(1 + 100/\text{TaxRate}) \qquad [2.9\text{-}6]$$

$$\text{TaxRate} = 100/(\text{TotalPrice}/\text{Tax} - 1) \qquad [2.9\text{-}7]$$

$$\text{TaxRate} = 100*(\text{TotalPrice}/\text{BasePrice} - 1) \qquad [2.9\text{-}8]$$

$$\text{TaxRate} = 100*\text{Tax}/\text{BasePrice} \qquad [2.9\text{-}9]$$

This completes the collection of formulas required in the problem statement.

The value of any of the four variables in this model can be expressed in terms of any two of the other three variables. Such equations not already listed above are

$$\text{BasePrice} = \text{TotalPrice} - \text{Tax} \qquad [2.9\text{-}10]$$

$$\text{Tax} = \text{TotalPrice} - \text{BasePrice} \qquad [2.9\text{-}11]$$

Expression 2.9-1 reflects the definitions in the English text and are the most commonly used relationships in calculating the tax on sales. In practice, equations 2.9-2 and 2.9-4 are convenient for converting base prices into total prices, and vice versa. Equation 2.9-6 is used to calculate the tax given the total price. The other equations are used occasionally as the need arises.

2.10 A HAND OF CARDS

During a card game, one player's hand contains the following cards: the nine of hearts, the two of clubs, the king of diamonds, the jack of spades, the three of clubs, the two of spades, the three of diamonds, the queen of spades, the ace of diamonds, and the five of diamonds.

This player's hand can be modeled by the mathematical expression

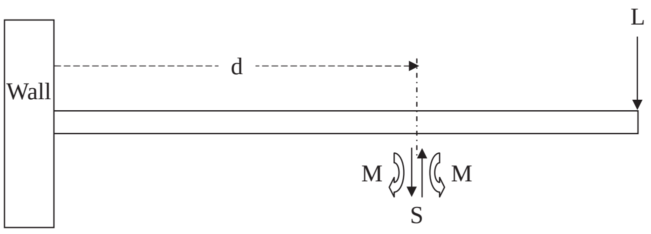
$$\begin{aligned}\text{hand}=\{&(9, \heartsuit), (2, \clubsuit), (\text{K}, \diamondsuit), (\text{J}, \spadesuit), (3, \clubsuit), (2, \spadesuit), (3, \diamondsuit), (\text{Q}, \spadesuit),\\ &(\text{A}, \diamondsuit), (5, \diamondsuit)\}\end{aligned} \tag{2.10-1}$$

The hand is modeled as a set of elements, each of which represents a card. Each card is represented by an ordered pair of the value and the suit.

This mathematical model of a player's hand of cards could be, for example, part of a model of the rules for playing a card within a larger model of a card game being played by several players (see Section 8.8).

2.11 SHEAR AND MOMENT IN A BEAM

In this example, one end of a beam is firmly fixed in a wall and a load is applied to the other end of the beam. A mathematical model is needed that gives the shear force and the moment at any location in the beam, to determine whether or not the beam will support the given load. For the sake of simplicity of this example, the weight of the beam is neglected. The diagram below illustrates the beam, its support, and its load.



The several variables and their interpretation are as follows:

Length:	the length in meters of the beam from the wall to its other end
L:	the load in kilograms applied to the end of the beam
S:	the shear force in kilograms in the beam at a location d meters from the wall
M:	the moment in kg-m in the beam at a location d meters from the wall
d:	at d meters from the wall the shear and the moment in the beam are S and M, respectively

Additional Implicit Information Fundamental laws of physics and mechanics, in particular Newton's laws, tell us that if a body is stationary, the sum of the forces applied to it must be zero and the sum of the torques applied to it must be zero. Applying this principle to that part of the beam to the right of the point d meters from the wall in the foregoing diagram, the load of L kilograms downward must be offset by an upward force of L kilograms applied to the right part of the beam by the left part. Correspondingly, the force applied to the left part of the beam by the right part is also L kilograms, but downward. Thus, a shear force of L kilograms is present at the cross section of the beam d meters from the wall. At this same point, the torque resulting from the load L kilograms at the end of the beam is L∗(Length−d) kg-m clockwise. This torque must be offset by a torque of the same amount but counterclockwise exerted on the right part of the beam by the left part of the beam. Thus, a moment of L∗(Length−d) kg-m is present at the cross section of the beam d meters from the wall, in the direction shown in the diagram.

The mathematical model relating the values of the variables above is therefore

$$0 \le d \le \text{Length} \wedge M = L * (\text{Length} - d) \wedge S = L \qquad [2.11\text{-}1]$$

From this model it can be seen that the moment M has its maximum value when d = 0 (i.e., at the point in the beam at the wall). If L = 100 kg and Length = 2 m, the maximum moment M in the beam is 200 kg-m. If the strength of the beam is not at least this much, the beam will break under the load. The shear S is a constant 100 kg at all locations in the beam. Similarly, if the beam's shear strength is not at least this much, the beam will break under the load L. Correspondingly, the attachment of the beam to the wall must be strong enough to withstand a shear (vertical load) of at least 100 kg and a moment of 200 kg-m; otherwise, the attachment will break under the load L at the end of the beam.

Other configurations of beams and loads can be analyzed in a similar manner. Of interest to do-it-yourself hobbyists is to determine where a bookshelf (effectively, a beam as above) should be supported at two locations along the shelf to minimize the maximum moment M in the shelf uniformly loaded along its length by books. This will minimize the bending of the shelf in the long term, because bending is proportional to the moment M. (*Hint*: Supporting the bookshelf at its extremities does not minimize the bending.)

This example illustrates the translation into the Language of Mathematics of an English text accompanied by a diagram. Such combinations of English text and diagrams arise often in engineering work. This example is typical of topics and problems covered in a course in statics taken by students in all disciplines of engineering early in their studies. Such examples arise so often in engineering studies that students can easily overlook the fact that their mathematical models are really nothing other than translations of English text accompanied by diagrams describing physical mechanisms or systems.

2.12 FORMING ABBREVIATIONS OF NAMES

In this example, a mathematically precise definition is formulated for a dynamic process to form the abbreviation of any given name. Such abbreviations are sometimes used in systems in which a possibly misspelled version of a name is to be matched with the correctly spelled name (e.g., airline reservation systems, processing handwritten orders from customers). If an effective method for forming the abbreviations is selected, misspellings of a name will usually have the same abbreviation as the correctly spelled name, whereas misspellings of a different name will rarely have the same abbreviation.

The process for forming the abbreviation of a name is modeled as a finite state machine (automaton).

Names are to be abbreviated as shown in the table that follows. In a sequence of vowels, only the first is to be considered. In a sequence of consonants, only the first is to be considered. The letters are to be converted as shown.

Replace:	by:
a, e, i, y	a
o, u	o
b, p	b
m, n	m
c, k, q	q
d, t	t
g, j	g
f, v, w	f
h	ignore as if not present
l, r	r
s, x, z	s

We will view the method for forming the abbreviation as a dynamic process in which each letter of the input name is examined, one after the other. Depending on each input letter, the process will, if and as appropriate, output (append) a letter to the abbreviation being formed and change state. The state will, in effect, keep track of whether a consonant, a vowel, or nothing has most recently been input.

From the description above of the abbreviation rules, a table can be constructed that gives, for every combination of current state and input letter, the appropriate output letter (if any) and the next state. Each column represents a group of input letters as defined in the table. Each row represents a current state. In each cell, the output letter and the next state are given [e.g., (x, 1)]. A dash (–) indicates that no letter is output (i.e., the empty sequence [] is appended to the output abbreviation formed previously).

Notice how Table 2.12-1 specifies the logic for all combinations of the several different cases of h at the beginning of the name, h after the beginning of the name, more than one vowel in a sequence, more than one consonant in a sequence, an h

TABLE 2.12-1 State Transition Table for Forming Abbreviations

	Input Letter										
Current State	a, e, i, y	o, u	b, p	m, n	c, k, q	d, t	g, j	f, v, w	h	l, r	s, x, z
0 initial	(a, 1)	(o, 1)	(b, 2)	(m, 2)	(q, 2)	(t, 2)	(g, 2)	(f, 2)	(–, 0)	(r, 2)	(s, 2)
1 after vowel	(–, 1)	(–, 1)	(b, 2)	(m, 2)	(q, 2)	(t, 2)	(g, 2)	(f, 2)	(–, 1)	(r, 2)	(s, 2)
2 after consonant	(a, 1)	(o, 1)	(–, 2)	(–, 2)	(–, 2)	(–, 2)	(–, 2)	(–, 2)	(–, 2)	(–, 2)	(–, 2)

within a vowel sequence, an h within a consonant sequence, an h between a vowel sequence and a consonant sequence, and so on.

The mathematical model is

$$
\begin{aligned}
&\mathbb{InputLetters} = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z\}\\
&\wedge\ \mathbb{Outputs} = \{[a], [o], [b], [m], [q], [t], [g], [f], [r], [s], []\}\\
&\wedge\ \mathbb{States} = \{0, 1, 2\}\\
&\wedge\ (\text{NextState} : \mathbb{InputLetters} \times \mathbb{States} \rightarrow \mathbb{States})\\
&\wedge\ (\text{NextOutput} : \mathbb{InputLetters} \times \mathbb{States} \rightarrow \mathbb{Outputs})\\
&\wedge\ [\wedge\ n : n{\in}\mathbb{Z} \wedge 1{\le}n{\le}\text{length(name)} : \text{input}(n){\in}\mathbb{InputLetters} \wedge \text{output}(n){\in}\mathbb{Outputs}]\\
&\wedge\ \text{state}(0){=}0\\
&\wedge\ [\wedge\ n : n{\in}\mathbb{Z} \wedge 0{\le}n < \text{length(name)} : \text{output}(n{+}1){=}\text{NextOutput}(\text{input}(n{+}1), \text{state}(n))\\
&\qquad\qquad\qquad\qquad\qquad\qquad \wedge\ \text{state}(n{+}1){=}\text{NextState}(\text{input}(n{+}1), \text{state}(n))]\\
&\wedge\ \text{name} = [\text{input}(1), \text{input}(2), \text{input}(3), \ldots, \text{input}(\text{length(name)})]\\
&\wedge\ \text{StateHistory} = [\text{state}(0), \text{state}(1), \text{state}(2), \ldots, \text{state}(\text{length(name)})]\\
&\wedge\ \text{abbreviation} = \text{output}(1)\ \&\ \text{output}(2)\ \&\ \ldots\ \&\ \text{output}(\text{length(name)}) \qquad [2.12\text{-}1]
\end{aligned}
$$

The expression 2.12-1 defines the values of the variables abbreviation and StateHistory given the value of the variable name. The functions NextState and NextOutput are defined by Table 2.12-1. The value of the function length is the number of letters in the argument of the function.

There is some redundancy in the mathematical model 2.12-1. The reader should identify which terms are redundant and could be deleted without changing the value of the entire expression.

This type of mathematical model is often used to specify a process for verifying the correctness of the syntax of many kinds of sequences of symbols, such as mathematical expressions as defined in Section 3.4, file names in computer systems, and statements in computer programs to be compiled.

2.13 THE ENERGY IN EARTH'S REFLECTED SUNLIGHT VS. THAT IN EXTRACTED CRUDE OIL

The sunlight arriving at Earth brings with it about $1.6 * 10^{17}$ watts of power, of which about 30% is reflected or scattered back into space. The rest is absorbed by the surface or the atmosphere—and mostly reradiated. Worldwide crude oil extraction averages about $70 * 10^6$ barrels per day. Burning the products of 1 barrel of crude oil releases approximately $1.7 * 10^6$ watt-hours of energy. (Estimates of the preceding numerical values vary significantly.) Therefore, the amount of solar energy reflected and scattered back into space is more than 9500 times as much as the amount of energy in crude oil extracted from underground reserves.

The truth of the last sentence above can be demonstrated by first translating the preceding paragraph into the Language of Mathematics, paying appropriate attention to the dimensions of the various numbers, giving

$$\begin{aligned}&\text{PowerArrivingWatts} = 1.6 * 10^{17} \wedge\\ &\text{Albedo} = 0.3 \wedge\\ &\text{PowerReturningWatts} = \text{PowerArrivingWatts}*\text{Albedo} \wedge\\ &\text{OilExtrBblperDay} = 70 * 10^6 \wedge\\ &\text{WHperBbl} = 1.7 * 10^6 \wedge\\ &\text{HperDay} = 24 \wedge\\ &\text{PowerOilExtrWatts} = (\text{OilExtrBblperDay} * \text{WHperBbl})/\text{HperDay}\\ &\Rightarrow \text{PowerReturningWatts} > 9500 * \text{PowerOilExtrWatts}\end{aligned} \tag{2.13-1}$$

The reader should verify that the value of the entire expression above is true: that the numbers given imply the truth of the last line in the mathematical model above, which says the same thing as the last sentence in the English statement.

The mathematical analysis above shows that the rate at which solar energy is being reflected and scattered back into space is quantitatively much, much more than enough to replace all the energy currently derived from extracted crude oil. The reader should contemplate what this implies. How can we take advantage of this fact by employing solar energy directly instead of burning fuels derived from crude oil? Into what forms of energy must the solar energy be converted? How can it be converted into those forms, stored until needed, and distributed to the places where needed? The Language of Mathematics is also useful in the analyses involved in answering these questions. The tasks involved are many and very challenging, but with serious, concerted effort and very considerable engineering and technical talent, they are feasible.

PART B
Mathematics and Its Language

3 Elements of the Language of Mathematics

The fundamental elements in expressions written in the Language of Mathematics are:

- *Values*: basic constants, arbitrary components not requiring further mathematical definition
- *Variables*: abstract representations of values, often unknown or undetermined
- *Functions*: ways of determining a value from other values
- *Expressions*: combinations of values, variables, and functions

3.1 VALUES

Values are the basic building blocks in the Language of Mathematics. Mathematical expressions and structures of all types are made up of values, references to values, and compositions thereof. Common examples of values are such numbers as 4, 89, −3, 1.414, 83/99, 0, and 1. Numbers used for counting were apparently the first values to be used in mathematics, so values were often called *quantities*, and this term is still used frequently.

Other values introduced in early mathematics are geometrical figures such as points, lines, triangles, rectangles, circles, ellipses, and polyhedrons, in addition to and apart from numerical measurements characterizing them, such as position, orientation, length, area, and volume.

In the course of time, still other values became important in mathematical work. Among them are the logical values *false* and *true* (also called *Boolean values*), letters (e.g., a, b, c, …), special symbols (e.g., ;, :, #, ∗, /, ?, +, %, …), sequences of letters and symbols (e.g., to form names and addresses, or words in a language), and any other values convenient and appropriate for any particular application of mathematics (e.g., colors such as red and green; states such as open, closed, rising, and falling).

Values are not restricted to numbers or any other particular category of things. A person applying mathematics to any problem area may define additional values if

The Language of Mathematics: Utilizing Math in Practice, First Edition. Robert Laurence Baber.

desired. The only restriction on a value is that either it cannot be subdivided further or else every one of its subcomponents is a value. Examples of various types of values and of structures of values (values composed of other values) are presented in several sections of the book.

The term *constant* is often used in mathematical writing. It usually means a value, but it can also refer to a variable whose value remains unchanged, either always or over some period of time or over a number of applications of the mathematical model containing references to the constants in question. In this book, the term *constant* alone will not be defined mathematically and will be used only infrequently and in a normal English sense. It will sometimes be used in combination with other words to refer to particular categories of constants, such as *physical constants* (e.g., the speed of light in a vacuum), *mathematical constants* [e.g., pi, also written π (Greek lowercase letter pi), the ratio of the circumference to the diameter of a circle], or *chemical constants*.

3.2 VARIABLES

Often, in the course of mathematical analysis or reasoning, one cannot or does not need to refer to particular values. This can arise, for example, if a particular value is not yet known but is to be determined. In other cases, a mathematical property may apply to many different values, so one specifically wants to avoid referring to a particular value so that the results of the analysis are more generally applicable.

Therefore, a name is often substituted for a value in a mathematical analysis, and the pair consisting of the name and the (possibly unknown or undetermined) value is called a *variable*. Thus, a variable is the association between a name and a value. One should always be careful to distinguish among three different things:

- The *name* of the variable
- The *value* of the variable
- The *variable* itself: that is, the pair (name, value), the association of the name and the value

An example is (x, 3.2). The variable is the pair (x, 3.2). The name of this variable is x. The value of this variable is 3.2.

Another example is (y, ?). The variable is the pair (y, ?). The name of this variable is y. The value of this variable is undetermined; it is unknown.

Numerical values are usually associated with particular dimensions or units, depending upon their interpretation in the application domain. For example, a number representing the length of a side of a triangle could have the dimension of centimeters, kilometers, inches, feet, yards, miles, furlongs, fathoms, and so on. Numerical terms in expressions must be consistent with respect to such dimensions if the values of the expressions are to have any meaning in the application area. Therefore, the concept of a variable could be defined mathematically to include the dimension associated with a numerical variable (i.e., the term *variable* could be the association of a name, a value, and the dimension of that value). More commonly, however, the dimension of each numerical variable is specified in the interpretation of the mathematical model and its variables (i.e., outside the Language of Mathematics). This topic is discussed in more detail in Section 6.13 and, especially, in Section 6.13.1.

In this book, a name of a variable or function that consists of two or more English words is written without spaces between the words and with each word capitalized. This naming convention is for reading convenience only. In the Language of Mathematics there is no commonly accepted rule for naming variables or for capitalization.

Traditionally, the names of variables and functions are typeset in italics in the formal mathematical literature. Within English text this sometimes helps the reader a little to understand that text more quickly and precisely. However, within mathematical expressions and in the English interpretation of the variables and functions appearing in the mathematical model, italicizing the names is of no advantage. There, it is always clear whether or not the name of a variable or function is meant. Within mathematical expressions, other aspects of formatting contribute much more to readability, especially indenting and aligning corresponding parts of expressions. Reference numbers for expressions and even for lines within an expression also help the readers considerably in communicating with one another about the mathematical expressions.

3.3 FUNCTIONS

A *function* yields a value depending on other value(s). The other values on which the value of the function depends are called the *arguments* of the function. The value of the function is determined uniquely by the values of the arguments.

A common example is the function “sum,” the value of which is the sum of the values of its arguments: The value of sum(4, 5) is 9 (the sum of the values 4 and 5).

Another example of a reference to the value of a function is sum(x, y). The value of sum(x, y) is the sum of the value of the variable named x and the value of the variable named y. If the value of the variable named x is 4 and the value of the variable named y is 5, then the value of sum(x, y) is the same as the value of sum(4, 5), which is 9 (see above).

English phrases such as “the value of the variable named x” and “the value of the variable named y” are long and they occur often. To reduce the length of such English phrases and the sentences containing them, the phrases “the value of” and “the variable named” are often dropped completely. Other shortcuts are also often employed, such as dropping the phrase “the same as” in the last sentence in the preceding paragraph. Applying these abbreviations, the last sentence in the paragraph above becomes “If x is 4 and y is 5, then sum(x, y) is sum(4, 5), which is 9.” Such abbreviations are convenient and can improve readability, provided that the reader is aware of the full meaning of such abbreviated sentences. The reader who does not keep these implied but missing phrases in mind will, sooner or later, become confused. A careless beginner can easily become confused early in the learning process, and his or her confusion will become worse, not less, as he or she deals with more complicated expressions. Lack of attention to detail in mathematics is always dangerous and leads quickly to errors and confusion.

A function may be defined to have any particular number of arguments, including none. The value of a function having no arguments is, of course, always the same, so such a function is of limited usefulness. Usually, a particular function has a fixed number of arguments, but this need not always be the case.

The values of the arguments of the function “sum” must be numbers and the value of the function itself is a number.

An example of a function whose value is not numerical is the function “IsGreaterThan,” whose name we will shorten here to “IsGreater.” Because 6 is greater than 3, the value of IsGreater(6, 3) is the logical value “true.” The value of IsGreater(3, 6) is “false,” because 3 is not greater than 6. The values of the arguments of the function “IsGreater” are numbers, but the value of the function itself is logical (i.e., “true” or “false”), not numerical.

The order of the arguments is, in general, significant; that is, the value of func(x, y) is not necessarily the same as the value of func(y, x). In the case of the function sum above, the value of sum(4, 5) is the same as that of sum(5, 4), but the function “IsGreater” has different values for IsGreater(6, 3) and IsGreater(3, 6), as we have seen above.

For references to the value of a function, the notational form above is standard and can be used in general. That is, the value of the function is written in the form of the function name followed by a list of the arguments enclosed in parentheses. The individual arguments in the list are separated by commas. Examples are:

- sum(4, 5)
- sum(x, 5)
- sum(4, y)
- sum(x, y)
- IsGreater(x, y)

In the above, “4, 5” is a list of the two arguments 4 and 5. Similarly, “x, y” is the list of the two arguments x and y.

A shorter notational form is commonly used for a few frequently referenced functions of two arguments, such as the functions sum and IsGreater. A single symbol replaces the name (e.g., “sum,” “IsGreater”) and the symbol is written between the two arguments. Thus, instead of writing sum(x, y), one writes x+y, and instead of writing IsGreater(5, 3), one writes 5>3. This shorter form is called *infix* notation because the function symbol is placed between the arguments (*in* the sequence of the arguments). Relative advantages and disadvantages of these two and other notational forms are discussed in Section 3.4.10.

For definitions of the *domain* and the *range* of a function, and for a notation for expressing the relationship between a function and its domain and range, see the latter part of Section 4.1.1 and Section 3.4.9.

Functions of two arguments whose values are elements of the same set are common in mathematics. Such functions are called *binary functions* or *binary operators*. Examples are the numerical functions addition ($+$), subtraction ($-$), multiplication ($*$), and division (/); the logical functions and ($\wedge$), or ($\vee$), and implication ($\Rightarrow$); and the relational functions equals ($=$), is not equal to ($\neq$), is less than ($<$), is greater than ($>$), is less than or equal to ($\leq$), and is greater than or equal to ($\geq$). These and other binary operators will be encountered in many sections of the book. Except for the logical functions, it is assumed that the reader is already familiar with these functions.

The logical functions and ($\wedge$), or ($\vee$), negation ($\neg$), and implication ($\Rightarrow$) are defined in the following truth table:

TABLE 3.3-1 Definitions of the Logical Functions

X	Y	X $\wedge$ Y	X $\vee$ Y	$\neg$ X	X $\Rightarrow$ Y
false	false	false	false	true	true
false	true	false	true	true	true
true	false	false	true	false	false
true	true	true	true	false	true

Informally, X$\wedge$Y is true if both X and Y are true; otherwise, it is false. X$\vee$Y is true if either X or Y (or both) is true; otherwise, it is false. $\neg$X is true if X is false, and vice versa.

Some people question the appropriateness of the definition of the implication function ($\Rightarrow$) when they first encounter it. In particular, some consider the first two lines with X being false and the value of the function being true to be strange. One possible response is that, mathematically, a justification is not needed; the implication is simply defined that way. Another possible response is that the English sentence "if X is true, then Y is true" says nothing about the cases in which X is false, and therefore, (1) X is false and Y is false and (2) X is false and Y is true are both consistent with the given English sentence—they do not contradict it or represent examples invalidating it. Therefore, "true" is appropriate as the value of the implication function in these cases. Still another justification could be that the mathematical expression "X $\Rightarrow$Y" is to be translated directly and colloquially into the English sentence "if X is true, then Y is true; otherwise (i.e., if X is false), anything goes."

When mathematical expressions including the logical functions are intended to be read by people not familiar with these functions and the symbols $\wedge$, $\vee$, $\neg$, and $\Rightarrow$, the words **and**, **or**, **not**, and **implies**, respectively, are sometimes used instead of the more abstract symbols. For example, the mathematical expression

$$(X \wedge Y) \Rightarrow \neg Z$$

can be written

(X **and** Y) **implies not** Z

When used as mathematical symbols, these words are often set in boldface type to distinguish their use as mathematical symbols from their use as normal words in English text.

3.4 EXPRESSIONS

Expressions are the highest-level elements in the Language of Mathematics. Expressions can be combined in certain ways to form another expression, much as in English, phrases are combined to form a larger phrase, a clause, or a sentence and clauses are combined to form a complete sentence. An expression in the Language of Mathematics corresponds to a phrase, clause, or sentence in English, depending on whether the value of the mathematical expression is a logical value (false, true) or not (i.e., anything else) and whether and how it is combined with other expressions. See Section 6.2 for a more detailed discussion of the correspondence between mathematical expressions and phrases, clauses, and sentences in English.

An expression is one of the following:

- A value
- A reference to the value of a variable: that is, a variable name
- A reference to the value of a function: that is, a function name followed by a list of arguments separated by commas and enclosed in parentheses
- A certain combination of expressions

By composing functions as described in Section 3.4.1, expressions can be combined to form another expression. An expression can be written using any of several different notational forms, each of which is the subject of a later subsection. An expression represents a function and is a reference to the value of that function.

Sections 3.4.1 through 3.4.9 define different notational forms for expressions. These definitions are given in relatively precise English or in the form of equivalences between these notational forms (e.g., in the form of rules for converting from one notational form to another). More advanced works define these notational forms mathematically precisely using the Language of Mathematics, not English.

3.4.1 Standard Functional Notation

The standard notational form for references to the value of a function (see Section 3.3) is

function(x, y, z) [3.4.1-1]

Each of the arguments x, y, z, and so on, may be a value, a reference to the value of a variable (i.e., a variable name), or a reference to the value of a function. One may write references to values of functions as arguments of higher-level functions, embedded to any depth desired. Thus one can write, for example,

sum(product(a, sum(w, x)), product(b, sum(y, z))) [3.4.1-2]

and

and(IsLess(product(a, sum(w, 2)), sum(product(d, x), 3)),
IsGreater(sum(w, u), z)) [3.4.1-3]

where the value of the logical function "and" is defined in Table 3.3-1.

Building expressions as described above by including references to values of functions as arguments of higher-level references to values of functions is called *composing* functions. Using this method, any valid expression can be formed, so this provides a general way of writing expressions. Other notational forms are defined in later sections, but they are all semantically equivalent to—and are defined below in terms of—a composition of standard notational forms for references to values of functions.

Note that both numbers and logical values (false and true) occur as values in expression 3.4.1-3. The values of the functions "IsLess" and "IsGreater" are logical values, whereas their arguments are numerical values. The values of the functions "product" and "sum" as well as the values of their arguments are numerical values. The value of the function "and" as well as the values of its arguments are logical values. The types of these various values can be seen more clearly in a tree diagram for this expression in Section 3.4.3.

3.4.2 Infix Notation

The most commonly used form for writing mathematical expressions in practice is a mixture of partially parenthesized infix notation (i.e., "x+y"; see Section 3.3) and the standard notational form for references to values of functions [i.e., "function(x, y)"].

Infix notation is used for frequently occurring functions of two arguments. The name of the function is replaced by a special symbol and the symbol is placed between the two arguments (hence the name "infix," as opposed to "prefix" and "postfix"; see Section 3.4.4). If either argument is an infix expression, it may or may not be necessary to enclose the argument in parentheses, depending on the context in which the argument then appears. It is never wrong to enclose the argument in parentheses, so when in doubt, do so. Eliminating unnecessary parentheses is discussed later in this section.

Functions of one argument can also be written in a prefix form similar to infix notation. The negative function for numbers and the logical negation function are the most common examples. The value of "negative(x)" is written "−x." The value of "not(x)" is written "¬x."

Viewed simply, an expression in infix notational form consists of a sequence of references to values and of symbols representing functions of two arguments alternating with each other. An infix expression must begin and end with a reference to a value. In addition, an infix expression may be enclosed in a pair of parentheses. An infix expression, whether or not enclosed in a pair of parentheses, is a reference to the value of the function represented by the infix expression. A symbol for a function of one argument (e.g., $-$ or $\neg$) together with its following argument is a reference to the value of the function.

Infix notation is not used for references to the values of functions with no argument or with three or more arguments. Standard functional notation is normally used for such functions.

Spaces may be inserted in an infix expression to improve readability. Similarly, an expression may be written in several lines and indented. Spaces and line breaks in an infix expression have no mathematical meaning; the expression is interpreted as if they were not present (i.e., as if the expression were written with no spaces and no line breaks).

Following are some examples of infix expressions:

$$\mathrm{a}$$

$$\mathrm{a}+\mathrm{b}$$

$$(\mathrm{a}+\mathrm{b}-\mathrm{c})$$

$$\mathrm{a}+\mathrm{b}*\mathrm{c}$$

$$-\mathrm{a}+\mathrm{b}*\mathrm{c}$$

$$(\mathrm{a}+\mathrm{b}*-\mathrm{c})$$

$$\mathrm{a}+(\mathrm{b}*(\mathrm{c}+2))/\mathrm{x}$$

$$\mathrm{a}+\mathrm{sum}(\mathrm{b}, \mathrm{product}(\mathrm{c}, \mathrm{d}))+\mathrm{product}(\mathrm{f}, 5)$$

The symbol $*$ is used in this book as the infix symbol for the product of two numbers (multiplication). Other symbols are also in common use for this function, such as a centered dot ($\cdot$), the symbol $\times$, and no symbol at all: that is, simply writing the two variable names immediately after another (e.g., xy for x$*$y). In this book, the symbol $\times$ is used for another function (the Cartesian product of two sets, not the numerical product).

Notice that the meaning of the expression a$+$b$*$c is not defined by the description above. The functions $+$ and $*$ require two arguments each, but there are three arguments in the expression. It is not clear to which two arguments each function is to be applied. The expression could mean either

$$(\mathrm{a}+\mathrm{b})*\mathrm{c} \qquad [\text{i.e., product(sum(a, b), c)}] \tag{3.4.2-1}$$

or

$$\mathrm{a}+(\mathrm{b}*\mathrm{c}) \qquad [\text{i.e., sum(a, product(b, c))}] \tag{3.4.2-2}$$

whose values are not, in general, the same. Such ambiguities can be resolved by inserting parentheses into the unparenthesized infix expression (e.g., as in either of the two expressions above).

To improve readability in general by reducing the number of parentheses appearing in expressions typically arising in practice, conventions have evolved for interpreting unparenthesized expressions and parts thereof. Such conventions, in effect, supply the missing parentheses. Applied to the expression a+b∗c, the conventions state that this expression means a+(b∗c) [i.e., sum(a, product(b, c))]. If the other interpretation is meant, the parentheses must appear explicitly. The conventions are described in detail below.

In an infix expression the various functions are applied (the functions *bind*) in the following order unless otherwise indicated by parentheses:

TABLE 3.4.2-1 Binding Order for Functions in the Absence of Parentheses

Binding Order	Number of Arguments	Symbol and Function
1	2	$\uparrow$ (exponentiation, also written with superscripting, e.g., x^{exp})
2	1	$+$, $-$ (when used as the sign of a number, not for addition or subtraction)
3	2	$*$, / (multiplication, division)
		$\cap$ (set intersection)
4	2	$+$, $-$ (when used for addition or subtraction, not as the sign of a number)
		$\cup$, $\setminus$ (set union or set difference)
5	2	$<$, $>$, $=$, $\leq$, $\geq$, $\neq$ (relations)
		$\in$, $\notin$ (element of a set), $\supset$, $\supseteq$, $\not\subset$, $\subset$, $\subseteq$ (subset)
6	1	$\neg$ (logical negation, sometimes written **not**)
7	2	$\wedge$ (logical and (conjunction), sometimes written **and**)
8	2	$\vee$ (logical or (disjunction), sometimes written **or**)
9	2	$\Rightarrow$, $\Leftarrow$, $\Leftrightarrow$ (logical implications)

Each function of two arguments is applied to the two arguments immediately adjacent to the function's symbol. Each function of one argument is applied to the argument immediately to the right of the function's symbol. When functions at the same binding level above appear in immediate succession, they are applied from left to right; see below.

The lower the binding order (the higher up in the list above), the more tightly the function binds its arguments. Thus, in an expression involving addition and multiplication, the multiplication function(s) apply before the addition function(s). Parentheses are assumed first around each multiplication and its immediately neighboring arguments. Afterward, parentheses are assumed around each addition and its immediate arguments. An expression in parentheses is considered a single argument.

As a general memory aid, remember that the arithmetic and set operations bind most tightly, the relations less tightly, and the logical functions the least tightly.

If several functions with the same binding level occur in succession in an expression, parentheses are inserted in pairs from left to right. For example,

$$d*e/f*g \tag{3.4.2-3}$$

means

$$(((d*e)/f)*g) \tag{3.4.2-4}$$

and, similarly, the expression

$$h+i-j+k \tag{3.4.2-5}$$

means

$$(((h+i)-j)+k) \tag{3.4.2-6}$$

Expressions containing several functions at each of several binding levels are parenthesized by considering the functions with the lowest binding order first, then the functions with the second lowest binding order, and so on. For example, in the expression

$$m*n/p+r/s*t-u*v*w \tag{3.4.2-7}$$

parentheses are first placed around the $*$ and / infix symbols

$$((m*n)/p) + ((r/s)*t) - ((u*v)*w) \tag{3.4.2-8}$$

and then around the $+$ and $-$ infix symbols, whereby each already parenthesized expression is considered as a single argument. After inserting the parentheses around the $+$ and $-$ infix symbols from left to right, expression 3.4.2-8 becomes

$$((((m*n)/p) + ((r/s)*t)) - ((u*v)*w)) \tag{3.4.2-9}$$

or, eliminating the spaces,

$$((((m*n)/p)+((r/s)*t))-((u*v)*w)) \tag{3.4.2-10}$$

as the fully parenthesized expression for the original expression above.

Consider again an expression of the form

$$a+b+c \tag{3.4.2-11}$$

which, by usual convention, means

$$(a+b)+c \tag{3.4.2-12}$$

but could otherwise mean

$$a+(b+c) \tag{3.4.2-13}$$

In the case of some functions (the mathematical function $+$ is one of them), the values of the last two types of expressions above are always the same. Such a function is called *associative*. Thus, even if the structural meaning of the expression “a+b+c” were not covered by the convention of grouping parentheses from left to right, its value is always unambiguously determined; its value is always the same no matter which structural meaning is assumed. Because the ultimate meaning of any expression is its value for given values of its arguments, the Language of Mathematics permits one to write “a+b+c” as an abbreviation for either “(a+b)+c” or “a+(b+c).” The convention of parenthesizing from left to right expressions involving functions at the same binding level is really needed only for functions that are not associative.

Caution is called for when abbreviating expressions with numerical values in this way. While the mathematical function $+$ is associative, arithmetic functions implemented in computer systems are generally not; they are only approximations to the corresponding mathematical arithmetic functions. The deviations from the mathematical functions are usually small, but they can be critical in some cases. A purported mathematical proof of the behavior of a computer program can be invalidated by incorrectly assuming that the value of “(a+b)+c” is the same as the value of “a+(b+c)” when $+$ means computational addition in a computer system.

Not all functions are associative. The difference function (subtraction) is an example. The value of the expression “(a−b)−c” is not, in general, the same as the value of “a−(b−c).” (The two expressions have the same value only when the value of c is 0.) By the conventions for supplying missing parentheses, the expression “a−b−c” will be interpreted as “(a−b)−c,” not as “a−(b−c).” When in doubt, the writer should insert the parentheses explicitly.

Another special abbreviation is useful in expressions involving the functions $<$, $\leq$, $=$, $\geq$, and $>$. It is often convenient to write an expression such as

$$a < b \leq c = d \qquad [3.4.2\text{-}14]$$

when one really means

$$(a<b) \land (b\leq c) \land (c=d) \qquad [3.4.2\text{-}15]$$

which does not need further parentheses because the function $\land$ is associative (see above). This type of abbreviation is most appropriately considered to be an idiom in the Language of Mathematics, because its meaning is not the meaning the binding-level conventions described above would assign to it.

In expression 3.4.2-15, the parentheses can be dropped because the relational functions bind more tightly than the logical functions do.

Note that “a<b≤c=d” does not mean “(((a<b)≤c)=d)” or any other expression formed only by inserting parentheses into “a<b≤c=d.” None of the expressions formed in that way have any meaning. In each such expression a logical value (false or true) would be an argument to a relational function ($<$, $\leq$, $\geq$, or $>$) requiring numerical values as arguments, and such an argument would be of the wrong type. This would be a grammatical error in the Language of Mathematics. Such an inconsistency of argument types is discussed further in connection with tree notation in Section 3.4.3. Section 6.5 deals in greater depth with this topic and its implications for the semantics of infix notation without parentheses.

References to the following functions are usually written in infix notation.

TABLE 3.4.2-2 Transforming Standard Functional Notation to Infix Notation

Standard Notation for Functions	Fully Parenthesized Infix Notation
sum(arg1, arg2)	((arg1)+(arg2))
difference(arg1, arg2)	((arg1)−(arg2))
product(arg1, arg2)	((arg1)∗(arg2))
quotient(arg1, arg2)	((arg1)/(arg2))
IsGreater(arg1, arg2)	((arg1)>(arg2))
IsGreaterOrEqual(arg1, arg2)	((arg1)≥(arg2))
equals(arg1, arg2)	((arg1)=(arg2))
DoesNotEqual(arg1, arg2)	((arg1)≠(arg2))
IsLessOrEqual(arg1, arg2)	((arg1)≤(arg2))
IsLess(arg1, arg2)	((arg1)<(arg2))
and(arg1, arg2)	((arg1)∧(arg2))
or(arg1, arg2)	((arg1)∨(arg2))
implies(arg1, arg2)	((arg1)⇒(arg2))
IsImpliedBy(arg1, arg2)	((arg1)⇐(arg2))
negative(arg1)	(−(arg1))
not(arg1)	(¬(arg1))

In the table, "arg1" refers to the first (left) argument and "arg2" to the second (right) argument. In general, the order of the arguments is significant and must be the same in the two notations.

If arg1 is a value, a variable name (i.e., a reference to the value of a variable), or a reference to the value of a function in standard functional notation [i.e., in the form "function(…)"], it need not be enclosed in parentheses, and similarly for arg2.

If arg1 is an infix expression [including such forms as "func1(…)+func2(…)"], parentheses around arg1 might be needed and should, therefore, be included, at least initially. The same applies to arg2.

Finally, depending on the context in which the infix expression "(arg1)+(arg2)" will appear, a pair of parentheses around this entire expression may be needed, so if in doubt, include them. It is never wrong to do so.

In general, to convert a reference to the value of a function from standard functional notation to fully parenthesized infix notation, place the first argument in parentheses, write the symbol for the function afterward, and then write the second argument also enclosed in parentheses. Then place parentheses around the entire expression thereby formed. Substitute this final expression for the original reference to the value of the function. Table 3.4.2-2 illustrates this for each of the functions shown. Finally, one may eliminate every pair of parentheses that would be implied by the conventions above for binding levels and left-to-right grouping.

Example: Transforming Standard Functional Notation to Infix Notation Consider the last expression in Section 3.4.1:

and(IsLess(product(a, sum(w, 2)), sum(product(d, x), 3)),
    IsGreater(sum(w, u), z)) [3.4.2-16]

Assuming that “a,” “w,” “d,” “x,” “u,” and “z” are names of variables, expression 3.4.2-16 can be rewritten in the forms shown below, all of which mean the same.

In the following sequence of transformations, each subexpression to be transformed to infix notation is enclosed in a box. (Such boxes are not provided for in the Language of Mathematics; the expressions are to be interpreted mathematically as if the boxes were not present. The boxes are inserted here to indicate particular subexpressions.)

$$\text{and(IsLess(product(a, }\boxed{\text{sum(w, 2)}}\text{), sum(}\boxed{\text{product(d, x)}}\text{, 3)),}$$
$$\text{IsGreater(}\boxed{\text{sum(w, u)}}\text{, z))} \qquad [3.4.2\text{-}17]$$

The reference to “sum(w, 2)” can be replaced by “w+2.” The reference to “product(d, x)” can be replaced by “d∗x.” The reference to “sum(w, u)” can be replaced by “w+u.” Each of these becomes an entire argument of a reference to the value of a function in standard functional notation, so surrounding parentheses are not needed; if included, the meaning of the expression would be the same. The expression above can therefore be written in the form

and(IsLess(product(a, w+2), sum(d∗x, 3)), IsGreater(w+u, z)) [3.4.2-18]

In the transformation above we began with those functions whose arguments contained no references to functions. Viewed differently but equivalently, we began with references to the values of functions with simple arguments (values and variable names) only. This is not the only possibility. One can rewrite references to the values of functions with arguments in other forms, also. For example, we can rewrite the references to the functions “IsLess” and “IsGreater” in the expression

$$\text{and(}\boxed{\text{IsLess(product(a, w+2), sum(d∗x, 3))}}\text{, }\boxed{\text{IsGreater(w+u, z)}}\text{)} \qquad [3.4.2\text{-}19]$$

to obtain

and(product(a, w+2)<sum(d∗x, 3), (w+u)>z) [3.4.2-20]

In expression 3.4.2-20, parentheses are not needed around “product(a, w+2)” because it is a reference to the value of a function. Enclosing it in parentheses would not change the meaning of the expression within which it appears. For the same reason, “sum(d∗x, 3)” need not be enclosed in parentheses.

The infix expression “w+u,” however, is not so clear. Therefore, we enclose it in parentheses initially, as it is never wrong to do so. The parentheses around “w+u” in “(w+u)>z” could be removed only if “w+u>z” means “(w+u)>z” and not “w+(u>z).” The conventions for binding levels (see Table 3.4.2-1) provide that + binds before >, therefore “w+u>z” means, by convention, “(w+u)>z.” Therefore, the parentheses can be removed.

The reference to the value of the function “and” in the expression

$$\boxed{\text{and(product(a, w+2)<sum(d∗x, 3), w+u>z)}} \qquad [3.4.2\text{-}21]$$

can be replaced by its equivalent in infix notational form. For the same reason that applied to "w+u" in the previous transformation above, the two arguments in infix notation ["product(a, w+2)<sum(d∗x, 3)" and "w+u>z"] will each be enclosed, initially at least, in parentheses in the new expression:

$$\text{(product(a, w+2)<sum(d}*\text{x, 3))} \wedge \text{(w+u>z)} \qquad [3.4.2\text{-}22]$$

By the conventions for binding levels, the relational functions < and > bind more tightly than the logical and function ∧, so the parentheses around each argument of the ∧ function can be dropped [remember that "(w+u>z)" means "((w+u)>z)"]:

$$\boxed{\text{product(a, w+2)}} < \boxed{\text{sum(d}*\text{x, 3)}} \wedge \text{w+u>z} \qquad [3.4.2\text{-}23]$$

The references in standard functional notation in expression 3.4.2-23 can be transformed in the same way as above so that the entire expression is in infix notation:

$$((a*(w+2))<((d*x)+3)) \wedge w+u>z \qquad [3.4.2\text{-}24]$$

Because the function < binds more tightly (has a higher binding level) than the function ∧, the outer parentheses around the left argument of ∧ can be removed, yielding

$$(a*(w+2))<((d*x)+3) \wedge w+u>z \qquad [3.4.2\text{-}25]$$

as an equivalent expression. Because the functions ∗ and + bind more tightly than the relational function <, the outer parentheses around the two arguments of the function < can be removed, giving

$$a*(w+2)<(d*x)+3 \wedge w+u>z \qquad [3.4.2\text{-}26]$$

Because ∗ binds more tightly than +, the parentheses around "d∗x" can be removed. Removing the parentheses around "w+2" would change the meaning of "a∗(w+2)" to "(a∗w)+2," which, in general, does not have the same value, so this pair of parentheses may not be removed. The resulting expression is, therefore,

$$a*(w+2)<d*x+3 \wedge w+u>z \qquad [3.4.2\text{-}27]$$

which is the minimally parenthesized infix expression for the original expression 3.4.2-16:

and(IsLess(product(a, sum(w, 2)), sum(product(d, x), 3)),
    IsGreater(sum(w, u), z)) [3.4.2-16 repeated]

in standard functional notation.

Readers will find tree notation (presented in Section 3.4.3) helpful in understanding parenthesizing and for remembering when parentheses are necessary, when they are optional, and when they should not be used around certain subexpressions. Parentheses serve to define the tree structure associated with an expression written in infix notation.

An expression in infix notation is valid grammatically if and only if it can be derived from an expression in purely standard functional notation (i.e., containing no infix subexpression) as described above or is equivalent to such an expression by the rules given above for parentheses. Such an expression in infix notation has the same meaning (i.e., the same value) as the expression in purely standard functional notation.

It is suggested that the reader apply the rules for parentheses above to insert explicitly the implied parentheses in the infix expression

$$a*(w+2)<d*x+3 \wedge w+u>z \qquad [3.4.2\text{-}28]$$

to obtain the more extensively parenthesized expression

$$(((a*(w+2))<((d*x)+3)) \wedge ((w+u)>z)) \qquad [3.4.2\text{-}29]$$

for the expression

```
and(IsLess(product(a, sum(w, 2)), sum(product(d, x), 3)),
     IsGreater(sum(w, u), z))                    [3.4.2-16 repeated]
```

in standard functional notation from earlier in this section and from Section 3.4.1.

The comments above on parentheses in infix expressions can be summarized in the following rules for parentheses. The term "exp" in these rules stands for any expression in any notational form.

TABLE 3.4.2-3 Summary of the Rules for Parentheses

1. *Equality of transformed expressions* If a syntactically correct composed expression in standard functional notation only (i.e., all arguments are in standard functional notation) is transformed as specified in *Table 3.4.2-2 Transforming standard functional notation to infix notation,* then the resulting infix expression is syntactically correct and fully parenthesized and means the same as the original expression in standard functional notation.
2. *Value independent of surrounding parentheses* For any expression exp written in any notational form, the value of (exp) is the same as the value of exp [e.g., function((exp1), exp2) can be reduced to function(exp1, exp2), and similarly for exp2]. Note that this rule does *not* permit (exp) to be reduced to exp if (exp) is a part of an infix expression, such as …+(exp)∗…, and so on, because the structure of the entire expression can be changed by removing these parentheses.
3. *Removing extra parentheses* If exp is included in more than one pair of ( ), it can be reduced to (exp) [i.e., all but one outer pair of ( ) can always be removed from any (…(exp)…)].
4. *Removing parentheses from a simple expression* If exp is simple (i.e., is a value, a variable name, or a reference to the value of a function in standard functional notation), then (exp) can be reduced to exp. If exp is in infix notation, this reduction is not permitted by this rule.
5. *Removing parentheses implied by binding rules* Any pair of parentheses can be removed if it is implied by the binding orders of the relevant functions in the infix expression in question or by the left-to-right grouping rule as described earlier in this section.

Note that pairs of parentheses match from inside out. They do not necessarily match from outside in. For example, in the structure of parentheses

$$\begin{array}{ccccccc} (& (&) & + & (&) &) \\ a & b & c & & d & e & f \end{array}$$

in an infix expression, parentheses b and c match (i.e., enclose a subexpression). Similarly, parentheses d and e match. Parentheses a and f match.

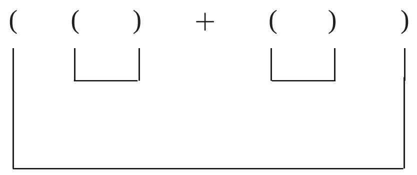
Examining the parentheses from outside inward, one might expect parentheses b and e to match, but they do not. The text between (but not including) the parentheses b and e is not a syntactically correct expression. That text contains two parentheses but not their matching parentheses.

In long infix expressions it is often difficult to see immediately which parenthesis matches with which other parenthesis. For this reason, different symbols are sometimes used for different parentheses, for example,

$$[\quad (\quad) \quad + \quad (\quad) \quad]$$

larger symbols being used for outer matching parentheses and smaller symbols being used for inner matching parentheses. This method is of limited usefulness, however, because the number of different available symbols is often insufficient to provide a different symbol for each level of parentheses in a long and complex expression.

Matching parentheses can be identified diagrammatically in various ways. One convenient way is to connect matching parentheses with lines:



Another method to identify matching parentheses is to number the parentheses according to their nesting level (i.e., enclosing level):

$$\begin{array}{cccccccccccccc} (& (&) & (& (& (&) &) &) &) & (& (&) &) \\ 1 & 2 & 2 & 2 & 3 & 4 & 4 & 3 & 2 & 1 & 1 & 2 & 2 & 1 \end{array}$$

Each left parenthesis matches with the next following right parenthesis with the same number. Conversely, each right parenthesis matches with the closest preceding left parenthesis with the same number.

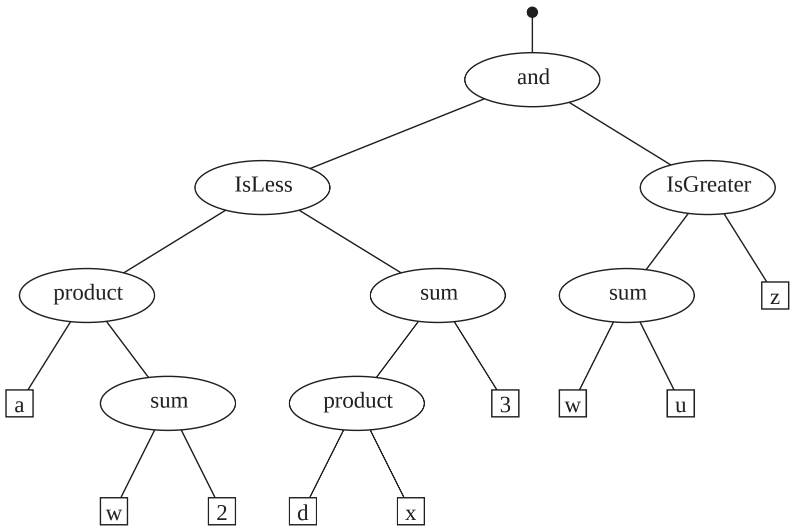
These numbers are written under the parentheses from left to right, starting with 1, to indicate that the following part of the expression is at nesting level 1. If the next parenthesis is a left parenthesis, the next-higher number is written under it, indicating that the following part of the expression is at the next-higher nesting level. If, instead, the next parenthesis is a right parenthesis, the same number (the number of the current nesting level) is written under it and the current nesting level is reduced by 1. This procedure continues from left to right until the last parenthesis is numbered. That last parenthesis will be a right parenthesis and will be numbered 1, and the lowest number ever assigned to any parenthesis will be 1; otherwise, the parenthesis structure is syntactically incorrect. If desired, one can proceed from right to left correspondingingly.

3.4.3 Tree Notation

Tree notation gives a good visual insight into the structure of an expression and facilitates understanding various aspects of it. Tree notation is, therefore, useful for learning purposes and for analyzing an expression in detail. It is not commonly used for practical work, however, because of the space required to represent an expression and because it takes longer to draw.

Consider again expression 3.4.1-3 (which is the same as 3.4.2-16):

and(IsLess(product(a, sum(w, 2)), sum(product(d, x), 3)),
IsGreater(sum(w, u), z)) [3.4.3-1]



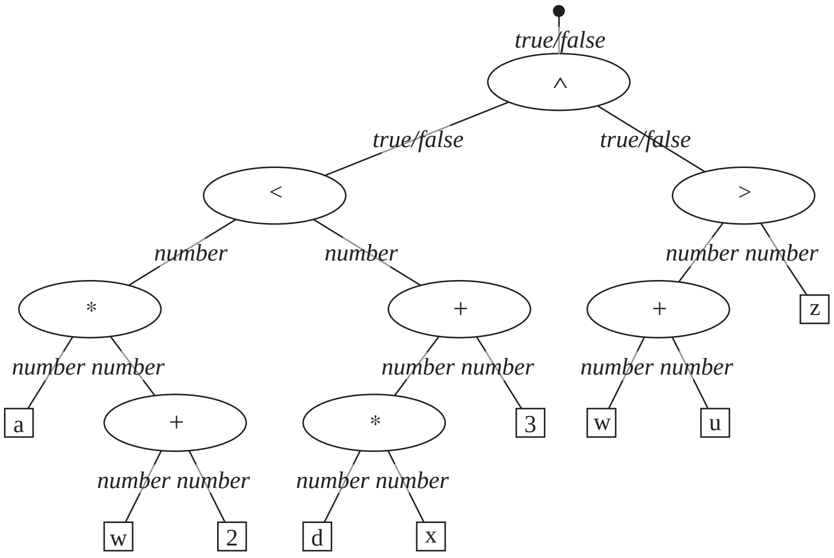
This expression refers to the value of the function “and” for two arguments. The first argument is, in turn, a reference to the value of the function “IsLess” for two arguments, each of which is composed further of references to the values of the functions “product” and “sum.” The second argument of the function “and” is a reference to the value of the function “IsGreater” for two arguments. The first argument of the function “IsGreater” is a reference to the value of the function “sum,” whose arguments are the values of the variables w and u. The second argument of “IsGreater” is the value of the variable z.

Note in expression 3.4.3-1 that all of the following appear as arguments to functions:

- Values (e.g., 2 and 3)
- References to the values of variables (the variable names a, d, w, x, u, and z)
- References to the values of functions (IsLess, product, sum, and IsGreater)

The hierarchical composition of these functions, values, and variables to form expression 3.4.3-1 is illustrated in the tree diagram above. It shows the logical structure of the expression (i.e., the relationships between the components of the expression). It also indicates the order in which the intermediate values must be determined and used when evaluating the expression.

In earlier sections it was pointed out that the types of values of arguments must agree with the types of values required by the functions in question. The tree shown below has been annotated with this information so that one can verify that the type of each value provided as an argument matches the requirements of the function at the next-higher node in the tree. The nodes of the tree below have also been relabeled with the infix symbols instead of with the function names.



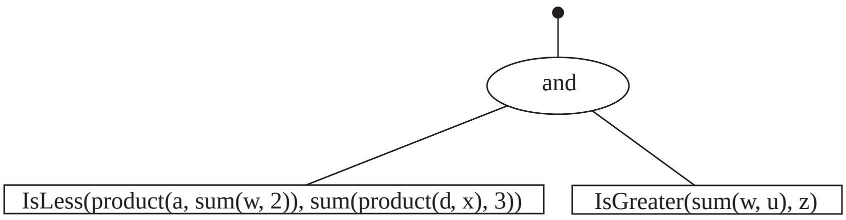
Example: Transforming Standard Functional Notation to Tree Notation An expression in standard functional notation can be transformed into a tree in the following way. Expression 3.4.3-1, repeated below, is used as an example.

and(IsLess(product(a, sum(w, 2)), sum(product(d, x), 3)),
    IsGreater(sum(w, u), z)) [3.4.3-1 repeated]

Begin by separating the first function name and its arguments from each other:

- and
- IsLess(product(a, sum(w, 2)), sum(product(d, x), 3))
- IsGreater(sum(w, u), z)

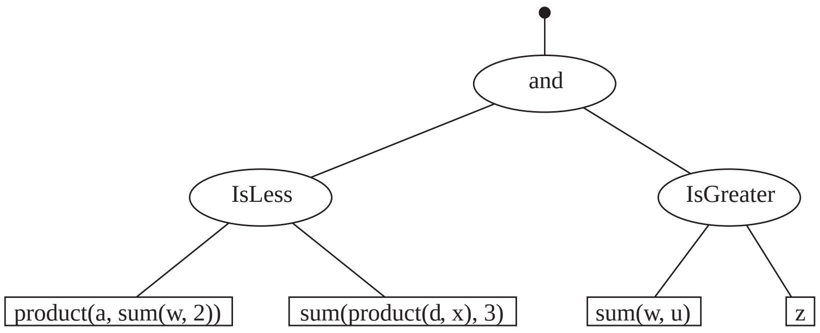
Draw a node and write the function's name in it. Then draw one line downward for each argument, and below each line, write the corresponding argument. Be careful to maintain the order of the arguments; the left argument in the reference to the function in the expression must be written under the left line from the node, and so on.



Next, repeat the process above for each of the two arguments above. That is, separate, for each argument above, its first function name from that function's arguments:

IsLess	IsGreater
product(a, sum(w, 2))	sum(w, u)
sum(product(d, x), 3)	z

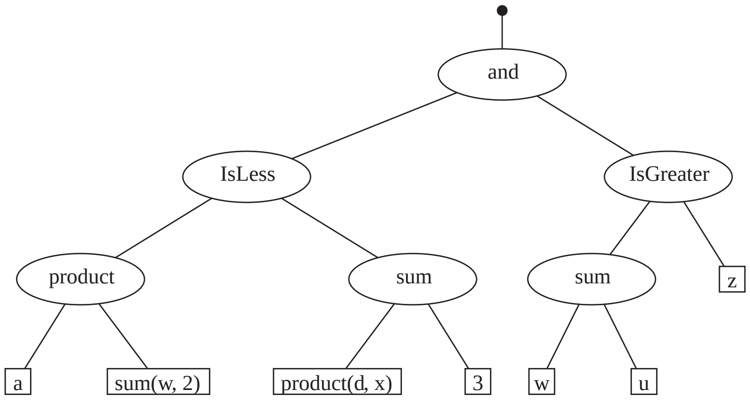
For each, draw a node, write the function name in it, and draw lines downward for the arguments. Place the node for each argument's first function under the line for that argument:



When a value or a variable name remains, that branch is complete (e.g., the rightmost branch above, ending with the variable "z"). Branches ending with a reference to the value of a function must be developed further by repeating the process described above. Separating each such argument above into its function name and its arguments:

product	sum	sum
a	product(d, x)	w
sum(w, 2)	3	u

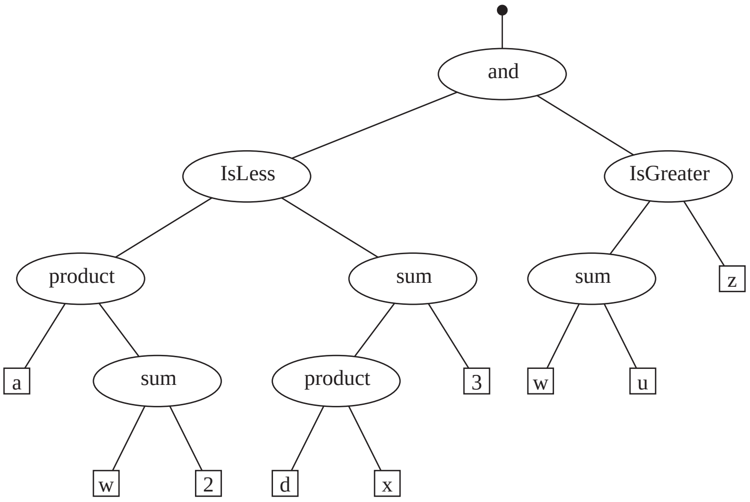
and adding the corresponding nodes and descending lines to the tree above results in the following tree:



Two branches still end with references to the value of a function, so they must, in turn, be developed further in the same way as above. Separating in each case the function name and its arguments from each other:

sum	product
w	d
2	x

and drawing the corresponding nodes and downward lines for the arguments yields the final tree:



Every branch in the tree above ends with a value or a variable name, so the tree is complete. This tree is the same as the first tree in this section.

The tree above was constructed "top down" from an expression in standard functional notation. A tree can also be constructed "bottom up" by starting with the innermost groups of arguments, which will be only values or variables. In general, there will be more than one such subtree at the bottom level (in the tree above, there are two such subtrees).

The parentheses in an extensively parenthesized infix expression correspond directly to the structure of the equivalent tree. Each pair of parentheses encloses the infix expression corresponding to one node and its arguments. For example, the fully parenthesized infix expression (cf. Table 3.4.2-2 for transforming standard functional notation to infix notation) equivalent to the tree above is

$$((((((a)*(((w)+(2)))))<(((((d)*(x)))+(3))))) \wedge (((((w)+(u)))>(z))))$$

[3.4.3-2]

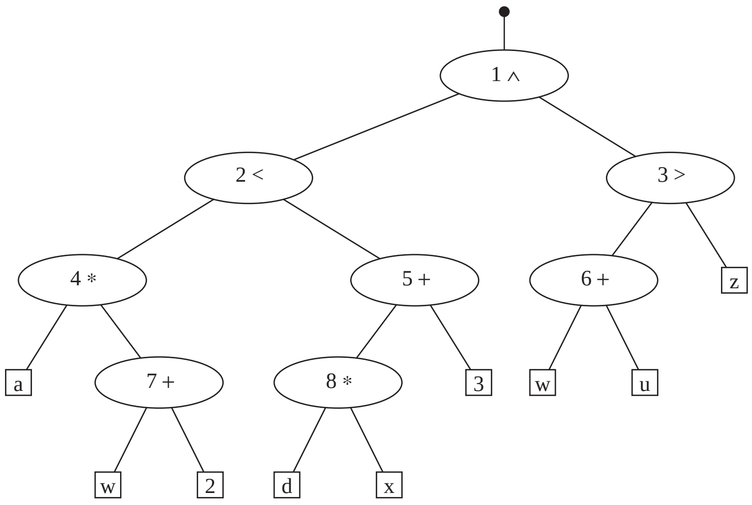
Removing the parentheses immediately surrounding the values and variable names simplifies the expression somewhat to

$$(((((a*((w+2))))<((((d*x))+3)))) \wedge ((((w+u))>z)))$$ [3.4.3-3]

Removing extra pairs of parentheses around single subexpressions [i.e., reducing "((exp))" to "(exp)"] simplifies the expression further to the expression

$$(((a*(w+2))<((d*x)+3)) \wedge ((w+u)>z))$$ [3.4.3-4]

in which one can see the correspondences between parenthesized infix subexpressions and subtrees. The tree is repeated here with the nodes numbered to facilitate referencing the subtrees.



The subtrees starting from the several nodes correspond to the infix subexpressions:

subtree 7:	$(\mathrm{w}+2)$	subtree 8:	$(\mathrm{d}\ast\mathrm{x})$
subtree 4:	$(\mathrm{a}\ast(\mathrm{w}+2))$	subtree 5:	$((\mathrm{d}\ast\mathrm{x})+3)$
subtree 2:	$((\mathrm{a}\ast(\mathrm{w}+2)) < ((\mathrm{d}\ast\mathrm{x})+3))$		
subtree 6:	$(\mathrm{w}+\mathrm{u})$		
subtree 3:	$((\mathrm{w}+\mathrm{u}) > \mathrm{z})$		
tree (1):	$(((\mathrm{a}\ast(\mathrm{w}+2)) < ((\mathrm{d}\ast\mathrm{x})+3)) \wedge ((\mathrm{w}+\mathrm{u}) > \mathrm{z}))$		

Notice how subtree 2 combines subtrees 4 and 5 and how subtree 1—the entire tree—combines subtrees 2 and 3.

Notice also that the nesting depth of parentheses around a subexpression in the infix expression is the same as the depth in the tree of the node heading the subtree for that subexpression. For example, consider the subexpression “d∗x.” To the left of this subexpression there are six left parentheses and two right parentheses, for a nesting depth of $6-2$, or 4. The subtree for this expression begins at node 8, which is at the fourth level of the tree. This relationship applies to every node in any tree and its corresponding subexpression provided that the infix expression is fully parenthesized except for those parentheses removable by rules 1 through 4 (only) in Table 3.4.2-3 of rules for parentheses. For this relationship to apply, parentheses may not be removed by rule 5 (binding order and left-to-right grouping).

3.4.4 Prefix and Postfix Notation

Instead of placing the symbol for a function between its arguments, as in infix notation (see Section 3.4.2), the symbol can be placed before or after the arguments. These notational forms are called *Polish prefix* or *Polish postfix*, respectively, or more simply, *prefix* or *postfix*. These notational forms have the advantages that functions with any number of arguments (not only 1 or 2 as in the case of infix notation) can be represented and that no parentheses are needed. There are also logical advantages that simplify converting expressions automatically from one notational form to another. Prefix and postfix expressions are, therefore, often used internally in compilers which translate infix expressions into computer program instructions for performing the corresponding calculations as well as in other computer programs for processing mathematical expressions in various ways for various purposes. Prefix and postfix notation is occasionally used for analytical purposes, but it is otherwise rarely used manually in practice.

A restriction on the use of prefix and postfix notation is that each function must have a fixed number of arguments. That is, it is not permitted that some function f sometimes has one number of arguments [e.g., f(a, b, c)] and sometimes some other number of arguments [e.g., f(a, b, c, d)].

To convert an expression from standard functional notation to prefix notation, one replaces each reference to the value of a function by the function name (or symbol) followed by its arguments, maintaining their order from left to right [e.g., “f (a, b, c)” becomes “f a b c”]. Variable names following one another must be separated from each other in some way [e.g., with a symbol (such as a space) not permitted within a name], in order to prevent two or more variable names becoming one (e.g., to prevent the three variable names x, y, and z from merging into the single variable name xyz).

For example, consider again expression 3.4.3-1:

and(IsLess(product(a, sum(w, 2)), sum(product(d, x), 3)),
IsGreater(sum(w, u), z)) [3.4.4-1]

The equivalent prefix expression is

$\wedge < * \text{ a} + \text{w } 2 + * \text{ d x } 3 > + \text{ w u z}$ [3.4.4-2]

where the function names have been changed to the same symbols as used in infix notation in Section 3.4.2.

No parentheses are needed because there is only one way of grouping functions and arguments. For example, the last + in the expression above requires two arguments and they must follow the symbol +. That is, the arguments for this + can only be w and u, leading to the term "sum(w, u)" in the following expression mixing prefix and standard functional notation:

$\wedge < * \text{ a} + \text{w } 2 + * \text{ d x } 3 >$ sum(w, u) z [3.4.4-3]

For the same reason, the arguments of the function > must be sum(w, u) and z:

$\wedge < * \text{ a} + \text{w } 2 + * \text{ d x } 3$ IsGreater(sum(w, u), z) [3.4.4-4]

The arguments of the function ∗ must be d and x, for the same reasons as in the cases above. Combining the prefix sequence "∗ d x" into standard functional notation as above, we obtain

$\wedge < * \text{ a} + \text{w } 2 +$ product(d, x) 3 IsGreater(sum(w, u), z) [3.4.4-5]

We continue as above, step by step, for the functions appearing in prefix notation in the remaining expression:

$\wedge < * \text{ a} + \text{w } 2$ sum(product(d, x), 3) IsGreater(sum(w, u), z) [3.4.4-6]

$\wedge < * \text{ a}$ sum(w, 2) sum(product(d, x), 3) IsGreater(sum(w, u), z) [3.4.4-7]

$\wedge <$ product(a, sum(w, 2)) sum(product(d, x), 3) IsGreater(sum(w, u), z)
[3.4.4-8]

$\wedge$ IsLess(product(a, sum(w, 2)), sum(product(d, x), 3)) IsGreater(sum(w, u), z)
[3.4.4-9]

and(IsLess(product(a, sum(w, 2)), sum(product(d, x), 3)),
IsGreater(sum(w, u), z)) [3.4.4-10]

Thus, expression 3.4.4-10 is the only way to interpret the prefix expression 3.4.4-2:

$\wedge < * \text{ a} + \text{w } 2 + * \text{ d x } 3 > + \text{ w u z}$ [3.4.4-2 repeated] so no parentheses are necessary. This applies generally to all prefix and to all postfix expressions.

Postfix expressions differ from prefix expressions in that the function name (or symbol) is placed after the list of arguments [e.g., "f (a, b, c)" becomes "a b c f"]. Otherwise, postfix expressions are formed and interpreted in the same way as prefix expressions. Note that a postfix expression is not the reverse of the corresponding prefix expression, because the order of the arguments of each function is the same (not reversed) in both prefix and postfix notation. The function symbols also appear in different positions relative to the arguments, as illustrated in the following example.

For example, the postfix expression for expression 3.4.4-1 can be constructed as follows. Step by step we replace a reference to the value of a function in standard functional notation by its equivalent postfix expression. The intermediate expressions below mix postfix and standard functional notation.

and(IsLess(product(a, sum(w, 2)), sum(product(d, x), 3)),
    IsGreater(sum(w, u), z)) [3.4.4-1 repeated]

IsLess(product(a, sum(w, 2)), sum(product(d, x), 3))
  IsGreater(sum(w, u), z) $\wedge$ [3.4.4-11]

product(a, sum(w, 2)) sum(product(d, x), 3) $<$ sum(w, u) z $>$ $\wedge$ [3.4.4-12]

product(a, sum(w, 2)) sum(product(d, x), 3) $<$ w u $+$ z $>$ $\wedge$ [3.4.4-13]

product(a, sum(w, 2)) product(d, x) 3 $+$ $<$ w u $+$ z $>$ $\wedge$ [3.4.4-14]

product(a, sum(w, 2)) d x $*$ 3 $+$ $<$ w u $+$ z $>$ $\wedge$ [3.4.4-15]

a sum(w, 2) $*$ d x $*$ 3 $+$ $<$ w u $+$ z $>$ $\wedge$ [3.4.4-16]

a w 2 $+$ $*$ d x $*$ 3 $+$ $<$ w u $+$ z $>$ $\wedge$ [3.4.4-17]

Compare the postfix expression 3.4.4-17 with the equivalent prefix expression 3.4.4-2 from earlier:

$\wedge$ $<$ $*$ a $+$ w 2 $+$ $*$ d x 3 $>$ $+$ w u z [3.4.4-2 repeated]

Notice that neither is simply the reverse of the other.

It is suggested that the reader convert the postfix and prefix expressions above into trees. The two trees will, of course, be the same as the complete tree in Section 3.4.3. The conversion process corresponds basically to the reverse of the transformation above from standard functional notation to postfix notation. It is probably easiest to start from the left of a postfix expression or from the right of a prefix expression (i.e., the bottom of the tree) and to work to the other end of the postfix or prefix expression. This effectively works upward in the tree. At each step of constructing the tree, work next with the leftmost function in the remaining postfix expression or with the rightmost function in the remaining prefix expression.

3.4.5 Tabular Notation

Many types and formats of tables are used to define functions. Their advantages are ease of writing, reading, and checking against nonmathematical statements of requirements, intentions, desires, and so on. A common characteristic of all tables is that they subdivide the range of concern by conditions and then give, for each condition, the value or an expression for the value of the function being defined. Beyond this common feature, they exhibit a wide variety of forms—one- and two-dimensional—as well as a variety of structures.

Tables can be illustrated with the example from previous sections. The infix expression 3.4.2-28 for this function was given earlier as

$$a * (w+2) < d*x+3 \wedge w+u > z \qquad [3.4.5\text{-}1]$$

This function could have been defined by the following table:

Condition		
$a*(w+2)<d*x+3$	$w+u>z$	Value of Function
false (no)	false (no)	false
false (no)	true (yes)	false
true (yes)	false (no)	false
true (yes)	true (yes)	true

In this table, all possible combinations of the conditions are listed, and for each combination of the conditions, the value of the function is given. The table includes one column for each condition and one column for the value of the function being defined. A header row contains the conditions. Each subsequent row contains values of the conditions and the value of the function for those values of the conditions.

Normally, the values of the conditions in the rows will be mutually exclusive and exhaustive, that is, exactly one row will always apply. Two rows will never apply in the same situation, but one always will. If this restriction is not met, the table might not define the function. If two or more rows ever apply in the same situation, and the values in the final column of those rows differ, the table is self-contradictory. If no row applies in some situation, the value of the function is not defined for the missing combination of values of the conditions.

In the table above, only two conditions appear. In situations arising in practice, such tables may contain a large number of conditions and, correspondingly, rows. In the table above, only one function is being defined, so only one column is present for the values of the function. In practice, one can define several functions with one table, in which case there will be one function value column for each function being defined.

The function defined in the table above can, alternatively, be defined with the following differently structured table with the same meaning:

	$a*(w+2)\geq d*x+3$	$a*(w+2)<d*x+3$
$w+u\leq z$	false	false
$w+u>z$	false	true

In a table of this type, any number of conditions may appear across the top, in different columns in the top header, and any number of conditions may appear down the left side, in different rows in the side header. The value of the function—or an expression for that value—is placed in the cell at the junction of the row and column whose conditions are true.

The conditions across the top should normally be mutually exclusive and exhaustive; that is, exactly one of these conditions should apply in every situation. Similarly, the conditions down the side header should be mutually exclusive and exhaustive. If these restrictions are not met, the table might either be self-contradictory or might fail to define the value of the function in some situation.

Many other layouts for tables defining a function (i.e., for writing an expression) can be found in the relevant literature. The term *decision table* is often applied to some types of tables. Any convenient structure can be used in practice, provided that its meaning is well defined.

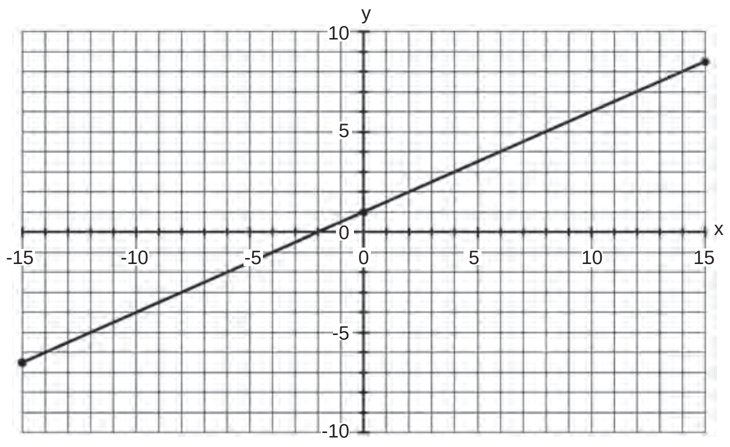
Remember that a table is another notational form for a mathematical expression. One should always be able to transform an expression in tabular form into one of the notational forms described in earlier sections. Use tabular notation when the people specifying the function find it easier to write, read, analyze, and verify than other notational forms and when a table facilitates communication between the people involved in the application problem and its solution.

3.4.6 Graphical Notation

Graphical notation differs from the other types of notation described above in that, inherently, it cannot be absolutely precise and accurate. Visualizing functional relationships between variables often helps people to understand those relationships much more clearly than other notational forms do. For that reason it is important in practice and used frequently. It can be used to aid thinking and preparing for more detailed analyses, but its direct application to analysis is limited because of its inherent lack of complete precision and accuracy.

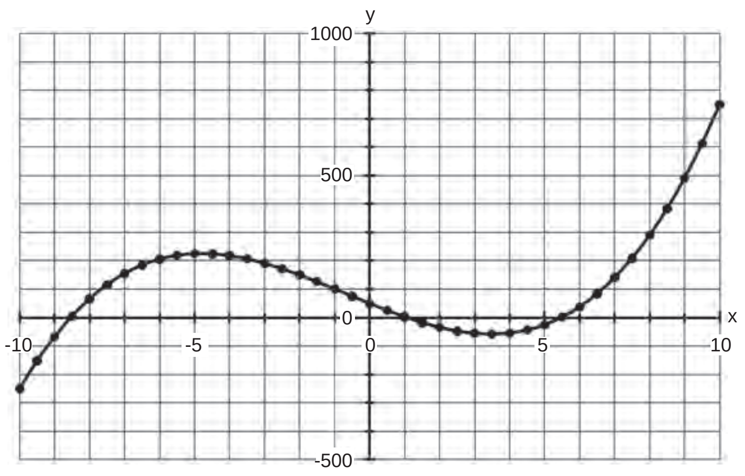
The simplest form of a graph is a two-dimensional plot on a piece of paper or similar medium. Horizontal distances represent values of one variable, and vertical distances represent values of a second variable. A reference point, called the *origin*, from which distances are measured, is defined and two straight lines, the *axes* along which distances corresponding to the values of the two variables involved are measured, are drawn through the origin. The two axes are usually drawn perpendicular to each other. Measurements along the two axes may, but need not, be to the same scale. The two distances thereby defining the position of each point in question are called the *Cartesian coordinates* of the point, named after the French mathematician *René Descartes*, to whom such graphs are attributed.

Example 1 A graph of the function $y=x/2+1$ is plotted below. For several values of x the corresponding values of y are calculated from this equation and the point corresponding to each pair of values for x and y is marked. The points are then connected with a smooth line: in this case, a straight line.



[3.4.6-1]

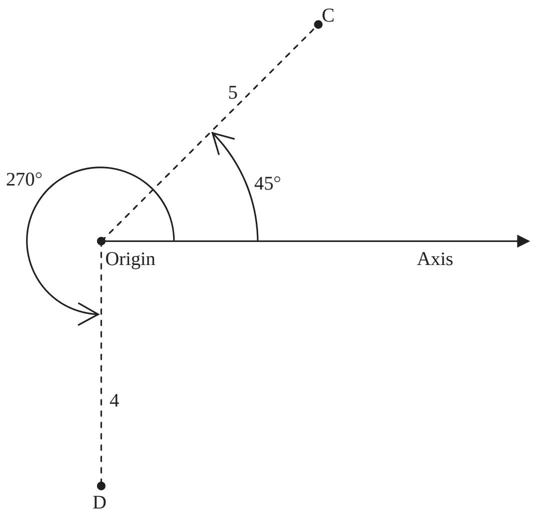
Example 2 Below is a plot of the function y=x^3+2∗x^2−50∗x+50. It is made in the same way as the graph in Example 1 was made, but here, more points should be marked because the graph is not a straight line. The greater the number of points marked, the better the accuracy of the graph between them will be.



[3.4.6-2]

Three-dimensional graphs can be conceived and, to a limited extent, modeled with various solid materials. They can be computed and various perspective views displayed on two-dimensional media such as paper, computer monitors, or projection screens. In principle, higher-dimensional graphs can be generated similarly and perspective views displayed.

The Cartesian coordinate system is not the only basis for graphs in mathematics. For some types of relations between variables, graphs based on *polar coordinates* are more convenient. A polar coordinate system is defined by a reference point (the origin) and an infinitely long straight line (the axis) beginning at the origin. The polar coordinates of a given point are (1) the length of the straight line from the origin to the given point and (2) the angle between that line and the axis. Any point can be represented with polar coordinates by a nonnegative length and an angle greater than or equal to $0°$ and less than $360°$.



[3.4.6-3]

For example, the polar coordinates of point C in the graph above are 5 and $45°$, typically written $(5, 45°)$. The polar coordinates of point D are (4, $270°$). For mathematical analyses the angles are usually measured in radians instead of degrees ($°$). If one superimposes the axes of a Cartesian coordinate system and the origin and axis of a polar coordinate system on one another, equations for converting one type of coordinates to the other type can easily be derived by geometric considerations.

Polar coordinates are especially convenient for circularly symmetric relationships. For example, if the distance from the origin to a point in question is called d, then the equation for a circle of radius r centered at the origin is simply $d=r$, independent of the angle. The equation for the same circle in Cartesian coordinates is $x^2+y^2=r^2$.

Polar coordinates can be generalized for three- and higher-dimensional spaces. The coordinates of a point are then the distance between the origin and the point and the two or more angles defining the direction of the line between the origin and the point. Still other coordinate systems can be and sometimes are defined for particular analyses and purposes.

One simple way to draw graphs is with a pencil (or pen) and ruler on paper already printed with lines representing the coordinates. For Cartesian coordinates, horizontal and vertical grid lines as in the examples above are preprinted. For polar coordinates, concentric circles centered on the origin (corresponding to different distances from the origin) and straight lines radiating from the origin (corresponding to different angular coordinates) are preprinted.

The accuracy with which a point can be placed is limited by the accuracy of the ruler, the width of the markings on it, the thickness of the pencil point, parallax between the markings on the ruler, the pencil, and the paper, and the eyesight of the person placing the point. Visual interpolation between markings on the ruler and the preprinted coordinate lines also limits accuracy. The limited accuracy with which the lines were printed on the paper, the thickness of the lines, and warping of the paper due to variations in temperature and humidity also reduce the accuracy of a graph.

In principle, the accuracy with which graphs can be drawn with fine mechanical assistance, digital computation and displays, and so on, can always be improved, but deviations from the ideal can never be eliminated completely. Therefore, despite their very considerable usefulness, graphs can never replace notational forms based on discrete symbols as described in previous sections of the chapter. Graphs play an important role in human interpretation of mathematical expressions, but in practice graphs remain a representation of an unrealizable ideal. Therefore, graphs do not play the same central role in the Language of Mathematics as do the notational forms based on discrete symbols discussed previously.

3.4.7 Figures, Drawings, and Diagrams

In addition to graphs as described in Section 3.4.6, many other types of figures, drawings, diagrams, and pictorial representations are used to illustrate mathematical relationships, functions, expressions, and so on, in visual form. Some of the common forms are technical and engineering drawings, maps, nautical charts, bar charts, and pie charts. Such figures can be drawn to scale, show perspective views, or be abstract illustrations indicating only general relationships between the various subcomponents. Like graphs as discussed earlier, all these figures facilitate human understanding through visualization, but they lack the absolute precision and accuracy inherent in the Language of Mathematics. Therefore, they have the same limitations as graphs, as pointed out in Section 3.4.6.

The advantages of visual illustrations in conveying information and understanding derive largely from the fact that vision is by far the fastest input medium to the human brain and that the majority of the brain is used for processing visual input. Visual input coming from a person’s two eyes is stereoscopic and therefore three-dimensional in nature. Text, however, being a single sequence of symbols, is only one-dimensional, so does not take good advantage of the potential of visual input to the human brain. The difference is captured in the old saying “a picture is worth a thousand words.” Consider, for example, a map vs. a verbal description of the road network and the spatial relationships between locations of places of interest.

Geometric figures are probably the first examples of nontextual writing in mathematics. They were certainly a well-established part of the mathematical language of geometry in early Greek mathematics and Euclidian geometry. Today, however, as graphs, it is better to consider them as very valuable adjuncts to mathematical language, but not to rely on them for accuracy, precision, and detailed analyses and proofs.

Like graphs as presented in Section 3.4.6, figures, diagrams, and drawings of all types should be viewed as adjunct representations of the mathematical model or even part of the interpretation of a mathematical model oriented to a particular application area. In some cases, they are perhaps best viewed as part of the application language itself and not of the Language of Mathematics at all.

Visual representations have not only advantages but also have pitfalls. For example, consider a geometrical figure illustrating a general theorem to be proved. The brain can notice some particular characteristic of the figure and subconsciously use that characteristic in the proof. Such a proof is not applicable to the general case and the theorem might not be true in general, despite the "proof." Many students have fallen into this trap in geometry classes in school.

Visual representations also are prone to the well-known phenomena of optical illusions. Some of the illusions are: Parallel lines clearly appear to the observer not to be parallel. Straight lines appear clearly curved. Dots appear white or gray, depending on which one the observer looks at. Two circles, one surrounded by small circles, the other surrounded by larger circles, clearly appear to be of different sizes, despite the fact that they are of the same size. The list of such optical illusions can be extended almost endlessly. After looking at a few dozen of these optical illusions, one realizes that visual perceptions are not reliable, even if they are totally convincing.

Engineering drawings of three-dimensional objects are typically projections of the object onto three perpendicular planes. Perspective views are also often used, especially in architecture. Drawings in these fields are almost always to scale, but drawings not to scale can be useful (e.g., to highlight certain features or aspects of the object being illustrated).

Some examples of figures, drawings, and illustrations of different types appear elsewhere in the book. See, for example, the drawing of a beam in Section 2.11, the pictures of dice and the drawings of urns containing colored balls in Sections 4.6.1, 4.6.3, and 4.6.4, and the drawings of a water reservoir and supply pipe with valve in Sections 8.10.1 and 8.11.1.

3.4.8 Notation for Series and Quantification

Often, one wants to write an expression adding, for example, a number of values, in which the number of values to be added is given by the value of some other variable or expression. None of the notational forms defined above satisfies this need adequately. One of the common examples is adding the values of expressions or array variables: for example,

$$\mathrm{x(1)} + \mathrm{x(2)} + \cdots + \mathrm{x(n)} \qquad [3.4.8\text{-}1]$$

where the value of the variable n is an integer greater than or equal to 1 and the x(…) are array variables. (Array variables are treated extensively in Section 4.1.2.) Because such expressions arise frequently in both theoretical and applied mathematics, expression 3.4.8-1 is often written more concisely, and unambiguously, as follows:

$$\sum_{i=1}^{n} x(i) \qquad [3.4.8\text{-}2]$$

where the capital Greek letter Σ (sigma) represents the function "sum." The expression "x(i)" is a template for an expression and the variable "i" is a special variable used to generate values to be applied to the template expression. The first value of the special variable is given by the expression to the right of the symbol = in the bottom line, and the last value is given by the expression above the Σ symbol. The last value is normally greater than or equal to the first value. All integer values equal to and between the first value and the last value inclusive are substituted for the special variable in the template expression to form the actual individual expressions whose values are to be added together. That is, the Σ expression above is defined to mean the expression $x(1) + x(2) + \cdots + x(n)$. Such an expression is often called a *series*.

The special variable (i in the example above) is sometimes called a *running* or *dummy variable*. In more formal mathematical terminology, it is the variable of "quantification": that is, the variable being quantified, the variable to which quantities (values) are being assigned. The variable of quantification is also called a *bound variable*, because its values are determined (bound, set) within the expression and referred to only within the Σ expression. In contrast, the values of n and the array variables x(…) must be determined externally to the Σ expression; these variables are often called *unbound* or *free variables*.

Note especially that expression 3.4.8-2 as a whole does not depend on any external variable named "i." This can be seen most clearly by considering the simpler example

$$\sum_{i=2}^{5} x(i) \qquad [3.4.8\text{-}3]$$

which, by the definition of this Σ notation, means

$$x(2) + x(3) + x(4) + x(5) \qquad [3.4.8\text{-}4]$$

Note that there is no reference to any variable "i" in expression 3.4.8-4. In the expression

$$\sum_{k=2}^{5} x(k) \qquad [3.4.8\text{-}5]$$

the running variable has a different name, but the meaning of the expression is defined to be the same [i.e., $x(2) + x(3) + x(4) + x(5)$]. The "running variable" is not really a variable at all, hence its alternative name "dummy variable." The name of a running variable can always be changed to any name not otherwise appearing within the quantified expression.

Consider the expression

$$\mathrm{i}+\sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{x}(\mathrm{i}) \qquad [3.4.8\text{-}6]$$

The first variable named "i" is not the same variable as the running variable "i" in the Σ expression. The first variable named "i" is a free, unbound variable, whose value does not influence the value of the Σ term in expression 3.4.8-6. The distinction between the two "variables" named "i" can be made clearer by renaming the running variable in expression 3.4.8-6 as in the earlier example, to obtain

$$\mathrm{i}+\sum_{\mathrm{k}=1}^{\mathrm{n}} \mathrm{x}(\mathrm{k}) \qquad [3.4.8\text{-}7]$$

In expression 3.4.8-7 the distinction between the free, unbound variable named "i" and the bound, quantified variable named "k" is explicit. Because using the same name for bound and unbound variables in an expression can be confusing, it is strongly recommended that different names be used for them. Although this is not a grammatical rule for writing in the Language of Mathematics, it is an accepted rule of good style. It helps human readers to understand the expressions in question.

In the general case, the initial and final values of the running variable can be given by any expressions. The template term can be any expression. Typically, the template expression will refer to and depend on the running variable as in the examples above, but it is not required to do so; for example,

$$\sum_{\mathrm{i}=2}^{5} 1 \qquad [3.4.8\text{-}8]$$

which, by the definition above, means 1+1+1+1. The first "1" is the template expression evaluated for i=2; the second "1," for i=3; the third "1," for i=4; and the last "1," for i=5. The fact that the running variable "i" does not appear in the template term does not affect the application of the definition of the meaning of a Σ expression. Such Σ expressions, in which the running variable does not appear in the template term, are not common in practice, but they do sometimes arise (e.g., after simplification of more complicated expressions or after evaluating expressions in special situations).

The examples above all involved series for sums. Series for products are also useful. The capital Greek letter Π (pi) is typically used for series for products:

$$\prod_{\mathrm{i}=1}^{4} \mathrm{x}(\mathrm{i}) \qquad [3.4.8\text{-}9]$$

which means x(1)∗x(2)∗x(3)∗x(4). Series based on other functions are also useful, such as the logical and (∧) and or (∨), and the set functions intersection (∩) and union (∪). Some examples are:

$$\bigwedge_{\mathrm{i}=1}^{\mathrm{n}-1} \mathrm{name}(\mathrm{i})<\mathrm{name}(\mathrm{i}+1) \qquad [3.4.8\text{-}10]$$

$$\bigvee_{i=1}^{n} \text{name}(i)=\text{customername} \qquad [3.4.8\text{-}11]$$

$$\bigcap_{i=1}^{n} S(i) \qquad [3.4.8\text{-}12]$$

$$\bigcup_{i=1}^{n} S(i) \qquad [3.4.8\text{-}13]$$

In the traditional mathematical literature, several different special notational forms for series involving the logical functions $\wedge$ and $\vee$ are common. For historical reasons, different symbols were and still are used:

- $\forall$, an upside-down "A," corresponding to "for all" in English, instead of $\wedge$ for "and"
- $\exists$, an upside-down "E," corresponding to "there exists" in English, instead of $\vee$ for "or"

The former ($\forall$, "for all") is called *universal quantification*. The latter ($\exists$, "there exists") is called *existential quantification*. As with sums and products in the earlier examples above, universal and existential quantification amount to the repeated application of a function—in these cases, of the logical "and" and the logical "or," respectively. Some of the notational forms in which $\forall$ and $\exists$ often appear are:

- $\exists i(\text{name}(i)=\text{customername})$
- $(\exists i)\text{name}(i)=\text{customername}$
- $\forall 1\leq i\leq n-1(\text{name}(i)<\text{name}(i+1))$
- $(\forall 1\leq i\leq n-1)\text{name}(i)<\text{name}(i+1)$
- $(\forall 1\leq i\leq n-1, \text{name}(i)<\text{name}(i+1))$
- $(\forall i\in\mathbb{Z}, 1\leq i\leq n-1: \text{name}(i)<\text{name}(i+1))$

Many other variations of the forms above are also seen in mathematical texts involving logic.

Above, $\mathbb{Z}$ is the set of integers (whole numbers). Sets are the subject of Section 4.1.1. Here and elsewhere in this section it suffices to read the subexpression $i\in\mathbb{Z}$ as "i in the set $\mathbb{Z}$," "i is an integer," or "integers i."

Most of the notational variations above for series and quantification have certain disadvantages; for example, the end of the template is not always clear in the Σ notation, and it is not always completely clear which are the quantified variables in some of the $\forall$ and $\exists$ expressions of the types listed above. Furthermore, the variety of notational forms can confuse newcomers and students. A more general notational form capable of expressing all types of quantification has, therefore, been introduced in the last few decades. Although it has not yet achieved universal acceptance, it is used extensively in certain branches of mathematics. It is possible to define each of the various notational forms presented above in terms of this more general notation, which has the following general form:

$$[\textit{function}\,\mathrm{i, j, k, \ldots : range(i, j, k, \ldots) : TemplateTerm(i, j, k, \ldots)}] \qquad [3.4.8\text{-}14]$$

where

- *function* is the name or infix symbol for the function being repeatedly applied
- i, j, k, … is a list of the quantified variables (the "bound," "running," "dummy" variables)
- range(i, j, k, …) is a Boolean function defining the values for the quantified variables
- TemplateTerm(i, j, k, …) is the template for forming the actual terms to which the function is to be applied

These, and only these, are the essential elements of any quantified expression. Throughout the book, quantified expressions are usually written in the form of expression 3.4.8-14.

Typically, the function in the series or quantified expression is commutative and associative, so the order of the arguments has no effect on the resulting value and the grouping of the implied parentheses does not matter [e.g., mathematical addition (+) is commutative because x+y always has the same value as y+x, and mathematical addition (+) is associative because ((x+y)+z) always has the same value as (x+(y+z))]. If either property does not apply to the function in question, the order of the terms and the grouping of parentheses must be defined. Usually, the range begins by indicating a set of values for each quantified variable, and if this set is ordered (as, e.g., the set $\mathbb{Z}$ of integers), this same order is applied to the terms formed from the template. If the function is not associative, implied parentheses are usually assumed to group the terms from left to right, the normal convention for infix expressions.

Each combination of values of the quantified variables satisfying the range [i.e., for which the value of the Boolean function range(i, j, k, …) is true] is used once to form an actual term from the template term. For example, the series expression

$$\sum_{k=2}^{5} x(k) \qquad [3.4.8\text{-}15]$$

is written in the general format for quantified expressions as

$$[+\ k : k\in\mathbb{Z} \wedge 2\leq k\leq 5 : x(k)] \qquad [3.4.8\text{-}16]$$

The series

$$\overset{n-1}{\underset{i=1}{\wedge}} \mathrm{name}(i)<\mathrm{name}(i+1) \qquad [3.4.8\text{-}17]$$

can be written in the general format for quantified expressions as

$$[\wedge\ i : i\in\mathbb{Z} \wedge 1\leq i\leq n-1 : \mathrm{name}(i)<\mathrm{name}(i+1)] \qquad [3.4.8\text{-}18]$$

or as

$$[\forall\ i : i \in \mathbb{Z} \wedge 1 \leq i \leq n-1 : \text{name}(i) < \text{name}(i+1)] \qquad [3.4.8\text{-}19]$$

If expression 3.4.8-18 is true, every term “name(i)<name(i+1)” in the range of quantification is true [i.e., the term is true “for all” values of i in the range of quantification], so expression 3.4.8-19 is true. Similarly, if expression 3.4.8-19 is true, then expression 3.4.8-18 is true (i.e., each expression is true if and only if the other is true); the two expressions always have the same value. Thus, the logical functions “and” and “for all” are the same.

In the same way, the expressions

$$\exists i,\ 1 \leq i \leq n(\text{name}(i) = \text{customername}) \qquad [3.4.8\text{-}20]$$

and

$$\bigvee_{i=1}^{n} \text{name}(i) = \text{customername} \qquad [3.4.8\text{-}21]$$

and

$$[\vee\ i : i \in \mathbb{Z} \wedge 1 \leq i \leq n : \text{name}(i) = \text{customername}] \qquad [3.4.8\text{-}22]$$

and

$$[\exists\ i : i \in \mathbb{Z} \wedge 1 \leq i \leq n : \text{name}(i) = \text{customername}] \qquad [3.4.8\text{-}23]$$

always have the same value.

If expression 3.4.8-22 is true, then at least one of the terms is true (i.e., there exists some value of i in the range of quantification for which a term is true). Therefore, expression 3.4.8-23 is true. The reverse also applies, so the two expressions are equivalent. Thus, the logical functions “or” and “there exists” are the same.

In summary, a general notational form for series and quantified expressions of all of the types mentioned in this book is

$$[\textit{function}\ i, j, k \ldots : \text{range}(i, j, k \ldots) : \text{term}(i, j, k \ldots)] \qquad [3.4.8\text{-}24]$$

For descriptions of the several parts of this format, see earlier parts of this section. The many other notational forms for series and quantified expressions can be defined in terms of this more general format. Some of the other notational forms for specific types of quantified expressions are shorter and therefore convenient, but they should be used only when they are defined in terms of the more general format and when both the writer and the reader are fully aware of that definition. Those other notational forms are best thought of as abbreviations or colloquialisms in the Language of Mathematics.

The reader might ask at this point what the value of a quantified expression is when the range of quantification is empty. Although this is, in the final analysis, a matter of definition, one definition stands out as particularly sensible. It derives from the observation that the value of a series of n terms can be expressed as the value of the function applied to a series of n−1 of the terms and the other remaining term. If this relationship is to apply to the situation in which n=1, then the value of a series of 0 terms must have the characteristic that if the function is applied to it and to a single term, the result is the value of the single term itself. In mathematics, such a value is called the *identity element* with respect to the function. For example, $0+x=x$ for any number x, so 0 is the identity element with respect to addition (+). The identity element with respect to multiplication is 1, because $1*x=x$ for any number x. Similarly, the identity element for the logical and ($\wedge$) function is true, because $\text{true}\wedge x=x$ for any Boolean value of x. The identity element for the logical or ($\vee$) function is false, because $\text{false}\vee x=x$ for any Boolean value of x. In general, the value of an empty series is the identity element with respect to the function in the series.

Quantification in the Language of Mathematics refers to many occurrences of the same type of object and, therefore, often corresponds to plural forms in English. Particularly when other words typical of quantification (such as "all," "every," "some," "any," etc.) occur in the same phrase with a plural noun or pronoun, one should consider translating plural forms with quantified expressions. See Section 7.3 for further discussion of this topic.

3.4.9 Specialized Notational Forms for Certain Expressions

The preceding sections in this chapter define notational forms for functions and expressions. Additional notational forms for structures such as sets, arrays, sequences, series, and quantified expressions are defined in Sections 3.4.8, 4.1.1, 4.1.2, and 4.1.3. These forms constitute the notational foundation of the Language of Mathematics.

In addition and like other languages, the Language of Mathematics has specialized abbreviations and idioms of various types. Some are in widespread use in mathematics, whereas others are used only in particular specialties. They are used because they are convenient; they facilitate writing, analysis, and communication between mathematicians. All of them can be defined in terms of the fundamental notational forms for functions, expressions, sets, arrays, sequences, and series. Linguistically, these other notational forms can be viewed as synonyms for the normal, usual forms they are defined to stand for.

One of these types of idiomatic expressions was defined in Section 3.4.2 on infix notation. An expression of the form

$$a \le b < c \le d \qquad [3.4.9\text{-}1]$$

is defined to mean

$$(a \le b) \wedge (b < c) \wedge (c \le d) \qquad [3.4.9\text{-}2]$$

and correspondingly for expressions of this form with other lengths. This abbreviation is in general use in all areas of mathematics.

Often in mathematics one needs to select one value or another depending upon whether or not a Boolean variable or expression is true. This need arises especially in definitions, for example, of functions. The traditional mathematical notation for such a selection is

$$X = \begin{cases} Y, & \text{if B} \\ Z, & \text{otherwise} \end{cases} \qquad [3.4.9\text{-}3]$$

This notational form can be extended for three or more Boolean conditions with the corresponding values. A significant disadvantage of this notation is that it is inconvenient for embedding within larger expressions. It is also inconvenient to enter via a keyboard.

The same need is met in some computer programming languages by the subexpression

$$\text{if B then Y else Z} \qquad [3.4.9\text{-}4]$$

which, after enclosing in parentheses, can be embedded in any expression. However, neither this notational form nor variations of it using special symbols has found general acceptance in the Language of Mathematics. Defining a function [e.g., choice(B, Y, Z)], using standard functional notation is probably the simplest way to provide such a selection in a mathematical model. Several other possibilities can be constructed, but they are generally clumsier and they do not fit so well into the general spirit and culture of the Language of Mathematics.

Another specialized notational form describes the relationship between a function and its domain and range:

$$\mathrm{f} : \mathbb{X} \rightarrow \mathbb{Y} \qquad [3.4.9\text{-}5]$$

It corresponds to the English statement “f is a function with domain $\mathbb{X}$ and range $\mathbb{Y}$.” It is used in all areas of mathematics. It can be defined formally and mathematically by the quantified expression

$$[\wedge\ \mathrm{a} : \mathrm{a}\in\mathbb{X} : \mathrm{f(a)}\in\mathbb{Y}] \qquad [3.4.9\text{-}6]$$

See the latter part of Section 3.4.8 and Section 4.1.1.

In probability theory, one often writes something of the form Pr{Bexp}, where Bexp is a Boolean expression including references to a random variable x. Although this looks like a reference to the value of a function Pr applied to the Boolean value of the expression Bexp, it is not. Instead, it is an abbreviation or idiomatic expression defined to mean the probability of the set defined by the Boolean expression Bexp: that is, $\mathrm{p}([\cup\, \mathrm{x} : \mathrm{x}\in \mathrm{S} \wedge \mathrm{Bexp} : \{\mathrm{x}\}])$, where p is a function to be applied to the value of the argument $[\cup\, \mathrm{x} : \mathrm{x}\in \mathrm{S} \wedge \mathrm{Bexp} : \{\mathrm{x}\}]$. See Sections 3.4.8 and 4.6.1 for further information on this particular notational form. This example illustrates that when working in a particular specialized area of mathematics, one must become familiar with the particular abbreviations and idioms defined and used in that area. They will all be based on, and defined in terms of, the fundamental notational forms in the Language of Mathematics, but they can exhibit their own linguistic variations.

Another notational form amounts to a different way of writing a function applied to an argument: f.x is defined to mean f(x). Here, the symbol . is viewed as an infix function meaning the application of a function to an argument (e.g., f.x means the application of the function f to the argument x). This notation distinguishes between an array variable named f and a function named f (cf. Section 4.1.4). The notation f.x is used extensively in some particular areas of mathematics, but the older notation f(x) is still used more widely.

An example of a notation used only in the specialized area of mathematics dealing with computer programs and segments thereof is

$$\{V\}\ S\ \{P\} \qquad [3.4.9\text{-}7]$$

where V and P are Boolean functions and S is the function representing the effect of executing a segment of a program. This expression is interpreted to mean "if V is true before S is executed, then P will be true afterward" in the language of the application domain. Mathematically, $\{V\}$ S $\{P\}$ is formally defined to mean

$$[\wedge\ d : V.d : P.(S.d)] \qquad [3.4.9\text{-}8]$$

The notation for this quantified structure is defined in Section 3.4.8. In this definition, the symbol . is used as the infix operator as presented in the paragraph above. Visually, the notation $\{V\}$ S $\{P\}$ suggests the interpretation in the application domain much more directly than does the more formal expression $[\wedge\ d : V.d : P.(S.d)]$. The notational form $\{V\}$ S $\{P\}$ is, therefore, much easier to recognize and to work with when manipulating expressions. This is the reason for defining this specialized notation.

In specific areas of mathematics additional symbols are used: for example, ∂ (partial derivative in differential calculus), $\int$ (integration in integral calculus), and ∇ (divergence and gradient in vector calculus).

In some cases in mathematics, the same symbol is used for different things. The use of the symbol $\times$ for both the product of two numbers and for the Cartesian product of sets is mentioned in Section 3.4.2. Potentially more confusing is the use of the symbol $\supset$ for both subset (As$\supset$Bs meaning that Bs is a subset of As) and logical implication (A$\supset$B meaning that A implies B). The symbol $\supset$ was used in the development of one branch of logic and is still used in modern texts in that specialized area. In many other application areas of logic, implication is written as Ab$\Rightarrow$Bb, meaning that Ab implies Bb. Confusion can and sometimes does result from the fact that if the Boolean expression Ab corresponds to the set As and the Boolean expression Bb corresponds to the set Bs and Ab$\Rightarrow$Bb, the set As is a subset of the set Bs, which can be written Bs$\supset$As. However, using $\supset$ as the implication symbol in logic, one would write Ab$\supset$Bb, just the reverse of Bb$\supset$Ab.

The lesson to be learned here is that one should always be completely aware of what the symbols mean in the context of the mathematical specialty within which one is working and the context of the application. In this regard, the Language of Mathematics is like other languages, too. For example, the word *jig* in English can mean either a type of dance or a mechanical fitting to guide or hold a tool, depending on the context. In mathematics, the words *ring* and *field* have very specific and well-defined meanings not related to their meanings in general English usage.

Restrictions on choices of additional notational forms in mathematics and guidelines for their interpretation are discussed in Section 6.5.

3.4.10 Advantages and Disadvantages of the Different Notational Forms

In this section the relative advantages and disadvantages of the several notational forms described in Sections 3.4.1 through 3.4.9 are identified. They are then summarized and compared in Table 3.4.10-1 below.

The notational forms for series and quantification described in Section 3.4.8 are not included explicitly in this comparison because they represent extensions to infix and standard functional notation and the combination of them. It is left as an exercise for the reader to identify the relative advantages and disadvantages of the different notational forms for series and quantification mentioned in Section 3.4.8.

In the following paragraphs an abbreviation is given for the name of each notational form. In Table 3.4.10-1, the notational forms are referred to by these abbreviations.

Standard Functional Notation (SFN) Standard functional notation is a compact and simple, yet general notational form which can represent any expression. It reflects the compositional structure of the expression directly, but with deep nesting the overall view can become difficult to follow. For this reason standard functional notation alone is not used extensively in practice. It requires parentheses only to enclose the argument list of each function and commas to separate the individual arguments from one another. Otherwise, no special symbols are needed. An expression in standard functional notation is a sequence of letters, parentheses, and commas only and is, therefore, easy to write by hand or with a standard keyboard.

Infix Notation (INF) Infix notation represents a function by a symbol for the function between its two arguments. Because of this structure, it is suitable only for functions with exactly two arguments, a severe limitation in general. Many of the most commonly used functions have two arguments (e.g., sum, difference, product, and quotient) and for expressions involving only such functions, infix notation is widely used. Nesting is represented by parentheses and is, therefore, generally clearer than with standard functional notation, but deep nesting can still be difficult to follow. An expression in infix notation is a sequence of letters, parentheses, and special symbols and is, therefore, relatively easy to write by hand or with a standard keyboard, provided that all special symbols needed are available in the editing program or system being used.

Combination of Infix and Standard Functional Notation (SFN+INF) Expressions can be written using infix notation for the most common functions of two arguments and standard functional notation for the others. Such a combination avoids the limitation of infix notation while retaining its advantages. For this reason, a combination of infix and standard functional notation is the most commonly used notational form for expressions.

Tree Notation (Tree) The main advantage of tree notation is that the compositional structure of the expression is especially clear, regardless of the nesting depth. This clarity is gained at the expense of a two-dimensional representation, with the accompanying difficulty of writing it by hand or with a standard keyboard. Because of the two-dimensional representation, a much larger area is required for an expression than with a one-dimensional notational form. For these reasons, tree notation is rarely, if ever, used in practice. Because of the clarity with which tree notation illustrates the compositional structure of an expression, it is very useful for teaching and learning.

Prefix and Postfix Notation (PPF) By placing each symbol for a function before or after its arguments instead of between them, both the restriction of infix notation to functions with two arguments and the need for parentheses can be eliminated. This makes it easier to analyze an expression to determine its compositional structure, but the lack of the visual clues provided by parentheses and commas in infix and standard functional notation make it more difficult for the human reader to discern this structure. Probably for this—and historical—reasons, prefix and postfix notation is rarely used in practice manually by humans. It is used within automated expression processors, such as compilers for computer programs. By eliminating the punctuation in infix and standard functional notational forms, expressions in prefix and postfix notation are shorter.

Tabular Notation (TBL) Basically, tabular notation is equivalent to an expression that has been factored in a certain way, with the several subexpressions arranged in a table. The result is an improved view of the compositional structure of the expression, an improved view of some of the functional dependencies, and a reduction in the length of each subexpression to be examined by a human reviewer. A human reader, especially one more familiar with the application area than with mathematics, will often find the tabular notation easier to read, understand, examine, and verify than the other notational forms. In short, a table is a human reader-friendly notational form for a mathematical expression. Because of the two (or even three)-dimensional format (and sometimes the repetition of subexpressions in several cells of the table), a table is usually less compact than an equivalent one-dimensional expression. In some application areas, tables are frequently used, whereas in others, they are rarely encountered.

Graphical Notation (GRA) The main advantage of a graph is, as with a table, its human reader-friendly form. The functional relationship between the function value and one or more arguments is presented visually very clearly. The compositional structure of the function is not shown at all. The usual graph is two-dimensional, but two-dimensional perspective views of three-dimensional graphs can be generated. Three-dimensional graphs are also sometimes constructed. A critical limitation of graphs for mathematical purposes is their lack of absolute accuracy and precision. A person can beneficially use a graph in considering what to look for or how to prove a theorem, but the graph cannot, for example, be used directly to prove the theorem. A graph can be used to determine an approximate value of a function, but never its exact value. In other words, a graph can be useful in planning a mathematical analysis, but it cannot be used in the formal analysis itself. Remember that a graph's usefulness derives only from human visual psychology. Graphs lack the precision of the other notational forms: Lines have nonzero width, measurement accuracy is limited and the paper on which they are printed can warp and stretch due to variations in temperature and humidity, folding, curling into rolls, and physical forces applied.

Figures, Drawings, and Diagrams (FDDs) Figures, drawings, and diagrams are much like graphs except that they refer to an object in the application area and not to a mathematical function or expression itself. They are less abstract and more concrete than a mathematical expression. Special symbols may or may not be needed, depending on the nature of the figures, drawings, or diagrams and the object to be represented.

Specialized Notational Forms (SPCs) Specialized notational forms are abbreviations for lengthier subexpressions arising frequently in certain areas or applications. Each abbreviation must be defined precisely and unambiguously. The intention of such an abbreviation is to improve human readability, even when an insight into the structure of the subexpressions they represent may be lost. Depending on the particular abbreviation, an insight into functional dependencies may or may not be lost. Many specialized notations are one-dimensional, but two-dimensional notations (e.g., tables) are not excluded. Specialized notations are normally defined in such a way that they are easy to write by hand, but whether or not they are easy to enter with a keyboard depends on the particular abbreviation and the symbols within it.

Of the different types of notational forms, standard functional notation, infix notation, combined standard functional and infix notation, prefix and postfix notation, and usually specialized notation are one-dimensional, even though they are written or printed on two-dimensional paper. In a sense, this represents a less than optimum use of the two-dimensional medium. Actually, a one-dimensional expression can be split into separate lines and indented to facilitate human readability without affecting the one-dimensional structure and, hence, the meaning of the expression. In practice, this is often done. This book contains examples of longer expressions split into lines and indented to improve readability.

Table 3.4.10-1 summarizes and compares the advantages and disadvantages of the various notational forms for mathematical expressions. Refer to the text above for finer distinctions in some cases. The degree to which a criterion is met or absent in a particular notational form is indicated roughly by the number of $+$ or $-$ symbols in the corresponding cell in the table. An asterisk (*) in a cell indicates that the degree to which the criterion is or is not met depends on the particular figure, drawing, diagram, or special notation. The abbreviations for the notational forms are given in the paragraphs above.

TABLE 3.4.10-1 Advantages and Disadvantages of the Different Notational Forms

	Notational Form								
Criterion	SFN	INF	SFN + INF	Tree	PPF	TBL	GRA	FDD	SPC
Overall readability	−	+	+	+	−	++	++	++	+
Clarity of structure of expression	+	+	+	+++	+	++	−	−	−
Clarity of functional dependencies	−	−	−	−	−	+	++	+	*
Compact representation	+	+	+	−−	++	−	−	−	+
Absolutely precise and accurate	yes	yes	yes	yes	yes	yes	no	no	yes
Frequency of use in practice	−	+	++	−−	−	+	+	+	+
Dimension	1	1	1	2	1	2 (3)	2 (3)	2 (3)	*
Can represent any expression	yes	no	yes	yes	yes	yes	yes	no	yes
Easy to write by hand	+	+	+	−	+	+	−	−	+
Easy to type in	+	+	+	−−	+	−	−−	−−	*
Requires parentheses	some	yes	yes	no	no	yes	no	no	*
Requires special symbols	no	yes	yes	no	yes	yes	no	*	*

3.5 EVALUATING VARIABLES, FUNCTIONS, AND EXPRESSIONS

The subject of this section is the definition of the value of a variable, function, or expression—*not* calculating it in a sequence of steps in time by a mechanistic process. Within the Language of Mathematics itself, the value of a variable, function, or expression *is* something—it is not calculated. No procedure for calculating the value of anything is included in the universe of discourse of the Language of Mathematics itself. A mathematical analysis of such a procedure is best viewed as an application of mathematics, requiring a mathematical model and its interpretation to bridge the gap between the temporal view of the calculation and the static view of the Language of Mathematics, just as any other application would.

A variable name appearing in an expression is a reference to the value of that variable. Therefore, the value of an expression is defined as the value of the expression obtained by replacing every variable name appearing in the expression by its value. The value of the resulting expression containing functions and values only is then determined by applying the definitions of the functions in the expression, the parenthesis rules, and the binding orders (see Section 3.4.2).

Thus, the value of the expression depends on the context of the values associated with the variables. These values are usually determined by other terms in the mathematical model in which the expression appears. The topic of assignment of values to variables is covered in more detail in Sections 6.10, 6.11, 6.13, and 6.13.2.

In determining the value of an expression, several different situations can arise. They lead to different types of results. The various situations are:

- The values of all variables in the expression are assigned (determined) and all arguments of each function are in the domain of that function. Evaluation is complete. See Section 3.5.1.
- The value of at least one variable in the expression is not assigned (determined), but no argument of any function is outside the domain of that function. In general, only partial evaluation is possible. See Section 3.5.2.
- One or more arguments of one or more functions in the expression are not in the domain of the function(s). The value of the expression is not defined. See Section 3.5.3.

3.5.1 Complete (Total) Evaluation

If the values of all variables in an expression are assigned (determined) and all arguments of each function are in the domain of that function, the value of the expression is defined to be the value of the expression obtained by replacing every variable name by its value.

Example 1 In the context

$$\mathrm{x}\in\mathbb{R} \wedge \mathrm{y}\in\mathbb{R} \wedge \mathrm{z}\in\mathbb{R} \wedge \mathrm{x}=2.5 \wedge \mathrm{y}=4 \wedge \mathrm{z}=1.5 \qquad [3.5.1\text{-}1]$$

the value of the expression

$$\mathrm{x} + \mathrm{y}*\mathrm{z} \qquad [3.5.1\text{-}2]$$

is

$$2.5 + 4*1.5 \qquad [3.5.1\text{-}3]$$

which is

$$2.5 + 6 \qquad [3.5.1\text{-}4]$$

which is, in turn, 8.5. Notice that the arguments of the function $*$ (product) are both numbers and, therefore, in the domain of $*$. Similarly, the arguments of the function $+$ (sum) are both numbers and, therefore, in its domain.

Example 2 In the context

$$\mathrm{a}\in\mathbb{B} \wedge \mathrm{y}\in\mathbb{R} \wedge \mathrm{z}\in\mathbb{R} \wedge \mathrm{a}=\mathrm{true} \wedge \mathrm{y}=4 \wedge \mathrm{z}=1.5 \qquad [3.5.1\text{-}5]$$

the value of the expression

$$\mathrm{a} \wedge \mathrm{y}\leq\mathrm{z} \qquad [3.5.1\text{-}6]$$

is

$$\text{true} \wedge 4\leq 1.5 \qquad [3.5.1\text{-}7]$$

which is

$$\text{true} \wedge \text{false} \qquad [3.5.1\text{-}8]$$

which is, in turn, false. Notice that the arguments of the function $\leq$ (is less than or equal to) are both numbers and, therefore, in the domain of $\leq$. The arguments of the function $\wedge$ are both Boolean values and, therefore, in its domain.

3.5.2 Partial Evaluation

If the value of at least one variable in an expression is not assigned (determined), but no argument of any function is outside the domain of that function, the expression can be *partially* evaluated. The partially evaluated expression is the expression after all the known values of variables have been substituted for their respective names and subexpressions have been evaluated wherever possible.

Example 1 In the context

$$x\in\mathbb{R} \wedge y\in\mathbb{R} \wedge z\in\mathbb{B} \wedge y=4 \wedge z=1.5 \qquad [3.5.2\text{-}1]$$

the value of the expression

$$x+y*z \qquad [3.5.2\text{-}2]$$

is

$$x + 4*1.5 \qquad [3.5.2\text{-}3]$$

so that the partially evaluated expression is

$$x + 6 \qquad [3.5.2\text{-}4]$$

Notice that the arguments of the function $*$ (product) are both numbers and, therefore, in the domain of $*$. Although the value of x is not known, the term $x\in\mathbb{R}$ ensures that the value of x is a number, so the arguments of the function $+$ (sum) are both numbers and, therefore, in its domain.

Example 2 In the context

$$w\in\mathbb{R} \wedge x\in\mathbb{R} \wedge y\in\mathbb{R} \wedge z\in\mathbb{R} \wedge w=3 \wedge x=5 \wedge y=4 \qquad [3.5.2\text{-}5]$$

the value of the expression

$$w<x \wedge y\leq z \qquad [3.5.2\text{-}6]$$

is

$$3<5 \wedge 4\leq z \qquad [3.5.2\text{-}7]$$

which is

$$\text{true} \wedge 4 \leq z \qquad [3.5.2\text{-}8]$$

so that the partially evaluated expression is

$$4 \leq z \qquad [3.5.2\text{-}9]$$

Although the value of z is not known, the term $z \in \mathbb{R}$ ensures that the value of z is a number, so the arguments of the function $\leq$ (is less than or equal to) are both numbers and, therefore, in the domain of $\leq$. The arguments of the function $<$ are both numbers and, therefore, in its domain. The arguments of the function $\wedge$ are both Boolean values and, therefore, in its domain.

For most purposes in mathematics, reduced expressions as in Examples 1 and 2 above are desired and accepted as results of partial evaluation. In some contexts, however, a partially evaluated expression is not desired as a result. In such cases, the result can be considered undefined (as in Section 3.5.3), instead of a reduced expression. A typical example occurs during the execution of a computer program, when it is expected that the values of all program variables be defined.

When the value of at least one variable is unknown, only partial evaluation is possible in general. However, in certain situations, the value of an unassigned variable is irrelevant and complete evaluation is possible, as in Examples 3 and 4 below.

Example 3 In the context

$$x \in \mathbb{R} \wedge y \in \mathbb{R} \wedge z \in \mathbb{R} \wedge x=5 \wedge y=0 \qquad [3.5.2\text{-}10]$$

the value of the expression

$$x+y*z \qquad [3.5.2\text{-}11]$$

is

$$5+0*z \qquad [3.5.2\text{-}12]$$

which is, for any numerical value of the variable z,

$$5+0 \qquad [3.5.2\text{-}13]$$

so that the evaluated expression is

$$5 \qquad [3.5.2\text{-}14]$$

Although the value of z is not known, the term $z \in \mathbb{R}$ ensures that the value of z is a number, so the arguments of the function $*$ (product) are both numbers and, therefore, in the domain of $*$. The arguments of the function $+$ (sum) are both numbers and, therefore, in its domain.

Example 4 In the context

$$w \in \mathbb{R} \wedge x \in \mathbb{R} \wedge y \in \mathbb{R} \wedge z \in \mathbb{R} \wedge w=5 \wedge x=3 \wedge y=4 \qquad [3.5.2\text{-}15]$$

the value of the expression

$$w<x \wedge y\leq z \qquad [3.5.2\text{-}16]$$

is

$$5<3 \wedge 4\leq z \qquad [3.5.2\text{-}17]$$

which is

$$\text{false} \wedge 4\leq z \qquad [3.5.2\text{-}18]$$

so that the evaluated expression is, for any numerical value of the variable z,

$$\text{false} \qquad [3.5.2\text{-}19]$$

Although the value of z is not known, the term $z\in\mathbb{R}$ ensures that the value of z is a number, so the arguments of the function $\leq$ (is less than or equal to) are both numbers and, therefore, in the domain of $\leq$. The arguments of the function $<$ are both numbers and, therefore, in its domain. The arguments of the function $\wedge$ are both Boolean values and, therefore, in its domain.

3.5.3 Undefined Values of Functions and Expressions

If one or more arguments of one or more functions in an expression are not in the domain of the function(s), the value of an affected function is not defined and, correspondingly, the value of the expression is not defined.

Example 1 In the context

$$a\in\mathbb{R} \wedge y\in\mathbb{R} \wedge z\in\mathbb{R} \wedge a=9 \wedge y=4 \wedge z=1.5 \qquad [3.5.3\text{-}1]$$

the value of the expression

$$a \wedge y\leq z \qquad [3.5.3\text{-}2]$$

is

$$9 \wedge 4\leq 1.5 \qquad [3.5.3\text{-}3]$$

which is not defined because the first argument of the function $\wedge$ is a number, which is not in the domain of the function $\wedge$. In the given context, this expression is not a valid expression in the Language of Mathematics.

Example 2 In the context

$$w\in\mathbb{R} \wedge x\in\mathbb{R} \wedge y\in\mathbb{R} \wedge z\in\mathbb{B} \wedge w=3 \wedge x=5 \wedge y=4 \qquad [3.5.3\text{-}4]$$

the value of the expression

$$w<x \wedge y\leq z \qquad [3.5.3\text{-}5]$$

is

$$3<5 \wedge 4\leq z \qquad [3.5.3\text{-}6]$$

which is

$$\text{true} \wedge 4 \leq z \qquad [3.5.3\text{-}7]$$

so that the reduced expression is

$$4 \leq z \qquad [3.5.3\text{-}8]$$

Although the value of z is not known, the term $z \in \mathbb{B}$ in the given context indicates that the value of z will be Boolean. Therefore, the second argument of the function $\leq$ (is less than or equal to) will be Boolean, which is not in the domain of $\leq$. That is, for all values of z consistent with the given context, the value of the expression is undefined. In the given context, this expression is not a valid expression in the Language of Mathematics.

Example 3 This example illustrates a situation that arises also in computer programs, particularly in the conditional expression of certain types of loops.

In the context

$$i \in \mathbb{Z} \wedge n \in \mathbb{Z} \wedge 0 \leq i \leq n \wedge k \in \mathbb{S} \wedge [\wedge\ j : j \in \mathbb{Z} \wedge 1 \leq j \leq n : a(j) \in \mathbb{S}] \wedge i = n \qquad [3.5.3\text{-}9]$$

where $\mathbb{S}$ is a linearly ordered set, the value of the expression

$$i < n \wedge a(i+1) < k \qquad [3.5.3\text{-}10]$$

is

$$n < n \wedge a(n+1) < k \qquad [3.5.3\text{-}11]$$

which is

$$\text{false} \wedge a(n+1) < k \qquad [3.5.3\text{-}12]$$

In this expression, neither the value of $a(n+1)$ nor the type of this variable is specified in the context. The context does not ensure that the value of $a(n+1)$ will be a valid argument of the function $<$, and, in turn, does not ensure that the value of this expression will be defined. Expressed differently, there are many possible values for the variable $a(n+1)$ which are consistent with the given context but for which the value of the function $<$ (and, therefore, the entire expression) is not defined.

In many situations it is useful and appropriate to consider the value of an undefined Boolean expression to be false. Such a convention is by no means universal or common in mathematics, but can be introduced when and where useful. One proper way to handle such situations in the Language of Mathematics is to *extend* each function involved so that its domain includes any arguments and so that its value for arguments outside the original domain is an appropriate value (e.g., either false as in this case, or a new value indicating that the arguments are not in the usual domain).

Such an extended function is defined in terms of an existing function, but it is important to recognize that the two functions are different. An extended function will not necessarily have all the properties that the original function upon which it is based has (e.g., commutativity, associativity, relationships with other functions).

In the case of Example 3 above, one might wish to define an extended function $\sqsubset$ "is less than" as follows:

$$\begin{aligned} x \sqsubset y &= x < y, && \text{if } (x, y) \in \text{Domain}(<), \\ &= \text{false}, && \text{otherwise} \end{aligned} \qquad [3.5.3\text{-}13]$$

Then, in the context

$$i \in \mathbb{Z} \wedge n \in \mathbb{Z} \wedge 0 \le i \le n \wedge k \in \mathbb{S} \wedge [\wedge\ j : j \in \mathbb{Z} \wedge 1 \le j \le n : a(j) \in \mathbb{S}] \wedge i = n \qquad [3.5.3\text{-}14]$$

the expression

$$i < n \wedge a(i+1) \sqsubset k \qquad [3.5.3\text{-}15]$$

evaluates to

$$\text{false} \wedge a(n+1) \sqsubset k \qquad [3.5.3\text{-}16]$$

and, because a(n+1) is unassigned and of unknown type, to

$$\text{false} \wedge \text{false} \qquad [3.5.3\text{-}17]$$

and, in turn, to

$$\text{false} \qquad [3.5.3\text{-}18]$$

Note that the functions $<$ and $\sqsubset$ do not share all their properties. If one defines the extended function $\sqsupseteq$ in terms of the function $\ge$ in the same way as the extended function $\sqsubset$ is defined above in terms of the function $<$, then

- $(x \sqsubset y)$ and $(x \sqsupseteq y)$ have different Boolean values if $(x, y) \in \text{Domain}(<)$, but
- $(x \sqsubset y)$ and $(x \sqsupseteq y)$ are equal and both false otherwise [i.e., if $(x, y) \notin \text{Domain}(<)$].

That is, it is true that $(x<y)=\text{not}(x \ge y)$, but it is not always true that $(x \sqsubset y)=\text{not}(x \sqsupseteq y)$.

In Example 3, the second argument of the logical and function is sometimes undefined when the first argument is false. This pattern arises moderately often in mathematical expressions. The more general form is

$$\text{Bexp1} \wedge \text{Bexp2} \qquad [3.5.3\text{-}19]$$

where Bexp2 is always defined when Bexp1 is true, but Bexp2 is not always defined when Bexp1 is false. One way to avoid the consequences of an undefined value of Bexp2 when Bexp1 is false is to replace the expression Bexp1 $\wedge$ Bexp2 by

$$\text{Bexp1} \wedge [\wedge\ i : \text{Bexp1} : \text{Bexp2}] \qquad [3.5.3\text{-}20]$$

or, equivalently,

$$\text{Bexp1} \wedge [\vee\ i : \text{Bexp1} : \text{Bexp2}] \qquad [3.5.3\text{-}21]$$

(where i is a variable name not appearing in either Bexp1 or Bexp2) or

$$\text{Bexp1} \wedge [\text{if Bexp1 then Bexp2 else false}] \qquad [3.5.3\text{-}22]$$

when such a case can arise. In expressions 3.5.3-20, 3.5.3-21, and 3.5.3-22, the value of Bexp2 (and whether or not the value of Bexp2 is even defined) is relevant only if Bexp1 is true. These expressions correspond to a Boolean function called *cand* (conditional and).

There are still other ways of handling situations in which values can be undefined. The writer(s) of a mathematical model must decide which approach is most appropriate for the application in question and then modify the expressions or define extensions of the functions in question accordingly. The ideal way of handling such special cases is to avoid them in the first place by writing all expressions so that all values referenced in an expression are defined. Unfortunately, this cannot always be achieved conveniently and easily.

In any event, the writer(s) of mathematical expressions should always consider the question of whether or not all terms are always defined. When undefined results can result, one should consider carefully what resulting values are appropriate for the possibly undefined expressions and modify the relevant expressions accordingly.

3.6 REPRESENTATIONS OF VALUES VS. NAMES OF VARIABLES

Notice that the name of a variable consists of a sequence of symbols. A value is also represented by a sequence of symbols (e.g., 425 for the number four hundred and twenty-five). One can view a sequence of symbols such as 425 either as a direct representation of the numerical value (i.e., as the value itself) or as the name of a variable whose value is the numerical value. The same can be said of a representation of a nonnumerical value such as red, rectangular, physician, true, or false. Whether a particular sequence of symbols is to be viewed as the name of a variable or as the value itself is usually clear from the context, but it should also be stated clearly in the interpretation of the mathematical model within which the sequence of symbols in question appears.

Mathematically, it makes no difference whether a sequence of symbols representing a value is considered to be a direct representation of the value or the name of a variable with that value, because the name of a variable is a reference to the value of that variable. Consequently, it does not really matter whether or not one makes the distinction at all. In this book, the distinction is made because it is sometimes helpful to do so, particularly in the framework of the application domain.

The Language of Mathematics, being a static language, has no mechanism for changing the value of a variable. A mathematical expression such as x=4 does not set or change the value of x; it is a statement—which may be true or false—that the value of the variable named x is 4. Not even the English statement “Let x=4” means “set the value of the variable named x to 4”; instead, it really means “suppose” or “assume that the value of the variable named x is 4.” Similarly, definitions of the values of functions (defined in Section 3.3) are static; they simply apply, without any reference to time. Therefore, whether the sequence of symbols 425 is viewed as the name of a variable with the numerical value or as a direct representation of the value is of no consequence.

In some contexts outside the Language of Mathematics, however, it can be important to distinguish between a direct representation of a value and a variable associated with that value. For example, in dynamic languages such as computer programming languages, this distinction can be important and, therefore, should be made explicitly. For example, in one compiler for a programming language with which the author once worked, this distinction was not made and numerical constants were handled as program variables with predefined values. The compiler applied some restrictions to prevent the values of such constants from being changed, but not enough, and it was possible for an executing program to change the value of a numerical constant such as 2. Subsequent arithmetic operations gave correspondingly incorrect results. After this anomaly was reported to the computer manufacturer, the compiler was modified to make the distinction between names of variables and direct representations of values.

4 Important Structures and Concepts in the Language of Mathematics

In addition to the basic elements introduced in Chapter 3, the Language of Mathematics includes a number of structures and concepts composed of those basic elements. The simpler, more fundamental, and most common of them are defined and described in Section 4.1. More advanced ones are the subjects of the remaining sections.

The material in this chapter is intended to present and explain certain additional *mathematical terminology* only: certain notational forms in the Language of Mathematics and certain associated English terms used in mathematical texts. None of the material in this chapter is intended to be a tutorial on the mathematical topics covered themselves. The explanations and examples are intended to convey an understanding of certain notational forms, terms, and terminology, *not* a complete mathematical understanding of the topics themselves. Many of these topics are the subjects of entire books themselves.

Some of the extensions to the Language of Mathematics represented by these structures can be viewed as interpretations of the more fundamental structures provided for major classes of applications. Probability theory, the subject of Section 4.6, is a good example. It introduces no new component into the Language of Mathematics, but provides a structured “package” of previously defined elements for dealing with, and reasoning about, random events and processes. Included in this package are interpretations of elements of the Language of Mathematics in the terminology of the many application areas that involve random processes.

4.1 COMMON STRUCTURES OF VALUES

In mathematics, one object is often built out of other, already defined objects. Within the object being built, the components may be structured in some particular way, characterizing the object being built. This approach is used in mathematics in order to facilitate working with the new object and its component objects. It is particularly useful when the newly defined type of structure arises in many different applications.

A particularly common and useful structure is a *set*: a collection of different values, without any other particular internal structure (defined in Section 4.1.1). Often in an application, different variables of the same kind arise (e.g., temperatures at different places in a nuclear reactor). In such cases it is often useful to consider the closely related variables as a collection of *indexed* variables, a structure also called an *array* (defined in Section 4.1.2). In other cases it is useful to arrange a number of values one after the other; such an ordered collection of values is called a *sequence* (defined in Section 4.1.3). A structure can also consist of other structures, functions, and so on. An example is the *finite state machine*, a structure consisting of three sets, two functions, and a value of one of the sets (defined and discussed in Section 4.1.7). A finite state machine is used, in turn, to define three sequences of relevance in typical applications.

Equivalence and *direct correspondence*—important relationships between particular structures—are defined and discussed in Sections 4.1.4, 4.1.5, and 4.1.6.

4.1.1 Sets

A *set* is a collection of different values. The word *different* here emphasizes that the values in a set are distinguishable from one another. Put still another way, any particular value occurs at most once in a set. Each value in a set is called an *element* or a *member* of the set. No two elements of a set are the same—no two elements of a set are equal to each other. A set may be *empty*, that is, it contains no element.

One way of writing a set is to enclose a list of its elements in braces (e.g., $\{1, 2, 3, 4\}$). Another way to define a set is to specify a property that its elements, and only its elements, have. This property is typically written as a Boolean expression. One commonly used notation for this is "$\{x \mid \text{property}(x)\}$," meaning that the set contains all values x for which the value of the function property(x) is true. Any notational form can be used to write the expression for property(x). Other more general notational forms for writing an expression defining a set are presented in Section 3.4.8.

A set is a structure of values. A set itself is also a value. Hence, one of the elements in a set can, itself, be a set.

Some common sets arising in mathematical work, the symbols used to refer to them, and their definitions are:

- *Natural numbers*: $\mathbb{N} = \{1, 2, 3, \ldots\}$
- *Natural numbers with zero*: $\mathbb{N}_0 = \{0, 1, 2, 3, \ldots\}$
- *Integers*: $\mathbb{Z} = \{\ldots -3, -2, -1, 0, 1, 2, 3, \ldots\}$
- *Rational numbers*: $\mathbb{Q} = \{x \mid x{=}a/b, \text{ where a and b are integers and b is not } 0\}$
- *Real numbers*: $\mathbb{R}$ (see the paragraph below)
- *Boolean*: $\mathbb{B} = \{\text{false, true}\}$

The real numbers include all the rational numbers and, in addition, the irrational numbers, numbers that cannot be expressed as the ratio of two integers. Some examples of irrational numbers are $\sqrt{2}$ (the square root of 2), π (the ratio of the circumference of a circle to its diameter), e (the base of the natural logarithms), and many other important mathematical constants. A more formal definition of the real numbers is given in Appendix C. To distinguish more clearly between the sets of the natural numbers with and without zero, the symbol $\mathbb{N}_1$ is often used for the set $\{1, 2, 3, \ldots\}$.

The expression above defining the set of rational numbers is written in a mixture of English and mathematics. In Section 3.4.8 a more general notational form is presented for expressions involving "dummy" variables ("a" and "b" in the description above of rational numbers).

An ordinary set has no internal arrangement or organization; it is an amorphous collection of its elements. Various arrangements of the elements of a set can be defined, if desired. A common arrangement is ordering a set as described in the next paragraph. Other more extensive structures can be built upon a set, some of which are described in Appendix D.

An *order* can be defined on a set by defining a Boolean function (a function whose value is an element of $\mathbb{B}$) exhibiting certain properties. An ordering function, called "orderf" below, must have two arguments, each of which is a value in the set in question and must have the following two properties, where the variables x, y, and z have values in the set.

- If orderf(x, y) is true, the values of x and y are different and the value of orderf(y, x) is false.
- If orderf(x, y) and orderf(y, z) are both true, orderf(x, z) is true. This is called the *transitive property*.

The infix symbol $<$ is frequently used for an ordering function, especially where the elements of the set in question are numbers. For orders on sets whose elements are not numbers, a similar symbol such as $\sqsubset$ or $\prec$ is also sometimes used.

If, in addition to the two properties above,

- exactly one of the expressions orderf(x, y), x=y, or orderf(y, x) is true for every pair of elements x and y in the set

the set is called a *linearly ordered* set. A set that is ordered but not linearly ordered is sometimes called *partially ordered.*

Note that the functions represented by the symbols $\leq$ and $\geq$ are not orders, because they do not satisfy the definition of order above. The reader should identify why not: What property of an order do they violate? These symbols are used for a convenient composition of the functions $<$ and $=$ in the first case and of the functions $>$ and $=$ in the second case. The functions $\leq$ and $\geq$ are defined as follows:

- $x \leq y$ means $x < y \vee x = y$.
- $x \geq y$ means $x > y \vee x = y$.

The combinations "is less than or equal to" and "is greater than or equal to" arise so often in practice that a single function is warranted for each of these combinations of functions.

Functions of sets arise often in applications of mathematics. The functions described below are among the most common functions on sets.

The function "… is an element of the set …" has the value true or false, depending on whether or not the value of the first argument (which may be any value) is a member of the value of the second argument (which may be any set). The infix symbol $\in$ is used for this function. For example, the value of the expression "$4\in\mathbb{Z}$" is true, but the value of the expression "$1.5\in\mathbb{Z}$" is false.

The *union* of two sets is the set containing those elements that are in either (or both) of the argument sets. The infix symbol $\cup$ is used for this function. For example, the value of the expression "$\{1, 3, 5\}\cup\{2, 4, 5, 6\}$" is the set $\{1, 2, 3, 4, 5, 6\}$.

The *intersection* of two sets is the set containing those elements that are in both of the argument sets. The infix symbol $\cap$ is used for this function. For example, the value of the expression "$\{1, 3, 5\}\cap\{2, 4, 6\}$" is the set containing no element (i.e., the empty set, written $\{\ \}$ or, more commonly, $\emptyset$). The value of the expression "$\{1, 3, 5\}\cap\{1, 2, 3, 4\}$" is the set $\{1, 3\}$.

A memory aid for remembering that $\cup$ stands for "union" is the similarity between the symbol "$\cup$" and the letter "U" for union. Another memory aid is that a bucket placed with its opening upward, as the symbol "$\cup$," will catch more rain than a bucket placed with its opening downward, as the symbol "$\cap$." From the definitions above, it is clear that the union of two sets will generally contain more elements than the intersection; sometimes the union and the intersection will contain the same number of elements, but never will the union contain fewer elements than the intersection.

The *Cartesian product* of two sets is the set of ordered pairs of elements in the two sets. The infix symbol $\times$ is used for this function. By *ordered pair* is meant a pair of values, the first value being in the first set and the second value being in the second set. For example, the value of the expression $\{1, 3\}\times\{2, 4, 6\}$ is the set $\{(1, 2), (1, 4), (1, 6), (3, 2), (3, 4), (3, 6)\}$, consisting of six elements, each being a pair of values in the first and second sets, respectively. The pair $(2, 4)$ is not in the set $\{1, 3\}\times\{2, 4, 6\}$ because 2 is not an element of the set $\{1, 3\}$. The pair $(1, 3)$ is not in the set $\{1, 3\}\times\{2, 4, 6\}$ because 3 is not an element of the set $\{2, 4, 6\}$. The pair $(2, 3)$ is not in the set $\{1, 3\}\times\{2, 4, 6\}$ because 2 is not an element of the set $\{1, 3\}$ and 3 is not an element of the set $\{2, 4, 6\}$. Note that the Cartesian product is not commutative (i.e., $S\times T$ is not, in general, the same as $T\times S$).

The *difference* of two sets is the set of elements in one set but not in the other. The infix symbols $\setminus$ and $-$ are used for this function. The symbol $\setminus$ is the more traditional one, but the normal minus sign $(-)$ has come into increasing use. For example, the value of the expression $\{1, 2, 3, 4\}\setminus\{2, 4, 6, 8\}$ is the set $\{1, 3\}$, the set of elements in the first set that are not in the second set. In other words, the set $X\setminus Y$ is the set X with the elements that are also in Y removed.

One set is a *subset* of another set if every element of the first set is also an element of the second set. The infix symbol $\subset$ is used for the corresponding function, whose value is Boolean (i.e., is either "false" or "true"). Thus, the value of the expression "$A\subset B$" is true if and only if every element of the set A is also an element of the set B. If any element of the set A is not an element of the set B, the value of the expression "$A\subset B$" is false. Note that every set is a subset of itself by this definition.

Two sets are equal to each other if and only if they contain exactly the same elements, that is, if and only if each is a subset of the other. This condition is often used as a test for the equality of two sets: $(A{=}B) = ((A{\subset}B) \wedge (B{\subset}A))$

If the set A is a subset of the set B, the sets A and B may or may not be equal to each other. Sometimes one wants to distinguish between these two situations. The symbols $\subsetneqq$ and $\subseteq$ are used to make this distinction. These functions are defined as follows:

- $(A{\subsetneqq}B)$ means $((A{\subset}B) \wedge (A{\neq}B))$.
- $(A{\subseteq}B)$ means $((A{\subset}B) \vee (A{=}B))$.

In the first case, A is a subset of B but A and B are not equal (i.e., are not the same set). In this case, A is called a *proper subset* of B. In the second case, A is a subset of B and A and B may, but need not, be equal to each other. Note that the Boolean *and* function $\wedge$ appears in the first expression above, while the Boolean *or* function $\vee$ appears in the second.

The list above of sets of different types of numbers yields several examples of sets being subsets of other sets: $\mathbb{N}$, the set of natural numbers, is a proper subset of $\mathbb{N}_0$, the set of natural numbers with zero. In turn, $\mathbb{N}_0$ is a proper subset of $\mathbb{Z}$, the set of integers. $\mathbb{Z}$, in turn, is a proper subset of $\mathbb{Q}$, the set of rational numbers, and $\mathbb{Q}$ is, in turn, a proper subset of $\mathbb{R}$, the set of real numbers.

Strictly speaking, the functions $\subset$ and $\subseteq$ as defined above are the same function (the expressions "$A{\subset}B$" and "$A{\subseteq}B$" always have the same value), so the two symbols are redundant. However, in some mathematical writing, especially older material, the symbol $\subset$ was used in the sense of a proper subset (i.e., $\subset$ meant $\subsetneqq$, not $\subseteq$ as is more common today). To prevent confusion arising from these different conventions, it is recommended that one use the functions and symbols $\subsetneqq$ and $\subseteq$ only, avoiding $\subset$ completely.

The change in meaning of the symbol $\subset$ illustrates that the Language of Mathematics has undergone changes in the course of time, just as English has, and that some ambiguity can arise from different writers using symbols or words in the different senses in vogue at different times. For example, when the author was young, the English noun *fag* meant in slang usage "cigarette," nothing else. Today, its main slang meaning is completely different, even unrelated. When writing for an audience including all age groups, ambiguity can best be avoided by not using this word, but by using instead an unambiguous synonym for the meaning intended. Similarly, ambiguity can best be avoided by not using the symbol $\subset$, but by using $\subsetneqq$ or $\subseteq$ instead.

Two sets are part of the definition of every function: the function's domain and its range. The *domain* of a function is the set of values of the argument(s) for which the value of the function is defined. Every value of the function is in the *range* of the function. Consider, for example, the function $+$ ("normal" mathematical addition), which has two arguments, each of which must be a real number, and whose value is a real number. The same applies to the function $*$. The function / is similar, but its second argument may not be 0, because division by zero is not defined. The relational function $<$ has two arguments, each of which must be a real number, while the value of the function is either false or true (i.e., is a Boolean value). The same applies to the other relational functions: $\leq$, $>$, and $\geq$. Therefore, the domains and the ranges of these functions are as follows:

- The function $+$ has the domain $\mathbb{R}\times\mathbb{R}$ and the range $\mathbb{R}$.
- The function $*$ has the domain $\mathbb{R}\times\mathbb{R}$ and the range $\mathbb{R}$.
- The function / has the domain $\mathbb{R}\times(\mathbb{R}\backslash\{0\})$ and the range $\mathbb{R}$.
- The function $<$ has the domain $\mathbb{R}\times\mathbb{R}$ and the range $\mathbb{B}$.
- The function $\leq$ has the domain $\mathbb{R}\times\mathbb{R}$ and the range $\mathbb{B}$.
- The function $>$ has the domain $\mathbb{R}\times\mathbb{R}$ and the range $\mathbb{B}$.
- The function $\geq$ has the domain $\mathbb{R}\times\mathbb{R}$ and the range $\mathbb{B}$.

The name of a set (more precisely, the name of a variable whose value is a set) is often printed in a particular type style, such as $\mathbb{R}$ and $\mathbb{B}$ above. This is done for the convenience of human readers only. The particular type style itself conveys no meaning or information in the Language of Mathematics.

As pointed out in Section 3.4.9, the relationship between a function f, its domain $\mathbb{X}$, and its range $\mathbb{Y}$ is often written

$$\mathrm{f} : \mathbb{X} \rightarrow \mathbb{Y} \qquad [4.1.1\text{-}1]$$

which is an abbreviation for—and is defined to mean the same as—the following quantified expression:

$$[\wedge\ \mathrm{a} : \mathrm{a}\in\mathbb{X} : \mathrm{f(a)}\in\mathbb{Y}] \qquad [4.1.1\text{-}2]$$

Translated to English, this means “for every element a in the set $\mathbb{X}$, the value of f(a) is an element of $\mathbb{Y}$. Quantified expressions such as this one are dealt with in detail in Section 3.4.8.

4.1.2 Arrays (Indexed Variables), Subscripted Variables, and Matrices

An *array* is a group of closely related variables, usually with the same basic interpretation. An *array variable* is a variable whose name is written in a particular way: the name of the array followed by a value, called an *index*, in parentheses [e.g., x(4)]. Any expression with a suitable value (see below) can be written for the index [e.g., x(y+z∗3)].

To determine the value of an array variable, the index expression is evaluated. The value of the array variable with that value of the index is the desired value. For example, if y=2, z=1, x(4)=8, x(5)=9, and x(6)=7, then x(y+z∗3) is x(5), which is 9.

A subscripted variable is essentially the same as an array variable, but it is written in a different form. The index follows the name of the array as a subscript (e.g., x_4,

x_{y+z*3}). The difference between arrays and subscripted variables is notational and historical only. It is not linguistically significant.

Notice that an array is, in effect, a function that maps the value of the index to the value of the array variable. The index of the array variable is the argument of the function, and the value of the array variable is the value of the function. The conceptual views of a collection of array variables and of a corresponding function are different, but this is the only difference. The effects and consequences of the two are the same.

The different ways of writing the index—f(0), f_0, and even f0—can be viewed as only a notational difference. However, if one clearly distinguishes the index from the function name, as in the first two of the three notational forms above, an expression can be allowed for the index. This is often very convenient when writing the various expressions in a mathematical model.

The variables in an array are usually of the same type; that is, their values are elements of the same set. This restriction is not necessary, but it is convenient in many situations, both from a mathematical and from an application viewpoint.

Index values are usually elements of a finite set of consecutive nonnegative integers, but this restriction is also not necessary. The index need not even be a number. The set of index values may be infinite, although such usage is rare.

In the paragraphs above, the names of array variables were shown with exactly one index. More generally, array variables may have more than one index [e.g., a(x, y), a(x, y, z)]. One can even view an ordinary variable as an array variable with 0 indices: for example, the variable a as the array variable a().

Example In a nuclear reactor, the temperature is measured at n different points. The temperature at point i is the value of the array variable Temp(i) for i ranging from 1 to n inclusive. The average temperature is, therefore,

$$\text{average temperature} = (\text{Temp}(1) + \text{Temp}(2) + \cdots + \text{Temp}(n))/n \qquad [4.1.2\text{-}1]$$

which can also be written as (see Section 3.4.8)

$$\text{average temperature} = (1/n) * \sum_{i=1}^{n} \text{Temp}(i) \qquad [4.1.2\text{-}2]$$

The convenience of being able to write the latter expression illustrates the advantage offered by array variables.

Other examples of array variables are names, addresses, and related data on customers, the power of each of the several engines in an airplane, and flow rates at various places in an oil pipeline network.

Other examples of indices to an array are “pointer” variables in some computer programming languages and their implementations. The structure formed by pointer variables and the objects to which they point is seldom thought of as an array, but viewing it as such can facilitate understanding and working with pointer structures.

The index (pointer) values are generally elements of a finite set of nonconsecutive integers.

If the values of an array are written in a table, where the row number is one index to the array and the column number is the other index to the array, the resulting table is called a *matrix* (plural, *matrices*). The definition can be generalized to one or three or more dimensions (number of indices). In mathematics, several operations on matrices are defined, including addition, multiplication, inversion, and transposition (interchanging rows and columns).

4.1.3 Sequences

A sequence is, as the name suggests, values arranged in order, one after another. The linear order is the distinguishing characteristic. The three most common notational forms for sequences are a *list* of the terms of the sequence, simple *juxtaposition* of the terms, and an *array*. An *ordered pair*, a *triple*, and, more generally, an *n-tuple* make up another form of a sequence. These are described in the following paragraphs.

List A sequence is frequently written by listing the values, one after another, separated by commas and enclosing the list in brackets. Each value in the list may be written as the value itself, the name of a variable with the value, or an expression: for example, [4, x+z, 5∗a, 5, 4, 8].

Juxtaposition A sequence may also be written by juxtaposing the terms of the sequence. For example, the word or name "George Brown" is a sequence of the letters and symbols G, e, o, r, g, e, a blank space, B, r, o, w, and n. This notational form is most frequently used when the entire sequence is to be interpreted as a single object, name, symbol, or value. The usual notation for the number one hundred forty-six is 146, the sequence of digits 1, 4, and 6, but "146" is normally interpreted as a single symbol for the single numerical value.

Array An array (see Section 4.1.2) can represent a sequence, the order of the terms of the sequence being given by the order of the index values. For example, the array a with its array variables and values

a(1)=46
a(2)=2
a(3)=8
a(4)=7
a(5)=34
a(6)=46
a(7)=21

can represent the sequence [46, 2, 8, 7, 34, 46, 21]. Such an array is, in effect, a sequence.

Because a sequence can be viewed as an array and an array can be viewed as a function, a sequence can be viewed as a function. A sequence of values can, itself, be viewed as a single value. Such a single value may be a term of another sequence (i.e., a sequence may be a term of a sequence). For example, in the sequence [3, 82, 7, [x, y, z], 55], the fourth term is [x, y, z], which is, itself, a sequence.

Notice that the arguments of a function can be viewed as a single sequence of arguments. In this view, a function can always be considered to have exactly one argument, which is a sequence of 0 or more values. That is, the functional reference f(x, y, z+w, 4) can be viewed as a function of the single sequence [x, y, z+w, 4], that is, as f([x, y, z+w, 4]).

In contrast to a set, the same value may appear more than once in a sequence. Another important difference between a set and a sequence is that the elements of a set are not necessarily ordered, but the terms of a sequence are always linearly ordered.

A sequence may be empty, that is, contain 0 terms. A sequence may contain exactly 1 term. A sequence may contain any number of terms. A sequence may be infinite: have no end, have no beginning, or have neither a beginning nor an end.

Two sequences can be combined to form one sequence by writing the terms of the second sequence immediately after the terms of the first sequence, in the original order. The resulting sequence is called the *concatenation* of the two sequences. The infix symbol & is often used for this function. For example, [1, 4, 6]&[3, 7, 9]=[1, 4, 6, 3, 7, 9]. The order of the arguments is significant, that is, A&B is not, in general, the same as B&A.

If the terms of a sequence are rearranged (i.e., put into a different order), the new sequence is called a *permutation* of the original sequence. The relationship is mutual; that is, each sequence is a permutation of the other. For this relation the infix symbol Perm is sometimes used. For example, [3, x, 6, 6, y]Perm[x, y, 3, 6, 6], and vice versa. The sequence [x, y, 3, 6] is not a permutation of [3, x, 6, 6, y] because one of the two terms “6” is missing from [x, y, 3, 6].

Concatenation & is a function that maps two sequences to a sequence. The function Perm defined in the preceding paragraph maps two sequences to a Boolean value: true if the two sequences are permutations of each other and false if they are not.

Tuple A tuple is a sequence usually written with parentheses ( ) instead of brackets [ ]. If a tuple contains two terms, it is usually called an *ordered pair*; if three terms, a *triple*; and if four or more terms, an *n-tuple*. The terms *tuple* and *sequence* are typically used in different contexts, but the meaning is essentially the same. The term *tuple* is often used in definitions of structures, where the structure being defined consists of several more basic components of different types. The structure is defined as a tuple of the components, each of which has certain stated properties. The elements of a sequence are typically simple values, whereas the terms of a tuple are often more complex structures, but this distinction is not defined sharply. The difference between a sequence and a tuple is comparable to the difference between the words *sick*, *ill*, and *indisposed* in English. These words all mean the same thing, but they are used in different contexts (e.g., the monarch of a country would not be described as sick, but instead, as indisposed).

4.1.4 The Equivalence of Array Variables, Functions, Sequences, and Variables

The notational forms for variables, array variables, functions, and sequences are equivalent in certain ways, as outlined briefly in Sections 4.1.2 and 4.1.3 and as described more systematically below. Any of them can be formulated as a function of a single argument or index. Therefore, one has considerable freedom when choosing which of these elements and structures to use when constructing a mathematical model. Notational convenience, correspondence with the application area being modeled, and ease of reading are the main criteria for choosing among these alternatives. These criteria are usually closely related in the eyes of the people reading the mathematical model.

A function and an array with the same list of arguments or indices are mathematically equivalent in the sense that they have the same meaning (values). A function or an array with any number of arguments or indices is equivalent to a function or array with exactly one argument or index, that argument or index being a sequence of the original arguments. A variable is equivalent to a function or array with an empty list of arguments or indices. A variable corresponds in a certain sense to a sequence with exactly one term (see also the end of Section 4.1.5). A nonempty sequence is equivalent to a function of one argument, that argument being an integer in a set of consecutive integers.

The following statements summarize the several corresponding forms between variables, array variables, functions, and sequences. In the statements below, var represents a variable; arr, an array; func, a function; […], a sequence; and …, a list of one or more arguments or indices:

- var, [var], arr(), arr([]), func(), and func([]) correspond with each other.
- arr(…), arr([…]), func(…), and func([…]) correspond with each other.
- arr(i), arr([i]), func(i), func([i]), and […] correspond with each other, where the value of i selects a term of the sequence.

Each of the array variables and functions of the form arr([]), arr([…]), func([]), and func([…]) has a single index or argument, a sequence.

Thus, these several structures are all fundamentally the same. This is one of the many comparable examples of correspondences between various objects and structures in mathematics. A few more are identified in Section 4.1.5. They illustrate the generality inherent in mathematics and in the Language of Mathematics. They are also examples of a characteristic shared with many other languages: the existence of different words, essentially synonymous, but connoting different aspects of the context.

4.1.5 Direct Correspondence of Other Mathematical Objects and Structures

In Section 4.1.4 several mathematical structures were shown to be equivalent in a certain sense, that is, to *correspond directly* to one another. Also in other situations one finds similar correspondences between different mathematical structures and objects. Identifying such correspondences consciously and explicitly can often facilitate simplifying, analyzing, interpreting, and understanding mathematical expressions. Corresponding structures are alternatives between which the writer(s) of mathematical models can choose.

Boolean Functions and Subsets One common example is the correspondence between a Boolean function and the subset of its domain on which the function has the value "true." If the Boolean function is named Zf and the subset of its domain where the function is true is named Zs, it is sometimes useful to refer to both Zf and Zs by the same name Z, even though this is not formally correct. The context of each reference to Z will make it clear whether Zf (a Boolean function) or Zs (a set) is meant. Because of this correspondence between a set and a Boolean function, anything defined in terms of a Boolean function can be defined alternatively in terms of a set, and vice versa. This correspondence also means that a set and a Boolean function are not fundamentally different and separate ideas, but are, instead, closely related concepts.

If the domain of Zf is $\mathbb{S}$ and Zs is a subset of the set $\mathbb{S}$, then Zs and Zf correspond directly in the foregoing sense if any of the following equivalent conditions are true:

$$[\wedge\ x : x{\in}\mathbb{S} : \mathrm{Zf}(x) = (x{\in}\mathrm{Zs})] \qquad [4.1.5\text{-}1]$$

$$\mathrm{Zs} = [\cup\ x : x{\in}\mathbb{S} \wedge \mathrm{Zf}(x) : \{x\}] \qquad [4.1.5\text{-}2]$$

$$\mathrm{Zf} = [\cup\ x : x{\in}\mathbb{S} : \{(x, x{\in}\mathrm{Zs})\}] \qquad [4.1.5\text{-}3]$$

Expression 4.1.5-1 states that (1) Zf is true on the set Zs and false elsewhere and that (2) Zf(x) is true if and only if $x{\in}\mathrm{Zs}$. Expression 4.1.5-2 states that Zs consists of those elements and only those elements of $\mathbb{S}$ for which Zf is true. Expression 4.1.5-3 states that Zf maps all elements of Zs to true and all other elements of $\mathbb{S}$ to false. It defines Zf as a particular subset of $\mathbb{S}\times\mathbb{B}$. Viewing a function as a subset of the Cartesian product of its domain and range and as a particular kind of relation is discussed further in Section 4.1.6.

Given the set $\mathbb{S}$, expression 4.1.5-1 can be used to define either Zf or Zs in terms of the other. Expression 4.1.5-2 can be used to define Zs in terms of Zf. Expression 4.1.5-3 can be used to define Zf in terms of Zs.

Relations, dealt with in more detail in Section 4.1.6, are sometimes defined as Boolean functions and sometimes as subsets. This direct correspondence shows that either kind of definition is equivalent to the other kind.

Logical Functions and Set Functions Following from the paragraphs above, the logical function $\wedge$ and the set function $\cap$ correspond directly with one another, as do the logical function $\vee$ and the set function $\cup$. Assume that the following are true:

- Xs and Ys are subsets of some set S.
- Xf and Yf are Boolean functions on S.
- Xs and Xf correspond directly with each other as described in the paragraphs above.
- Ys and Yf correspond directly with each other as described above.

Then the Boolean function $Xf \wedge Yf$ and the set $Xs \cap Ys$ correspond directly with one another. Also, the Boolean function $Xf \vee Yf$ and the set $Xs \cup Ys$ correspond directly with one another. The proof of this theorem is left as an exercise for the reader after reading Section 5.2.

In this sense, the logical function $\wedge$ and the set function $\cap$ correspond directly with one another, and the logical function $\vee$ and the set function $\cup$ correspond directly with one another.

Images, Preimages, and Related Functions If the function f maps elements of the set $\mathbb{X}$ to elements of the set $\mathbb{Y}$, that is, if

$$f\colon \mathbb{X} \to \mathbb{Y} \qquad [4.1.5\text{-}4]$$

it is often useful to talk about the *image* of a subset $\mathbb{X}\mathbb{1}$ of $\mathbb{X}$ *under* the function f. It is defined to be the set of those elements of $\mathbb{Y}$ to which f maps elements of $\mathbb{X}\mathbb{1}$:

$$\text{image}(\mathbb{X}\mathbb{1}, f) = [\cup\ a : a \in \mathbb{X}\mathbb{1} : \{f(a)\}] \qquad [4.1.5\text{-}5]$$

Note that the image is a function of the subset $\mathbb{X}\mathbb{1}$ and the function f. Because of the direct correspondence between this new image function and the original function f, the image is often written as $f(\mathbb{X}\mathbb{1})$. Strictly speaking, this is not correct, because the two functions are different; f maps an element of $\mathbb{X}$ to an element of $\mathbb{Y}$, whereas the image function maps a subset of $\mathbb{X}$ to a subset of $\mathbb{Y}$, so we should give a different name to the image function. However, the context of every reference to f will make it clear whether the original function f or the image function is meant.

Similarly, it is often useful to talk about the *preimage* of a subset $\mathbb{Y}\mathbb{1}$ of $\mathbb{Y}$ *under* the function f. It is usually written $f^{-1}(\mathbb{Y}\mathbb{1})$ and is defined to be the set of those elements of $\mathbb{X}$ that f maps to elements of $\mathbb{Y}\mathbb{1}$:

$$f^{-1}(\mathbb{Y}\mathbb{1}) = [\cup a : a \in \mathbb{X} \wedge f(a) \in \mathbb{Y}\mathbb{1} : \{a\}] \qquad [4.1.5\text{-}6]$$

The inverse function of f (if it exists), also written f^{-1}, maps each element y of $\mathbb{Y}$ to that element x of $\mathbb{X}$ that f maps to y. (The inverse function of f does not exist if f maps two or more different elements of $\mathbb{X}$ to the same element of $\mathbb{Y}$.) The preimage function (which always exists) maps a subset of $\mathbb{Y}$ to a subset of $\mathbb{X}$. The inverse of the function f and the preimage function correspond to each other in the same way that the function f and the image function correspond with each other. Both the inverse function and the preimage function are therefore often written in the same way, that is, f^{-1}.

Summarizing, a function f and its image function, different from f but usually also written as f, correspond directly with each other. Similarly, f^{-1}, the inverse function of f (if it exists), and the preimage function, also written as f^{-1}, correspond directly with each other.

A Singleton Set and Its Element A set consisting of only one element is called a *singleton set*. Strictly speaking, a singleton set and its element are two different things, two different mathematical objects. For example, consider a singleton set consisting of the number 7. The singleton set $\{7\}$ and its element, 7, are different objects. The number 7 can be added to another number, but a set (e.g., {7}) cannot be added to anything. The union of the set {7} and some other set can be formed, but the union of a number and a set is not defined. In some analyses, some authors prefer not to have to make a notational distinction between a singleton set and its element, because the distinction between the two is always clear from the context, in particular, from the functions applied to them. A singleton set and its element can be viewed as corresponding directly to one another. They can be written in the same way without introducing any ambiguity.

Using the convention of not distinguishing notationally between a singleton set and its element, expression 4.1.5-6 can be written without enclosing the variable a in the set parentheses { }, as follows:

$$f^{-1}(\mathbb{Y}\mathbb{1}) = [\cup a : a{\in}\mathbb{X} \wedge f(a){\in}\mathbb{Y}\mathbb{1} : a] \qquad [4.1.5\text{-}7]$$

Similarly, one could write {7}+5=12, although this is not done, mainly because it is simpler to write only 7 instead of {7}. Therefore, the element alone is sometimes written instead of the singleton set, but not vice versa.

In a similar way, a sequence consisting of only one term (a singleton sequence) and its term can be considered to be in direct correspondence to another and the notational distinction dropped. This correspondence was mentioned in Section 4.1.4.

4.1.6 Relations

It is often useful to consider specific relationships, associations, or connections between values. The mathematical structure *relation* is frequently used for this purpose. Some examples of relations and some of the ways of expressing them are given in Table 4.1.6-1.

TABLE 4.1.6-1 Notational Forms for Relations

English	Infix	Standard Functional	Element of a Subset
2 is less than 4	$2 < 4$	IsLess(2, 4)	$(2, 4) \in \mathbb{L}$, where $\mathbb{L} \subset \mathbb{R}\times\mathbb{R}$
12 is greater than 11	$12 > 11$	IsGreater(12, 11)	$(12, 11) \in \mathbb{G}$, where $\mathbb{G} \subset \mathbb{R}\times\mathbb{R}$
3 is equal to 3	$3 = 3$	Equals(3, 3)	$(3, 3) \in \mathbb{E}$, where $\mathbb{E} \subset \mathbb{R}\times\mathbb{R}$
George is the father of Sarah	George f Sarah	IsFather(George, Sarah)	(George, Sarah) $\in \mathbb{F}$, where $\mathbb{F} \subset \mathbb{P}\times\mathbb{P}$
Jenny is the wife of Thomas	Jenny w Thomas	IsWife(Jenny, Thomas)	(Jenny, Thomas) $\in \mathbb{W}$, where $\mathbb{W} \subset \mathbb{P}\times\mathbb{P}$

The two middle notational forms, infix and standard functional, both express relations as functions with the values true and false. The column on the right expresses the relations as subsets of a Cartesian product of appropriate sets. Both provide a basis for defining a relation. The more common general mathematical definition of a relation is based on the latter: A *relation* is a subset of the Cartesian product of two sets. The two sets may, but need not be, the same.

The observation that a relation can be expressed as either a Boolean function or a subset is an example of the direct correspondence between a Boolean function and a subset as pointed out in Section 4.1.5.

A function was defined in Section 3.3. From that definition it follows that a function is a special kind of relation. A function relates each value of its argument with only one value of the function, while a relation relates each value of its argument (the left value in the pair of values being related) with one or more values (right value in the pair). That is, a function is a relation that relates every left value with only one right value. Expressed more mathematically, a relation (set) $\mathbb{R}\mathbb{e}\mathbb{l}$ of ordered pairs of values is a function if and only if for all values of x, y and z, $(x, y)\in\mathbb{R}\mathbb{e}\mathbb{l}$ and $(x, z)\in\mathbb{R}\mathbb{e}\mathbb{l}$ imply that $y=z$.

Among the examples above, the relation "Equals" is a function. In a monogamous society, the relation "IsWife" is also a function. The other relations above are not functions.

The reader should consider in what ways an order (see Section 4.1.1) and a relation are similar and different. Is an order a relation? Is a relation an order? Which is a special case of the other?

Although convenient and often useful, the introduction of the separate concept of a relation is not necessary. Any particular relation can be expressed either as a Boolean function or as a set of ordered pairs of values. For such a Boolean function any of the notational forms described in Section 3.4 may be used (e.g., infix or standard functional notation).

Note that a relation on the sets $\mathbb{X}$ and $\mathbb{Y}$ can be viewed alternatively as a function fx that maps each element of $\mathbb{X}$ to a subset of $\mathbb{Y}$. It can also be viewed as a function fy that maps each element of $\mathbb{Y}$ to a subset of $\mathbb{X}$ (cf. Section 4.1.5). The reader should write a definition for each of the functions fx and fy using only mathematical expressions (i.e., solely in the Language of Mathematics).

4.1.7 Finite State Machines

A *finite state machine*, often called an *automaton* in the theoretical literature, is a mathematical structure often used to model dynamic processes in engineering applications and to model controllers for such processes. Finite state machines have been found useful in many other application areas as well.

Finite state machines are suitable for modeling processes in which discrete events occur at successive points in time. To model continuous processes, differential and integral calculus is more suitable and is used instead.

The basic idea of a finite state machine is that it exists in some state and, upon receiving an input symbol, outputs a symbol and undergoes a transition to a new state. Then, upon receiving another input symbol, the process continues, potentially indefinitely. The output symbol depends on both the input symbol and the previous state. A sequence of input symbols is transformed into a sequence of output symbols and, internally, a sequence of states.

In a finite state machine, the state is its memory. It makes the input-to-output transformation more general than it would otherwise be. Each state effectively remembers the information in the preceding part of the input sequence that is needed to determine the future part of the output sequence.

In practical applications, the “input symbols” typically correspond to physical events such as a person pressing a button, a person entering an elevator, or a door reaching its closed position. The “output symbols” correspond to actions taken by the machine on its environment, such as opening or closing an elevator door, applying power to a motor to pull an elevator upward, or turning a light on or off.

Mathematically, a finite state machine consists of three finite, nonempty sets, two functions related to these sets as described below, and an element of one of the sets. These components are interpreted as follows. One of the sets is the set of *states*. Another set is the set of *input symbols*. The third set is the set of *output symbols*. One function determines the next state as a function of the preceding state and the input symbol. The other function determines the output symbol as a function of the preceding state and the input symbol. An element of the set of states is the *initial state*.

More formally, a finite state machine consists of the following:

- A finite, nonempty set $\mathbb{States}$
- A finite, nonempty set $\mathbb{Inputs}$
- A finite, nonempty set $\mathbb{Outputs}$
- A function NextState: $\mathbb{States} \times \mathbb{Inputs} \rightarrow \mathbb{States}$
- A function NextOutput: $\mathbb{States} \times \mathbb{Inputs} \rightarrow \mathbb{Outputs}$
- An initial state $\mathrm{s}(0) \in \mathbb{States}$

This mathematical structure is used to define a function mapping a sequence of input symbols into a sequence of output symbols in the following way. If the input sequence is

$$[\mathrm{in}(1), \mathrm{in}(2), \mathrm{in}(3), \mathrm{in}(4), \ \ldots]$$

where each $\mathrm{in}(k) \in \mathbb{Inputs}$, the sequence of states is defined to be

$$[\mathrm{s}(0), \mathrm{s}(1), \mathrm{s}(2), \mathrm{s}(3), \mathrm{s}(4), \ \ldots]$$

and the sequence of output symbols is defined to be

$$[\mathrm{out}(1), \mathrm{out}(2), \mathrm{out}(3), \mathrm{out}(4), \ \ldots]$$

where for every value of $k=0, 1, 2, 3, 4, \ \ldots$,

$$\mathrm{s}(k+1) = \mathrm{NextState}(\mathrm{s}(k), \mathrm{in}(k+1))$$
$$\mathrm{out}(k+1) = \mathrm{NextOutput}(\mathrm{s}(k), \mathrm{in}(k+1))$$

For some applications, the definition above of the function “output” is modified so that its value is a (possibly empty) sequence of output symbols, not necessarily exactly one output symbol.

Examples using finite state machines are the subjects of Sections 2.12, 8.10, 8.11, and 8.13.

4.2 INFINITY

Many students, including the author of this book in his student days, have had considerable difficulty with the concept of infinity in mathematics. Most of this difficulty is due to the fact that the English word *infinity* is sometimes used in ways that suggest that "infinity" is a thing, an object, or a value in the sense of Section 3.1. However, the word *infinity* does not mean a thing, an object, or a value. The implicit feeling that it does, and the fact that it does not can and often does lead to confusion.

What does *infinity* mean? Let's break the word down into its component parts:

- The prefix "in-" means no, not, without, negation of whatever follows.
- The word stem "fin" means end.
- The suffix "-ity" means a property, quality, characteristic, state, or condition.

The noun *infinity* means, therefore, the *property of having no end*, the *quality of being endless (without end)*—no more, no less. It means a property, not a thing.

Some dictionaries include an additional meaning: an indefinitely large number or amount. If your dictionary gives such a meaning, strike it out and forget it before it confuses you, as it confused me for too long a time. Such a "definition" is at best very misleading. It is mathematically wrong. It is self-contradictory and hence meaningless, because no number, no amount, is indefinitely large.

The corresponding adjective *infinite* means without end, boundless. Again, if your dictionary includes a definition such as "greater than any number," strike it out and forget it. "Greater than any number" suggests the existence of some other number or object which is greater than any number, and such a suggestion can be very misleading to a new student of mathematics.

Also, the use of the symbol "∞" in mathematics in connection with the concept of infinity suggests to some people that it represents a thing or a value. This, in turn, also confuses many beginners and even advanced students. This symbol is used in expressions best thought of as idioms. For example, the pair of symbols $\to\infty$ occurs often and means "increases without bound," "increases unendlessly." It does not mean "approaches something very large" (a statement that is ambiguous and therefore has no place in the Language of Mathematics). Similarly, it does not mean "approaches infinity" or "goes to infinity," English phrases that would be literally meaningful only if the word *infinity* meant a value, which it does not. In other words, "$\to\infty$" in the Language of Mathematics and "approaches infinity" or "goes to infinity" in English are all best viewed as unfortunate idiomatic expressions.

One will sometimes hear or read the phrase "an infinite number of … ." No number is "infinite," or without end, so the noun phrase "infinite number" is meaningless and incorrect. The terms "infinitely many" or "unendlessly many" are perhaps better. Still better, the entire clause in which the phrase appears should be appropriately restructured and reworded.

Occasionally, a large but finite set of numbers (a set containing a definite number of elements) will be extended by an additional value representing any other number not already in the set. The symbol ∞ is sometimes used for the additional value. Some computer arithmetic systems operate on a set defined in this way. In such a system, the division function is defined so that the value of 5 divided by 0, for example, is this additional value, often labeled "∞." Note that this is not the standard mathematical division. It is a similar—but different—computer division.

Examples: The sequence [1, 4, 9] is finite. The sequence [1, 2, 3, 4, …] is infinite because there is no end to the sequence; after every term in the sequence there is another term following it. As $n \to \infty$, $(1/n) \to 0$. In English, as the value of n increases without bound, the value of 1/n approaches 0 (i.e., becomes ever closer to 0). See the discussion of limits in Section 4.4.

In summary, the English adjective *infinite* means *endless* or *having no end* and the English noun "infinity" means the *property of having no end*. "Infinity" is **not** a value.

4.3 ITERATIVE DEFINITIONS AND RECURSION

It is sometimes convenient to define sequences and functions in terms of themselves. In other references to terms in a sequence and to functions it is also sometimes useful to include references to themselves in their subscripts or arguments. Such repetitive self-references in mathematical contexts are called *iteration* or *recursion*.

Example 1 Consider a sequence of numbers in which each term is the sum of its index and the preceding term, whereby the value of the first term is 1. The terms in this sequence can be defined iteratively as follows:

$$x_1 = 1 \qquad [4.3\text{-}1]$$

$$x_j = j + x_{j-1}, \qquad \text{for all } j \neq 1 \qquad [4.3\text{-}2]$$

This defines the sequence

$$[1, 3, 6, 10, 15, 21, \ldots] \qquad [4.3\text{-}3]$$

Example 2 The *factorial* function of a positive integer n is defined as the product of all integers between 1 and n inclusive. This English definition can be translated into a recursive statement in the Language of Mathematics as in Example 1:

$$\mathrm{fac}(1) = 1 \qquad [4.3\text{-}4]$$

$$\mathrm{fac}(n) = n * \mathrm{fac}(n-1), \qquad \text{for all } n \neq 1 \qquad [4.3\text{-}5]$$

Note that the definitions above define the values of the terms of the sequence or the function for positive integer values of the subscript or argument only. For other values of the subscript or argument, the value of the term or function is "defined" in terms of the preceding value, but that value is not defined. Note also that a base case [e.g., $x_1=1$ and fac(1)=1 above] is always needed to provide a basis for an iterative definition.

In Examples 1 and 2, iterative or recursive references were to the single preceding value of the index or argument. References to several other values can also be included, as illustrated in the following example.

Example 3 The *Fibonacci function* or *sequence* is commonly defined mathematically by the following recursive expression:

$$\text{fib}(0) = 0 \qquad [4.3\text{-}6]$$

$$\text{fib}(1) = 1 \qquad [4.3\text{-}7]$$

$$\text{fib}(n) = \text{fib}(n-1)+\text{fib}(n-2), \qquad \text{for } n\neq 0 \text{ and } n\neq 1 \qquad [4.3\text{-}8]$$

The above defines the sequence

$$[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, \ldots] \qquad [4.3\text{-}9]$$

The Fibonacci function is of interest in certain areas of mathematics and of nature.

In the definitions in the examples above, different expressions applied depending on the value of the argument or the index in question. The format above is common for selecting the appropriate expressions in mathematical texts written in a combination of English and the Language of Mathematics. The choice of the applicable expression can be written entirely in the Language of Mathematics by employing a selection function, although this is not common practice. If we define the value of the function reference choice(Bexp, exp1, exp2) to be the value of the expression exp1 if the value of the Boolean expression Bexp is true, and the value of the expression exp2 otherwise, the three definitions above can each be written as a single expression solely in the Language of Mathematics:

$$x_j = \text{choice}(j=1, 1, j+x_{j-1}) \qquad [4.3\text{-}10]$$

$$\text{fac}(n) = \text{choice}(n=1, 1, n*\text{fac}(n-1)) \qquad [4.3\text{-}11]$$

$$\text{fib}(n) = \text{choice}(n=0 \vee n=1, n, \text{fib}(n-1)+\text{fib}(n-2)) \qquad [4.3\text{-}12]$$

The base case or the iterative part of each definition is selected by the function “choice” instead of by the separate parts of the definitions as in the examples above. The iterative or recursive nature of each definition remains.

It is implicit from the context that the three equalities above are to apply for all appropriate values of the index j and the arguments n. To be precise and complete, one should express this explicitly by placing each of the three equations above in an appropriate quantified expression:

$$[\wedge\ j\ :\ j\in\mathbb{Z} \wedge 1\leq j : x_j = \text{choice}(j=1, 1, j+x_{j-1})] \qquad [4.3\text{-}13]$$

$$[\wedge\ n\ :\ n\in\mathbb{Z} \wedge 1\leq n : \text{fac}(n) = \text{choice}(n=1, 1, n*\text{fac}(n-1))] \qquad [4.3\text{-}14]$$

$$[\wedge\ n\ :\ n\in\mathbb{Z} \wedge 0\leq n : \text{fib}(n) = \text{choice}(n=0\ \vee\ n=1, n, \\ \text{fib}(n-1)+\text{fib}(n-2))] \qquad [4.3\text{-}15]$$

Iteration and recursion are related structurally to series (defined in Section 3.4.8). A series can be defined iteratively and, conversely, an iteratively defined function can often be expressed as a series. For example, consider the function s(·) defined by the following series:

$$s(n) = \sum_{k=1}^{n} x(k) \qquad [4.3\text{-}16]$$

Note that

$$s(n) = x(n) + \sum_{k=1}^{n-1} x(k) \qquad [4.3\text{-}17]$$

can be written as

$$s(n) = x(n)+s(n-1) \qquad [4.3\text{-}18]$$

Therefore, the function s(·) can alternatively be defined iteratively by the expressions

$$s(0) = 0 \qquad [4.3\text{-}19]$$

$$s(n) = x(n)+s(n-1), \qquad \text{for all } n\neq 0 \qquad [4.3\text{-}20]$$

or, more completely and explicitly, by the corresponding quantified expression

$$[\wedge\ n : n\in\mathbb{Z} \wedge 0\leq n : s(n) = \text{choice}(n=0, 0, x(n)+s(n-1))] \qquad [4.3\text{-}21]$$

Thus, a translator and developer of a mathematical model for a practical application has the choice of defining some functions either iteratively or as a series. The relationship between the function in question and various aspects of the given application and its environment will determine which approach is more natural, more understandable, and easier.

4.4 CONVERGENCE, LIMITS, AND BOUNDS

Often, the terms in a sequence become closer and closer to a certain value, even though no term ever becomes equal to it. An example is the infinite (endless) sequence

$$[1, 1/2, 1/3, 1/4, \ \ldots] \qquad [4.4\text{-}1]$$

The terms in this sequence become—and remain—arbitrarily close to 0, but no term is exactly equal to 0.

Expressed more formally: For every arbitrarily small positive number ε, some term in the sequence deviates from 0 by less than ε, and so do all following terms in the sequence. One says that the *limit* of 1/n is 0 as n increases without bound (as n "goes to infinity" in colloquial, but incorrect, language; see Section 4.2). The sequence *converges* to 0. A common notation for this limit is

$$\lim_{n\to\infty} (1/n) = 0 \qquad [4.4\text{-}2]$$

In the phrase above, “for every arbitrarily small positive number ε,” the word *small* is helpful to the reader in suggesting the concept or idea motivating the notion of limit, but the word *small* is logically superfluous. If the deviation in question is less than some number ε, it is clearly less than any other value larger than ε. Therefore, the word *small* can be eliminated from a mathematical definition. The statement “for every arbitrary positive number ε, some term in the sequence deviates from 0 by less than ε and so do all following terms in the sequence” is logically sufficient. The word *arbitrary* adds subjective emphasis but makes no objective contribution to the statement, so also the word *arbitrary* can be eliminated from this mathematical definition without changing its meaning.

Formally, L is the limit of the sequence [f(1), f(2), … f(i), … f(j), …] of numbers if, for every positive number ε, some term in the sequence deviates from L by less than ε and so do all following terms in the sequence. We translate this into a quantified expression: “For every positive number ε,”

$$[\forall\ \varepsilon : \varepsilon\in\mathbb{R} \land \varepsilon>0 : \ldots] \qquad [4.4\text{-}3]$$

“some term in the sequence” (or, in more mathematically oriented wording, “there exists a term in the sequence that”)

$$[\forall\ \varepsilon : \varepsilon\in\mathbb{R} \land \varepsilon>0 : [\exists\ i : i\in\mathbb{Z} \land i\geq 1 : \ldots]] \qquad [4.4\text{-}4]$$

“deviates from L by less than ε and so do all following terms in the sequence”

$$[\forall\ \varepsilon : \varepsilon\in\mathbb{R} \land \varepsilon>0 : [\exists\ i : i\in\mathbb{Z} \land i\geq 1 : [\forall\ j : j\in\mathbb{Z} \land i\leq j : |f(j)-L| < \varepsilon]]] \qquad [4.4\text{-}5]$$

which, as mentioned in Section 3.4.8, can also be written as

$$[\land\ \varepsilon : \varepsilon\in\mathbb{R} \land \varepsilon>0 : [\lor\ i : i\in\mathbb{Z} \land i\geq 1 : [\land\ j : j\in\mathbb{Z} \land i\leq j : |f(j)-L| < \varepsilon]]] \qquad [4.4\text{-}6]$$

Notice how the English sentence above has been translated to a quantified expression step by step, quantification by quantification. The following slight revision of the English sentence above might make the last step clearer: “For every positive number ε, some term in the sequence and all following terms deviate from L by less than ε.” Often, restructuring an English statement in such a way makes it easier to translate the statement into an expression in the Language of Mathematics.

The quantified expressions 4.4-5 and 4.4-6 are equivalent to the English sentence “For every positive number ε, some term in the sequence [f(1), f(2), … f(i), …] and all following terms deviate from L by less than ε.” The expression using the limit notation above is

$$\lim_{i\to\infty} f(i) = L \qquad [4.4\text{-}7]$$

which by definition means expression 4.4-5 or 4.4-6.

Note that expressions 4.4-6 and 4.4-7 are both Boolean functions of the value of the variable L and of the function f. In particular, in expression 4.4-7 in limit notation, the variable i is not a free variable; it is a quantified variable.

Expressions 4.4-5, 4.4-6, and 4.4-7 evaluate to either true or false, depending on whether or not the value of L is the limit of the function f as its argument increases without bound. Sometimes, one would like to know whether or not a sequence converges (i.e., has a limit) without determining the limit (if it exists). A theorem in mathematics states that a sequence converges if and only if after some term, every two terms are arbitrarily close to each other. Expressed more precisely and in a way similar to the statement for a limit in earlier paragraphs in this section, a sequence converges if and only if for every positive number ε, some term in the sequence exists such that every two later terms deviate from each other by less than ε.

This statement can be translated step by step, quantification by quantification, as earlier: For every positive number ε,

$$[\forall\ \varepsilon : \varepsilon\in\mathbb{R} \wedge \varepsilon>0 : \ldots] \tag{4.4-8}$$

some term in the sequence exists

$$[\forall\ \varepsilon : \varepsilon\in\mathbb{R} \wedge \varepsilon>0 : [\exists\ i : i\in\mathbb{Z} \wedge i\geq 1 : \ldots]] \tag{4.4-9}$$

such that every two later terms deviate from each other by less than ε.

$$\begin{aligned}&[\forall\ \varepsilon : \varepsilon\in\mathbb{R} \wedge \varepsilon>0 : [\exists\ i : i\in\mathbb{Z} \wedge i\geq 1: \\ &\qquad [\forall\ j, k : j\in\mathbb{Z} \wedge k\in\mathbb{Z} \wedge i < j \wedge i < k : |f(j)-f(k)| < \varepsilon]]]\end{aligned} \tag{4.4-10}$$

If the value of expression 4.4-10 is true for the function f, the sequence [f(1), f(2), … f(i), …] converges and has a limit. Such a sequence is called a *Cauchy sequence*. If the value of expression 4.4-10 is false, the sequence does not converge; it has no limit.

An obvious example of a sequence that does not converge is [1, 2, 3, 4, …]. Clearly, the values in this sequence increase without bound; they do not become closer and closer to some value. The terms in this sequence do not become and remain arbitrarily close to any particular value.

A less obvious example of a sequence that does not converge is the sequence [g(1), g(2), … g(i), … g(j), …], where g is the function

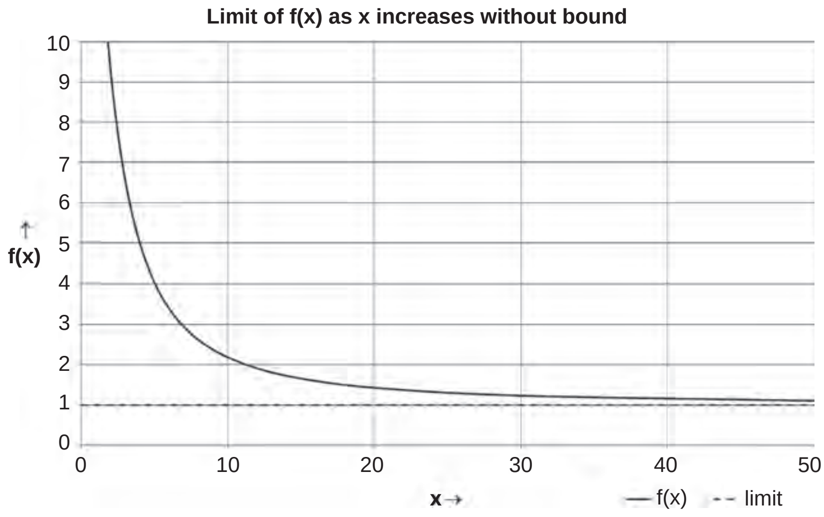
$$g(n) = \sum_{i=1}^{n} 1/i \tag{4.4-11}$$

For large values of n, this function increases very slowly with n, but it does increase without bound.

The definition above of the limit of a sequence can be generalized to a function of a continuous variable. In other words, the argument of the function (f in the example above) need not be an integer, but can be any real number. If f is a function of a real number, L is the limit of the function f as its argument increases without bound if

$$[\wedge\ \varepsilon : \varepsilon\in\mathbb{R}\ \wedge \varepsilon>0 : [\vee\ b : b\in\mathbb{R} : [\wedge\ x : x\in\mathbb{R}\ \wedge\ b\leq x : |f(x)-L| < \varepsilon]]] \tag{4.4-12}$$

Below is a graph illustrating such a function. In this example, 1 is the limit of the function as the argument increases without bound.



In the examples above, the limit of the value of a function was defined as the argument of the function increased without bound. A limit can also be defined as the argument decreases without bound (e.g., becomes more and more negative). As an exercise for the reader, modify expression 4.4-12 for a function g as its argument decreases without bound.

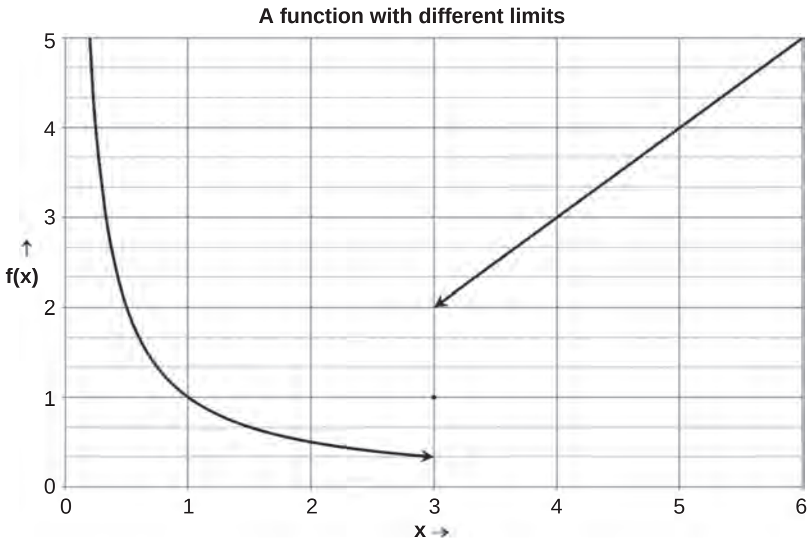
The notion of convergence to a limit can also apply to a function as its argument approaches some particular value. The argument need not increase or decrease indefinitely. The argument need not actually become equal to the particular value it approaches (e.g., if the function is not defined for that value of the argument). Similarly, as in the case examined above, the value of the function need not become equal to the limit. The value of the function need only become and remain arbitrarily close to the limit.

For example, consider the value of the expression $(x^2-4)/(x-2)$. When the value of x is 2, the value of this expression is not defined, because division by 0 is not defined. However, as the value of x approaches 2, either from above (values of x greater than 2) or from below (values of x less than 2), the value of this expression becomes and remains arbitrarily close to 4 (i.e., converges to 4). Therefore, mathematicians say that the limit of the value of the expression $(x^2-4)/(x-2)$ is 4 as the value of x approaches 2.

Functions defined in different regions of the argument can converge to different limits, depending on whether the value of the argument approaches a particular value from below or from above. Consider, for example, the function defined as follows:

$$\begin{array}{ll} f(x) = 1/x, & \text{for } 0<x<3 \\ f(x) = 1, & \text{for } x=3 \\ f(x) = x-1, & \text{for } 3<x \end{array}$$

As x increases toward 3, the value of f(x) decreases to 1/3. That is, the limit of f(x) is 1/3 as x approaches 3 from below. As x decreases toward 3, the value of f(x) decreases to 2. That is, the limit of f(x) is 2 as x approaches 3 from above. When the value of the argument is equal to 3, the value of the function is 1, which is neither of the two limits. This function and its limits are shown in the graph below.



Mathematical expressions corresponding to those given earlier but for a function converging to a limit as the argument approaches a particular value x0 from one side or the other are given below.

L is the limit of f(x) at x0 as x approaches x0 from below if and only if the expression

$$[\wedge\ \varepsilon : \varepsilon{\in}\mathbb{R} \wedge \varepsilon{>}0 : [\vee\ d : d{\in}\mathbb{R} \wedge d > 0 : \\ [\wedge\ x : x{\in}\mathbb{R} \wedge x0{-}d < x < x0 : |f(x){-}L| < \varepsilon]]] \qquad [4.4\text{-}13]$$

is true.

L is the limit of f(x) at x0 as x approaches x0 from above if and only if the expression

$$[\wedge\ \varepsilon : \varepsilon{\in}\mathbb{R} \wedge \varepsilon{>}0 : [\vee\ d : d{\in}\mathbb{R} \wedge d > 0 : \\ [\wedge\ x : x{\in}\mathbb{R} \wedge x0 < x < x0 + d : |f(x){-}L| < \varepsilon]]] \qquad [4.4\text{-}14]$$

is true.

The reader should notice how the two expressions above describe the limits of f(x) in the graph above (1) as x approaches 3 from below and (2) as x approaches 3 from above.

If a function is smooth (in mathematical terminology *continuous*) at a point, the limits as x approaches the point from below and from above will be equal to each other and to the value of the function at that point. This statement is not the usual definition of *continuous*, but is equivalent to it. By the usual definition, a function f is *continuous* at x0 if (and only if)

$$\begin{aligned}&[\wedge\ \varepsilon : \varepsilon\in\mathbb{R} \wedge \varepsilon>0 : [\vee\ d : d\in\mathbb{R} \wedge d > 0 :\\ &\qquad [\wedge\ x : x\in\mathbb{R} \wedge |x-x0| < d : |f(x)-f(x0)| < \varepsilon]]]\end{aligned} \tag{4.4-15}$$

is true. The reader should understand why this definition and the statement in the first sentence of this paragraph are equivalent ways of defining the mathematical term *continuous*.

In the examples above, the values of the function approach the limit smoothly and monotonically. This is not necessary; the values of the function can and sometimes do increase and decrease as they approach the limit. The only requirement is that the range between the peaks and dips decreases sufficiently so that for any small, arbitrary deviation ε, the value of the function comes and remains within ε of the limit for any positive ε, no matter how small.

Notice the structural similarity among the several definitions above of a limit: For any (every) arbitrarily small range (range F) of values of the function, there exists a range (range A) of values for the argument of the function, such that the value of the function for every argument in the range A is within the range F. This structure arises often in mathematical expressions for convergence or limits of various kinds. It will arise again in Section 4.5.

Sometimes one is interested in the largest (maximum) or the smallest (minimum) value a variable or a function can have. Alternatively, upper or lower bounds are sometimes of interest, even if these bounds are not actually taken on by the variable or function of interest. The corresponding theoretical mathematical literature deals with upper bounds, lower bounds, least upper bounds, greatest lower bounds, and so on. These concepts are introduced briefly in the paragraphs below. Conceptually and in the forms of the expressions arising, these ideas are similar to convergence and limits dealt with above.

A value UB is said to be an *upper bound* of a set $\mathbb{S}$ if UB is greater than or equal to every element in $\mathbb{S}$, that is, if the following quantified expression is true:

$$[\wedge\ x : x\in\mathbb{S} : x\leq UB] \tag{4.4-16}$$

Note that UB may, but need not, be an element of the set $\mathbb{S}$.

Any value larger than one upper bound is also an upper bound, so the concept of an upper bound is rather weak and inadequate for many purposes. The smallest of the upper bounds of a set is a stronger and more meaningful characteristic of the set. Therefore, the *least upper bound* LUB (also called the *supremum*, abbreviated "sup") is defined to be that upper bound of $\mathbb{S}$ which is less than or equal to every upper bound of $\mathbb{S}$. The least upper bound LUB must, therefore, satisfy the following expression:

$$[\wedge\ x : x\in\mathbb{S} : x\leq LUB] \wedge [\wedge\ b : [\wedge\ x : x\in\mathbb{S} : x\leq b] : LUB\leq b] \tag{4.4-17}$$

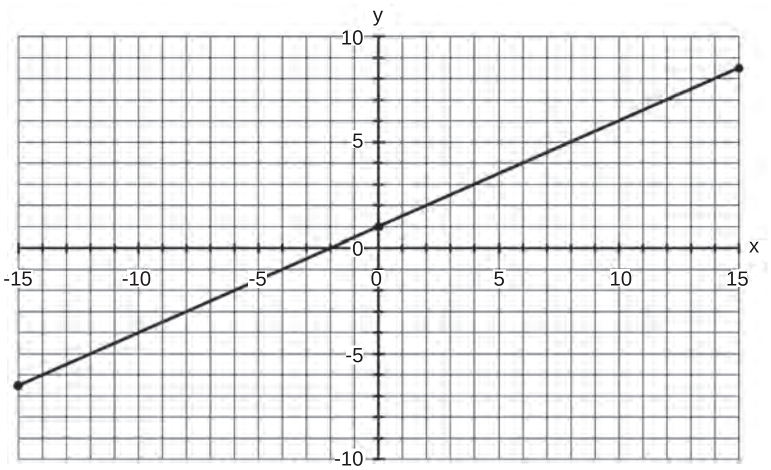
In most contexts arising in practical applications, it can be proved that if $\mathbb{S}$ is not empty and that if any upper bound of $\mathbb{S}$ exists, a least upper bound LUB satisfying expression 4.4-17 exists.

Lower bounds and greatest lower bounds are defined in the corresponding ways. In many applications, the set $\mathbb{S}$ above is a subset of $\mathbb{R}$, the real numbers.

4.5 CALCULUS

Calculus is a subdiscipline of mathematics that deals with the *rate of change* of the value of one function or variable as the value of another variable changes. The rate of change is the *slope* of the graph of the function or variables in question. The mathematical term is the *derivative* of the one function or variable *with respect to* the other.

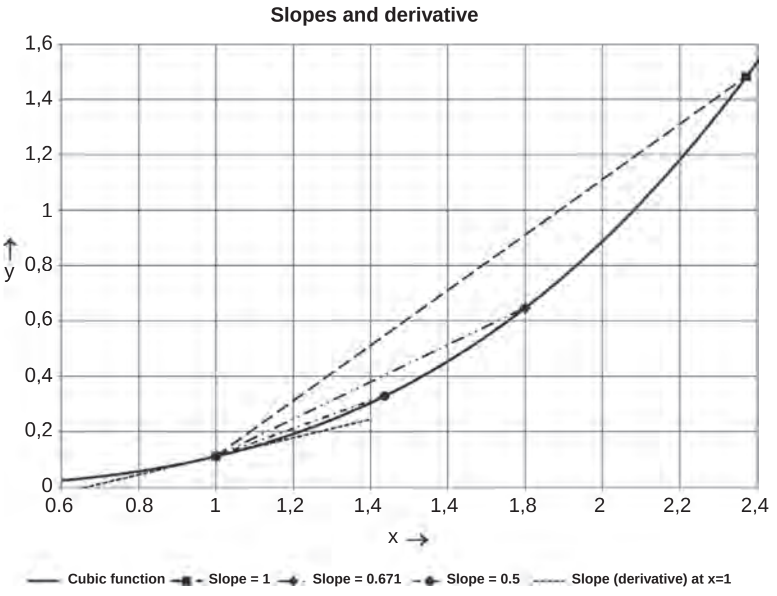
The graph of $y=x/2+1$ in Example 1 in Section 3.4.6 is a straight line:



When x increases by 1, y increases by 1/2; when x increases by 2, y increases by 1; and so on. The slope of the graph is defined to be the change in y divided by the corresponding change in x (i.e., in this example, the slope is 1/2). The slope is the same at all places along the function because the graph is a straight line.

This notion can be generalized for functions whose graphs are not straight lines, but curves. As an example, consider $y=x^2$. If x increases from 0 to 1, y increases from 0 to 1, so the slope over this interval is 1. If x increases from 1 to 2, y increases from 1 to 4, so the slope over this interval is 3. If x increases from 1 to 1.5, y increases from 1 to 2.25, so the slope over this interval is 1.25/0.5, or 2.5. Thus, the slope over an interval of a function whose graph is not a straight line varies depending on both the position and the length of the interval selected. If x increases from 1 to 1.1, y increases from 1 to 1.21, so the slope is 0.21/0.1, or 2.1. If x increases from 1 to 1.001, y increases from 1 to 1.002001, so the slope is 0.002001/0.001, or 2.001. The limit of the slope as the interval between the two values of x becomes smaller and smaller is 2. This limit is called the *slope*, or *derivative*, of the function x^2 at the point $x=1$.

The following is a graph of $y=x^3/9$ and several straight lines from the base point (x=1, y=1/9) to different points to the right. The several straight lines connect the base point with points successively closer to the base point. The uppermost straight line has a slope of 1. The second straight line from the top has a slope of 0.671. The next-lower straight line has a slope of 0.5. The limit of the slope of such lines as their rightmost points approach x=1 from above is 1/3. This limit is the slope, or derivative, of the function at x=1.



The lowest straight line in this graph has a slope of 1/3, the slope of the function at x=1, and touches (includes) the base point. Note that in contrast to the other straight lines shown in the graph, the lowest line never goes above the curve. It is *tangential* to the graph of the function at x=1.

The general formula for the slope of a function f along the interval from x to x1 is

$$\text{slope} = (f(x1)-f(x))/(x1-x) \qquad [4.5\text{-}1]$$

or, equivalently,

$$\text{slope} = (f(x+dx)-f(x))/dx \qquad [4.5\text{-}2]$$

where dx is defined as the difference x1−x.

The derivative of f(x) with respect to x is most commonly written

$$\frac{df(x)}{dx} \qquad [4.5\text{-}3]$$

but is also often written typographically more simply as df(x)/dx. Each letter “d” indicates the “difference” or “differential” of the following value.

The derivative of the function f(x) with respect to x is the limit of the slope (expression 4.5-2) as dx approaches 0:

$$\frac{df(x)}{dx} = \lim_{dx \to 0} (f(x+dx)-f(x))/dx \qquad \text{[definition of derivative, 4.5-4]}$$

The corresponding quantified expression defining the derivative of a function is as follows. The function g(x) is the derivative of the function f(x) with respect to x if and only if

$$[\wedge\ \varepsilon : \varepsilon\in\mathbb{R} \wedge \varepsilon>0 : [\vee\ d : d\in\mathbb{R} \wedge d>0 :$$
$$[\wedge\ dx : dx\in\mathbb{R} \wedge 0 < |dx| < d : |((f(x+dx)-f(x))/dx)-g(x)|<\varepsilon]]] \quad [4.5\text{-}5]$$

is true. Note that the only free variables in this quantified expression are the functions f and g and the argument value x. Thus, the derivative g is a function of the function f and the argument value x only. More generally, the derivative can be defined for complex variables and functions also.

As with limits as discussed in Section 4.4, the derivative can be defined either as dx approaches 0 only from above or only from below. This is meaningful if the function f has a kink, a sudden, abrupt bend, at the argument value x in question. If only the derivative from above is to be considered, the term $0<|dx|<d$ in the quantified expression above must be changed to $0<dx<d$. If only the derivative from below is to be considered, this term must be changed to $-d<dx<0$. If the function f is not defined for the argument value x, but is defined for all neighboring argument values, a corresponding modification can be made to the quantified expression above.

The derivative of a particular function f can be derived algebraically starting with the expression 4.5-4 "definition of derivative." For example, if f(x) is defined to be x^2, then

$$\begin{aligned}
& (f(x+dx)-f(x))/dx \\
= \quad & ((x+dx)^2-x^2)/dx \\
= \quad & (x^2 + 2*x*dx + (dx)^2-x^2)/dx \\
= \quad & (2*x*dx + (dx)^2)/dx \\
= \quad & \qquad\qquad\qquad\qquad \text{[provided that } dx \neq 0] \\
& 2*x + dx
\end{aligned}$$

The expression $2*x+dx$ clearly becomes arbitrarily close to $2*x$ as dx approaches 0; that is,

$$\lim_{dx \to 0} ((x+dx)^2 - x^2)/dx = 2*x \qquad [4.5\text{-}6]$$

and, therefore, the derivative of the function $f(x) = x^2$ with respect to x is $2*x$.

A derivative is a function, and one can consider its derivative, that is, the derivative of a derivative, also called the *second derivative*. The second derivative of f with respect to x is written $d^2f(x)/dx^2$. It is the derivative of df(x)/dx, which can be written as (d/dx)(df(x)/dx). This suggests the view that the derivative operator or function, written symbolically as d/dx, is applied to the first derivative df(x)/dx. Derivatives of derivatives can be continued indefinitely: for example, $d^2f(x)/dx^2$, $d^3f(x)/dx^3$, and $d^4f(x)/dx^4$.

In many applications one works with functions of two or more variables, and the derivatives with respect to any or all of them are of interest. For example, in dealing with waves (e.g., the displacement of a string in a stringed instrument, the voltage across the wires in a transmission line, or an electromagnetic field in space) one might define a function amp (amplitude) with arguments x (representing the position along the string or line or in space) and t (representing time). The derivative of amp with respect to x is then written ∂amp(x, t)/∂x or simply ∂amp/∂x and the derivative of amp with respect to t, ∂amp(x, t)/∂t or simply ∂amp/∂t. The symbol ∂ is used instead of d to indicate that the derivative in question is a *partial derivative*; that is, it is the derivative with respect to only one of the several variables (arguments of the function) in question. The different partial derivatives are often related in ways depending on the physics or other underlying characteristics of the phenomena in question. For example, Appendix F illustrates some aspects of waves that can be derived from the fundamental wave equation, which relates the second partial derivative with respect to x with the second partial derivative with respect to t.

As pointed out above, the usual notational form for the derivative is dy/dx or a typographical variant thereof. Other notational forms are or have been in use, such as f' for the derivative of f and $\dot{y}$ for the derivative of y. It was also pointed out above that the derivative of a function f at argument value x is a function of f and x. The notational forms f' and $\dot{y}$ do not explicitly indicate the variable with respect to which the derivative is to be taken and are, perhaps therefore, not widely used. The notational form dy/dx does make this explicit and clearly suggests the differential and quotient nature of the terms underlying its definition, but it is not an otherwise typical notation for a function. Something like d(f, x) would be a more typical notation for the derivative function, but it has not come into accepted use. Apparently, the fact that the notation dy/dx suggests the quotient of a deviation in y divided by the corresponding deviation in x is a significant reason for this notation becoming and remaining the standard notational form for the derivative.

The *integral* of a function f is a function g such that f is the derivative of g. That is, the integration operation is the inverse of taking the derivative. The integral of f with respect to the variable x is written

$$\int f(x)\,dx \qquad [4.5\text{-}7]$$

This integral is a function g such that f is the derivative of g. That is, g is the integral of f and f is the derivative of g:

$$g(x) = \int f(x)\,dx \qquad [4.5\text{-}8]$$

$$f(x) = dg(x)/dx \qquad [4.5\text{-}9]$$

Notice that the derivative f is unique for a given g. However, the integral g is unique for a given f except for a constant. If the function f(x) is the derivative of g(x), f(x) is also the derivative of (g(x)+1), (g(x)+7), and so on. This follows from the fact that any two functions that differ only by a constant have the same slope, and hence the same derivative, at every value of the argument. For example, the derivative of x^2 + any constant is $2*x$ and, therefore, the integral of $2*x$ is x^2 + any constant.

The derivative of a function was interpreted graphically above as the slope (rate of change) of the function at any argument value in question. The integral of a function f is interpreted graphically as the area under the curve of the function f between two values of the argument. This follows from the observation that the rate of change of this area as one of the argument values changes is the height of the changing border (i.e., the value of f at that argument value). Thus, the value of f is the rate of change (derivative) of the area, so the area is given by the integral of f (i.e., the function named g above). The two argument values bounding the area are called the *limits of integration* and the integral between the two limits is called a *definite integral*. It is written

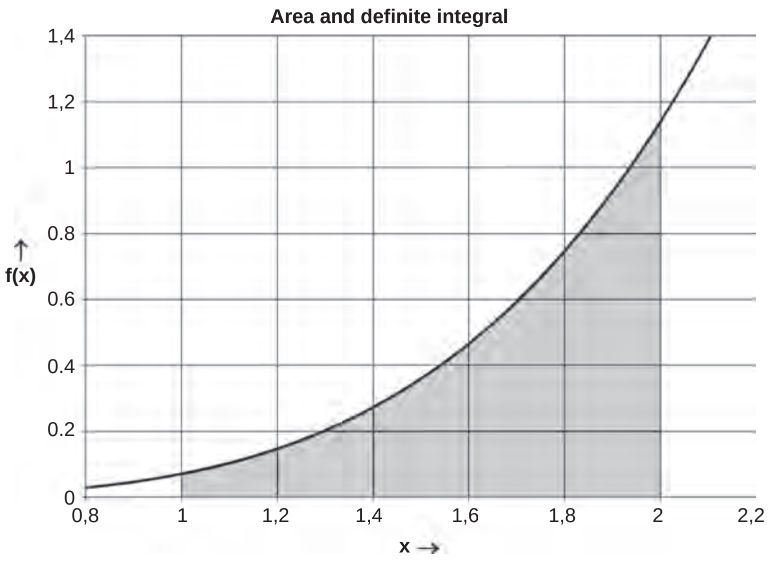
$$\int_a^b f(x)\,dx \qquad [4.5\text{-}10]$$

where a and b are the limits of integration. The value of this definite integral is the area under the curve f(x) between the argument values a and b.

For example, the area corresponding to the definite integral

$$\int_1^2 f(x)\,dx \qquad [4.5\text{-}11]$$

is shaded in the graph that follows.



Derivatives and integrals are important in many applications of mathematics. They arise in very basic physical laws such as Newton's laws of motion. For example, if a net force F is applied to a body of mass m, the acceleration of that body will be proportional to that force and inversely proportional to the mass m. Acceleration is the rate of change of velocity with respect to time. The velocity is, in turn, the rate of change of position of the body with respect to time. Thus, acceleration is the second derivative of position with respect to time. Newton's law can, therefore, be expressed as

$$\mathrm{F} = \mathrm{m}{*}\mathrm{d}^2\mathrm{x}/\mathrm{dt}^2 \qquad [4.5\text{-}12]$$

provided that the force F, mass m, position x, and time t are measured in corresponding physical units.

Derivatives and integrals appear in connection with many other physical phenomena. For example, energy is the integral of force with respect to the distance through which the force is applied:

$$\mathrm{Energy} = \int_{\mathrm{a}}^{\mathrm{b}} \mathrm{F(x)\,dx} \qquad [4.5\text{-}13]$$

while power is the derivative of energy with respect to time:

$$\mathrm{Power} = \mathrm{dEnergy/dt} \qquad [4.5\text{-}14]$$

or, equivalently,

$$\mathrm{Energy} = \int_{\mathrm{t0}}^{\mathrm{t1}} \mathrm{Power(t)\,dt} \qquad [4.5\text{-}15]$$

When electrical energy is delivered to a user, a meter measures the amount of energy delivered. The power delivered is the product of the voltage e and the current i, each as a function of time. The traditional meter effectively measures the voltage and current, multiplies these values, and integrates the product over time by an appropriate configuration of coils, a magnet, a rotor, and other parts, but without a computer! The meter effectively determines the value of the definite integral

$$\int_{\text{initial time}}^{\text{current time}} \mathrm{e(t){*}i(t)\,dt} \qquad [4.5\text{-}16]$$

and displays this value continuously on a set of dials or digital wheels.

The three most common passive elements in electrical circuits are the resistor, the capacitor, and the inductor. The mathematical model of each is a relation between the voltage e across the element and the current i flowing through it. The mathematical models for two of these three elements involve derivatives with respect to time:

for the inductor: $\mathrm{e} = \mathrm{L} * \mathrm{di/dt}$ [4.5-17]

for the capacitor: $\mathrm{i} = \mathrm{C} * \mathrm{de/dt}$ [4.5-18]

Other examples of derivatives arise in mathematical models of the storage and flow of all types of liquids, such as water, oil, or chemicals. The flow of a liquid into a storage tank or reservoir minus the flow of the liquid out of the tank or reservoir is the derivative with respect to time of the volume of liquid stored. The storage and flow of gases, heat, and energy give rise to the same kinds of relationships involving derivatives.

In business and commerce, derivative relationships arise in many places. Many optimization problems involve differential equations. Delivery from inventory represents changes (the derivative with respect to time) of inventory levels. Interest and return on investment are the derivative with respect to time of the investment value. Comparable notions of levels and flows appear even in double-entry bookkeeping, with its two categories of accounts: balance sheet accounts (assets, liabilities, and equity) and profit and loss accounts (income and expenditures). The latter category represents short-term changes (sales, purchases) to the first-category levels (things of value on hand, things owed or owned).

In this section we presented basic concepts of calculus—derivatives and integrals—and introduced basic terminology commonly used in this mathematical discipline and especially in its application. Further aspects of calculus, such as solving equations and other expressions involving derivatives or integrals, are to a lesser extent language topics, and to a greater extent mathematical topics, and hence go beyond the scope of the book. They are the subject of a very extensive mathematical literature.

4.6 PROBABILITY THEORY

Many important applications of mathematics require analyzing and reasoning about events or outcomes of processes involving uncertainty (i.e., nondeterministic results that cannot be predicted exactly with the information available). A common structure for appropriate mathematical models has been developed within the area of applied mathematics and it has become known as *probability theory*. It can be viewed as a sort of generalized template for frequently repeating parts of mathematical models for events and processes involving uncertainty. Much of the associated terminology belongs to the interpretation of the mathematical model in the application domain rather than to the Language of Mathematics proper. See the sections immediately below for examples of interpretations of mathematical models and their parts and Section 6.13 for a definition of *interpretation* as used here.

The idea underlying the application of probability theory is that of the relative frequency (likelihood) of occurrence of events that cannot be predicted exactly in advance. A simple example is the result of rolling a die. Joint events resulting from rolling two or more dice, or rolling one die many times in succession, are more complex examples. Examples in important commercial applications include demand for and sales of products, supplying articles from inventory, prices and volumes of goods in financial markets, failure rates of mechanisms and systems, and events leading to insurance claims. Examples in scientific and engineering work include noise in electrical communication systems; nuclear decay; failure modes and frequencies in all kinds of physical components, devices, systems, and structures; loads (such as from earthquakes) on such structures and systems; genetics; mutations in DNA; and statistical analyses of letters and words in texts in various languages.

From the standpoint of the Language of Mathematics, probability theory is nothing other than a structure of sets and a function as already introduced in earlier sections, primarily Sections 4.1.1 and 3.3. How these sets and functions are combined and interpreted to form probabilistic mathematical models for many applications is shown in the following sections.

Probability theory is a special case of *measure theory* in mathematics. In probability theory, every probability is required to be less than or equal to 1. In measure theory, the measure of a set can be any nonnegative real value; it is not bounded above. Otherwise, the mathematical model introduced in the following section applies to measure theory also.

It must be emphasized that most of the new terminology introduced in this section and its subsections does not belong to the Language of Mathematics but to the jargon of the application areas. This begins with the term *relative frequency* and goes throughout the text in this section.

4.6.1 Mathematical Model of a Probabilistic Process

Mathematically, a nondeterministic process and the probabilities of its possible resulting events are modeled with a combination of the following:

- A nonempty set S
- A nonempty set A of subsets of S
- A function p mapping each element of A (subset of S) to a real number

all subject to the following requirements:

- $\emptyset \in A$.
- $S \in A$.
- $p(\emptyset)=0$.
- $p(S)=1$.
- For every $B \in A$, $0 \leq p(B) \leq 1$.
- For every $C \in A$ and every $D \in A$, $C \cup D \in A$ and $C \cap D \in A$.
- For every $C \in A$ and every $D \in A$, $p(C \cup D) = p(C) + p(D) - p(C \cap D)$.

The parts of this model are interpreted as follows. The elements of S are the individual possible results (outcomes) of the nondeterministic process. The set S includes every possible result of the process. So that every probability value can be interpreted as a relative frequency, it must be a real number between 0 and 1 inclusive. The value of p(X) is the *probability* (likelihood) that the result of the process is an element of the set X; p(X) is a measure of the set X. This probability is interpreted as the relative frequency with which some element of X occurs in a large number of trials of the process. The probability that something results is 1, and that nothing results is 0. The probability of the union of two disjunct (nonoverlapping) subsets of S is the sum of the probabilities of the two subsets. In the case of two overlapping subsets of S, the sum must be corrected for the otherwise double counting of the intersection of the two subsets. To permit these calculations, both the union and the intersection of every pair of subsets in A must also be in A (so that the probabilities of the union and the intersection are defined).

The structure (S, A, p) defined above is called the *probability space* of the nondeterministic process in question. The set S is called the *sample space*. Sometimes the term *probability space* refers just to the set S. In the theoretical literature, the Greek capital letter omega (Ω) is often used instead of S. Each element of A (a subset of S) is often called an *event*.

The result of a nondeterministic process is often represented by a variable that either takes on values in the set S of the corresponding probability space (S, A, p) as defined above or is a function of the elements of S. Such a variable (or function) is called a *random variable*. Conditions or categories (sets) of values of the random variable are typically formulated as Boolean expressions [e.g., $x=5$, $x<4$, IsEven(x), $x=5 \vee x=6$, x=red]. Each such Boolean expression identifies an element of A (a subset of S) (i.e. an event). The Boolean expression representing an event must correspond directly (in the sense of the correspondence of a set and a Boolean function presented in Section 4.1.5) to the subset of the sample space representing the event in question. That is, the Boolean expression representing an event must be equivalent to an expression of the form $x \in B$ (e.g., $x \in \{5\}$, $x \in \{1, 2, 3\}$, $x \in \{2, 4, 6\}$), where x is the random variable in question and B is the subset of the sample space representing the event in question.

If the sample space S is finite, A is typically defined to be the set of all subsets of S, and p is defined by assigning a probability to each element of S. The probabilities of subsets of S containing two or more elements of S follow from the last requirement listed above in the definition of a probability space. This approach can generally be used to define the probability function p for a countably infinite sample space S also. (A set is *countably infinite* if its elements can be put into a one-to-one correspondence with the positive integers.) Otherwise, and particularly if the sample space S is infinite and not countable, other approaches to defining the set A and the probability function p must be employed.

The probability of the event $x \in B$ is p(B), where p is the function p of the corresponding probability space (S, A, p) as defined above. This probability is often written as $\Pr\{x \in B\}$. The notation $\Pr\{x \in B\}$ may suggest to some that Pr is a function whose argument is a Boolean value, but the definitions above require that the argument of a function whose value is a numerical probability be a subset of the sample space S. Note that $\Pr\{x \in B\}$ is not the value of a function of a Boolean value. The probability in question depends on the set B, not on a Boolean value true or false. More generally, Pr{Bexp}, where Bexp is a Boolean expression including one or more references to a random variable, should be interpreted as p(B), where B is the set of all values of the random variable for which the value of the expression Bexp is true. In other words, Pr{Bexp} is the probability of the set corresponding directly to the Boolean expression Bexp as described in Section 4.1.5.

Notational forms such as $\Pr\{x{=}5 \vee x{=}6\}$ and $\Pr\{x{<}4\}$ are used because they are readable, simple in form, and easy to work with. However, it should always be kept in mind that they are written instead of a term such as p(B), where B is a subset of S. More specifically, if Bexp is a Boolean expression containing references to a random variable x whose values are elements of a set (sample space) S, then Pr{Bexp} is an abbreviation or an idiom for

$$p([\cup\ x : x{\in}S \wedge \text{Bexp} : \{x\}]) \qquad [4.6.1\text{-}1]$$

Notice that the numerical value of the probability depends on the *function* represented by Bexp (and the set corresponding to that Boolean function; see Section 4.1.5), not on the *value* of Bexp for any particular value of the random variable.

Note also that if the Boolean expression Bexp contains references to more than one variable, the notation Pr{Bexp} does not distinguish between random variable(s) and other variables. Often, the context within the mathematical model will make the distinction between these two different types of variables clear. If not, the distinction will follow from the English statements in the interpretation of the mathematical model defining the variables in question. When using the notational form Pr{Bexp}, one must ensure that this distinction is made explicitly somewhere. From a purely mathematical standpoint, the notational form $p([\cup\ x : x{\in}S \wedge \text{Bexp} : \{x\}])$ is preferable because it distinguishes unambiguously between random and other variables in the expression Bexp. On the other hand, the notation Pr{Bexp} is, to a human reader, generally clearer. It facilitates understanding the mathematical model in its entirety and its connection with the practical application.

A notational form such as $\Pr\{S, p, x, \text{Bexp}\}$, defined to mean $p([\cup\ x : x{\in}S \wedge \text{Bexp} : \{x\}])$, would represent a compromise between the two extremes and distinguish between random and other variables. However, this notation has not become an accepted standard in the Language of Mathematics.

Defining the sample space completely and in detail is the most important step in formulating a mathematical model of a nondeterministic (probabilistic) process. In practice, this step is too often handled carelessly or only implicitly—or even not at all—with the consequence that the model is inadequate or even wrong, and the results of the analysis are not applicable. Each element of S should be defined in such a way that it cannot, even conceptually, be subdivided further.

For example, when modeling the process of rolling two dice, one should consider the result of rolling each individual die separately, even if the two dice are identical in appearance. One can conceptually identify each die individually, and the sample space should take this possibility into consideration. If one does not subdivide each possible result in full detail, one is likely to make errors in assigning probabilities to the resulting events (i.e., in defining the probability function p). Such errors will generally lead to incorrect conclusions about the probabilistic process involved.

Example Rolling a Dice:

After the die is rolled and it finally comes to rest, one, and only one, of the six faces will be up. We can identify these six possibilities with the numbers 1, 2, 3, 4, 5, and 6. The sample space S is, therefore, the set {1, 2, 3, 4, 5, 6}. If each of these events is equally likely, the probability of each is 1/6 [i.e., p(1)=1/6, p(2)=1/6, etc.].

The probability of the event that the result of rolling the die is an even number can be calculated from the requirements of a probability space listed at the beginning of this section. The event that an even number results is the event that either 2, 4, or 6 results. That is, the probability that the result is even is the probability of the set {2, 4, 6}, which is p(2)+p(4)+p(6), which is 1/2. Correspondingly, the probability that either 1 or 6 results is p(1)+p(6), which is 1/3.

The reader should consider a standard deck of 52 cards and the process of drawing a single card at random (i.e., each card with the same probability). Calculate the probability of each of the following:

- Drawing a spade
- Drawing a red card
- Drawing a face card
- Drawing a red face card
- Drawing a card that is either red or a face card

4.6.2 Mean, Median, Variance, and Deviation

If the elements of the sample space of a probabilistic process are numbers, it is meaningful to define several numerical characteristics of the probability space. Some of these characteristics indicate the average or "typical" value generated while others measure the typical variation of the values generated.

A common measure of the "typical" value generated by a probabilistic process is the average value. In the terminology of probability theory, it is called the *expected* *value* or *mean*. The idea underlying the formal definition given below is that the mean is the average of the numbers that will result if the probabilistic process is repeated many times.

In the case of rolling a die as described in Section 4.6.1, we expect a 1 to result 1/6 of the times the die is rolled; a 2, 1/6 of the times; and so on. The average would therefore be

$$1*1/6 + 2*1/6 + 3*1/6 + 4*1/6 + 5*1/6 + 6*1/6 \qquad [4.6.2\text{-}1]$$

which is equal to

$$(1+2+3+4+5+6)*1/6 \qquad [4.6.2\text{-}2]$$

and, in turn, to 21/6 and to 3.5. Note that the mean (expected value) is not necessarily a value in the sample space S.

Formally, the mean is defined by the formula

$$\text{mean} = \sum_{v \in S} v*p(v) \qquad [4.6.2\text{-}3]$$

or, using the more general form for a series (quantified expression) as introduced in Section 3.4.8,

$$\text{mean} = [+\ v : v \in S : v*p(v)] \qquad [4.6.2\text{-}4]$$

provided that S is a finite or a countably infinite set. Otherwise, the mean is defined in a corresponding way using integral calculus. Calculus is outlined briefly in Section 4.5, but is otherwise outside the scope of this book.

The expected value (mean) of the random variable S is often written as E(S), using the name of the sample space to represent the random variable in question.

Another measure of the "typical" value generated by a probabilistic process is the *median*, which is a value in the middle of the probability distribution (i.e., a value below which and above which the random variable occurs with equal probability). In the case of rolling a die above, a value 1, 2, or 3 results with probability 1/2, and a value 4, 5, or 6 results with probability 1/2, so the median in this case is any number between 3 and 4. In such a situation, the midpoint is typically chosen (here, 3.5). Strictly speaking, the term *median* is usually defined mathematically with regard to a (finite) collection of numbers, not a probability space. Applying this term to a probabilistic process requires in some situations adapting the definition above appropriately; for example, M is the median if the probability that the random variable is greater than or equal to M is at least 1/2 and the probability that the random variable is less than or equal to M is also at least 1/2. The example below of the newsboy problem requires such an interpretation of the definition of the term *median*.

If the probability function p is symmetric about the mean, the mean and the median will be equal. The mean and the median can be different when the probability function p is skewed, or assymetric. In the newsboy example below the mean and median are different.

The variation, or spread, of the values generated by a probabilistic process can be defined in any of several different ways. The most common is the *variance,* the expected value of the square of the difference between the value and the mean:

$$\text{variance} = [+\ v : v \in S : (v - \text{mean})^2 * p(v)] \qquad [4.6.2\text{-}5]$$

Squaring the difference effectively treats positive and negative deviations from the mean equally. It also weights large deviations more heavily than smaller ones, which for some types of analyses may be considered to be inappropriate. Note that the dimension of the variance is the square of the dimension of the random variable, so the two are not directly comparable. To facilitate such a comparison, the square root of the variance is often taken; it is called the *standard deviation* or sometimes the *root mean square deviation*.

To avoid the variance's overemphasis on large deviations caused by squaring the deviation, the *mean absolute deviation* (abbreviated MAD) is sometimes used instead:

$$\text{mean absolute deviation} = [+\ v : v \in S : |v - \text{mean}| * p(v)] \qquad [4.6.2\text{-}6]$$

In theoretical analyses the variance and the functionally related standard deviation are by far the most commonly used of these measures of variation. The expressions are easier to work with algebraically than the expressions for the mean absolute deviation. In some practical applications the mean absolute deviation is used for the reason mentioned above.

Example The newsboy problem is a simple and classical example of optimizing a small business process. Early each day, the newsboy buys a number of newspapers, which he then hopes to sell during the morning to passersby on their way to work. Any newspapers left unsold are worthless and he scraps them. His profit is the amount of money he receives for the newspapers he sells during the morning less his cost of the newspapers he purchased at the beginning of the day. If he buys too many, the unsold newspapers represent a potentially avoidable loss, and if he buys too few, the potential profit on the newspapers he could not supply represents a lost opportunity. The newsboy wants to calculate how many newspapers to buy each day to maximize his profit.

The newsboy has collected data on the demand for newspapers each day for the last 60 days. He recorded not only the actual number of newspapers he sold each day, but also the additional number he could have sold had he had enough newspapers. He believes that the demand varies randomly from day to day with no identifiable pattern; that is, he believes that the data he has collected adequately represents the probability distribution for the demand on any day. The following table summarizes his data and shows on how many days he could have sold each particular number of newspapers. The final column shows the fraction of days on which each number of newspapers were or could have been sold (i.e., his best estimate of the probability of that number of newspapers being demanded on any day).

N	Number of Days on Which N Newspapers Were Demanded	Probability
5	1	1/60
6	5	5/60
7	15	15/60
8	10	10/60
9	8	8/60
10	6	6/60
11	3	3/60
12	4	4/60
13	5	5/60
14	2	2/60
15	1	1/60

Thus, the sample space S for this probabilistic process is the set of numbers in the column "N" in the table. The probability of each element in S is shown in the column "Probability."

The expected value or mean of the number of newspapers sold on any one day can be calculated by applying the definition above to the problem as stated above:

$$\text{mean} = \sum_{N=5}^{15} N * p(N) \qquad [4.6.2\text{-}7]$$

Inserting the numbers from the table gives

$$\text{mean} = 5*(1/60) + 6*(5/60) + \cdots + 14*(2/60) + 15*(1/60) \qquad [4.6.2\text{-}8]$$

and

$$\text{mean} = 541/60 = 9.0166666\ldots \qquad [4.6.2\text{-}9]$$

That is, on average, the newsboy will sell slightly fewer than 9.02 newspapers per day, provided that he bought enough to supply each day's demand.

The median (middle value of the distribution) can also be determined from the table above. The probability that N≥8 is more than half and the probability that N≤8 is also more than half; the "middle" value of N is one of the 10 occurrences of 8 as the value of N.

Another way to see that 8 is the "middle" value of N is to look at the 60 days of data the newsboy collected. From the table it is evident that the data (when sorted in ascending order and written with 15 days per line) was

5, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7, 7, 7, 7, 7,
7, 7, 7, 7, 7, 7, 8, 8, 8, 8, 8, 8, 8, 8, 8,
8, 9, 9, 9, 9, 9, 9, 9, 9, 10, 10, 10, 10, 10, 10,
11, 11, 11, 12, 12, 12, 12, 13, 13, 13, 13, 13, 14, 14, 15

The middle point in this sequence of numbers is between the next-to-last 8 and the last 8. The median is, therefore, 8.

The variance can be calculated by applying the definition of variance above to the data in the table and the mean as already calculated:

$$\text{variance} = \sum_{N=5}^{15} (N-\text{mean})^2 * p(N) \qquad [4.6.2\text{-}10]$$

Inserting the numbers from the table gives

$$\begin{aligned}\text{variance} = {} & (5-\text{mean})^2*(1/60) + (6-\text{mean})^2*(5/60) + \cdots \\ & + (14-\text{mean})^2*(2/60) + (15-\text{mean})^2*(1/60)\end{aligned} \qquad [4.6.2\text{-}11]$$

and

$$\text{variance} = 5.85 \qquad [4.6.2\text{-}12]$$

The standard deviation is the square root of 5.85, or 2.42. It indicates how large the variation from the mean typically is.

Similarly, the mean absolute deviation can be calculated from the numbers in the table by applying its definition:

$$\text{mean absolute deviation} = \sum_{N=5}^{15} |N-\text{mean}| * p(N) \qquad [4.6.2\text{-}13]$$

Performing the calculation in the same way as above gives a value of 1.99 for the mean absolute deviation. That is, the newsboy can expect his customers' daily demand to be slightly more than 9 on average, but variations of 2 from this average will be typical. Roughly speaking, daily demand varying from 7 to 11 will be normal and usual. Such variations will represent an entirely normal risk in his business.

The normal risk in this business is significant, so the newsboy, intending to become an astute businessman, will want to control this risk in a rational way and to prevent it from destroying the viability of his business—and wasting his time. The only decision that he can make is the quantity of newspapers to buy at the beginning of each day. If he buys Q newspapers and during the day experiences a demand N, he will sell the lesser of these two quantities [i.e., min(N, Q) newspapers]. If each newspaper he buys costs him C and if he sells each newspaper for the price Pe each, his profit for the day will be

$$\text{profit} = Pe*\min(N, Q) - C*Q \qquad [4.6.2\text{-}14]$$

His expected profit, or mean profit, will be the average of expression 4.6.2-14 weighted by the probabilities of the various values of N:

$$\text{expected profit} = \sum_{N=5}^{15} (Pe*\min(N, Q) - C*Q)*p(N) \qquad [4.6.2\text{-}15]$$

The newsboy will want to choose that value for Q which maximizes his expected profit.

If Pe=1 and C=0.3, calculating the profit expected for every value of the purchase quantity Q yields the following numerical results:

Purchase Quantity Q	Expected Profit
4	2.80
5	3.50
6	4.18
7	4.78
8	5.13
9	5.32
10	5.37
11	5.32
12	5.22
13	5.05
14	4.80
15	4.52
16	4.22

The newsboy should, therefore, buy 10 newspapers each day to maximize his expected (average) profit, which will then be 5.37 per day. He will then have on average 1.63 newspapers left over at the end of his sales day. How can one calculate the expected number of newspapers left over?

Note that buying fewer than 5 newspapers cannot be optimum, because at least 5 can always be sold. Similarly, buying more than 15 cannot be optimum, because no more than 15 will ever be sold.

4.6.3 Independent Probabilistic Processes

If two probabilistic processes are modeled with the two probability spaces (S1, A1, p1) and (S2, A2, p2) as described above, their combination can be modeled by the Cartesian product of the two probability spaces and a probability function in which the probability of every pair of results is the product of the individual results in their individual probability spaces. That is, the probability space (Sc, Ac, pc) for the combination of the two probability spaces above is

$$\mathrm{Sc} = \mathrm{S1} \times \mathrm{S2} \qquad [4.6.3\text{-}1]$$

$$\mathrm{Ac} = \mathrm{A1} \times \mathrm{A2} \qquad [4.6.3\text{-}2]$$

$$[\wedge\ X, Y : X \in \mathrm{A1} \wedge Y \in \mathrm{A2} : \mathrm{pc}(X, Y) = \mathrm{p1}(X) * \mathrm{p2}(Y)] \qquad [4.6.3\text{-}3]$$

provided that the two processes are *independent*. “Independent” means that the two processes do not influence each other, that their results are not influenced by something in common, and that knowing the result of one process does not affect the probabilities of the results of the other. An example is throwing two dice in such a way that neither affects the other, such as two different people rolling the two different dice, or one person rolling one die first and another later, or one person rolling both dice simultaneously but in such a way that neither die affects the other in such a way as to alter the expected relative frequencies (the probabilities) of the results.

Note that the probability of a pair of results from each of the two processes is defined to be the product of the two probabilities of the individual results. The notion of independence in the paragraph above is an argument for the plausibility of the definition, but not a logical justification in a rigorous sense.

Example Rolling Two Individually Identifiable Dice:

In this example, each die—the gray die and the white die—is rolled independently, that is, in such a way that the probability of any number appearing on one die is not affected by the number that appears on the other die.

The probabilistic model for each die is the same as that for the one die in the example in Section 4.6.1. The probabilistic model for the combination of rolling both the gray and white dice is formed by applying the formulas at the beginning of this section. The combined sample space Sc is, therefore,

$$\begin{aligned} Sc=\{&(1, 1), (1, 2), \ldots (1, 6), (2, 1), (2, 2), \ldots (2, 6), \ldots (6, 1),\\ &(6, 2), \ldots (6, 6)\} \end{aligned} \quad [4.6.3\text{-}4]$$

where the first number in each pair is the number appearing on one die (e.g., the gray die) and the second number in each pair is the number appearing on the other die (e.g., the white one). The probability of each combination is the same, 1/36 [i.e., pc(1, 1)=1/36, …, pc(6, 6)=1/36].

From these values one can calculate the probabilities of various events of interest. One such event is "snake eyes," the event that a 1 is rolled on each die. This is the event consisting of only one element in Sc, namely (1, 1), so its probability is 1/36.

Another event of interest is that the sum of the numbers on the two dice is 7. The subset of elements of Sc corresponding to this event is

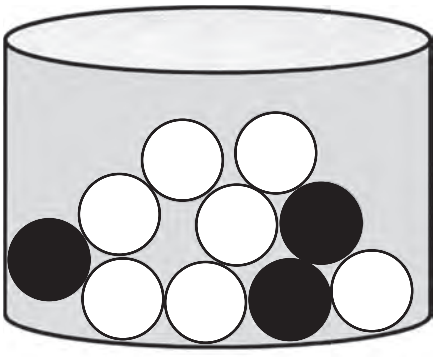
$$\{(1, 6), (2, 5), (3, 4), (4, 3), (5, 2), (6, 1)\}$$

and because this set contains six elements, each with a probability of 1/36, the probability of rolling a 7 is 6∗1/36, which is 1/6. This is the most likely sum that can appear on the two dice. Why? The reader should calculate the probability of rolling some other number, such as 1, 2, 4, 11, 12, or 15.

4.6.4 Dependent Probabilistic Processes and Conditional Probabilities

If two processes are not independent, they can be modeled by considering all individual possible results and their interaction explicitly and constructing the sample space S, the set of subsets A, and the probability function p accordingly. The following example illustrates this procedure.

Example Consider an urn containing b black balls and w white balls, where $b \geq 1$ and $w \geq 1$. The balls in the urn are thoroughly mixed and then a ball is drawn randomly from the urn (i.e., each ball is selected with the same probability). Afterward, a second ball is drawn randomly from the remaining balls. Among themselves, the black balls are indistinguishable, and similarly, the white balls are indistinguishable among themselves.



Urn with b=3 and w=7

The first ball drawn will be either black or white. The second ball will be either black or white. So the combined sample space contains only four elements: {(black, black), (black, white), (white, black), (white, white)}.

The various probabilities can be determined from the given numbers b and w. The probability that the first ball drawn is black is $b/(b+w)$ and the probability that the first ball drawn is white is $w/(b+w)$.

The probability that the second ball drawn is black depends on the number of black and white balls remaining after the first drawing, and those numbers depend on which color of ball was drawn the first time. This introduces dependence into the process.

If the first ball drawn was black, there are $b-1$ black balls and w white balls remaining in the urn. In this case, the *conditional probability* that the second ball drawn is black *given* that the first ball drawn was black is, therefore, $(b-1)/(b-1+w)$, and the conditional probability that the second ball drawn is white given that the first ball drawn was black is $w/(b-1+w)$.

If the first ball drawn was white, there are b black balls and $w-1$ white balls remaining in the urn. In this case, the conditional probability that the second ball drawn is black is b/(b+w−1), and the conditional probability that the second ball drawn is white is (w−1)/(b+w−1).

Note that if b=1 or w=1 (or both), certain of the probabilities above is 0, indicating that the event in question is impossible (e.g., if no white balls remain in the urn, drawing a white ball impossible and the corresponding probability is 0).

The *joint probability* that the first ball drawn is black and the second ball drawn is black is the product of the following two probabilities:

- The probability that the first ball drawn is black, which is (b/(b+w))
- The probability that the second ball drawn is black given that the first ball drawn was black, which is ((b−1)/(b−1+w))

That is, the joint probability that the first ball drawn is black and the second ball drawn is black is (b∗(b−1))/((b+w)∗(b−1+w)). The other joint probabilities are calculated in the same way.

The following table shows the various events and their probabilities.

TABLE 4.6.4-1 Probabilities for a Dependent Probabilistic Process

Color of First Ball Drawn	Probability of Color of First Ball Drawn	Color of Second Ball Drawn	Conditional Probability of Color of Second Ball Drawn Given the Color of the First Ball	Colors of the Two Balls Drawn in Order of Drawing	Joint Probability of the Colors of the Two Balls Drawn
black	b/(b+w)	black	(b−1)/(b−1+w)	(black, black)	(b∗(b−1))/((b+w)∗(b−1+w))
		white	w/(b−1+w)	(black, white)	(b∗w)/((b+w)∗(b−1+w))
white	w/(b+w)	black	b/(b+w−1)	(white, black)	(w∗b)/((b+w)∗(b−1+w))
		white	(w−1)/(b+w−1)	(white, white)	(w∗(w−1))/((b+w)∗(b−1+w))

For the numerical example b=3 and w=7 illustrated in the drawing of the urn above, Table 4.6.4-1 is as follows:

Color of First Ball Drawn	Probability of Color of First Ball Drawn	Color of Second Ball Drawn	Conditional Probability of Color of Second Ball Drawn Given the Color of the First Ball	Colors of the Two Balls Drawn in Order of Drawing	Joint Probability of the Colors of the Two Balls Drawn
black	3/10	black	2/9	(black, black)	6/90
		white	7/9	(black, white)	21/90
white	7/10	black	3/9	(white, black)	21/90
		white	6/9	(white, white)	42/90

The last two columns in each of the tables above show all the elements of the sample space S and the probabilities of each. From the entries in those columns one can calculate the probability of any possible outcome of the process of drawing two balls from the urn as described above. To do so, one must first express the outcome of interest in terms of the elements of the sample space and then add the corresponding probabilities.

For example, to calculate the probability of drawing at least one black ball, one expresses this outcome as the subset {(black, black), (black, white), (white, black)} of the sample space. The three probabilities are 6/90, 21/90, and 21/90, respectively, and their sum is 48/90, which equals 8/15. Similarly, the outcome that one draws one black ball and one white ball is the subset {(black, white), (white, black)} of the sample space, and its probability is 42/90, which equals 7/15.

A common notational form for the conditional probability of the event Bexp2 given the occurrence of the event Bexp1 is Pr{Bexp2|Bexp1}. In the example above, if c1 and c2 are random variables whose values are the colors of the first and second balls drawn, respectively, then the probability that the second ball drawn is black given that the color of the first ball drawn was white would be written Pr{c2=black|c1=white}.

More general conditional probabilities as well as notational forms such as Pr{x|y} (where x and y are random variables) are formally defined in probability theory. For example, the reversed conditional probability Pr{c1=white|c2=black} can also be defined and an expression for calculating it can be derived. (Pr{c1=white|c2=black} does not mean the same as Pr{c2=black|c1=white} and their numerical values are different.) These topics go beyond the scope of this book.

Among the many applications of conditional probabilities and related serial correlation are linguistic analyses. Letter frequencies and their sequential dependencies vary characteristically between languages and language groups. They can be helpful in identifying the language of an unknown text and in identifying relationships between different languages. They can also be used to correct or fill in gaps in damaged texts. Extreme examples of different conditional probabilities in letter sequences are the following: The letter sequence “cz” is much more common in Slavic languages than in English. If the first letters in an English word are “throu,” it is very likely that the next two letters are “gh.” If a short sample of text in a contemporary language contains the letter sequence “ough,” the sample is likely to be English, and most other European languages can be excluded as possibilities. The statistical nature of sequences of words can be used in similar ways.

4.7 THEOREMS

A theorem in mathematics is a statement that is true for all values of the variables appearing in the statement. In the mathematical literature a theorem is often written in a mixture of English and mathematical expressions, but every theorem can be formulated as a mathematical expression only. A theorem can be any statement (provided that it is always true), but it is usually in the form “if X then Y”; that is, “if

X is true, then Y is true” or, more precisely, “if the value of the expression X is true, then the value of the expression Y is (must be) true.” This can be written in the form of the mathematical expression

$$X \Rightarrow Y \qquad [4.7\text{-}1]$$

where X and Y are Boolean expressions. The expression X is called the *antecedent*, the *condition*, or the *hypothesis* of the theorem and the expression Y, the *consequent* or *thesis*. The infix symbol $\Rightarrow$ represents the logical *implication* function. The domain of this function is $\mathbb{B}\times\mathbb{B}$ and its range is $\mathbb{B}$. This function is defined in Table 3.3-1.

Any Boolean expression purported to be a theorem must be proved to be true for every combination of values of the variables appearing in the statement of the theorem. Proving theorems is the subject of Section 5.2.

Note that a theorem is, in the Language of Mathematics, just a Boolean expression: nothing more, nothing less. The Language of Mathematics itself provides no way of specifying a Boolean expression to be a theorem—this must be stated in English, typically in preceding explanatory text or simply by prefixing the mathematical expression by the label “Theorem:.” More generally, the interpretation of a mathematical expression—be it a theorem purported to be true, a statement that may or may not be true, an expression whose value is unknown or unimportant within some context—lies outside the scope of the Language of Mathematics. See Sections 6.9, 6.12, and 6.13 and Chapter 7.

4.8 SYMBOLS AND NOTATION

In all written language, symbols are used to represent both concrete and abstract things and concepts. The actual subjects of the writing never appear in the writing itself. Sequences of symbols are used to form names—representations—of various objects, such as variables, and to form other objects, such as expressions of several kinds. Representations in the form of tables, diagrams, drawings, pictures, and so on, are common in English texts and are also often used to supplement mathematical models (e.g., graphs, as described in Section 3.4.6).

The distinction between actual things and their representation is particularly great in the case of values in the Language of Mathematics. Values themselves can never be included in texts in the Language of Mathematics; values are always referred to with intermediate symbols, names, or representations of one type or another. One should always keep in mind, for example, that the sequence “241” of the symbols “2,” “4,” and “1” is *not* the numerical value, but only *represents* it. Many other sequences of symbols can represent this value, such as, CCXLI, 41, C1, F1, 361, 11110001, two hundred forty-one, zweihunderteinundvierzig, σμα, and ٢٤١. The single digit 2 is not the numerical value; it only represents or symbolizes it. The same distinction applies in English; the word “cat” only represents a cat. It also applies to names representing values. “George Smith” is not the person (value); it represents a person named George Smith.

One can say that nothing in any language is really defined precisely. The definition of every word relies on the definitions of other words, so there is no solid, independent foundation upon which to build. Instead, one relies on general consensus on the meanings of basic terms, symbols, and words and then builds on that consensus. Typically, that consensus is based on certain observable things, but many abstract terms cannot be based completely and only upon commonly agreed things observed in the physical environment.

In principle, the foundations of the Language of Mathematics are similarly problematic, but the Language of Mathematics is founded on fundamental concepts that have been kept as few in number and as simple as possible. The most important of these basics are values, the definition of which is left open (see Section 3.1), and certain symbols used to form names of variables and functions, representations of values, and expressions. The rules by which the symbols may be combined are defined precisely, and the rules by which expressions and their components may be manipulated are also defined precisely. Symbols and their manipulation play a greater and more central role in the Language of Mathematics than they do in English and other natural languages. The linguistic emphasis in the Language of Mathematics is on symbols and notation. Within the Language of Mathematics, their meaning plays a simple and very limited role. We assign meaning to mathematical objects through our interpretation of them, and this takes place outside the Language of Mathematics.

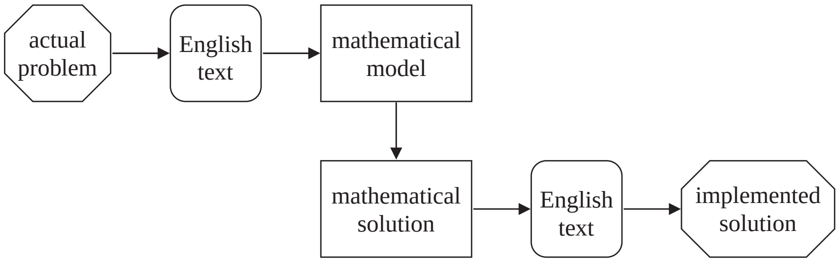
The very limited number of fundamental basics upon which the Language of Mathematics is built leads not only to the limitations of the Language of Mathematics, particularly its limited universe of discourse, but also to its precision and unambiguity.

In Section 1.1 the main purposes of languages were identified as communicating, thinking, and reasoning. The relatively precise definitions of the foundations of the Language of Mathematics compared to those of natural languages make the Language of Mathematics much more suitable for reasoning and analyzing. In fact, such reasoning and analyzing can be reduced to manipulating mathematical expressions according to precise, well-defined rules. Because the rules are so well and precisely defined, mathematical expressions can be transformed mechanistically. Subjectivity has been eliminated from the process and has been replaced by objectivity. Deciding which transformations to apply in any particular case is not so mechanistic, and typically requires human knowledge, insight, and experience, but whether or not the proposed transformations are permitted is clearly defined.

In other words, the strengths of the Language of Mathematics in the area of reasoning and analyzing are founded in the precise definitions of its notation system and the rules for manipulating symbols deriving from those definitions. Thus, symbols and notation play an even more important and critical role in the Language of Mathematics than in the natural languages.

5 Solving Problems Mathematically

The steps in the overall process of going from an actual problem to a mathematically founded solution are illustrated in the following diagram:



In this chapter only the mathematical aspects of this process, that is, transforming the mathematical model into a mathematical solution, are considered. This step is represented by the vertical arrow in the diagram. Translating the English text into a mathematical model is the subject of Chapter 7.

Finding a mathematical solution from the mathematical model involves transforming the expression constituting the mathematical model into a form that is either equivalent to or that follows logically from (is implied by) the mathematical model. This is done by modifying the mathematical model and subexpressions in it in a series of steps. Each step transforms a subexpression according to standard mathematical rules. The rules guarantee that either the value of the overall expression is preserved or that the initial expression logically implies the resulting expression.

Mathematically, the goal of this transformation is usually to find values for certain variables or definitions of certain functions. Ultimately, one strives for a final expression that contains subexpressions giving the desired values directly. In the process, it may be necessary to formulate and prove new transformation rules: that is, to state and to prove theorems.

The initial mathematical model can be thought of as a specification of the solution. The final, transformed expression contains the solution in the form of values for the previously unknown variables and functions.

In some cases the goal is to optimize some quantity: to find values of variables and functions so that some quantity is maximized or minimized. Parts of the mathematical model will restrain the solution found in some way.

More detailed descriptions of how the desired solution can be derived from the original mathematical model are the subjects of the following sections.

5.1 MANIPULATING EXPRESSIONS

As outlined above, the purposes of manipulating subexpressions in the mathematical model are (1) to transform one expression into another form with the same meaning (value) and (2) to draw logical conclusions relevant to the desired solution. In either case, the manipulation should be valid for all values of the variables appearing in the expression.

It should be emphasized that transforming mathematical expressions from one form to another does not, and cannot, generate new information. Manipulating an expression into an equivalent form maintains all the information in the original expression but expresses it in a different way. The different form is often intended to express information in a way that human readers can understand or interpret more easily. Manipulating an expression into a form that logically follows from the original expression can, and usually does, lose information. Again, the purpose of the manipulation is often to express the information in a form more useful to human readers. Manipulating a mathematical expression can help one to discover facts and information that were already there but not readily apparent. It can help to improve insight into a problem or a solution represented by the mathematical expression, but it cannot create new information or facts.

Every transformation of a mathematical expression is ultimately justified by one of only three principles:

- Equals may be substituted for equals.
- Applying the same function to equal arguments gives equal results.
- Particular properties of specific function(s) ensure that certain transformations are valid.

The first two principles are general and apply to all types of expressions. Because the third type of transformation depends on particular properties of parts of the expression in question, such transformations are applicable only in specific circumstances.

The first principle for manipulating mathematical expressions—equals may be substituted for equals—is simple, but the devil lies in the details: What is equal to what? Also, the opposite of the rule does not always apply: Even if two subexpressions are not equal for all values of the variables appearing therein, one may sometimes be substituted for the other without changing the value of the overall expression. Whether or not such a substitution is permitted depends on the context within which the substitution takes place. This is discussed further below, with examples.

The second principle—applying the same function to equal arguments gives equal results—is a direct consequence of the definition of a function (see Section 3.3). This principle can be expressed mathematically as

$$(a=b) \Rightarrow (f(a)=f(b)) \qquad [5.1\text{-}1]$$

where a and b are values, variables, or expressions and f is any function. This principle is the basis for many common transformations of expressions, including those often described as “canceling” and “moving a term” from one side of an equation to the other. Neither “canceling” nor “moving a term” is, itself, a valid transformation. Only when the desired effect can be achieved by the application of the second principle is the transformation justifiable.

The third principle gives rise to many specific transformation rules based on certain properties of the relevant functions. Such properties, in turn, follow from the definitions of the functions in question. Among these properties are the commutative, associative, and distributive properties, satisfied by functions such as addition, multiplication, logical and, and logical or. An example of another particular property of multiplication is that the product of two numbers is zero if and only if at least one of the numbers is zero. In mathematical notation this can be written as

$$(x*y=0) = (x=0 \vee y=0) \qquad [5.1\text{-}2]$$

This identity, or transformation rule, is useful, for example, in solving some types of equations. Evaluation of subexpressions containing only values is still another example of applying the third principle.

The following points are overriding guidelines to observe when transforming expressions or parts thereof.

- Watch both implied and explicit parentheses to be sure that you are really substituting what you think you are substituting. For example, x+y=y+x, but 2∗x+y does not always equal 2∗y+x, because x+y is not a subexpression of 2∗x+y. The subexpressions of 2∗x+y are 2∗x, y, 2, and x. 2∗(x+y) does equal 2∗(y+x), because x+y is a subexpression of 2∗(x+y). This problem can arise with any functions, not just addition and multiplication.
- By applying the same operation to both sides of an equation to obtain a second equation, the first equation implies the second. The reverse is not always true; that is, the two equations are not necessarily equal. They do not necessarily have the same value for all values of the variables appearing therein. For example, x=y implies that 0∗x=0∗y, but 0∗x=0∗y does not imply that x=y; that is, it is not generally true that (x=y)=(0∗x=0∗y). If both sides of an equation are multiplied by any number other than zero, the two equations are equivalent. If the multiplier can be zero, the two equations are not equivalent. This phenomenon arises with many functions, not just multiplication. With nonarithmetic operations, even more care is called for. Only if the operation is reversible will the two equations be equivalent in general.

- “Canceling” a term is often mentioned as a possible transformation, but it is *not* a legitimate operation. Only if the effect can be achieved in some other legitimate way is the result correct. For example, $a \wedge c = b \wedge c$ is not equivalent to $a=b$ and it does not imply that $a=b$. The equations $x+z=y+z$ and $x=y$ are equivalent because the first can be transformed to the second by adding $-z$ to each side, which is always allowed, and the second can be transformed to the first by adding z to each side, which is always allowed. Note that this transformation can be performed only if an inverse (here, $-z$) of the term to be canceled exists. There are no inverses of Boolean values with respect to either the logical and or the logical or, so Boolean values cannot be “canceled” from each side of an equation.
- “Moving” a term from one side of the equation to the other is also often mentioned as a convenient transformation, but it also is *not* a legitimate operation. With numerical equations the effect can be achieved by adding or subtracting the term to be moved to both sides of the equation. Again here, as with “canceling” above, an inverse must be added to both sides of an equation to effect the transfer of a term from one side of the equation to the other. When no inverse exists, the term cannot be “moved” from one side of the equation to the other. For example, $x+y=z$ is equivalent to $x=z-y$, but $a \wedge b = c$ cannot be transformed into any equation with a on the left side of the equation and b and c on the right side.
- If equals are added to equals, the original equations will imply the result, but not vice versa. Similarly, if equals are subtracted or multiplied or divided by equals, the original equations imply the result, but the result does not imply the original equations. See the example in expression 5.1-3.

Note especially the difference between equality and implication. If one needs to maintain equality of the original and resulting expressions but wants to use a transformation for which the first expression only implies the second, then one should include the original expression in the second expression. This has the effect of retaining all original information, permitting implication in both directions, thereby ensuring equality. For example,

$$(a=b \wedge c=d) \Rightarrow (a+c=b+d) \qquad [5.1\text{-}3]$$

but $(a=b \wedge c=d)$ is not in general equal to $(a+c=b+d)$. It is, however, always true that

$$(a=b \wedge c=d)=(a=b \wedge c=d \wedge a+c=b+d) \qquad [5.1\text{-}4]$$

This is an application of the transformation rule that if $x \Rightarrow y$, then $x = x \wedge y$. In English: If x implies y, then x and $x \wedge y$ always have the same value. This can be seen easily by considering the two cases x=false and x=true. If x is false, x and $x \wedge y$ are both false, regardless of the value of y. If x is true, so is y, so x and $x \wedge y$ again have the same value.

Another example is

$$(z=2+x \wedge x=4+y) \Rightarrow (z=6+y) \qquad [5.1\text{-}5]$$

in which equals have been substituted for equals (the subexpression 4+y has been substituted for x in the leftmost equality) to obtain the equality on the right side of the implication. This implication goes in one direction only; the two sides of the implication are not, in general, equal. From z=6+y one can conclude nothing about the value of x. But

$$(z=2+x \wedge x=4+y) = (z=2+x \wedge x=4+y \wedge z=6+y) \qquad [5.1\text{-}6]$$

Having noted the overall principles and potential pitfalls above, we now turn to a series of useful identities that are useful in transforming mathematical expressions. Each serves as a basic pattern. The variables in each pattern can be applied to values, variables, functions, or subexpressions in the expression that one wishes to transform.

The first group of identities deals with numbers and the operations of addition and multiplication. Readers will have learned them in school.

TABLE 5.1-1 Identities for Sums and Products

Sum	Product	Comments
$x+y = y+z$	$x*y = y*z$	+ and ∗ are commutative
$(x+y)+z = x+(y+z)$	$(x*y)*z = x*(y*z)$	+ and ∗ are associative
—	$x*(y+z) = (x*y)+(x*z)$	+ does **not** distribute over ∗ ∗ distributes over +

The commutative, associative, and distributive laws also apply to the logical functions and and or (∧ and ∨). In fact, each of these functions distributes over the other.

TABLE 5.1-2 Identities for the Logical Or and the Logical And

Logical Or	Logical And	Comments
$x \vee y = y \vee z$	$x \wedge y = y \wedge z$	∨ and ∧ are commutative
$(x \vee y) \vee z = x \vee (y \vee z)$	$(x \wedge y) \wedge z = x \wedge (y \wedge z)$	∨ and ∧ are associative
$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$	$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$	each distributes over the other

Because of the associative laws for the functions +, ∗, ∧ and ∨, parentheses can be dropped in expressions involving sequences of any of these functions; that is, because (x+y)+z = x+(y+z), either may be abbreviated as x+y+z. The two expressions with this abbreviation always have the same value, so this type of ambiguity is of no consequence. Many other functions are not associative, however, and expressions with parentheses involving them either may not be abbreviated or require additional conventions for dropping parentheses (see Section 3.4.2). Subtraction and division are commonly used functions that are not associative; for example, (a−b)−c and a−(b−c) mean different things and often have different values. Note that each of the identities above is valid for all values of the variables appearing therein. Additional identities often used in transforming Boolean expressions are given in Section 5.2.4.

The following examples show how an expression can be transformed using the principles and identities above. The three examples also illustrate extremes of manipulative steps: In Example 1, full detail is given, showing each individual justification. In Example 3, many individual steps that are obvious to someone with moderate experience are combined into a single step. Example 2 is intermediate between these two extremes.

Example 1 The following expression contains two equations involving the variables x and y. The goal of the transformations is to determine values of x and y that satisfy the expression (i.e., for which it is true). In this example, each and every transformation step is shown in full detail. In practice, several of the steps below would be combined into one step in several places in the transformations below, as in Examples 2 and 3.

$x\in\mathbb{R} \wedge x\in\mathbb{R} \wedge x+y=4 \wedge x+2*y=6$

= [insert equality derived from definition of multiplication by 2]

$x\in\mathbb{R} \wedge x\in\mathbb{R} \wedge x+y=4 \wedge x+2*y=6 \wedge 2*y=y+y$

= [substitute equals for equals (i.e., y+y for 2∗y in the middle equation)]

$x\in\mathbb{R} \wedge x\in\mathbb{R} \wedge x+y=4 \wedge x+2*y=6 \wedge x+y+y=6$

= [substitute equals for equals (i.e., 4 for x+y)]

$x\in\mathbb{R} \wedge x\in\mathbb{R} \wedge x+y=4 \wedge x+2*y=6 \wedge 4+y=6$

= [apply the function sum to the equal arguments (4+y, −4) and (6, −4)]

$x\in\mathbb{R} \wedge x\in\mathbb{R} \wedge x+y=4 \wedge x+2*y=6 \wedge (4+y)+(-4)=6+(-4)$

= [+ is commutative, so exchange terms on the left side of the last equation]

$x\in\mathbb{R} \wedge x\in\mathbb{R} \wedge x+y=4 \wedge x+2*y=6 \wedge (y+4)+(-4)=6+(-4)$

= [+ is associative; regroup parentheses on the left side of the last equation]

$x\in\mathbb{R} \wedge x\in\mathbb{R} \wedge x+y=4 \wedge x+2*y=6 \wedge y+(4+(-4))=6+(-4)$

= [substitute equals for equals: 4+(−4) is zero, property of inverse]

$x\in\mathbb{R} \wedge x\in\mathbb{R} \wedge x+y=4 \wedge x+2*y=6 \wedge y+0=6+(-4)$

= [substitute equals for equals (i.e., 2 for 6+(−4) in the last equation)]

$x\in\mathbb{R} \wedge x\in\mathbb{R} \wedge x+y=4 \wedge x+2*y=6 \wedge y+0=2$

= […+0 is …, property of +0]

$x\in\mathbb{R} \wedge x\in\mathbb{R} \wedge x+y=4 \wedge x+2*y=6 \wedge y=2$

= [substitute equals for equals (i.e., 2 for y in the first equation)]

$x\in\mathbb{R} \wedge x\in R \wedge x+y=4 \wedge x+2*y=6 \wedge y=2 \wedge x+2=4$

= [apply the function sum to the equal arguments (x+2, −2) and (4, −2)]

$x\in\mathbb{R} \wedge x\in\mathbb{R} \wedge x+y=4 \wedge x+2*y=6 \wedge y=2 \wedge (x+2)+(-2)=4+(-2)$

= [+ is associative; regroup parentheses on the left side of the last equation]

$x\in\mathbb{R} \wedge x\in\mathbb{R} \wedge x+y=4 \wedge x+2*y=6 \wedge y=2 \wedge (x + (2+(-2))=4+(-2)$

= [substitute equals for equals: 2+(−2) is zero, property of inverse]

$x\in\mathbb{R} \wedge x\in\mathbb{R} \wedge x+y=4 \wedge x+2*y=6 \wedge y=2 \wedge x+0=4+(-2)$

= […+0 is …, property of +0]

$x\in\mathbb{R} \wedge x\in\mathbb{R} \wedge x+y=4 \wedge x+2*y=6 \wedge y=2 \wedge x=4+(-2)$

= [substitute equals for equals (i.e., 2 for 4+(−2) in the last equation)]

$x\in\mathbb{R} \wedge x\in\mathbb{R} \wedge x+y=4 \wedge x+2*y=6 \wedge y=2 \wedge x=2$

The expression above contains the desired result: The only solution of the original expression above is the value 2 for x and the value 2 for y (i.e., only for these values of x and y is the original expression above true).

Example 2 Example 1 is repeated here, but the solution is found using different transformations. Also, several steps below are actually combinations of several more detailed steps.

$x\in\mathbb{R} \wedge x\in\mathbb{R} \wedge x+y=4 \wedge x+2*y=6$

= [apply the function sum to the equal arguments (x+y, −y) and (4, −y)]

$x\in\mathbb{R} \wedge x\in\mathbb{R} \wedge x=4-y \wedge x+2*y=6$

= [apply the function sum to the equal arguments (x+2∗y, −2∗y) and (6, −2∗y)]

$x\in\mathbb{R} \wedge x\in\mathbb{R} \wedge x=4-y \wedge x=6-2*y$

= [substitute equals for equals (i.e., 4−y for x in the last equation above)]

$x\in\mathbb{R} \wedge x\in\mathbb{R} \wedge x=4-y \wedge x=6-2*y \wedge 4-y=6-2*y$

= [apply the function sum to the equal arguments (4−y, 2∗y−4) and (6−2∗y, 2∗y−4)]

$x\in\mathbb{R} \wedge x\in\mathbb{R} \wedge x=4-y \wedge x=6-2*y \wedge y=2$

= [substitute equals for equals (i.e., 2 for y in the first equation above)]

$x\in\mathbb{R} \wedge x\in\mathbb{R} \wedge x=4-y \wedge x=6-2*y \wedge y=2 \wedge x=2$

Again, the expression above contains the desired result: The only solution of the original expression above is the value 2 for x and the value 2 for y.

Example 3 The examples above are repeated here as a moderately experienced person would combine steps.

$x{\in}\mathbb{R} \wedge x{\in}\mathbb{R} \wedge x+y=4 \wedge x+2*y=6$

$=$ [apply the function difference to the equal arguments (x+2∗y, x+y) and (6, 4)]

$x{\in}\mathbb{R} \wedge x{\in}\mathbb{R} \wedge x+y=4 \wedge x+2*y=6 \wedge y=2$

$=$ [subtract the last equation from the first equation]

or: [substitute equals for equals (i.e., 2 for y in the first equation above)]

$x{\in}\mathbb{R} \wedge x{\in}\mathbb{R} \wedge x+y=4 \wedge x+2*y=6 \wedge y=2 \wedge x=2$

Note that because each step in the transformations above is valid for all values of the variables in the expressions, the transformation consisting of the entire sequence of steps is also valid for all values of the variables.

These three examples illustrate that although a large number of individual justifications are logically required to manipulate the original expression into the solution, with some practice and experience one can group a fairly large number of individual steps into one and reduce very considerably the effort required to find the solution.

5.2 PROVING THEOREMS

As discussed in Section 4.7, a theorem is a Boolean mathematical expression that is purported to be true for all values of the variables appearing in the expression, and that has been or is to be proved. A theorem is often of the form X⇒Y, where X and Y are Boolean expressions. The expression X is called the *antecedent* (or hypothesis) and the expression Y is called the *consequent* (or thesis).

Unless stated explicitly otherwise, a theorem is assumed to apply for all values of the variables appearing in it. This is sometimes stated in the English formulation of a theorem but is usually not included in the mathematical expression of the theorem. One should be consciously aware that the expression is, at least implicitly, embedded in a corresponding quantified expression as defined and described in Section 3.4.8. Because a theorem is intended to apply for all values of the variables appearing therein, each step in the proof of a theorem must be valid for all values of the variables.

The following sections outline and describe various ways of proving theorems and convenient mathematical notation for recording and presenting proofs. Examples illustrate the application of the ideas. Frequently used equalities for transforming expressions are given.

5.2.1 Techniques and Guidelines for Proving Theorems

Frequently used techniques and guidelines for proving a theorem (an expression of the form $X \Rightarrow Y$) include the following:

- Transform the antecedent X of the theorem into the consequent Y (the classical approach). The transforms may show equivalence or that one expression implies the next.
- Show that the theorem $X \Rightarrow Y$ is equivalent to the logical constant true for all values of the variables appearing in the theorem. Transformations demonstrating equivalences or reverse implications may be used. However, transformations demonstrating forward equivalences may not be used, because anything implies true, so a transformation showing forward implication would prove nothing using this technique.
- Prove by contradiction, that is, prove that the theorem $X \Rightarrow Y$ cannot be false. This is usually done by assuming that the theorem is false (i.e., that X is true and Y is false) and showing that a known fact is contradicted. Then, the assumption $(X \wedge \neg Y)$ must be false [i.e., $\neg(X \wedge \neg Y)$ must be true]. This is equivalent to $\neg X \vee Y$, which, in turn, is equivalent to $X \Rightarrow Y$, thereby proving the theorem.
- Prove by induction. Prove the theorem for a base case. Then prove that if the theorem is true for one case, it is true for the next case. For example, prove that the theorem is true when $n=0$, and then prove that if it is true for n, it is also true for $n+1$. In this way, one proves that the theorem is true for $n=0$, for $n=1$, for $n=2$, and so on (i.e., for all nonnegative integer values of n).
- Construct a *truth table*, a table of all possible combinations of values for the variables appearing in the theorem and the corresponding values (false or true) of the theorem. If all values of the theorem are true, the theorem is proved; otherwise, counterexamples are identified. This approach is feasible for theorems referring to values in small sets (e.g., Boolean values) or (rarely) when the combinations of values can be grouped into a small number of categories.

With all of the techniques above, expressions are, where appropriate, transformed as described in Section 5.1 using additional identities in Section 5.2.4.

There is no standard, general way to find a sequence of steps to prove theorems. Finding a proof is, therefore, in general, a creative process. For most theorems arising in practice, this is not a problem standing in the way of applying mathematics. Many helpful approaches and guidelines are available. With increasing experience, proving theorems arising in practical applications of mathematics becomes ever easier.

Examples illustrating the use of the techniques above are included in Section 5.2.3.

5.2.2 Notation for Proofs

The following format is particularly convenient for documenting proofs:

```
       expression 1                    [comment about the expression to the left]
=           [justification for the step between the expressions above and below]
       expression 2                    [comment about the expression to the left]
=           [justification for the step between the expressions above and below]
       expression 3                    [comment about the expression to the left]
…                                                                           […]
       …                                                                    […]
```

The above is defined to mean (expression 1 = expression 2) and (expression 2 = expression 3), and so on.

Similarly,

```
       expression 1                    [comment about the expression to the left]
⇒           [justification for the step between the expressions above and below]
       expression 2                    [comment about the expression to the left]
⇒           [justification for the step between the expressions above and below]
       expression 3                    [comment about the expression to the left]
…                                                                           […]
       …                                                                    […]
```

is defined to mean (expression 1 ⇒ expression 2) and (expression 2 ⇒ expression 3), and so on.

The text or expressions in brackets at the right margin are comments intended for the human reader to facilitate understanding the proof. They are not part of the mathematical expressions.

Using the classical proof technique outlined in Section 5.2.1, the first expression in the sequence of expressions is the antecedent X. The final expression is Y. Each individual step may prove equality (=) or forward implication (⇒). No reverse implication (⇐) is permitted.

Using the second proof technique outlined in Section 5.2.1, the first expression in the sequence of expressions is the entire theorem (X⇒Y) itself. The final expression is the logical constant true. Each individual step may prove equality (=) or reverse implication ($\Leftarrow$). No forward implication ($\Rightarrow$) is permitted.

Using the technique proof by contradiction outlined in Section 5.2.1, the first expression in the sequence of expressions is the negation of the theorem $(X\wedge\neg Y)$ or, equivalently, $(\neg(X\Rightarrow Y))$. The final expression is the logical constant false or, alternatively, any expression known to be false for all values of the variables appearing therein. Each individual step may prove equality (=) or forward implication ($\Rightarrow$). No reverse implication ($\Leftarrow$) is permitted.

Using the technique proof by induction outlined in Section 5.2.1, the proof is divided into two parts. The first part proves that the theorem is true for some value of some variable, called the *variable of induction*. The second part proves that if the theorem is true for some value of the variable of induction, it is true for the next value of that variable. The first and final expressions in each part of the proof are as outlined above, depending on the technique used in each part of the proof. See the example of a proof by induction in the latter part of Section 5.2.3.

5.2.3 Lemmata and Examples of Proofs

A theorem used in the proof of another theorem is sometimes called a *lemma* (plural *lemmata*). In the literature on logic the first three lemmata below are often presented as basic rules of logic. They follow from the definitions of the Boolean functions $\wedge$, $\vee$, $\neg$, and $\Rightarrow$ given in Table 3.3-1. If X, Y, and Z are expressions with Boolean values (i.e., if $X\in\mathbb{B} \wedge Y\in\mathbb{B} \wedge Z\in\mathbb{B}$), the following lemmata are true. Note that the expression $X\in\mathbb{B} \wedge Y\in\mathbb{B} \wedge Z\in\mathbb{B}$ is an antecedent of each of the following lemmata.

$$[(X\Rightarrow Y) \wedge (Y\Rightarrow Z)] \Rightarrow (X\Rightarrow Z) \qquad \text{[Lemma 1]}$$
$$(X\Rightarrow Y) = (\neg Y\Rightarrow\neg X) \qquad \text{[Lemma 2]}$$
$$(X\Rightarrow Y) = \neg(X\wedge\neg Y) \qquad \text{[Lemma 3]}$$
$$(X\Rightarrow Y) = (\neg X\vee Y) \qquad \text{[Lemma 4]}$$

These lemmata are proved below. The proofs serve not only to allow the lemmata to be used in proofs of other theorems, but also to show how the above-mentioned principles and techniques can be used in proofs of theorems.

Lemma 1 is an important basis for essentially all proofs. By applying it iteratively, it enables a proof to be made in any number of steps in which each expression implies the next. The original expression is the antecedent X of the theorem to be proved, and the final statement is the consequent (here, Z).

Proof of Lemma 1 Lemma 1 can be proved by constructing a truth table as shown below. For every possible combination of values for X, Y, and Z, the table shows that the value of the expression $[(X\Rightarrow Y) \wedge (Y\Rightarrow Z)] \Rightarrow (X\Rightarrow Z)$ representing Lemma 1 is always true.

X	Y	Z	X⇒Y	Y⇒Z	(X⇒Y) ∧ (Y⇒Z)	X⇒Z	[(X⇒Y) ∧ (Y⇒Z)] ⇒ (X⇒Z)
false	false	false	true	true	true	true	true
false	false	true	true	true	true	true	true
false	true	false	true	false	false	true	true
false	true	true	true	true	true	true	true
true	false	false	false	true	false	false	true
true	false	true	false	true	false	true	true
true	true	false	true	false	false	false	true
true	true	true	true	true	true	true	true

Lemma 2 will be proved after Lemma 4 because Lemma 4 will be used in the proof of Lemma 2.

Lemma 3 underlies a proof by contradiction (also called *reductio ad absurdum*) of a theorem X⇒Y (see Section 5.2.1). Lemma 3 implies that a proof by contradition is a valid proof method. Lemma 3 itself states that (X⇒Y) = ¬(X∧¬Y).

Proof of Lemma 3 Lemma 3 can be proved by constructing the truth table below. For every possible combination of values for X and Y, the table shows that the value of the expression ¬(X∧¬Y) is the same as the value of the expression X⇒Y [i.e., that the value of the expression (X⇒Y) = ¬(X∧¬Y) representing Lemma 3 is always true].

X	Y	(X∧¬Y)	¬(X∧¬Y)	(X⇒Y)	(X⇒Y) = ¬(X∧¬Y)
false	false	false	true	true	true
false	true	false	true	true	true
true	false	true	false	false	true
true	true	false	true	true	true

Lemma 4 is similar to Lemma 3 and can be viewed as a minor modification of the rule for proving a theorem by contradiction. Lemma 4 can also be used to eliminate the function implies (⇒) from expressions to transform them into a more regular form involving only logical and, or, and negation functions.

One proof of Lemma 4 uses one of de Morgan’s rules for the negation of a logical ∧ or ∨ expression:

$$\neg(A \wedge B) = (\neg A \vee \neg B)$$

$$\neg(A \vee B) = (\neg A \wedge \neg B)$$

These identities can easily be proved: for example, with appropriate truth tables as shown above.

Proof of Lemma 4 This lemma, $(X \Rightarrow Y) = (\neg X \vee Y)$, can be proved by the following steps:

$$\begin{array}{ll} & X \Rightarrow Y \\ = & \hfill [\text{Lemma 3}] \\ & \neg(X \wedge \neg Y) \\ = & \hfill [\text{de Morgan's rule}] \\ & \neg X \vee Y \end{array}$$

Lemma 2 is a reversing rule for logical implications. Many people tend at first to assume implicitly that if $X \Rightarrow Y$, it is also true that $Y \Rightarrow X$. This is not, in general, true. It is true, however, that if $X \Rightarrow Y$, then $\neg Y \Rightarrow \neg X$, and vice versa. That is, $X \Rightarrow Y$ and $\neg Y \Rightarrow \neg X$ are equal; if either is true, the other is also true. In some cases it is easier to prove that $\neg Y \Rightarrow \neg X$ than $X \Rightarrow Y$, and Lemma 2 permits one to prove that $\neg Y \Rightarrow \neg X$ when a proof of $X \Rightarrow Y$ is needed.

Proof of Lemma 2 We prove Lemma 2 by reducing the expression representing it to the logical constant true. The expression for Lemma 2 is

$$\begin{array}{lll} & (X \Rightarrow Y) = (\neg Y \Rightarrow \neg X) & [\text{Lemma 2}] \\ = & & [\text{Lemma 4 applied to left side of equation}] \\ & (\neg X \vee Y) = (\neg Y \Rightarrow \neg X) & \\ = & & [\text{Lemma 4 applied to right side of equation}] \\ & (\neg X \vee Y) = (Y \vee \neg X) & \\ = & & [\vee \text{ is commutative}] \\ & (\neg X \vee Y) = (\neg X \vee Y) & \\ = & & [\text{a value is equal to itself, definition of function equals}] \\ & \text{true} & \end{array}$$

Example: Proof by Induction This example deals with a calculation often attributed to Johann Carl Friedrich Gauss (1777–1855), a famous German mathematician. As the story goes, when Gauss was nine years old, his teacher instructed the class to add the numbers from 1 to 100 inclusive. Gauss finished the calculation in an unbelievably short time by, in effect, applying the formula in this theorem. Gauss was not, however, the first person aware of this simple method for calculating such a series. Such a calculation is the subject of an Egyptian papyrus in Demotic script dated to the first millenium B.C.E., which is now in the British Museum (BM 10520). The subject of this papyrus is the sum of the first 10 integers, not the first 100, but the formula effectively behind it is the same.

The theorem to be proved here states that the sum of the first n natural numbers is given by the formula $n*(n+1)/2$. This theorem can be written as the expression

$$n \in \mathbb{Z} \wedge n > 0 \wedge [+\ i : i \in \mathbb{Z} \wedge 1 \leqslant i \leqslant n : i] = n*(n+1)/2$$

Dividing this theorem into two parts, the first for the base case and the second for the inductive step, gives two subtheorems to be proved. The first part is the base case,

$$1\in\mathbb{Z} \wedge 1>0 \wedge [+\ i : i\in\mathbb{Z} \wedge 1\leqslant i\leqslant 1 : i]=1*(1+1)/2 \qquad \text{[base case, n=1]}$$

and the second part is the inductive step,

$$n\in\mathbb{Z} \wedge n>0 \wedge [+\ i : i\in\mathbb{Z} \wedge 1\leqslant i\leqslant n : i] = n*(n+1)/2$$

[If the theorem is true for n,]

$$\Rightarrow n+1\in\mathbb{Z} \wedge n+1>0 \wedge [+\ i : i\in\mathbb{Z} \wedge 1\leqslant i\leqslant n+1 : i] = (n+1)*(n+2)/2$$

[then it is true for n+1.]

Proof, Part 1, Base Case, n = 1 When n=1, the statement of the theorem is

$$1\in\mathbb{Z} \wedge 1>0 \wedge [+\ i : i\in\mathbb{Z} \wedge 1\leqslant i\leqslant 1 : i]=1*(1+1)/2$$

which is

$$\text{true} \wedge \text{true} \wedge 1=1$$

which is clearly true.

Proof, Part 2, Inductive Step We begin with the antecedent of part 2 of the theorem and transform it into the consequent:

$$n\in\mathbb{Z} \wedge n>0 \wedge [+\ i : i\in\mathbb{Z} \wedge 1\leqslant i\leqslant n : i] = n*(n+1)/2$$

[If the theorem is true for n,]

$$\Rightarrow \qquad [n\in\mathbb{Z} \Rightarrow n+1\in\mathbb{Z},\ n>0 \Rightarrow n+1>0]$$

$$n+1\in\mathbb{Z} \wedge n+1>0 \wedge [+\ i : i\in\mathbb{Z} \wedge 1\leqslant i\leqslant n : i] =n*(n+1)/2$$

$$= \qquad \text{[adding n+1 to each side of the equation]}$$

$$n+1\in\mathbb{Z} \wedge n+1>0 \wedge [+\ i : i\in\mathbb{Z} \wedge 1\leqslant i\leqslant n : i]+(n+1) = n*(n+1)/2+(n+1)$$

$$= \qquad \text{[simplifying both sides of the equation]}$$

$$n+1\in\mathbb{Z} \wedge n+1>0 \wedge [+\ i : i\in\mathbb{Z} \wedge 1\leqslant i\leqslant n+1 : i]=(n+1)*(n+2)/2$$

[then it is true for n+1.]

The last expression above is the consequent of the inductive step, so the proof is complete.

Example: Proof by Contradiction The theorem to be proved here states that the square root of 2 is irrational. [Briefly, a rational number is a number that can be expressed as the ratio of two integers. Any number that is not rational (i.e., that cannot be expressed as the ratio of two integers) is irrational. See Appendix C for a further explanation of rational and irrational numbers.]

Proof (English) Assume that the square root of 2 is rational. Then, by the definition of a rational number, there exist two integers a and b such that $\sqrt{2}$=a/b. Then 2=a^2/b^2 and $2*b^2=a^2$. The number of occurrences of 2 as a prime factor of a^2 must be even. Similarly, the number of occurrences of 2 as a prime factor of b^2 must be even and, therefore, the number of occurrences of 2 as a prime factor of $2*b^2$ must be odd. Because $2*b^2=a^2$, and because the prime factorization of any number is unique, the number of occurrences of 2 as a prime factor of a^2 and $2*b^2$ must be the same. However, as argued earlier, one must be even and the other must be odd. Since no number is both odd and even, there is a contradiction. Therefore, the square root of 2 cannot be rational, so it must be irrational.

Proof (Language of Mathematics) Again, assume the negation of the theorem to be proved, that is, that the square root of 2 is rational.

$\sqrt{2}\in\mathbb{Q}$ [5.2.3-1, $\sqrt{2}$ is rational]

= [definition of rational]

$[\vee\ a, b : a\in\mathbb{Z} \wedge b\in\mathbb{N}_1 : a\in\mathbb{Z} \wedge b\in\mathbb{N}_1 \wedge \sqrt{2}=a/b]$

[5.2.3-2, $a\in\mathbb{Z} \wedge b\in\mathbb{N}_1$ repeated for convenience below]

To determine the value of the quantified expression above, the template term will be evaluated first. In the expressions below, NOcc(p, N) is the number of occurrences of p as a prime factor of N, where p is a prime number and N is an integer. (This is a function because the prime factorization of any integer is unique.) IsOdd(N) and IsEven(N) are Boolean functions that have the value true when and only when N is odd or even, respectively.

$a\in\mathbb{Z} \wedge b\in\mathbb{N}_1 \wedge \sqrt{2}=a/b$ [5.2.3-3, template term of 5.2.3-2]

= [square both sides of the equation above]

$a\in\mathbb{Z} \wedge b\in\mathbb{N}_1 \wedge \sqrt{2}=a/b \wedge 2=a^2/b^2$

= [multiply both sides of the last equation above by b^2]

$a\in\mathbb{Z} \wedge b\in\mathbb{N}_1 \wedge \sqrt{2}=a/b \wedge 2*b^2=a^2$

= [$\wedge$ true terms to expression above]

$a\in\mathbb{Z} \wedge b\in\mathbb{N}_1 \wedge \sqrt{2}=a/b \wedge 2*b^2=a^2 \wedge \text{IsOdd(NOcc}(2, 2*b^2))$
$\wedge\ \text{IsEven(NOcc}(2, a^2))$

= [substitute equals for equals (i.e., a^2 for $2*b^2$)]

$a\in\mathbb{Z} \wedge b\in\mathbb{N}_1 \wedge \sqrt{2}=a/b \wedge 2*b^2=a^2 \wedge \text{IsOdd(NOcc}(2, a^2))$
$\wedge\ \text{IsEven(NOcc}(2, a^2))$

= [a number (e.g., NOcc(2, a^2)) cannot be both odd and even.]

false [5.2.3-4]

Expressions 5.2.3-3 through 5.2.3-4 show that the template term of 5.2.3-2 is false. Returning to 5.2.3-1 and 5.2.3-2, we have

$\sqrt{2}\in\mathbb{Q}$

= [5.2.3-1 repeated]

$[\vee\ a, b : a\in\mathbb{Z} \wedge b\in\mathbb{N}_1 : a\in\mathbb{Z} \wedge b\in\mathbb{N}_1 \wedge \sqrt{2}=a/b]$ [5.2.3-2 repeated]

= [5.2.3-3 through 5.2.3-4, substitute false for template term]

$[\vee\ a, b : a\in\mathbb{Z} \wedge b\in\mathbb{N}_1 : \text{false}]$

=

false

Thus, the square root of 2 is not rational. It must, therefore, be irrational, by the definition of irrational.

Steps 5.2.3-3 through 5.2.3-4 are, in effect, the proof of a lemma used in the proof that the square root of 2 is irrational.

The structure of the proof above applies not just to 2, but to any prime number. It can be generalized easily to other numbers as well. To what type of numbers can it be generalized?

5.2.4 Additional Useful Identities

In addition to the equalities presented in the paragraphs on the commutative, associative, and distributive laws in Section 5.1, the following identities are also often used as lemmata in proofs of theorems. Some have been used in the proofs in Section 5.2.3. As an exercise the reader should prove that each of the following expressions are true for all $X, Y, Z \in \mathbb{B}$, where X, Y, and Z can be values, variables, expressions, or functions. Your proofs should assume only the definitions of the Boolean functions in Table 3.3-1 and, where appropriate, already proved identities. Use the proof table approach in some proofs and the algebraic approach for others. Apply both approaches to some identities to be proved and notice their relative advantages and disadvantages.

$(X \wedge \neg X)=\text{false}$ [$\wedge$ with complement is false, 5.2.4-1]

$(X \wedge \text{false})=\text{false}$ [$(X \wedge)$ can be dropped in this context, 5.2.4-2]

$(X \wedge X)=X$ [$(X \wedge)$ can be dropped in this context, 5.2.4-3]

$(X \wedge \text{true})=X$ [$(\wedge$ true) can be dropped in this context, 5.2.4-4]

$(\neg(\neg X))=X$ [double negation can be dropped, 5.2.4-5]

$(X \vee \text{false})=X$ [($\vee$ false) can be dropped in this context, 5.2.4-6]

$(X \vee X)=X$ [(X $\vee$) can be dropped in this context, 5.2.4-7]

$(X \vee \text{true})=\text{true}$ [(X $\vee$) can be dropped in this context, 5.2.4-8]

$(X \vee \neg X)=\text{true}$ [$\vee$ with complement is true, 5.2.4-9]

$(X \vee (X \wedge Y))=X$ [($\vee$ (X $\wedge$ Y)) can be dropped in this context, 5.2.4-10]

$(X \vee (\neg X \wedge Y))=(X \vee Y)$ [($\neg$ X) can be dropped in this context, 5.2.4-11]

$(\neg(X \wedge Y)) = ((\neg X) \vee (\neg Y))$ [de Morgan's rule, 5.2.4-12]

$(\neg(X \vee Y)) = ((\neg X) \wedge (\neg Y))$ [de Morgan's rule, 5.2.4-13]

$\text{false} \Rightarrow X$ [false implies anything, 5.2.4-14]

$X \Rightarrow \text{true}$ [anything implies true, 5.2.4-15]

$(X \Rightarrow Y) = ((\neg X) \vee Y)$

[useful for eliminating $\Rightarrow$ from an expression, 5.2.4-16]

$((X \Rightarrow Y) \wedge (\neg X \Rightarrow Z)) = ((X \wedge Y) \vee (\neg X \wedge Z))$

[useful for eliminating $\Rightarrow$ from an expression, 5.2.4-17]

$(X \Rightarrow Y) = ((\neg X) \Leftarrow (\neg Y))$

[reverse direction of implication, negate both terms, 5.2.4-18]

$(X \Rightarrow Y) \Rightarrow (X=(X \wedge Y))$

[($\wedge$ weaker term) can be dropped or inserted, 5.2.4-19]

$(X \Rightarrow Y) \Rightarrow (Y=(Y \vee X))$

[($\vee$ stronger term) can be dropped or inserted, 5.2.4-20]

$X \Rightarrow (X \vee Y)$ [any Boolean expression implies a weaker one, 5.2.4-21]

$(X \wedge Y) \Rightarrow X$ [any Boolean expression is implied by a stronger one, 5.2.4-22]

$(X=Y)=((X \wedge Y) \vee (\neg X \wedge \neg Y))$

[useful for eliminating = within a subexpression, 5.2.4-23]

$(X \Rightarrow (Y=Z)) = ((X \wedge Y) = (X \wedge Z))$

[condition for substituting Z for Y when not equal, 5.2.4-24]

Note that the terms "X $\wedge$" may not be dropped in the last identity above. They are needed when X is false and Y and Z are not equal.

After proving each of the identities above, the reader should try to prove that

$$(X \Rightarrow (Y=Z)) \Rightarrow (Y=Z)$$

Note how the failure of the attempted proof shows the need for the "X $\wedge$" terms in this identity.

5.3 SOLVING EQUATIONS AND OTHER BOOLEAN EXPRESSIONS

In technical applications of mathematics one often wants to "solve the equations" for the values of certain variables. This is an instance of the more general task of finding the solution of a Boolean expression, of which an equation is one type. The solution of a Boolean expression is defined to mean those combinations of values of the variables appearing in the Boolean expression for which the value of the expression is true.

A solution depends only on the expression, and not on the interpretation of the expression in the application domain. Consequently, solutions can be derived without regard to the interpretation. Thus, solving a Boolean expression is an exercise that takes place strictly and solely in the mathematical domain, completely outside the application domain. This characteristic leads to several advantages: Solutions can be derived without regard to unnecessary factors and details and without regard to possible ambiguities in the application domain. Solutions can be derived by mathematical specialists without specific knowledge and experience in the application domain. The solutions found apply to any interpretation (i.e., to any application domain to which the mathematical model may apply), so the solutions are often more generally applicable than a solution derived specifically for the application problem would be. This, in turn, means that fewer solution methods are needed than would otherwise be the case.

Not only *can* a solution be derived without regard to the interpretation of the expression in the application domain, it *should* be so derived. If additional information from the application domain appears to be helpful or required, the mathematical model—and the English statement of the requirements from which it was translated—are probably incomplete. Both should be completed or corrected before proceeding to derive the solution desired.

In some cases, the definitions of functions referred to in the Boolean expression are to be determined as part of the solution. The functions in question can usually be handled in the same way as array variables (see Section 4.1.4).

The solution of a Boolean expression is found by manipulating the Boolean expression in suitable ways as outlined in Section 5.1. The Boolean expression or parts thereof can be transformed in two ways: One type of transformation preserves the value of the expression for all values of the variables appearing therein. The second type of transformation results in a new expression that the original expression implies. The most useful transformations to apply depend on the form and structure of the original expression, but the general goal is to derive an expression containing terms of the form

$$\text{variable name} = \text{value} \qquad [5.3\text{-}1]$$

for the names of all variables whose values are to be determined.

Example A rectangle with area 600 m^2 is to be enclosed with fencing material with total length 100 m. What are the length x and width y, both in meters, of the rectangle?

Implicit in this statement is the following information from elementary geometry: The area 600 m^2 of the rectangle is the product of the length x and width y. The perimeter 100 m of the rectangle is twice the sum of x and y.

Combining the given statement and the implicit information above, the mathematical model is:

$$x\in\mathbb{R} \wedge y\in\mathbb{R} \wedge x*y=600 \wedge 2*(x+y)=100 \qquad [5.3\text{-}2]$$

From this expression, values for x and y in terms of the given area A and the given length of the fencing material L can be derived as shown below.

$$x\in\mathbb{R} \wedge y\in\mathbb{R} \wedge x*y=600 \wedge 2*(x+y)=100 \qquad [5.3\text{-}3]$$

$=$

$$x\in\mathbb{R} \wedge y\in\mathbb{R} \wedge x*y=600 \wedge x+y=50 \qquad [5.3\text{-}4]$$

$=$

$$x\in\mathbb{R} \wedge y\in\mathbb{R} \wedge x*y=600 \wedge x+y=50 \wedge y=50-x \qquad [5.3\text{-}5]$$

$=$

$$x\in\mathbb{R} \wedge y\in\mathbb{R} \wedge x*y=600 \wedge x+y=50 \wedge y=50-x \wedge x*(50-x)=600 \qquad [5.3\text{-}6]$$

$=$

$$x\in\mathbb{R} \wedge y\in\mathbb{R} \wedge x*y=600 \wedge x+y=50 \wedge y=50-x \wedge x*(50-x)=600 \wedge x*50-x^2=600 \qquad [5.3\text{-}7]$$

$=$

$$x\in\mathbb{R} \wedge y\in\mathbb{R} \wedge x*y=600 \wedge x+y=50 \wedge y=50-x \wedge x*(50-x)=600 \wedge 0=x^2-50*x+600 \qquad [5.3\text{-}8]$$

$=$

$$x\in\mathbb{R} \wedge y\in\mathbb{R} \wedge x*y=600 \wedge x+y=50 \wedge y=50-x \wedge x*(50-x)=600 \wedge 0=(x-20)*(x-30) \qquad [5.3\text{-}9]$$

$=$

$$x\in\mathbb{R} \wedge y\in\mathbb{R} \wedge x*y=600 \wedge x+y=50 \wedge y=50-x \wedge x*(50-x)=600 \wedge (x=20 \vee x=30) \qquad [5.3\text{-}10]$$

$\Rightarrow$

$$y=50-x \wedge (x=20 \vee x=30) \qquad [5.3\text{-}11]$$

$=$

$$(x=20 \wedge y=50-x) \vee (x=30 \wedge y=50-x) \qquad [5.3\text{-}12]$$

$=$

$$(x=20 \wedge y=30) \vee (x=30 \wedge y=20) \qquad [5.3\text{-}13]$$

The last line above is the solution: Either x is 20 and y is 30 or x is 30 and y is 20.

The reader should insert the justification for each transformation step in the derivation of the solution above.

In many of the lines in the derivation above, two or more terms were equal to one another and the repetitions could have been eliminated. They were repeated in order that each step, each manipulation of a subexpression, can be clearly seen. In practice, one would write many of the lines above more concisely, and two or more steps would be taken in one larger step. Although such shortcuts are legitimate, they do place more of a burden on less experienced readers. When developing such solutions in practice, the repetition and small steps help to reduce the chances of making mistakes and provide better opportunities to check intermediate results. When documenting derivations of solutions in practice, the writer should keep the experience and manipulative skills of the intended readers in mind. Too much detail and repetition bore and distract the experienced reader, while too many shortcuts strain the inexperienced reader. A good approach to verifying a derivation of a solution is first to read the derivation, noting mentally the basic approach and techniques used, and then, without referring to the existing derivation, to derive the solution anew.

Several examples in Chapters 2 and 8, especially Sections 2.2, 2.3, and 2.8 and Sections 8.6 and 8.13, contain derivations of solutions of Boolean expressions. Sections 8.6.7 and 8.13.10 especially deal with deriving functions as the solutions.

Note that a Boolean expression may have no solution, any number of solutions, or infinitely many solutions. For example, the following Boolean expressions have no solution, that is, there are no values for the variables x and y for which these expressions have the value true.

$$x\in\mathbb{R} \wedge x^2=-1$$

$$x\in\mathbb{R} \wedge x^2<0$$

$$x\in\mathbb{R} \wedge y\in\mathbb{R} \wedge x>0 \wedge y>0 \wedge x+y=6 \wedge 2*x+y=5$$

The example in Section 8.9, the classical paradox of the Barber of Seville, also has no solution. The example in Section 8.6 has exactly one solution. The following Boolean expressions have two, three, and infinitely many solutions, respectively:

$$x\in\mathbb{R} \wedge x^2=4$$

$$x\in\mathbb{R} \wedge x^3-x=0$$

$$x\in\mathbb{R} \wedge y\in\mathbb{R} \wedge x>0 \wedge y>0 \wedge y>x$$

Always keep in mind that a given Boolean expression may have no solution, one solution, any finite number of solutions, or infinitely many solutions. Never assume exactly one solution. Don’t forget the possibility that there is no solution.

5.4 SOLVING OPTIMIZATION PROBLEMS

The goal of an optimization problem is to maximize or minimize some quantity subject to given restraints. If the quantity to be maximized or minimized is a continuous and differentiable function, it will take on its maximum or minimum value when its derivative (see Section 4.5) with respect to the appropriate independent variable is zero. This is most easily seen by considering a graph of the function and observing that at a maximum or minimum point, the derivative is zero.

Example Here a problem similar to that in Section 5.3 will be considered as an example. What are the length x and width y, each in meters, of the rectangle with the largest area A (m^2) that can be enclosed using L meters of fencing material? Making use of the implicit information from basic geometry in Section 5.3, the mathematical model can be written

$$x \in \mathbb{R} \wedge y \in \mathbb{R} \wedge x*y=A \wedge 2*(x+y)=L \qquad [5.4\text{-}1]$$

This expression can be solved for A in terms of x alone as follows:

$$x \in \mathbb{R} \wedge y \in \mathbb{R} \wedge x*y=A \wedge 2*(x+y)=L \qquad [5.4\text{-}1 \text{ repeated}]$$

$=$

$$x \in \mathbb{R} \wedge y \in \mathbb{R} \wedge x*y=A \wedge x+y=L/2 \qquad [5.4\text{-}2]$$

$=$

$$x \in \mathbb{R} \wedge y \in \mathbb{R} \wedge x*y=A \wedge y=L/2-x \qquad [5.4\text{-}3]$$

$=$

$$x \in \mathbb{R} \wedge y \in \mathbb{R} \wedge y=L/2-x \wedge A=x*(L/2-x) \qquad [5.4\text{-}4]$$

$=$

$$x \in \mathbb{R} \wedge y \in \mathbb{R} \wedge y=L/2-x \wedge A=x*L/2-x^2 \qquad [5.4\text{-}5]$$

$\Rightarrow$

$$A=x*L/2-x^2 \qquad [5.4\text{-}6]$$

$\Rightarrow$

$$\frac{dA}{dx}=L/2-2*x \qquad [5.4\text{-}7]$$

For a given L, A will be at a maximum only if the derivative of A with respect to x is equal to zero; that is, A will be at its maximum when $x=x_0$, where

$$0=L/2-2*x_0 \qquad [5.4\text{-}8]$$

which is equivalent to

$$x_0=L/4 \qquad [5.4\text{-}9]$$

in which case the maximum value of A is Amax, where

$$Amax=x_0*(L/2-x_0) \qquad [5.4\text{-}10]$$

which is equivalent to

$$Amax=(L/4)*(L/2-L/4) \qquad [5.4\text{-}11]$$

$=$

$$Amax=(L/4)^2 \qquad [5.4\text{-}12]$$

If $L=100$, as in the example in Section 5.3, the maximum enclosable rectangular area is 625 m^2. In this case, $x=y=25$ and the rectangle is a square, as one might have suspected on grounds of symmetry.

Finally, one must verify that a maximum, and not a minimum, has been found, because the condition for both is the same—the derivative being equal to zero. This can easily be seen from a graph of A as a function of x. Alternatively, take the second derivative (the derivative of the derivative) and evaluate it at the point in question; in this example it is negative, so a maximum has been found. (If it were positive, a minimum would have been found.) Still another alternative is to notice that A decreases when x deviates both above and below L/4 by a small amount, so a maximum has been found.

In the example above, equating the appropriate derivative to zero gave the condition for optimality. More generally, the optimality requirement can be expressed in the mathematical model by an appropriate term, including a quantified expression whose value is the optimum. In the case of the example above, this additional term could have been

$$A = [\max x : x\in\mathbb{R} : x*L/2-x^2] \qquad [5.4\text{-}13]$$

in which case the entire mathematical model would have been

$$x\in\mathbb{R} \land y\in\mathbb{R} \land y=L/2-x \land A=x*L/2-x^2$$
$$\land\ A = [\max x : x\in\mathbb{R} : x*L/2-x^2] \qquad [5.4\text{-}14]$$

or any one of several other equivalent expressions. Note that the x within the quantified expression above is not the same as the x outside the quantified expression (see Section 3.4.8). To avoid any possible confusion arising from the double use of the variable name x in expression 5.4-14, it could be rewritten using a new name (e.g., z) for the quantified variable:

$$\begin{aligned} &x{\in}\mathbb{R} \wedge y{\in}\mathbb{R} \wedge y{=}L/2{-}x \wedge A{=}x{*}L/2{-}x^2 \\ &\qquad \wedge A = [\max z : z{\in}\mathbb{R} : z{*}L/2{-}z^2] \end{aligned} \qquad [5.4\text{-}15]$$

In the theoretical mathematical literature, the "max" function above is often referred to as *supremum* or *least upper bound*, abbreviated "sup" or "l.u.b.," respectively (see Section 4.4).

PART C
English, the Language of Mathematics, and Translating Between Them

6 Linguistic Characteristics of English and the Language of Mathematics

Languages differ from one another in several ways. The most important way in which English and the Language of Mathematics differ is probably their universes of discourse, that is, *what* they enable their users to write about and express. The Language of Mathematics deals with few and purely abstract things, beginning with values and variables (see Sections 3.1 and 3.2), whereas English deals with essentially all aspects of the world in which their speakers live and, therefore, has a much richer vocabulary. Languages differ also in *how* statements intended to convey information can be composed and structured. Also in this regard, English and the Language of Mathematics differ from one another fundamentally. The Language of Mathematics provides only precisely defined mathematical expressions—combinations of values, variables, and structures built upon them—as media for communication, whereas English provides a large number of less precise words, sentences, paragraphs, and so on, intended not only to communicate specific information, but also to enable the author to stimulate the readers' thinking and to encourage readers to add their own interpretations, in the context of their previous experience.

This chapter deals with these contrasting differences in some detail and examines the consequences of these differences for translating between English and the Language of Mathematics. A few similarities will provide the bridges needed to connect the two different media for communication. Conscious awareness of the differences will enable the translator to avoid the potential pitfalls arising from the considerable differences between English and the Language of Mathematics.

6.1 UNIVERSE OF DISCOURSE

All those things that one talks or writes about in a language make up the *universe of discourse* of that language. The universe of discourse of a natural language such as English consists of the various aspects of the real, physical world in which its speakers live and the abstract concepts they think and communicate about. Subgroups of speakers with specialized interests based, for example, on vocations or hobbies will often have specialized terms ("jargon") for things of particular interest (i.e., will have specialized universes of discourse), extending the common, shared universe of discourse. Commonly used terms in such fields will not be understood by people not working in that field and having no need for those specialized terms.

A native speaker of Arabic told me that Arabic poetry is characterized by rhythms reflecting the different rhythms of a camel traveling at different speeds over sand, a common feature of the physical world in which many Arabic-speaking societies live. Arabic has a richer vocabulary for describing fine distinctions between different aspects of sand and sand dunes than do languages whose speakers live in areas with no sand deserts. Inhabitants of cold regions need to be able to describe different kinds of snow and ice conveniently, and their languages or dialects will cater to this need with specific words or standardized phrases. Languages spoken by people living in areas where it never snows do not have the vocabulary to enable one to make such distinctions so easily; their language may even lack a specific word for snow itself. Different groups living or working in different environments will need and develop correspondingly different universes of discourse, with special terms or standard phrases for important aspects of their own particular environments.

The universe of discourse of the Language of Mathematics is quite restricted and very different from the universes of discourse of English and other natural languages. Most of the basic terms of importance in the Language of Mathematics were introduced in earlier chapters. The universe of discourse of the Language of Mathematics in the pure, narrow sense does not include any of the aspects of the real, physical world in which people live. It includes only those abstract concepts mathematicians think and communicate about when doing mathematics. The universe of discourse of the Language of Mathematics is very limited, consisting only of values, variables, functions, expressions, and the various types of structures built up of these components. The universe of discourse of the Language of Mathematics does not include any elements of the areas to which mathematics can be applied. This has implications for interpreting mathematical models in terms of the application areas (e.g., things in the real, physical world). This is discussed in greater detail in the sections below and in Chapter 7.

The universes of discourse of natural languages overlap extensively. Text in one language whose meaning is also in the universe of discourse of another language can be translated into that other language using only more or less standard correspondences between the elements of the two languages in question. That is, the only rules and guidelines needed for translating the text from one language into the other are general ones, ones that are valid for all texts and not specific to the particular text in question itself. Although the rules and guidelines will not depend on the text, the application of those rules and guidelines to the text will, of course, depend on certain aspects of the text. For example, standard grammatical rules must be applied and generally accepted styles of expression observed. For words with different meanings or interpretations in the source language, one must select the one intended (which usually depends on the context) and express it in the target language. When an idiom is to be translated, a corresponding and appropriate expression must be identified in the target language—sometimes idioms translate directly, but often they do not.

The existence of "rules and guidelines" independent of the text to be translated does not mean that such translation is straightforward and mechanistic. It means only that a new set of rules and guidelines will not be required for every new text to be translated.

For example, consider the English sentence "Eleanor chose a girl for her team" and its translation into Esperanto: "Eleanor selektis knabinon por sia teamo." The correspondence between these two sentences follows from rules and guidelines that are not specific to this text but that are applied to particular aspects, such as tense of the verb and cases of nouns.

English Word	Esperanto Word	Rules and Guidelines
Eleanor	Eleanor (or Eleanoro)	Proper nouns are the same (or the noun ending -o can be added).
chose	selektis	Dictionary, past tense conjugation ending -is
a	—	Esperanto has no indefinite article.
girl	knabinon	Dictionary, feminine ending -in, noun ending -o, singular objective case ending -n
for	por	Dictionary, preposition
her	sia	Dictionary, reflexive, possessive, adjective ending -a
team	teamo	Dictionary, singular subjective case ending -o only

Several different word orders are possible for the Esperanto sentence, for example:

Knabinon Eleanor selektis por sia teamo.
Knabinon selektis Eleanor por sia teamo.
Por sia teamo Eleanor selektis knabinon.

In English, not all of these word orders would be correct translations of the Esperanto sentence. In particular, if "girl" precedes "Eleanor" in the English sentence, the subject and the object become reversed, with a correspondingly different meaning of the sentence: "A girl chose Eleanor for her team" has a very different meaning than "Eleanor chose a girl for her team." The reversal in the Esperanto sentence is possible because the case ending -n on "knabinon" identifies it as the object, no matter whether it appears before or after the subject.

Note that all of the rules cited in the table above are general rules and guidelines; they are not specific to this text.

If a text in one language (e.g., English) has a meaning not expressible in a second language (e.g., the Language of Mathematics) (i.e., a meaning not in the common universe of discourse), translating the text into the second language will not be so straightforward—if it is meaningfully possible at all. In place of the missing standard correspondences between elements of the two languages, a correspondence specific to the text (and its context) must be defined. Such a correspondence defines how the elements of the text in the first language are to be related to elements in the second language. In this book such a correspondence is called an *interpretation* of the mathematical model. The interpretation itself must be written in the language with the more extensive, more encompassing universe of discourse. An interpretation can be thought of as relating the universes of discourse of the two languages with one another.

When a text in the Language of Mathematics is to be understood in the context of a real-world application, it is necessary to provide an interpretation for the mathematical text in question, and that interpretation will be specific to the text and to the context and application in which it is to be interpreted. Another mathematical text—or another context or application—will require another interpretation.

A text in the Language of Mathematics can always be translated into English using only a standard interpretation independent of the mathematical text, but such a translation does not convey information of additional usefulness or interest. Such a translation will not relate to the application area. For example, an expression such as “$a=2*(b+c)$” can be translated directly, without a text-specific interpretation, into “the value of the variable a is twice the sum of the values of the variables b and c.” If a connection to some application is intended, the meanings of the values of the variables a, b, and c must be defined in terms meaningful in the application world. That definition will constitute the interpretation of the mathematical text in question. Such an interpretation might be, for example, “the values of b and c are the lengths in meters of adjacent sides of a rectangular piece of land and the value of a is the length in meters of a fence enclosing that piece of land.” This interpretation is not unique; many other interpretations of the expression “$a=2*(b+c)$” are possible, each within another context or application.

To the extent that things in the universe of discourse of English can be modeled with elements of the universe of discourse of the Language of Mathematics (i.e., in terms of values, variables, functions, expressions, and the various types of structures built up of these), one can use the Language of Mathematics to describe things otherwise described in English. In the Language of Mathematics one cannot say or write anything that cannot be expressed in English, but one can express things unambiguously and usually more concisely in the Language of Mathematics. Most importantly, mathematical expressions can be used as a basis for logically reasoning about things and in ways for which English language texts are inadequate.

6.2 LINGUISTIC ELEMENTS IN THE LANGUAGE OF MATHEMATICS AND IN ENGLISH

The fundamental elements in the Language of Mathematics are:

- Values (see Section 3.1)
- Variables (see Section 3.2)
- Functions (see Section 3.3)

- Expressions (see Section 3.4)
- Structures of the elements above (see Section 4.1)

Letters and words are the building blocks of written English. The types of words and sequences of words that will be of concern to us are:

- Nouns [names of people, things, concepts, etc. (e.g., house, George, length)]
- Pronouns [shortened references to names (e.g., I, you, he, she, it, they)]
- Articles (e.g., the, a, and an)
- Adjectives [words qualifying nouns (e.g., large, long, green, beautiful, dry, hot)]
- Verbs [words indicating actions or state or being (e.g., goes, wrote, drove, is, were)]
- Adverbs [words qualifying verbs (e.g., slowly, illegibly, always, erratically)]
- Past participles and present participles [words derived from verbs, used to form perfect tenses and also used as adjectives (e.g., gone, coming, lifted, lifting, closed)]
- Prepositions [words relating or connecting other words (e.g., of, in, to, at)]
- Phrases [certain combinations of the above (e.g., the green house, the color of Anne's dress)]
- Predicate [a verb phrase (e.g., a verb together with its adverbs and object, predicate adjective, or predicate noun)]
- Clauses [combinations of the above including a conjugated verb (e.g., George hit the ball, the length of the box is greater than its height)]
- Conjunctions [words combining words or phrases of the same grammatical type or clauses (e.g., and, or, when, if … then, but)]
- Sentences (clauses or combinations of several clauses (e.g., George hit the ball and it flew over the fence)]

Some of the categories above are subdivided further in the study of grammar (e.g., possessive adjectives, relative pronouns, demonstrative pronouns), but these subdivisions will not be important for our purposes of translating between English and the Language of Mathematics. For translating between English and other natural languages, however, such subdivisions are often very important in order to construct grammatically correct sentences in the target language.

The elements of English above are used to form the subject, the verb or predicate, and the object of a clause whose verb is an action verb. They are also used to form the subject, the verb or predicate, and the predicate adjective or the predicate noun of a clause whose verb is a verb of state or being. For example, in the clause "George hit the ball," "George" is the subject, "hit" is the verb of action, and "ball" is the object. In the clause "Anne's dress is green," "dress" is the subject, "is" is the verb of state, and "green" is the predicate adjective. In the clause "Susan is an editor," "Susan" is the subject, "is" is the verb of state, and "editor" is the predicate noun.

Note that the basic linguistic elements of the Language of Mathematics differ completely and fundamentally from the basic linguistic elements of English. Establishing in detail an appropriate correspondence between the linguistic elements of English in a given text and the linguistic elements of the Language of Mathematics in a mathematical model is the task of the translator, but many useful general guidelines can be deduced from the characteristics of the different types of linguistic elements in each language. These general guidelines are developed in the rest of this chapter and in Chapter 7.

All of the linguistic elements of English listed above are common to many natural languages, and most of those elements are common to all natural languages. The main exception is probably the article, which some languages do not have. The ways in which these elements are formed and combined in the various natural languages differ significantly from language to language, however.

6.2.1 Verbs, Clauses, and Phrases

A sentence is a clause or a combination of clauses joined by conjunctions. In English and many other languages, each clause must contain a conjugated verb. In some languages, however, verbless clauses and sentences are permitted in some cases. Arabic, Hebrew, and Russian are among the languages of this type in current use, and Egyptian was an example of an ancient such language.

For translating between a natural language and the Language of Mathematics, the important distinction between a clause and a phrase is that a clause is a statement that is either true or false, and whose question form is answered by yes or no. A phrase is not simply true or false, and its question form (if there is any) cannot be answered by yes or no. For example, the sentence “John’s house is brown” is a statement that is either true or false, so it is a clause (and a sentence). A verbless version of this English sentence, which corresponds to a valid sentence in some languages, would be “John’s house brown.” Because English does not permit verbless clauses, the presence of a conjugated verb is used in this book as the distinguishing characteristic of a clause. If the reader also wishes to translate between a natural language permitting verbless clauses and the Language of Mathematics, the presence or absence of a conjugated verb will not be a satisfactory distinguishing characteristic of a clause and relevant passages in this book must be adjusted accordingly.

In English, a clause contains a finite (finished, complete, conjugated) verb. An independent clause can stand on its own as a grammatically correct and complete sentence, as can a dependent clause without the conjunction joining it with another clause. A phrase does not satisfy these conditions; it is not a grammatically correct and complete sentence. In the Language of Mathematics, expressions corresponding to phrases and clauses can be similarly distinguished from one another, but the criterion is different, because in the Language of Mathematics verbs are not explicitly present. A Boolean expression corresponds to a clause, and a non-Boolean expression corresponds to a phrase.

Examples of phrases are “the color of Anne’s dress,” “the number of elevators in the Eiffel Tower,” “Sam’s profession,” “the state of the door,” and “the altitude of the rocket in meters at time t seconds after ignition.” None of these are statements that are true or false; the values of these phrases might be green, 7, lawyer, opening, and $2*t^2$, respectively.

Boolean variables and functions and expressions with a Boolean value and one or more non-Boolean arguments correspond to clauses in English because both are either true or false. Therefore, such a variable, function, or expression effectively contains a verb. For example, the expression “$x>y$” corresponds to the English clause “the value of x is greater than the value of y,” which contains the verb “is.” A variable, function, or expression whose value is not Boolean does not correspond linguistically to a clause in English.

Functions with a Boolean value and only Boolean arguments correspond to conjunctions in English and, hence, do not themselves contain implicit verbs, but their subsidiary Boolean components do. (The implication function $\Rightarrow$ can be viewed as an exception; see below.) For example, in the expression “$(x>y) \wedge (z=2)$,” the function $\wedge$ corresponds to the English conjunction “and,” combining two independent clauses to form the sentence “The value of x is greater than the value of y and the value of z is equal to 2.” Each of the relational functions $>$ and $=$ implicitly contains the verb “is,” but the function $\wedge$ does not.

Thus, a mathematical expression with a Boolean value (and non-Boolean subexpressions) corresponds to either a dependent clause or an independent clause, depending on whether and how it is combined with another expression. An independent clause may be a sentence or may be combined with one or more other clauses by appropriate conjunctions to form a sentence. English sentences in a sequence are, logically, connected with “and,” so the mathematical expressions corresponding to the individual English sentences in a sequence are, therefore, connected with the logical “and,” usually written $\wedge$.

One English translation of the expression “$(x>y) \Rightarrow (z=2)$” is the sentence “If the value of x is greater than the value of y, then the value of z is equal to 2.” Instead of “if” one could write “when,” “whenever,” and so on. The first part (“If the value of x is greater than the value of y”) is a dependent clause. The second part (“the value of z is equal to 2”) is the independent clause. The conjunction “if … then” combines the dependent clause and the independent clause to form the complete sentence.

Some English translations of expressions involving the implication function $\Rightarrow$ include the verb “implies,” as in the translation “X implies Y” or “The truth of X implies the truth of Y” for the mathematical expression “$X \Rightarrow Y$.”

Verbs of state or being not referring to any particular time (e.g., present, past, future) are often called *stative verbs* in grammatical terminology. All verbs in the Language of Mathematics are stative verbs. This and other aspects of verbs are discussed in more detail in Sections 6.2.6 and 6.6. In a verbless clause (see the beginning of this section), there is no verb expressing tense, so the corresponding missing verb can be considered to be tensless, and hence stative.

6.2.2 Nouns and Pronouns

A mathematical expression with a non-Boolean value corresponds to a phrase in English. The phrase is often a noun phrase. Sometimes it is only a single noun.

In mathematical expressions, the names of variables and functions serve as relative pronouns, referring to the values, mathematical definitions (the set of values), and/or interpretations of these names. The definitions referred to may, in principle, be anywhere in the expressions, but good style indicates that each definition should precede all references to it (i.e., before all other appearances of the name of the variable or function in question). Using names to refer to the corresponding definitions and interpretations in this way enables mathematical expressions to be more concise and clearer than English text with the same meaning. English has very few relative pronouns, and references to a distant target tend to be unclear, so that a longer phrase reiterating the definition must be repeated, often many times, in an equivalent English text.

Nouns and pronouns in English text can refer to things or properties. Things can normally be represented in mathematical models by values. Some properties can be represented by values, such as the colors red and blue. Other properties mentioned in English text are not normally represented by values in mathematical models, such as "infinity." When translating English text into a mathematical model written in the Language of Mathematics, one must be careful to distinguish between those nouns and pronouns associated with values in the mathematical model and those not associated with such values. The two categories of nouns and pronouns will be handled differently and translated differently, if translated at all.

6.2.3 Adjectives, Adverbs, and Prepositional Phrases

The two main parts of any English sentence are the subject (a noun/pronoun phrase) and the predicate (a verb phrase). Subcomponents of these two parts (e.g., adjectives, adverbs, and prepositional phrases) appearing in English text related to expressions in a mathematical model are normally considered together with the noun, pronoun, or verb they modify or complement. How adjectives, adverbs, and prepositional phrases are handled when translating between English and the Language of Mathematics and when naming variables is discussed in Section 6.2.6.

6.2.4 Conjunctions

A conjunction in English connects two or more like things: nouns and pronouns, adjectives, verbs, clauses, and sentences. The combination retains the grammatical character of the two like things. Common conjunctions in English are "and" and "or." These conjunctions can be used with any part of speech.

In the Language of Mathematics the situation is quite different. In mathematics, a function with two or more arguments is used to connect or combine two or more things. In contrast to conjunctions in English:

- Different functions are used to connect different types of arguments.
- A function can connect or combine unlike (different) types of arguments.

For example, the mathematical functions "and" or "or" do not connect two or more numbers; instead, arithmetic functions such as $+$, $-$, $*$, and / connect numbers. When and where useful, a function can be defined to connect Boolean variables and numbers or any other combination of different types of arguments.

Therefore, a particular conjunction in English does not always translate to a single conjunction in the Language of Mathematics. A particular conjunction in English will translate into different mathematical functions depending on what the conjunction in English is combining. The English conjunction "and" applied to two clauses will translate into the Boolean function "and," but when the English conjunction "and" is applied to other parts of speech, it will not translate into any Boolean function. Its translation will depend on other parts of the English sentence in question.

For example, the "and" in "the sum of the length and the width of a rectangle …" can be translated into the infix function $+$ in an expression or implicitly into the connection of length and width in the sequence of arguments of the function sum(length, width). In other cases, the conjunction can lead to a function (e.g., union, intersection, sequence).

When a conjunction in English connects or combines two things other than clauses, the translator should first identify what mathematical object should represent the combination. The most common mathematical structures containing two or more values are the set and the sequence. These are, therefore, primary candidates to represent a combination of values. The phrase "John, George, and Miriam" could, therefore, be translated into the Language of Mathematics as $\{$John, George, Miriam$\}$ or as [John, George, Miriam]. The sentence "John, George, and Miriam are physicians" could then be translated as "$\{$John, George, Miriam$\} \subset \mathbb{Physicians}$."

Alternatively, the translator can reformulate a clause in which a conjunction appears into separate clauses connected by one or more conjunctions (e.g., rewrite "John and George are physicians" as "John is a physician and George is a physician"). This sentence, consisting of the conjunction of two clauses, can then be translated as "John$\in\mathbb{Physicians} \wedge$ George$\in\mathbb{Physicians}$."

In other cases, other key words, such as "is" or "are" (see Sections 7.2 and 7.3) or words and phrases suggesting a quantified expression (e.g., all, for all, any, for any, each, every, there exists, there is, there are, for some, at least one), will indicate how to translate an English text containing a conjunction into an appropriate mathematical expression.

Logically, the English conjunction "but" means the same as "and." "But" is used to draw attention to some sort of a contrast between the two things it joins. Sometimes, it points out that the things it joins would often seem at first to contradict each other, although they do not. This extra connotation of "but" over "and" is rarely, if ever, translated into a mathematical expression. When translating text containing the word "but," it should normally be treated in the same way as "and."

6.2.5 Negation

In the Language of Mathematics the term *negation* has two quite different meanings: numerical negation and logical negation. *Numerical negation* is, in effect, multiplication by -1. *Logical negation* reverses the values true and false. Because these two negations are mathematically different, two different prefix symbols are used: $-$ for numerical negation and $\neg$ for logical negation. When translating, one must be careful to distinguish between these two meanings.

The English negation “not” is an adverb that corresponds to the logical negation in the Language of Mathematics. The numerical negation in mathematics is usually better translated by the English word “minus.” The word “negative,” as in “negative 2,” is also sometimes used.

A translator should pay careful attention to what is being negated. Note that “negative 2” means “minus 2,” not “not 2.” A phrase such as “not 2” will always appear in a clause (e.g., “x is not 2”), and the “not” will refer to the verb, not to the number “2.” (The clause can be implicit; see Section 6.4 for an example.) The positive formulation of the clause will correspond to a Boolean expression (e.g., $x=2$) in the Language of Mathematics, and the logical function “not” will be applied to that Boolean expression: for example, $\neg(x=2)$, which can also be written $x\neq 2$. The symbol $\neq$ can be translated as “is not equal to.”

In English, negative forms of questions and even statements are sometimes used when the positive forms are actually meant. When translating from English to the Language of Mathematics, one must be careful in such cases to translate what is meant, not what is written. For example, one sometimes asks “Isn’t that a nice painting?” when the question could be formulated positively “Is that a nice painting?” with exactly the same intended meaning. Questions of the form “don’t you…” often fall into this category, such as the question “Don’t you think that her dress is elegant?” which usually suggests—and even emphasizes—a positive meaning (i.e., that her dress is elegant). Such negatively phrased questions are sometimes not even meant as questions but instead as statements of the questioner’s positive opinion.

Conversely, positive statements in English sometimes have a negative meaning. For example, the sentence “That’s a fine kettle of fish” usually means just the opposite: that one is confronted with a miserable mess, not with something fine and nice.

Different languages have different conventions for answering a negatively phrased question. If the painting is, in fact, nice, the correct answer in English to the question “Isn’t that a nice painting?” is “yes” (that is a nice painting). In Chinese and other Asian languages, the correct answer is “no” (the negatively phrased corresponding statement “That is not a nice painting” is not true). The English answer refers always to the positively phrased statement corresponding to the question. The Chinese answer refers to the statement derived by rephrasing the question, including the “not” if present. This aspect of different languages can and does cause confusion and misunderstanding—and laughs—when native speakers of the two types of languages converse with each other. For correct communication between native speakers of such different languages, questions should always be phrased positively, even when this is not normal usage. An old joke – and a line in a song – is derived from this difference between languages: To the question “Don’t you have any bananas today?” the immigrant grocer replies: “Yes, we have no bananas today.”

When translating English text into a mathematical model, be careful to understand what is meant, not just what is written, particularly when negatives appear in the text. Especially if people from different language backgrounds are working together in a team must one pay close attention to the intended meaning of negatively formulated parts of text.

6.2.6 Parts of Speech and Naming Conventions for Functions and Variables

In general, the name of a variable or a function should be selected to reflect both:

- The part of speech of the key word in its definition or interpretation and
- The type of value it can take on

Selecting names of variables and functions according to the guidelines below will make the mathematical model easier to read, to understand, and to work with. The guidelines below will help those who formulate the mathematical model to avoid structural and logical problems, oversights, inconsistencies, and errors.

Each combination of the parts of speech and the types of values of variables and functions are discussed below. Combinations that should generally not occur and should be avoided are identified. Naming conventions and suggestions for the other combinations are given. Any exception to the guidelines given below should be carefully thought out and justified.

The following parts of speech of the key word(s) in the definition or interpretation of a variable or function are discussed below:

- Noun or pronoun
- Verb of state or being (stative verb)
- Verb of action
- Adjective (including predicate adjective)
- Adverb
- Prepositional or other phrase

For each of these parts of speech, the following types of values taken on by the variable or function are discussed:

- Boolean
- Numerical
- Other

The following table summarizes the recommended naming conventions for variables and functions, depending on the type of their value.

TABLE 6.2.6-1 Naming Conventions for Variables and Functions

		Type of Value of the Variable or Function	
		Boolean	Other (e.g., Numerical)
part of speech of name of variable or function	noun or pronoun	only together with a stative verb	recommended
	stative verb	recommended	avoid
	action verb	avoid	avoid
	adjective	only together with a stative verb	only as part of a noun or pronoun phrase
	adverb	only together with a stative verb	only as part of an adjective phrase in a noun or pronoun phrase
	prepositional or other phrase	only together with a stative verb	only as part of a noun or pronoun phrase

If the proposed

- name of a variable or function
- definition or description in its interpretation
- type of the variable or function

are inconsistent with the guidelines outlined in the table and discussed in more detail below, one or more of the three items above should be revised accordingly. It may be necessary to reformulate the English text in the interpretation of the variable or function, especially when translating English text to the Language of Mathematics and in order to eliminate action verbs. Only very rarely and only after careful and extensive consideration and justification should exceptions to these guidelines be allowed.

In the foregoing classification of verbs as stative or action, one type of verb has been left out: verbs of state or being but with a reference to time (e.g., was, had been). Like action verbs, state verbs cannot be translated directly into the Language of Mathematics because there is no concept of time in the Language of Mathematics. Any reference to time must be embedded in the interpretation of a variable or function or must be brought into the mathematical model by a variable whose value represents time.

Before examining the relationships above in detail, it is useful to consider the various main parts of the structure of English sentences and the types of values in the Language of Mathematics to which they correspond.

There are two types of sentences, a statement and a question. Each consists of one or more clauses. Every clause contains a subject, a verb, and often some combination of an object, adjectives, adverbs, prepositional phrases, and so on. A clause is also either a statement or a question, but phrases within a clause are not.

A *statement* in English can be thought of as being a proposition that is either true or false (i.e., as having a Boolean value). A phrase or word within a statement is typically associated with some type of value other than Boolean, common examples being numbers, things, properties, and characteristics.

A *question* can be thought of as having a value corresponding to the answer to the question. If the question is a sentence in the form of a question (e.g., "Is Rome the capital of Italy?"), the answer is yes or no, corresponding to the Boolean value of the statement. If the question is formulated with an interrogative word, the type of value of the answer corresponds to an interrogative word:

- Who?—a person
- What?—a thing
- When?—a date or time (e.g., expressed by numbers)
- Where?—a place
- Why?—a reason
- How?—a procedure or process
- How many?—a number
- How much?—a quantity expressed as a number and a unit of the quantity in question

When formulating a mathematical model in the Language of Mathematics from an English text, statements and their component phrases typically play a more important role than do questions. Similarly, statements occur more often in interpretations of mathematical values, variables, and functions than do questions.

Keep the comments above in mind when reading the following paragraphs about the correspondence between parts of speech and naming conventions for variables and functions in a mathematical model.

Noun or Pronoun In English texts associated with mathematical models, many nouns refer to things represented or measured by numbers. Commonly appearing nouns include "number," "quantity," and "amount." Other nouns associated with aspects of physical objects or systems measured with one or more numbers are "distance," "length," "height," "width," "weight," "mass," "position," "velocity," "rate," "force," "pressure," "stress," "strain," "voltage," "current," "field strength," "gravitational attraction," and "angle." This illustrates that the meaning and interpretation of numerical variables in a mathematical model are best described by nouns or noun phrases.

Nonnumeric characteristics, properties, and categories can be handled in a similar way. Nouns and noun phrases such as "color," "shape," "direction of movement," "structure of bridge," "material," "poetic meter," "name," and "address" describe variables or functions whose values can be nouns, adjectives, adverbs, or corresponding phrases. For example, a variable described by the noun phrase "structure of bridge" might take on such values as "suspension," "cantilever," "cable-stayed," or "truss." Another variable described by the noun phrase "poetic meter" might take on such values as "iambic pentameter" or "anapestic trimeter." A variable or function named "shape" might take on such values as "rectangular," "circular," "oval," or "triangular."

This leaves the question: Can nouns, pronouns, or phrases based on them describe a variable or function that takes on Boolean values? A description that is either true or false must relate at least two things with each other, but a noun or pronoun expresses only one thing (or a collection of things, but not the relationship between them). This, in turn, raises the question: What kinds of words can connect two or more nouns or pronouns such that the combination takes on Boolean values?

Two or more nouns or pronouns can be related with each other by a verb or verb phrase, leading to a clause or a sentence in English. Thus, a distinguishing characteristic of a variable or function with Boolean values is a verb in its English description. See the section "Verb of State or Being" below.

As pointed out in Section 6.2.4, a conjunction in English also connects two or more like things, such as nouns and pronouns, but the combination retains the character of the two like things. Thus, when a conjunction connects two or more nouns or pronouns, the combination is a noun or pronoun phrase, not a clause with a Boolean value. Only if the nouns or pronouns being connected already had Boolean values would the combination have a Boolean value. Except for nouns or pronouns referring to a clause or a sentence, such nouns or pronouns are neither evident nor apparent. In exceptional cases where nouns or pronouns refer to clauses or sentences, the clauses or sentences referred to should be the description of the variable or function in question. See also the section "Conjunction" below.

In summary, a variable or function with a non-Boolean value should be described by a noun, pronoun, or phrase based on a noun or pronoun. Variables or functions with a Boolean value should not be described by a noun, pronoun, or phrase based on a noun or a pronoun, but instead, they should be described by a clause or a sentence.

Verb of State or Being As pointed out above, a variable or function with a Boolean value should be described by a clause or sentence, not by a noun, pronoun, or a phrase based on a noun or a pronoun. The name of a variable or function with a Boolean value may be shortened to a verbal phrase, thereby maintaining the distinction between a Boolean and a non-Boolean variable or function, the latter being given a name based on a noun or pronoun.

A Boolean variable or function should be described by a clause or sentence containing a verb of state or being. Depending on whether or not the verb is a conjugated form of "to be," the name of the Boolean variable or function is best formed as follows. If the verb of state or being is "is" or "are" (or some other tense of "to be"), it will be followed by a predicate adjective or noun. Form the name of the Boolean variable or function by combining the verb and the predicate adjective or noun. The combination can be abbreviated when no ambiguity results. Often, the subject of the sentence describing the Boolean function will be the argument or one of the arguments of the Boolean function. When naming a Boolean variable, the subject can also be included in the name. Some examples are:

- DoorIsOpen
- AllCardsAreDealt
- MotorPowerIsOn
- IsGreen(object)
- IsColor(object, color)
- IsPhysician(name)
- IsProfession(name, profession)
- IsKingOf(name, country)
- IsOff(device)
- IsOn(device)
- IsOpen(door)
- IsOpening(door)
- IsClosed(door)
- IsClosing(door)
- IsInState(subsystem, state)

Sometimes one is tempted to shorten the name of a Boolean variable or function by dropping the verb. The remaining part of the name will be a noun or adjective. This can lead to misinterpretation of the nature of the variable or function. Therefore, such an abbreviation should be avoided. As an example of the possible confusion that can result, consider the names "IsKingOf" and "KingOf." The name "IsKingOf" includes a verb and, therefore, one would expect the value of the function to be true or false (i.e., Boolean). The name "KingOf" is the beginning of a noun phrase, so one would expect the value of the function to be the person who is the king of the given country. Thus, abbreviating the function name "IsKingOf" to "KingOf" misleads the reader. Similarly, one would expect "PowerIsOn" to be the name of a Boolean variable or function. The name "PowerOn" can easily be expected to be the name of a variable or function whose value is the object of the preposition "on," that is, whose value is the device or devices to which power is being applied (e.g., "rear wheels only," "front wheels only," "front and rear wheels," "headlights," "all").

To avoid such possibilities of misunderstanding, one should not abbreviate the names of Boolean variables or functions of non-Boolean arguments by dropping the verb from the name.

If the verb of state or being is not a conjugated form of "to be," the name of the Boolean variable or function can be the verb alone or the verb followed by part or all of the verbal phrase of the English sentence describing the Boolean variable or function. In the case of a Boolean variable, it is often desirable to include the subject of the descriptive sentence in the name of the variable. In the case of a Boolean function, the subject of the descriptive sentence will often be one of the arguments of the function. Some examples are:

- GeorgeLovedSarah
- Eats(animal, food)
- ResidedIn(person, city)
- WorksAt(person, company)
- Shaves(name of person shaving, name of person being shaved)
- WillPlay(player, card)

Remember that the Boolean variable or function itself is not associated with any time, the concept of time being absent in the Language of Mathematics. Any association between the value of the variable or function and time is in the interpretation, not the mathematical variable or function itself.

Note that many verbs of the type noted above represent habitual, regular action and can be viewed as verbs of state or being. “George loves Sarah” and “Cows eat grass” are examples of sentences containing verbs that can be viewed as stative verbs (i.e., timeless, tenseless verbs of state or being). Such verbs do not refer to time, but express a static relationship.

In summary, the name of a variable or function should signal the type of value of that variable or function. A name based on a verb signals a Boolean value. A name not based on a verb signals a non-Boolean value.

Verb of Action English statements or phrases about action in the application domain cannot be translated to variables and functions in the Language of Mathematics. Therefore, action verbs are not appropriate and should be avoided in the interpretation of variables and functions and should not appear in the names of variables or functions. Statements about action in the application domain must be reformulated in terms of statements of state or being before defining and naming variables or functions. This is not difficult, but it is necessary that one pays conscious attention to this idiosyncracy of translating between English and the Language of Mathematics.

Action verbs often describe or represent a change of state. In such cases, a sentence with an action verb can be reformulated to refer to the state before or after the transition. The states will be described by noun or adjective phrases and will be represented by variables and sequences of values. Further details of modeling dynamic processes in the Language of Mathematics are discussed in Section 7.5.

Some examples of sentences with action verbs and how they can be reformulated in terms of states are:

- If the pedestrian pushes the button to cross the street, … :
  - If button X is depressed, … or
  - If button contact X is closed, … .

- When the pilot pulls the stick back, … :
  - When the control stick is in position (v, h), … .
- If a person pushes against the elevator door while it is closing, … :
  - If the blockage sensor on the elevator door is on … or
  - If the signal from the blockage sensor on the elevator door is active …
- If the reactivity rises above the permissible limit, shut the reactor down:
  - If the reactivity is greater than the reactivity safety threshold, the reactor's control state is "shut down."

Basically, every clause containing an action verb is reworded to refer to the state after (and possibly also the state before) the transition represented by the action. The reworded clauses contain only verbs of state or being. The reworded statements are the basis for formulating the interpretations and names of relevant variables and functions.

Adjective and Predicate Adjective Adjectives and predicate adjectives normally appear in names and interpretations of variables and functions as parts of noun or verb phrases. They are discussed in the corresponding sections on nouns, pronouns, and verbs above.

Adverb Adverbs play a subordinate role in the names and interpretations of variables and functions. There, they modify adjectives and verbs as needed and appropriate. It is usually not necessary to consider them separately when formulating names and interpretations of variables and functions.

Conjunction Conjunctions connect two or more clauses with Boolean values to form a sentence with a Boolean value; the English conjunctions "and" and "or" (corresponding to the mathematical functions of the same names) are two common examples. In such cases, the name of the function will be the conjunction. The interpretation of the function will be the conjunction or a phrase based on it.

A mathematical function corresponding to an English conjunction not connecting clauses should be named and interpreted according to the type of the value of the function and the part of speech of the phrases being joined by the English conjunction. See the corresponding paragraphs above and Section 6.2.4.

The question of naming and interpreting a variable does not arise in the context of a conjunction, because a variable does not join two or more things.

Examples The following table contains examples of functions and shows the relationship between the types of their values and the parts of speech of their names and interpretations.

TABLE 6.2.6-2 Conventions for Names and Interpretations of Functions: Examples

Infix notation	$x < y$	$x \vee y$	$x + y$	(not applicable)
Reference to function	IsLessThan (x, y)	or(x, y)	sum(x, y)	ShapeOf(object)
Function name	IsLessThan	or	sum	ShapeOf
Type of value of function	**Boolean**	**Boolean**	**number**	**other (adjective)** (e.g., triangular)
Type of values of arguments	**number**	**Boolean**	**number**	**geometrical object**
Part of speech of the name	**verb phrase**	**conjunction**	**noun**	**noun**
Alternative names	IsLess	disjunction	addition	
English interpretation (in full)	The value of x is less than the value of y.	The value of $x \vee y$ is the disjunction of the values of x and y.	The sum of the values of x and y	The shape of the object given as the argument
Abbreviated English interpretation	x is less than y.	x or y (or both).	sum of x and y	shape of object
Part of speech of the interpretation	**sentence**	**sentence**	**noun phrase**	**noun phrase**

6.3 CAUSE AND EFFECT

Cause–effect relationships can be expressed in English. They can be expressed explicitly, as in the sentence “Rain falling on the grass makes the grass wet.” The verb “makes” indicates clearly that “Rain falling on the grass” is the cause of the effect that the grass becomes wet. Such causal relationships are expressed so often in English that noncausal sentences are often used to imply causality, as in the sentence “If it rains, the grass will become wet.” Analyzed precisely and pedantically, the latter sentence does not state at all that the rain *causes* the grass to become wet, only that there is a relationship between rain and the grass becoming wet. Which causes which, whether both are caused by something else, or the absence of a causal relationship between them is not stated. However, the context implied by the sentence and the generally recognized causal relationship leads most people to interpret that sentence to mean that the rain causes the grass to become wet. That is, most people will understand that sentence to imply a causal relationship, even though, strictly and precisely, it does not.

Causality cannot be expressed in the Language of Mathematics. There is no provision for causal relationships in the definitions of values, variables, expressions, and functions. Cause–effect relationships may be defined in the interpretation of a mathematical model to the application area, but this must be done in English—outside the Language of Mathematics. A mathematical expression relates values to one another but does not state anything about cause and effect. The concept of cause and effect is simply missing from the Language of Mathematics. For example, the Boolean expression “$x=y+2$” corresponds to the English sentence “The value of x equals the sum of the value of y and the value 2.” A value of 3 for y does *not* cause the value of x to be 5. Neither does a value of 5 for x cause the value of y to be 3. Neither does a value of 3 for y and a value of 5 for x cause the value of the expression to be “true”; in this case, the value of the expression *is*, by definition, “true.” One should be careful not to attribute cause–effect relationships to mathematical expressions.

The logical implication function (infix notation $\Rightarrow$) is sometimes used in a mathematical model to represent a causal relationship in the application area, but even in this case the causality is introduced in the interpretation of the model as mentioned in the paragraph above. The concept of causality always lies outside the Language of Mathematics, never within it.

In English texts the notion of cause and effect is often used more extensively than is really appropriate, especially in technical contexts. One often finds it convenient to think of a voltage across a resistor (e.g., the filament of a light bulb) as causing current to flow through it. However, it is just as meaningful physically to think of the current flowing through the resistor as causing a voltage across its terminals. Physics, as mathematics, says nothing, really, about which is the cause and which is the effect. The very concept of cause and effect in this case is not really physically meaningful; it is an artifact of human thinking. There are many similar examples in the scientific, engineering, and technical fields.

6.4 WORD ORDER

When analyzing an English sentence to determine its meaning when preparing to translate it, the order of words and phrases is important—but sometimes also misleading. Before translating an English sentence into the Language of Mathematics, one must be consciously aware that the structure of the given sentence and its word order might not reflect the intended meaning. Common usage, although usually adequate for general communication, can often lead to ambiguity that is easily overlooked and that can lead to mathematical expressions that do not express the meaning intended. In particular, the position of words and phrases modifying other words in the sentence can lead to problems that are often not recognized immediately.

For example, consider the sentence “He is tied to a fence with white shorts.” Grammatically, it is not at all clear what the phrase “with white shorts” modifies. Proximity to the word “fence” suggests first that the phrase “with white shorts” modifies “fence,” in which case the fence has or is wearing white shorts, which is semantically very unusual. The word “shorts” here presumably means an item of clothing, which suggests that the person tied to the fence is wearing white shorts. In this case, it is not clear whether he is wearing only the white shorts or the white shorts under outer trousers. Another possibility is that the prepositional phrase “with white shorts" is used as an adverb, modifying "tied," in which case the white shorts were used instead of a rope to tie him to the fence. Thus, this sentence can be interpreted to have any one of several meanings; for example:

- He is tied to a fence. The fence has white shorts.
- He is tied to a fence. The fence is wearing white shorts.
- He is tied to a fence. He is wearing white shorts under his trousers.
- He is tied to a fence. He is wearing only white shorts.
- He is tied to a fence. White shorts were used to bind him to the fence.

Considering all aspects of this sentence, the last possibility above is probably the most justifiable interpretation, but the next-to-last interpretation above is also very likely to be the intended meaning. In any case, the sentence is ambiguous.

The word order in the original sentence above strongly suggests either the first or second interpretation above, but the meanings of the words "he," "tied," "fence," and "shorts" argue just as strongly against these interpretations.

As the example above illustrates, considerable ambiguity can be introduced into an English sentence by careless word order. The adverb "only" is often placed in an inappropriate position in a sentence, increasing the possibility of misinterpreting the sentence. It is often placed within or next to the verb, although it really is intended to restrict the meaning of some other part of the sentence.

For example, consider the sentence "The light should only be on if condition A is true" in a context in which two conditions, A and B, are mentioned. How should this sentence be interpreted? Which of the following meanings did the author intend?

- The light should be on if condition A is true, regardless of condition B. Otherwise, the light should be off.
- The light should be on if condition A is true and condition B is false. Otherwise, the light should be off.

If the first interpretation above was meant, a much clearer formulation of the original sentence would be "The light should be on only if condition A is true" or, even better, "The light should be on if and only if condition A is true."

If the second interpretation above was meant, a much clearer formulation of the original sentence would be "The light should be on if only condition A is true" (note the different position of the word "only") or, still clearer, "The light should be on if and only if only condition A is true." Because this last formulation includes the word "only" twice, it is stylistically perhaps poorer, but it is in a form closer to mathematical language and, therefore, less ambiguous.

Still clearer and completely unambiguous would be a table representing the intended meaning in mathematically explicit form, such as, for the first interpretation above,

Condition A	Light
false	off
true	on

or, for the second interpretation above,

Condition A	Condition B	Light
false	false	off
false	true	off
true	false	on
true	true	off

representing the mathematical expressions

$$\text{light}\in\{\text{off}, \text{on}\} \wedge \text{conditionA}\in\{\text{false}, \text{ true}\} \wedge (\text{light}=\text{on}) = \text{conditionA}$$

and

$$\text{light}\in\{\text{off}, \text{on}\} \wedge \text{conditionA}\in\{\text{false}, \text{ true}\} \wedge \text{conditionB}\in\{\text{false}, \text{ true}\} \\ \wedge (\text{light}=\text{on}) = (\text{conditionA} \wedge \neg\text{conditionB})$$

respectively. Such tables can be structured and labeled so that they can be read and understood by nonmathematically inclined people. For logical relationships that are difficult to express clearly and unambiguously in English, tables should always be considered as an alternative descriptive mechanism in otherwise English texts.

The reader should identify still other possible interpretations of the original sentence above. How can they be expressed unambiguously in English?

The position of the word "not" for logical negation in an English sentence can also lead to confusion and misunderstanding. Normally, "not" is an adverb modifying a verb and it is, therefore, typically written adjacent to the verb or some part of it. In some situations, however, it is meant as a restriction on an adjective or even on a noun. Sentences in which the word "not" refers most directly to an adjective or to a noun are usually abbreviated sentences with major parts of one or more clauses implied. This should be kept in mind explicitly when analyzing such a sentence.

For example, consider the sentence

Not Bill, but either George or Sam is the criminal.

Here, "not" refers most directly to the person named "Bill," not to the verb "is." The sentence above suggests immediately the mathematical expression

$$(\neg\text{Bill} \wedge (\text{George } \vee \text{Sam})) = \text{criminal}$$

which is obviously meaningless and nonsense. The negation function $\neg$ yields a Boolean value, and its argument must be Boolean. The values of the arguments of the $\vee$ function must also be Boolean. Presumably the names Bill, George, and Sam are values in this context (i.e., are not variables with Boolean values), so are not valid arguments for the $\neg$ and $\vee$ functions. The value of the variable criminal should presumably be a name (e.g., Bill, George, Sam), not a Boolean value. The necessity that arguments of certain functions have certain types of values was discussed in several parts of Chapter 3 and is dealt with in more detail in Section 6.5.

These inconsistencies arise because the sentence above is in a highly abbreviated form that disguises the grammatical structure of its nonabbreviated intended meaning, which is

> The criminal is not Bill, but is either George or Sam.

or, more fully,

> The criminal is not Bill, but either the criminal is George or
> the criminal is Sam.

In this form, it is clearer that "not" does not modify "Bill," but instead, that "not" modifies the verb "is" and hence effectively the entire clause "The criminal is Bill." The words "but" and "or" are conjunctions, each combining clauses. This form of the sentence leads directly to the correct mathematical expression

$$\neg(\text{criminal}=\text{Bill}) \wedge (\text{criminal}=\text{George} \ \vee \ \text{criminal}=\text{Sam})$$

whose value is Boolean, as the value of an expression corresponding to an English sentence must be. The values of all terms in the expression above are also consistent with the functions applied to them.

As pointed out in Section 6.2.4, the English conjunction "but" means, logically, "and."

In summary, when "not" appears to refer most directly to something other than a verb, look for implicit clauses. Reformulate the given sentence, at least mentally, into the full, intended sentence, with all verbs expressed explicitly. Whenever the word "only" appears in an English text, identify exactly the word to which it logically refers and its intended meaning. More generally, ask yourself consciously which words and phrases modify which other words and phrases in a sentence and whether or not they modify what the author actually *intended* them to modify.

6.5 GRAMMATICAL AGREEMENT

In English and other natural languages, various grammatical rules must be satisfied. These rules pertain to relationships between various parts of sentences. For example, in English the verb must agree with the subject in person and in number:

- I am happy.
- You are happy.
- Sarah is happy.
- They are happy.

The sentences “I are happy,” “I is happy,” “You am happy,” “You is happy,” “Sarah am happy,” “Sarah are happy,” and “They am happy” are grammatically incorrect, although the meanings are clear.

The reader should identify which of the following examples are grammatically correct and incorrect and why.

- George saw he.
- George saw him.
- George saw his.
- George saw she.
- George saw her.
- George saw hers.
- George saw they.
- George saw them.
- I saw George.
- Me saw George.
- She saw George.
- Her saw George.
- Hers saw George.
- They saw George.
- Them saw George.

These examples illustrate the requirement that the object of a sentence be in the objective case, not the nominative or genitive case. Similarly, a pronoun must be in the subjective (nominative) case when it is the subject of the sentence. In English, only pronouns have case distinctions. Many other languages have distinctive case forms for still other pronouns, nouns, adjectives, and articles as well.

Similarly, in the Language of Mathematics the types of arguments must be consistent with the functions applied to them. This need for type agreement between different parts of an expression was covered initially in Section 3.4.2 on infix notation and in Section 3.4.3 on tree notation. See also Section 3.3 and the latter part of Section 4.1.1 for the definition of the domain of a function. In the paragraphs below, this aspect of the grammar of the Language of Mathematics is discussed in greater depth and more extensively. In particular, its influence on the possible meanings of an expression consisting of a sequence of infix functions without parentheses is examined.

Consider an infix symbol op and the following expression involving it:

x op y op z

The justification for writing this expression without parentheses is the observation that in many cases, the expression (x op y) op z and the expression x op (y op z) are both defined (the requirements for type agreement are met) and the two expressions have the same value. Thus, it makes no difference where one writes the parentheses.

This leads to the view that the parentheses are irrelevant and redundant, so they can be dropped completely.

More generally, the expression x op y op z can be interpreted to mean any one of several things. The requirement that the types of arguments agree with the function being applied to them influences which meaning can be associated with the expression above.

First priority is given to the interpretation above, which motivated dropping the parentheses in the first place: that is, that the expression x op y op z is interpreted to mean

$$(\mathrm{x\ op\ y})\ \mathrm{op\ z}$$

or

$$\mathrm{x\ op}\ (\mathrm{y\ op\ z})$$

provided that (1) both of these expressions satisfy the requirement of type agreement and (2) they both have the same value for all values of the variables x, y, and z (i.e., the function op is associative). Requirement (1) is satisfied if and only if the types of the arguments and of the function value are the same. This implies that the variables x, y, and z are all of the same type.

If both of the parenthesized expressions above satisfy the requirement that all types are consistent, but the function op is not associative, then the rule "parenthesize from left to right" for functions with the same binding order (see Table 3.4.2-1 and the accompanying text) can be employed, but this can easily lead to a reader misinterpreting the expression. In this case, the intended parentheses should be written explicitly in the expression.

If the interpretation with first priority above cannot apply because the requirements are not fulfilled, then **second priority** is given to the interpretation

$$(\mathrm{x\ op\ y}) \wedge (\mathrm{y\ op\ z})$$

provided that the requirement for type agreement is satisfied.

If the requirements for both interpretations above are violated, the original expression is syntactically incorrect—it is not a valid construct in the Language of Mathematics. It must be rewritten.

An example of the expression $\mathrm{a}+\mathrm{b}+\mathrm{c}$, where a, b, and c have numerical values, was presented and discussed in Section 3.4.2. The requirements for the first convention above are met, so that convention applies.

The relational operators (e.g. $<$, $>$, $\leq$, $\geq$, $=$, $\neq$) and the implications ($\Rightarrow$, $\Leftarrow$, $\Leftrightarrow$) are important in this context, as they exhibit pecularities of the type mentioned above. Below are examples of their usage and meaning in certain forms of expressions.

Example 1: $=$ Applied to $\mathbb{B}\times\mathbb{B}$ Consider the expression

$$\mathrm{x} = \mathrm{y} = \mathrm{z}$$

where x, y, and z are variables or expressions with Boolean values. The requirements for the first interpretation above are met—the arguments and the value of the function are the same (Boolean) and the function $=$ on $\mathbb{B}\times\mathbb{B}$ is associative. The reader should prove that this is the case. The expression $x{=}y{=}z$ is, therefore, defined to mean either $(x{=}y){=}z$ or $x{=}(y{=}z)$, both of which have the same value for all Boolean values of x, y, and z.

Example 2: $=$ Applied to $\mathbb{R}\times\mathbb{R}$ In this example, the expression $x{=}y{=}z$ does not fulfill the requirements for the first convention because the arguments of $=$ are numbers, but the value of the function $=$ is Boolean (false or true). The requirement (type agreement) for the second convention does apply, so that in this case the expression $x{=}y{=}z$ is defined to mean $(x{=}y)\wedge(y{=}z)$. This last expression implies that $x{=}z$, so that the expression $x{=}y{=}z$ can be interpreted to mean that all three variables (or expressions) x, y, and z have the same value. In this example, the set $\mathbb{R}$ is not critical; the same conclusions apply if any set other than $\mathbb{B}$ is substituted for $\mathbb{R}$.

Example 3: $<$ Applied to $\mathbb{R}\times\mathbb{R}$ In this example, the expression $x{<}y{<}z$ does not fulfill the requirements for the first convention because the arguments of $<$ are numbers, but the value of the function is Boolean (false or true). The requirement (type agreement) for the second convention does apply, so that in this case the expression $x{<}y{<}z$ is defined to mean $(x{<}y)\wedge(y{<}z)$. This last expression implies that $x{<}z$. In this example, the set $\mathbb{R}$ is not critical; the same conclusions apply if any linearly ordered set (other than $\mathbb{B}$) is substituted for $\mathbb{R}$.

Example 4: $\Rightarrow$ Applied to $\mathbb{B}\times\mathbb{B}$ In this example, the expression $x{\Rightarrow}y{\Rightarrow}z$ satisfies the first convention's requirement of type agreement, but $\Rightarrow$ is not associative. One could apply the first convention and the "parentheses from left to right" rule, but as mentioned above, this is not desirable, as it can too easily lead to misinterpretation. Applying the second convention would lead to the expression $(x{\Rightarrow}y)\wedge(y{\Rightarrow}z)$. Because it is not really clear to the reader whether the writer intended the first or the second convention to be applied, the writer should avoid an expression of the form $x{\Rightarrow}y{\Rightarrow}z$ and, instead, write the intended expression with appropriate parentheses.

Example 5: $\Leftrightarrow$ Applied to $\mathbb{B}\times\mathbb{B}$ The functions $\Leftrightarrow$ and $=$ are both equality functions; that is, the value of each function is true if both arguments have the same value and false otherwise. The functions differ in that the domain of $\Leftrightarrow$ is $\mathbb{B}\times\mathbb{B}$, while the domain of $=$ is $\mathbb{S}\times\mathbb{S}$, where $\mathbb{S}$ is any nonempty set (i.e., including $\mathbb{B}$). Also, their binding orders are different (see Table 3.4.2-1). The comments on Example 1 above apply here. Correspondingly, the expression $x{\Leftrightarrow}y{\Leftrightarrow}z$ is to be interpreted to mean either $(x{\Leftrightarrow}y){\Leftrightarrow}z$ or $x{\Leftrightarrow}(y{\Leftrightarrow}z)$.

Depending on one's viewpoint, the various interpretations of the several subexpressions appearing in infix expressions discussed above can be considered to be definitions, abbreviations, idioms, irregularities, peculiarities, idiosyncracies, or oddities of the Language of Mathematics. In this regard, the Language of Mathematics is just like English or any other natural language; they all have their own peculiarities. The need for parentheses to match and ways to check that they do were discussed in Section 3.4.2.

6.6 VERBS: TENSE, MOOD, VOICE, ACTION VS. STATE OR BEING, STATIVE

In English and other natural languages, different conjugated forms of verbs are used to distinguish the characteristics tense, mood, voice, and action vs. state or being. In English, some of the differences in the conjugated forms have disappeared as the language has evolved, with the consequence that fluent and even native speakers of English are sometimes unaware of some of the grammatical distinctions. In the case of auxiliary verbs, even infinitive forms have disappeared from English (e.g., the infinitive forms for the verbs may, must, can, and shall). In other languages (e.g., French and German) these infinitives still exist and are used.

Tense The tense of a verb expresses the time when the action took place or the state existed. The tense can also indicate if the action or state ended and whether the action or state was continuous, habitual, or a single event. The temporal relationship between actions or states can also be expressed by the tenses of the corresponding verbs.

In the Language of Mathematics, the meanings of mathematical expressions depend only on the values associated with the variables appearing in them. There is no reference to time—a concept missing completely from the Language of Mathematics. The Language of Mathematics is a timeless language. Verbs directly associated with mathematical expressions are, therefore, timeless—they are tenseless.

In the Language of Mathematics, verbs arise only in Boolean-valued functions, and there only implicitly, see Section 6.2, especially Sections 6.2.1 and 6.2.6. These verbs are always verbs of state or being. Most commonly, they are expressed by conjugated forms of “to be” in English. Other examples are “equals” (is equal to), “exceeds” (is greater than), and “implies.” Action verbs do not appear in direct, literal translations from the Language of Mathematics to English.

In the grammar of natural languages such as English, verbs in statements that express states (i.e., relationships, characteristics, or properties as opposed to actions) independent of time or without reference to time are called *stative verbs*. All implicit verbs arising in Boolean functions are such stative—timeless, tenseless—verbs. Direct, literal translations from the Language of Mathematics into English describe a timeless, static model. The present tense in English is usually the most suitable tense for such verbs in a direct translation from the Language of Mathematics to English, but it is only an approximation to the true timelessness of the implicit verbs in the Language of Mathematics. The tense (reference to time) associated with every conjugated verb in English can lead one to think of a verb in the Language of Mathematics as expressing a state or relationship valid in the past, present, and future, or a state or relationship that has no beginning or end point of concern in time, but this is introducing a concept actually absent from the verb in the Language of Mathematics.

Some examples of stative verb constructs in English are: Lemons are yellow. Giraffes’ necks are long. Giraffes have long necks. The sun is hot. Mt. Everest is high. The specific weight of water is 1 kg/liter. Water weighs 1 kg/liter.

English and many other contemporary natural languages do not have specific grammatical forms for stative verbs. In such languages, stative verbs can be identified only by their meanings and within context. The irrelevance of time for the truth or falsity of the statement is one of the two criteria for identifying a stative verb. The other criterion is that the verbal part (predicate) of the sentence describes a state or condition rather than an action. The verb in a predicate expressing a static relationship is typically stative.

The ancient Akkadian, written in cuneiform on clay tablets and the oldest known Semitic language, had a particular conjugation for stative verbs. Middle Egyptian, written with hieroglyphs and derived scripts, also had specific grammatical forms for stative verbs. In such languages, stative verbs can be distinguished grammatically from nonstative verbs, without having to rely on the intended meanings of the statements in which they appear. A stative verb can be viewed as an inflected verbal form of an adjective. The subsequent development of natural languages has, apparently, tended to drop the grammatical distinction between stative and nonstative verbs.

In Akkadian, stative verb forms of many words were used instead of the verb "to be" with a predicate adjective, so the verb "to be" was used infrequently. Such a structure in the verb system corresponds well with verbs in the Language of Mathematics, and if English had explicit forms for stative verbs, they would be used in English translations of mathematical models expressed in the Language of Mathematics. Lacking explicit forms for stative verbs, however, many translations of Boolean expressions and functions into English will consist of the verbs "is" or "are" followed by a predicate adjective or predicate adjective phrase describing a state, characteristic, or property. However, even these will be only approximations to the timeless meanings of the mathematical expressions in question.

In order to model temporal relationships in the application world (e.g., the physical world), mathematical variables representing time in the application world and functions of these variables are employed. The mathematical representation (model) itself remains a static one. Lack of conscious awareness of this distinction has caused much confusion and many errors in the practical application of mathematics to dynamic processes: for example, in the mathematical specification and logical analysis of computer programs.

English and natural languages have the capability to express both static and dynamic views in their universe of discourse. The Language of Mathematics itself enables only a static view to be expressed. Some technical languages (e.g., many computer programming languages) are oriented to a dynamic view of a process, such as a computer executing a program. Some processes of concern to engineers in the traditional fields are fundamentally dynamic. These differences must be kept in mind when applying mathematics in practice.

Mood In English and other natural languages, the mood of a verb expresses a category of purpose or intended interpretation of the verb and the statement in which it appears. The *indicative mood*, typically the most commonly used mood, expresses a fact or asks a question of fact.

The *subjunctive mood* expresses supposition, speculation, condition, wish, possibility, uncertainty, irreality, and so on. A sentence with the verb in the subjunctive mood is not to be understood as a statement of fact. For example, in the sentence "If I were you, I would study medicine," the first clause expresses an unreal supposition, and the second clause, an action that is not taking place and that will not take place. Both verbs are, therefore, in the subjunctive mood. (Some other languages, and some English grammarians, distinguish between the subjunctive mood and the *conditional mood*.)

The *imperative mood* expresses a command: for example "Come here."

Mathematical expressions may also have various purposes or intended interpretations. The antecedent in an implication (e.g., "A" in the expression "A⇒B") corresponds directly to the subjunctive (or conditional) mood in many natural languages. In this structure, it is not necessary to distinguish explicitly between the subjunctive and the indicative moods because the meaning is completely clear from the context. Furthermore, the antecedent A above will at least sometimes refer to a very real condition, in which case the subjunctive mood is not really appropriate. A corresponding English sentence might be "When a 100-kg weight is attached to the cable, the cable will not break," in which both verbs are in the indicative mood.

For these reasons, a distinction corresponding to the subjunctive mood is not made in the Language of Mathematics. Such distinctions are often neglected in English, also. In fact, so many subjunctive forms have been dropped in the evolution of English that one could say that the subjunctive mood has almost disappeared from common usage. Many other natural languages, however, still require that this distinction be expressed explicitly and systematically. Such languages have retained the distinctive conjugational forms of the verbs for the subjunctive mood.

Semantically, the Language of Mathematics does give rise to other needs for moods of the verbs. Different purposes and intended interpretations (moods) of mathematical expressions are listed and described in Sections 6.9 and 6.12. If there were a mechanism in the Language of Mathematics to indicate the intended purpose or interpretation of an expression, the several purposes could be viewed as moods of the verbs in question. However, there is no such mechanism in the Language of Mathematics, so the purpose or grammatical "mood" in this sense must be explained in English in the interpretation of the mathematical expressions in question. See Section 6.13 and the examples in Chapter 7.

Voice The *voice* of a verb refers to the relationship between the verb and the subject of the clause either as the acting agent (actor, performer of the action) or the receiver of the action. In the sentence "The boy threw the ball," the verb "threw" is in the *active voice* because the subject of the sentence, "boy," is the performer of the action. The *passive voice* version of this sentence is "The ball was thrown by the boy"; here the subject of the sentence, "ball," is the receiver of the action.

In mathematical expressions, this distinction could occasionally be made, particularly in logical implications. The expression "A⇒B" (A implies B) could be viewed as the active voice version and the reversed implication "B⇐A" (B is implied by) as the passive voice version. This distinction is unnecessary and contributes nothing of apparent value in terms of insight, so the voice of a verb can be neglected in the Language of Mathematics.

Action vs. State In natural languages, the distinction between verbs of action and verbs of state or being is useful in many respects. As pointed out earlier, there is no equivalent to verbs of action in the Language of Mathematics, all verbs being not only timeless, but also verbs of state or being. Therefore, in the Language of Mathematics no distinction between verbs of action and verbs of state can be made.

To model an action mathematically, one must formulate an appropriate static representation of the dynamic process involving the action. A mathematical structure commonly used to represent discrete time steps is based on a sequence (see Section 4.1.3), with the terms in the sequence representing states resulting from actions, not the actions themselves. The steps between the terms in the sequence correspond to actions. Related to this is the use of a finite state machine (see Section 4.1.7), in which the transitions between states are associated with actions. Thus, the mathematical expressions model the results of the actions, not the actions themselves. In other cases, a variable can be interpreted as a command to perform the action in the physical world: for example, a variable with the meaning “power to the motor” or the interpretation “apply power to the motor” (see the example in Section 8.13).

In summary, in the Language of Mathematics it is neither meaningful nor possible to distinguish verbs based on tense, voice, or action vs. state. It is meaningful to distinguish between the moods (i.e., the purposes or intended interpretations) of mathematical expressions. The Language of Mathematics has no mechanisms for making these distinctions. Such distinctions can be made only in accompanying text in English, in the interpretation of the mathematical model in question.

In translations from the Language of Mathematics into English, verbs will appear in a tense other than the present or in a mood other than than the indicative only when the text is intended to reflect the interpretation of the mathematical model in the context of the application world. Such forms of verbs will appear only when the English text is not intended to be a direct, literal restatement of the mathematical model itself. See also Sections 6.9, 6.12, and 6.13 and Chapter 7.

6.7 AMBIGUITY

Ambiguity in the logical sense has no place in the world of mathematics. The Language of Mathematics does not provide for ambiguous or vague expressions or statements. Probability theory, a subdiscipline of mathematics, does enable one to make statements about uncertain events, but those statements themselves are neither ambiguous nor vague. Similarly, “fuzzy” theory provides a way of modeling incompletely or vaguely defined objects or properties, but the statements about those incompletely or vaguely defined objects or properties are, themselves, unambiguous.

Statements written in natural languages such as English are almost always ambiguous in some way, that is, are subject to different interpretations. For most purposes for which natural languages are used, this can actually be an advantage. Most writers and even more readers do not want to go into the excrutiating, pedantic precision and detail that would be necessary to even attempt to eliminate ambiguity. Authors of literary works often intend to invoke the reader's imagination to fill in details, and this requires a language that permits ambiguity—that leaves certain details open.

Because sentences in English are unconsciously interpreted within certain contexts, the communicating parties are usually not consciously aware of the ambiguity of their communication and the misunderstanding it can cause. The ambiguity in typical English sentences is often underestimated.

One example sometimes cited is the sentence

> Time flies like an arrow.

The most common interpretation is probably that this sentence is an observation about how fast time seems to pass. In this interpretation, "time" is a noun and the subject of the sentence, "flies" is a verb in the indicative mood, and "like an arrow" is a prepositional phrase used as an adverb, indicating how fast time seems to go.

A second interpretation, grammatically and semantically fully justifiable, is that this sentence is a command to measure the speed at which flies fly ("time" their flight), just as one would measure the speed of an arrow in flight. In this case, the word "time" is a verb in the imperative mood, "flies" is a noun and the object, and "like an arrow" is a prepositional phrase used as an adverb.

A third interpretation, also grammatically and semantically fully justifiable, but in a context somewhat difficult to imagine, is the following. "Time" is an adjective indicating which type of flies is meant, "flies" is a noun and the subject of the sentence, "like" is a verb, and "an arrow" is a noun phrase and the object.

Most people will immediately interpret the sentence above in the first way described. When pressed for another meaning, some will recognize the second interpretation as valid. Few will recognize the third interpretation, but when it is explained, most will agree that it is, grammatically, a possible interpretation, although a rather odd one.

Similarly, many unexpected ambiguities exist in English sentences describing a problem to be solved or an application to be analyzed. Sometimes, the author will see only one interpretation and the reader will also see only one interpretation, but their two interpretations will be different. The discrepancy will often go unnoticed until much later in the project, when resolving the matter will be more time consuming and expensive and will require redoing some of the intermediate work.

For logical and technical analyses, ambiguity is counterproductive and undesired. For those purposes, a language is needed that permits—or even better, requires—statements to be unambiguous. The Language of Mathematics has evolved to fulfill this need. Where ambiguities could arise (e.g., when abbreviating), conventions have been laid down for eliminating otherwise possible ambiguities (see Sections 3.4 and 6.5 for examples).

Do not try to express ambiguous statements in the Language of Mathematics. Either you will not be able to do so or the expressions you do write will not be legitimate mathematical expressions—they will violate rules of the Language of Mathematics.

The languages and notations used in mathematics were not always unambiguous. Initially, mathematical statements consisted of numbers embedded in natural language. Very early number systems were positional systems but without a symbol for zero. Furthermore, a symbol for separating the whole-number part from the fractional part was not written. For example, “257” could have stood for two hundred and fifty-seven, two thousand five hundred and seven, twenty-five and seven tenths, and so on. Because named variables did not exist, references to the various values to be used in the calculation were expressed in words or numbers. In Babylonian texts describing calculations, for example, this leads to ambiguity regarding the targets of such references, making it more difficult for us to decipher the ancient algorithms correctly and precisely.

6.8 STYLE

In any language there are many ways of expressing any particular thing. Selecting one of the grammatically and semantically valid possibilities is a matter of style. The overall goal is to make it easy for the reader to read and understand the intended message fully with a minimum of time and mental effort. Good style is just as important in mathematics as in English and other natural languages—perhaps even more important, because for readers of mathematical expressions the Language of Mathematics is not their first, their native language.

The authors of English texts pursue a variety of goals. Literary and poetic texts aim to convey a combination of ideas, concepts, facts, impressions, feelings, emotions, enjoyment, and entertainment. The reader is often expected to fill in images and details based on imagination stimulated by the text. The style in which the text is written should encourage and enable the reader to participate actively and effectively in this communication.

The goals of technical texts are similar, but with a rather different distribution of emphasis. Depending on the specific text, conveying facts, concepts, or ideas is of primary importance. Impressions, feelings, emotions, enjoyment, and entertainment are of comparatively little or even no concern, but this does not mean that the reader should find the text unenjoyable or even disagreeable. Mathematical models and expressions are normally intended to convey primarily facts, and secondarily concepts and ideas, relating to some rational problem so that a solution can logically be deduced.

Feelings and emotions are rarely the concern of authors of mathematical expressions. The universe of discourse of the Language of Mathematics does not, itself, include feelings and emotions. Mathematicians do, however, sometimes perceive beauty and corresponding feelings and emotions when reading mathematical material, but these perceptions derive from the way the Language of Mathematics is used to deal with aspects of the world outside mathematics. These perceptions themselves exist only outside the world of mathematics in the narrow sense.

In summary, the goals of most mathematical translations of English text are to convey to the reader facts and related concepts, ideas, and understanding of selected rational aspects of the application area in question. The facts and understanding conveyed should be in a form suitable for mechanistic, symbolic reasoning leading to a solution to a problem. The style in which the mathematical model is written should help readers to understand the problem and deduce a solution to it. Expressed differently, the author should predigest the material for the reader.

Although the overall goals for style are much the same for English and mathematical texts—to help the readers to understand the text quickly—the specific guidelines differ in many cases considerably.

One guideline for English text is to avoid repetition. Repetition is often perceived as boring. It does not offer alternative views that might stimulate the readers' own imagination. Instead, the use of other words with similar meanings and, for the sake of brevity, pronouns is suggested. Limits are posed by the rather small number of pronouns available in English, the limited number of other words with appropriate meanings, and the fact that distant references of pronouns to their targets can reduce, rather than increase, clarity. In mathematical models, there are no near synonyms (either the names of variables or functions are the same or they are different) and there are no pronouns as such, so the author of a mathematical model has no choice but to repeat. There is one exception: Define a function to represent an expression that is either long or that must be repeated often. Repeating the name of a variable or function is desirable because it makes it easier for readers to see that the same thing is meant in the various places.

Alternative views can and, where appropriate, should be expressed in mathematical text. Such alternatives can usually be expressed most clearly by alternative formulations of the mathematical model or of parts of it, not by embedding the different views in one conglomerate model. It should be made explicitly clear to the reader what the alternative views are.

An essential guideline for style is: Be especially careful to write precisely what you mean, not just something that contains the same symbols and "looks sort of" right. For example, when writing references to the functions f and g and a variable x, choose between such expressions as

f(g)
f(g(x))
f(g, x)

carefully. Mean what you write and write what you mean.

The first expression above, f(g), means a function f whose value is determined by the function g, not by the value of the function g for some argument of g.

The second expression above, f(g(x)), means the value of the function f for the value of the function g for the argument value of the variable x.

The third expression above, f(g, x), means the value of the function f for its two arguments, the first of which is the function g (not the value of g for some argument) and the second of which is the value of the variable x.

Still another variation of the expressions above is (f(g))(x). The reader should identify how this expression differs from the three expressions above and to which one(s) it relates and how.

At first glance the differences between these meanings might appear to some to be subtle, but in mathematics there is no such thing as a subtle difference. Two things are either the same or they are different. A "slight" difference is a difference.

Some additional guidelines for the style of a translation of English text into a mathematical model are:

- Introduce new variables and functions where helpful to achieve the goals outlined above.
- Write an interpretation for each variable used in the model.
- Define and write an interpretation for each function used in the model that is not already a well-known standard mathematical function.
- For every variable in the model, include a definition of its value set (e.g., in the form $x \in \mathbb{R}$) at the beginning of the model, before any reference to the value of any variable. In this section of the model, group the independent variables first and the dependent (derived) variables second.
- Structure the entire mathematical model as a single Boolean expression.
- Group and organize the various subexpressions in the mathematical model in a logical way, much as one sequences sentences in English text to follow from previous sentences.
- Indent subexpressions to facilitate reading. Indent in ways related to the nesting of parentheses so that it is clear which parentheses match.
- Include reference numbers or brief comments at the right margin and enclose them in a standard type of parentheses.
- Include redundancy only when it improves clarity and facilitates reading and understanding.
- Simplify expressions and subexpressions where possible.
- Be consistent. For example, sequence references to variables the same way throughout the model. Write subexpressions that occur repeatedly in the same way so that the reader can easily see that they are the same. Apply naming conventions and indentation guidelines consistently.
- Define abbreviated notational forms only when helpful. Use widely accepted terminology and notational forms.
- Choose names of variables and functions to achieve a balance between clarity of meaning and brevity. Follow the guidelines for naming variables and functions given in Section 6.2.6.
- Include a glossary and relevant citations.
- Include an index to names of variables and functions.

6.9 LIMITATIONS AND EXTENDABILITY OF THE LANGUAGE OF MATHEMATICS

The Language of Mathematics is a language of uninterpreted expressions: the mathematical expressions presented in Chapter 3, particularly in Section 3.4. These expressions deal with abstract entities, with no inherent reference to anything in the concrete, real world.

The Language of Mathematics provides no way of indicating how an expression or any of its components is to be interpreted or related to anything in a particular application area, what they mean in the context of the application in question or the motivation behind them. This information can be provided only by English text supplemented appropriately by the special terminology and jargon of the application area in question (see Sections 6.10 and 6.13).

There is an additional ambiguity regarding mathematical expressions. A mathematical expression may represent any of the following:

- A statement that may or may not be true
- A variable or an expression to be evaluated
- A problem to be solved
- An assertion purported to be true
- An assumption (axiom) accepted without proof as true
- A theorem to be proved
- A definition of a mathematical object or structure
- Something whose value is unknown or unimportant within some context

(see also Section 6.12). Which of these applies to any particular expression cannot be indicated within the Language of Mathematics. Often, it is evident from the context, but sometimes it is not. Although the definition of a mathematical object or structure can—and should—be written entirely in the Language of Mathematics, the motivation for it and common applications cannot.

As mentioned above, the Language of Mathematics is a language of uninterpreted expressions. That does not mean that the expressions are uninterpretable. In fact, they frequently are interpreted in the description of the mathematical model of which they are a part. Such interpretations cannot be written in the Language of Mathematics; they must be explained in English text.

In contrast to the limitations of the Language of Mathematics stands its very considerable extendibility. Functions were composed to form other functions in Section 3.4. Different mathematical objects can be combined to form still other structures, such as a finite state machine (defined in Section 4.1.7) or a *group*, which consists of a set and a function of two elements of that set yielding a function value also in that set, and exhibiting certain properties specified by expressions called the *group axioms*. In turn, a *field* is defined as a structure based on a group. Such structures are useful in various analyses and mathematical models. A number of standard structures are defined in mathematics, and additional ones can be defined as needed. This process can be continued without any inherent limit.

The ability to *generalize* is another very valuable advantage of the Language of Mathematics. Generalizing reduces the amount of detail that one must cope with in learning and applying mathematics to practical problems, thereby simplifying problems and making their solutions applicable to a wider variety of problems. For example, any structure of values can, itself, be considered to be a value. A set of values can be handled as a single value. Similarly, a sequence of values can, itself, be viewed as a single value.

Applying this idea to the arguments of functions, the several arguments of a function can be viewed as a sequence of values, which, in turn, can be viewed as a single value. Thus, every function can be viewed as having only a single argument. Even a function with no arguments (i.e., a constant value) can be viewed as a function of one argument, the empty sequence.

As another related example, generalization can be applied to the function "sum" used frequently in Chapter 3. There it was considered to be a function of two numbers. As mentioned above, it can be viewed as a function of one argument, that argument being a sequence of two numbers. This suggests generalizing the definition of the function "sum" to allow the argument to be a sequence of any (finite) length. Even the sum of the terms in an infinitely long sequence (in a sequence of unbounded length) of numbers can, under certain conditions, be suitably defined: for example, by considering the limit of a sequence of sums of a finite number of arguments as the number of arguments increases without bound as outlined in Section 4.4.

Some other examples of unifying mathematical concepts by generalizing them to a common concept are given in Section 4.1.4.

6.10 THE LANGUAGES USED IN MATHEMATICAL TEXT

Mathematical texts, articles, analyses, and so on, are typically written in an unstructured mixture of the following:

- The Language of Mathematics as presented in this book.
- Normal English supplemented by special mathematical terms and jargon.
- The interpretation (as described below and defined in Section 6.13) connecting the two. This interpretation is written in English.

When reading such material, the reader should clearly distinguish mentally between these three components and especially between the precise, unambiguous parts of the messages formulated in the Language of Mathematics and the less precise, ambiguous parts formulated in English. When documenting mathematical models and their application, these three components should be distinguished from one another clearly and structurally in order to help the reader.

There are several reasons for writing mathematical texts in a mixture of these two rather different languages. As discussed in Section 6.9, the Language of Mathematics provides no way to do any of the following:

- Express reasons or the motivation for making the various decisions about what to express in the mathematical model and how to express it
- Explain many characteristics or aspects of the various mathematical expressions, variables, functions, terms, and other parts of the mathematical model
- Interpret the various parts of the mathematical model in terms of the application area: that is, to establish the connection between the mathematical model expressed in the Language of Mathematics and the relevant aspects of the application area

The third reason is particularly important. The Language of Mathematics does not include any of the special terms and jargon used in the application area and needed to understand the relevance of the mathematical model to the application area. For example, a mathematical model for optimizing the inventory of finished goods will contain various variables representing sales, purchases, costs of various types, quantities of the various articles ordered, in stock, arriving, being shipped, and so on. Such a mathematical model will be of use to people only if the various variables are explained and, especially, linked to the quantities in the real world that they represent. The Language of Mathematics does not provide the terminology and vocabulary needed to do this. Natural languages such as English do.

Every other application area provides examples of this inability to explain and express the connection between the mathematical model and the real-world application area. In traditional engineering disciplines, one talks about force, stress, strain, voltage, current, energy, speed, power, temperature, pressure, rate of flow of gases and liquids, and so on. The Language of Mathematics has no vocabulary for such things; they are all represented by numerical variables and values. The Language of Mathematics deals only with a completely abstract world. When applying mathematics, one must establish explicitly the connection between the abstract world of mathematics and the real world of the application. This requires English or some other natural language, supplemented by the special terminology ("jargon") of the application area.

A document presenting a mathematical model for an application will, therefore, typically contain the following:

- A *statement of the problem* to be solved, written in English and mathematical expressions as appropriate
- An *interpretation* of the variables in the mathematical model in terms of entities in the application world, written in English

- The *mathematical model* itself, written in the Language of Mathematics
- An *assignment* of values to variables, written either as mathematical equations or in the form of an equivalent list of the variable names together with the value of each
- One or more *solutions to the problem*, also written in an appropriate combination of English and mathematical expressions

These parts of the documentation of a mathematical model are discussed in more detail in Section 6.13.

6.11 EVALUATING STATEMENTS IN ENGLISH AND EXPRESSIONS IN THE LANGUAGE OF MATHEMATICS

A statement in English, interpreted within the context of one's knowledge, is either true, false, or, if any essential information is missing from one's knowledge, undetermined (undefined, unknown). For example, the sentence "George Washington was the first president of the United States of America" is true within the context of the knowledge learned by schoolchildren in many countries. The sentence "The sum of 2 and 3 is 6" is false within the context of all people who learned arithmetic in school. The truth value of the sentence "At least one of my ancestors who lived in the second century B.C.E. was taller than 190 centimeters" is undetermined within the context of my knowledge, because I know nothing about any ancestor of mine who lived in that time period. My knowledge of my ancestors living in that time period does not enable me to evaluate that sentence as being either true or false. The statement is, clearly, either false or true, but I do not know—and have no way of knowing—which.

The same comments apply to mathematical expressions. They are evaluated within the context of assignments of values to variables. If those assignments lack values for certain variables referenced in the expression to be evaluated, the value of that expression cannot be determined. In many cases, the expression can be partially evaluated, that is, reduced to an expression in which only variables without assigned values appear (see Section 3.5).

6.12 MEANINGS OF BOOLEAN EXPRESSIONS IN AN ENGLISH LANGUAGE CONTEXT

Within the Language of Mathematics a Boolean expression is, like any other expression, an expression ultimately to be evaluated (see Section 3.5): no more, no less. When translating into English, however, one must keep in mind that Boolean expressions typically serve several somewhat different purposes, as mentioned in Section 6.9. An English language translation must usually distinguish between these various purposes. The lines distinguishing these purposes are not always sharp and well defined; they can be somewhat blurred.

A Boolean expression may be a statement that may or may not be true. Typically, the expression is to be evaluated.

A Boolean expression may represent a problem to be solved. The values of variables appearing in the expression are to be determined so that the value of the expression is true.

A Boolean expression may be an *assertion*, a statement purported to be universally true, that is, true for all values of the variables appearing in it. The assertion may or may not be accompanied by a proof of its truth or by an English language argument of its plausibility. If no proof or plausibility argument is given, a claim is implied that the assertion is provable. It is to be accepted as true in the text and analysis following it.

A Boolean expression may be an *axiom*. The statement is not assumed to be universally true, but is accepted without proof for the subsequent analysis. An axiom is intended to restrict attention to situations in which it is true. Such a statement restricts the subsequent universe of discourse.

A Boolean expression may be a *proposition*, a *conjecture*, or a *theorem*. Such a statement is often proved to be universally true in the subsequent text. These Boolean expressions usually take the form of an implication, such as $A \Rightarrow B$ (if the Boolean expression A is true, then the Boolean expression B is true). The proof amounts to transforming the expression $A \Rightarrow B$ into the logical constant "true," demonstrating that the value of the expression is always true.

A Boolean expression may constitute the *definition* of some term used later frequently. For example, the equation sum(a, b)=a+b might serve to define the function named "sum" in terms of the already defined infix symbol +. More precisely, this definition of the function "sum" might be written as

$$[\wedge\, a, b : a \in \mathbb{R} \wedge b \in \mathbb{R} : \mathrm{sum}(a, b) = a + b] \qquad [6.12\text{-}1]$$

Finally, an expression (Boolean or otherwise) may simply represent a quantity, entity, or restriction in the application world whose value is unknown or unimportant in the context in question.

The different purposes of a Boolean expression are comparable to the different moods of a verb in English as outlined in Section 6.6.

The intended purpose of a Boolean expression is not usually clear from the mathematical expression alone. Sometimes the purpose is clear from the context. English language comments indicating the purpose often accompany such Boolean expressions. For these reasons, mathematical works are almost never written in the Language of Mathematics alone, but are almost always supplemented with English explanations and comments. Therefore, the translator will rarely, if ever, translate from mathematical expressions alone into English.

Translating mathematical expressions into English is a useful exercise in the process of learning and becoming fluent in the Language of Mathematics. In actual practice, one will sometimes translate an individual mathematical expression—or even a part thereof—in order to check a translation from English to mathematics, that is, to check that a mathematical expression was written correctly and conveys the intended low-level meaning. In these cases, the mathematical expression will usually be examined at a rather detailed level and without regard to the full context within which it appears.

6.13 MATHEMATICAL MODELS AND THEIR INTERPRETATION

A *mathematical model* is a collection of the following:

- Variables
- Values (constants)
- Expressions determining the relationship between the values of the model's variables

As stated in Section 1.5, a mathematical model is a statement of a problem and the requirements that any solution must satisfy.

The variables in a mathematical model are typically of two types: *independent variables*, sometimes called "input" values, and *dependent variables*, the "output" variables. If values are given for all the independent variables, the expressions in the mathematical model determine possible values of the dependent variables.

A mathematical model typically represents selected aspects of some part of the real world, commonly called the *application* or *application domain* or *application world*. The application in question could be, for example, a physical system such as the structure of a building or an automobile, or procedures in an organization such as those for receiving orders, supplying goods ordered, or billing. See Section 1.2 for a lengthier list of some applications.

The expressions referred to above that make up a mathematical model are typically equations, inequalities, and other Boolean expressions. Normally, all are to apply. Thus, the mathematical model consists of all the expressions referred to above combined with the logical "and" function. The mathematical model is, then, a single Boolean expression.

Part of the mathematical model should specify the range of values that can be assumed by each variable appearing in the model. This part of the model we call the *header*, and the rest of the model, the *body*. The header should ensure that the value of every function and subexpression appearing in the body of the model is properly defined, that is, that every argument of each function or subexpression is within the domain of the respective function or subexpression.

The *interpretation* of a mathematical model relates the variables, functions, and, where helpful, expressions appearing in a mathematical model to objects and entities in the application world. It defines the meaning and explains the significance of each variable in the model in terms of the application being modeled. Because the interpretation must refer to things in the application world, it is written in English supplemented with accepted terminology and jargon of the application world as appropriate. The interpretation forms a bridge between the mathematical model and the application world. The interpretation belongs to the area to which or in which mathematics is being applied, not to the world of mathematics itself. Because the interpretation is written in English, the potential problems of vagueness, ambiguity, imprecision, and so on, inherent in any natural language must be considered and taken into account.

As pointed out in Section 1.1, mathematics provides a template language, the Language of Mathematics, which must be adapted to every application. The interpretation of a mathematical model, as described in the preceding paragraph, constitutes that adaptation. Different applications will generally require different interpretations.

The mathematical model itself is a collection of mathematical functions and expressions. These should be "uninterpreted," but this does not mean or imply that they should be "uninterpretable." The various functions and expressions can be and often are interpreted in terms related to the application world when this helps the reader to understand the functions and expressions and to relate them to the "real" world. Mathematically, however, it should never be necessary to interpret the functions and expressions this way; the mathematical model should stand on its own logically and mathematically without the support of an interpretation in English. See also the end of Section 4.7 and Section 6.9.

The *assignment* specifies values for some of the variables appearing in the mathematical model. These values may be physical constants or parameters that can be changed to adapt the model to different particular individual applications or calculations (e.g., the number of articles in an inventory system, the number of elevators in an elevator system). The assignment of values to variables can be expressed in the form of equations, connected with the logical and ($\wedge$).

The *statement of the problem* can have various forms, depending on the problem to be solved. It normally includes expressions that relate and restrict the values of the variables. It sometimes includes an expression whose value is to be maximized or minimized subject to the restrictions.

A *solution to the problem* specifies the values of the variables not already assigned values in the assignment. These remaining values must satisfy the requirements in the statement of the problem. There may be more than one solution, exactly one solution, or no solution. In the last case, the "solution" is the statement that there is no solution.

Note that two translations of a mathematical model are possible. One is a literal translation, a direct restatement of the mathematical expressions into English. For example, a literal translation of the expression $d=v*t$ is "the value of the variable d is the product of the values of the variables v and t." Such a translation refers only to concepts in the Language of Mathematics. It contains no reference to anything in any application. Alone, it is of little, if any, use.

An example of a translation of the second type of the expression $d=v*t$ is "the distance in kilometers traveled by a vehicle moving at a constant speed of v kilometers per hour for a time t hours is the product of v and t." Most of this translation refers to things in the application domain, outside the Language of Mathematics. This translation amounts to a reformulation of the literal translation in the paragraph above combined with the interpretation of the relevant parts of the mathematical model. Those relevant parts of the interpretation of the mathematical model are: "the value of the variable d is the distance in kilometers traveled by a vehicle," "the value of the variable v is the constant speed in kilometers per hour of a vehicle” and “the value of the variable t is the time in hours during which a vehicle travels.”

Many variables in mathematical models represent numbers (i.e., quantities). In the interpretation, such variables are typically described by the nouns to which the numerical values refer. In the example above, these nouns are “distance,” “speed,” and “time.” Including the dimensions of these numbers, the descriptions become the noun phrases “distance in kilometers,” “speed in kilometers per hour,” and “time in hours.”

The values of variables and functions in mathematical models need not always be numbers. They are also often adjectives such as “male,” “female,” “blue,” and “brown.” They can also be nouns used as descriptions, such as “physician,” “teacher,” and “farmer.” In all such cases, they are specific instances of the generic noun describing the variable in the interpretation (e.g., “gender,” “color,” “vocation”).

The interpretation of a Boolean variable or function is typically a clause describing a state or a relationship. Possible examples are “x is greater than y,” “the gate is open,” and “the door is closing” (in the state/process of changing from open to closed). Such clauses often contain a conjugated form of the verb “to be” and a predicate adjective or predicate noun.

The comments above suggest that one should look especially for (1) nouns and noun phrases and (2) clauses with predicate adjectives or predicate nouns when formulating a mathematical model from an English description of the application. The topic of translating from English to a mathematical model is dealt with extensively in Chapter 7.

6.13.1 Dimensions of Numerical Variables

An old saying reminds us that “you can’t add apples and oranges.” Correspondingly, it makes no sense to add a length expressed in feet to a length expressed in meters, or to add a length in feet to an area in square feet. In each case, one can add the two numbers, but the resulting sum is neither a combined length in any units nor a length nor an area in any units; the sum is meaningless as a length or an area—it represents nothing in the physical world.

The *dimensions*, or *units*, associated with the variables and terms in a numerical expression must be dimensionally consistent if the value of the expression is to have any meaning in the application area in question.

The definitions of various physical quantities enable measures to be converted from one unit to another. For example, a kilometer (km) is defined to be 1000 meters (m), and there are 3600 seconds (s) in an hour (h). Therefore, if Skmh is the velocity of an object in km/h and Sms is the same velocity in m/s, the relationship between the two numbers Sms and Skmh is

$$\mathrm{Sms} = \mathrm{Skmh}*1000/3600 \qquad [6.13.1\text{-}1]$$

or

$$\mathrm{Sms} = \mathrm{Skmh}/3.6 \qquad [6.13.1\text{-}2]$$

or

$$3.6*\text{Sms} = \text{Skmh} \qquad [6.13.1\text{-}3]$$

Similarly, the Fahrenheit and Celsius scales for measuring temperature are defined in such a way that a temperature F measured in degrees Fahrenheit is the same as the temperature C measured in degrees Celsius, where F and C are related by the equation

$$(F-32)*5/9 = C \qquad [6.13.1\text{-}4]$$

As pointed out in Chapter 2 and Section 3.2, the interpretation of every numerical variable in a mathematical model must include the specification of the dimension (units) in which that variable's numerical value is expressed. These units are critical in all quantities in the physical world. The selection of a convenient system of units is an important consideration in the several areas of physics [e.g., the meter-kilogram-second (mks) system vs. the centimeter-gram-second (cgs) system], in chemistry, in medicine, and so on. A numerical measurement of a physical quantity is meaningful only when the units of measure are stated explicitly.

Note the symbols used to connect the names of units when the dimension of a number is a combination of two or more other dimensions. For example, if a velocity is given in meters per second, the dimensional abbreviation is, as in the example above, m/s. The amount of effort that 10 workers expend in 15 days is 10∗15, or 150 worker days. This is usually written worker-days, although worker∗days or worker·days would be more logical. In the written abbreviation for a dimensional combination, a minus sign (−) never occurs (because such a combination would be physically meaningless), so no ambiguity is introduced by using a hyphen (-) instead of a multiplication sign.

6.13.2 An Example of a Mathematical Model and Its Interpretation

In this section an example illustrates how an interpretation connects with each other:

- A mathematical model written in the Language of Mathematics
- A description of a problem to be solved written in English

This example of a mathematical model, its interpretation, and its English language translation involves a problem of mixing two components (acrinium and lodus) in different ratios to make two different products (protana and satea) in such a way that the available quantities of both components are used completely. The quantities of each component and product are measured in appropriate (and possibly different) units (e.g., liters, kilograms, milligrams). Constants in the model will convert the various units of measure as required.

The chemical properties of acrinium, lodus, protana, and satea are such that:

- Acrinium and lodus must be mixed in the ratio of 3 units of acrinium to 5 units of lodus to make protana, in which case 7 units of protana are produced.

- Acrinium and lodus must be mixed in the ratio of 11 units of acrinium to 6 units of lodus to make satea, in which case 19 units of satea are produced.

Normalizing the given ratios can reduce the chance of making an error when writing the corresponding mathematical expressions. The information above can be rewritten as follows:

- With each unit of acrinium mix 5/3 units of lodus; 7/3 units of protana will be produced.
- With each unit of acrinium mix 6/11 unit of lodus; 19/11 units of satea will be produced.

The **interpretation** of the variables in this model is:

AP:	the number of units of acrinium used to make protana
LP:	the number of units of lodus used to make protana
AS:	the number of units of acrinium used to make satea
LS:	the number of units of lodus used to make satea
P:	the number of units of protana produced
S:	the number of units of satea produced
QA:	the number of units of acrinium available
QL:	the number of units of lodus available

Note that each variable in this mathematical model is a number and that the variable corresponds to a noun phrase ("number of units of …") in English. This correspondence between a numerical variable in the Language of Mathematics and a noun or noun phrase in English is typical.

The **mathematical model** is:

$$\mathrm{QA}\in\mathbb{R} \wedge \mathrm{QL}\in\mathbb{R} \wedge \mathrm{AP}\in\mathbb{R} \wedge \mathrm{LP}\in\mathbb{R} \wedge \mathrm{AS}\in\mathbb{R} \wedge \mathrm{LS}\in\mathbb{R} \wedge \mathrm{P}\in\mathbb{R} \wedge \mathrm{S}\in\mathbb{R} \wedge \quad [6.13.2\text{-}1]$$

$$\mathrm{QA}\geq 0 \wedge \mathrm{QL}\geq 0 \wedge \mathrm{AP}\geq 0 \wedge \mathrm{LP}\geq 0 \wedge \mathrm{AS}\geq 0 \wedge \mathrm{LS}\geq 0 \wedge \mathrm{P}\geq 0 \wedge \mathrm{S}\geq 0 \wedge \quad [6.13.2\text{-}2]$$

$$\mathrm{LP} = (5/3)*\mathrm{AP} \wedge \mathrm{P} = (7/3)*\mathrm{AP} \wedge \quad [6.13.2\text{-}3]$$

$$\mathrm{LS} = (6/11)*\mathrm{AS} \wedge \mathrm{S} = (19/11)*\mathrm{AS} \wedge \quad [6.13.2\text{-}4]$$

$$\mathrm{AP} + \mathrm{AS} = \mathrm{QA} \wedge \mathrm{LP} + \mathrm{LS} = \mathrm{QL} \quad [6.13.2\text{-}5]$$

Lines 6.13.2-1 and 6.13.2-2 in the mathematical expression above constitute the header of the mathematical model, and the other lines constitute the body of the model (see the definitions of header and body in Section 6.13).

The mathematical model above can be translated into English line by line and term by term, interpreting the variables appropriately:

$$\mathrm{QA}\in\mathbb{R} \wedge \mathrm{QL}\in\mathbb{R} \wedge \mathrm{AP}\in\mathbb{R} \wedge \mathrm{LP}\in\mathbb{R} \wedge \mathrm{AS}\in\mathbb{R} \wedge \mathrm{LS}\in\mathbb{R} \wedge \mathrm{P}\in\mathbb{R} \wedge \mathrm{S}\in\mathbb{R} \wedge$$

[6.13.2-1 repeated]

$$\mathrm{QA}\geq 0 \wedge \mathrm{QL}\geq 0 \wedge \mathrm{AP}\geq 0 \wedge \mathrm{LP}\geq 0 \wedge \mathrm{AS}\geq 0 \wedge \mathrm{LS}\geq 0 \wedge \mathrm{P}\geq 0 \wedge \mathrm{S}\geq 0 \wedge$$

[6.13.2-2 repeated]

can be translated to "The variables QA, QL, AP, LP, AS, LS, P and S are real numbers. They are all greater than or equal to zero."

Including the English interpretations of the variable names, this English text can be reformulated as "The number of units of acrinium available (QA), of lodus available (QL), of acrinium used to produce protana (AP), of lodus used to make protana (LP), of acrinium used to make satea (AS), of lodus used to make satea (LS), of protana produced (P), and of satea produced (S) are all nonnegative real numbers."

Next, line 6.13.2-3 is translated:

$$\mathrm{LP} = (5/3)*\mathrm{AP} \wedge \mathrm{P} = (7/3)*\mathrm{AP} \wedge \qquad \text{[6.13.2-3 repeated]}$$

"The number of units of acrinium times 5/3 is the number of units of lodus required to make protana. The number of units of protana produced is then 7/3 times the number of units of acrinium used."

This paragraph can be reformulated as "With every 3 units of acrinium mix 5 units of lodus to make 7 units of protana."

Next, line 6.13.2-4 is translated:

$$\mathrm{LS} = (6/11)*\mathrm{AS} \wedge \mathrm{S} = (19/11)*\mathrm{AS} \wedge \qquad \text{[6.13.2-4 repeated]}$$

"The number of units of acrinium times 6/11 is the number of units of lodus required to make satea. The number of units of satea produced is then 19/11 times the number of units of acrinium used."

This, in turn, can be rewritten as "With every 11 units of acrinium mix 6 units of lodus to make 19 units of satea."

Finally, we translate line 6.13.2-5 in the mathematical model above:

$$\mathrm{AP} + \mathrm{AS} = \mathrm{QA} \wedge \mathrm{LP} + \mathrm{LS} = \mathrm{QL} \qquad \text{[6.13.2-5 repeated]}$$

"The sum of the quantity of acrinium used to make protana and the quantity of acrinium used to make satea must be equal to the quantity of acrinium available. The sum of the quantity of lodus used to make protana and the quantity of lodus used to make satea must be equal to the quantity of lodus available."

This can be rewritten in a more typical style as "The total quantities of acrinium and lodus used to make both protana and satea must be equal to the quantities of the components available" or even "The quantities of acrinium and lodus available must be used completely to make protana and satea."

The complete translation of the mathematical model into English is, then, "With every 3 units of acrinium mix 5 units of lodus to make 7 units of protana. With every 11 units of acrinium mix 6 units of lodus to make 19 units of satea. The quantities of acrinium and lodus available must be used completely to make protana and satea." The sentence "The values of the variables QA, QL, AP, LP, AS, LS, P, and S are real numbers" would typically be left out of the translation because the names of the variables are not otherwise mentioned explicitly in the translation and because the quantities of the components and products are implicitly numbers.

Notice that, first, the individual terms in the mathematical model were translated literally from the Language of Mathematics into English. The resulting relatively stilted style was then rephrased into a more typical natural language style, sometimes in more than one step. With practice and experience, one can perform these translational steps mentally, without writing them all down. After completing the translation, one should compare it with the original formulas in the mathematical model to verify that the translation reflects the meaning of the mathematical model. Some precision will be lost and some ambiguity will be introduced, due to the nature of every natural language, but this loss of precision and introduction of ambiguity should be held to an acceptable level. The translator's goal is an English text that will convey to the intended readers an accurate understanding of the meaning of the mathematical expressions.

Notice again that the variables in this model have numerical values and are interpreted in English as noun phrases referring to certain quantities. This is typical for numerical variables in mathematical models in general. Such quantities can be lengths, weights, or volumes of materials; the strength of electrical, magnetic, or gravitational fields; velocities; acceleration; the strengths of various types of radiation; earthquakes; and so on.

The **assignment** of values to variables is

$$\mathrm{QA} = 18 \qquad [6.13.2\text{-}6]$$

$$\mathrm{QL} = 19 \qquad [6.13.2\text{-}7]$$

The variables QA and QL are the *input* variables in this problem, and the remaining variables, AP, LP, AS, LS, P, and S, are the *output* variables.

The **statement of the problem** is, in mathematically oriented terms: Determine the values of the variables AP, LP, AS, LS, P, and S.

That is, the purpose of this example of a mathematical model is to determine how to mix the two components to make the two products so that the components available are used completely. That is, the task is to determine values for the variables AP, LP, AS, LS, P, and S that satisfy the expression—the mathematical model—above, given the values for the variables listed in the assignment section above.

Solving the Problem More precisely, we are to determine values for the variables such that the value of the expression

$$\mathrm{QA}\in\mathbb{R} \wedge \mathrm{QL}\in\mathbb{R} \wedge \mathrm{AP}\in\mathbb{R} \wedge \mathrm{LP}\in\mathbb{R} \wedge \mathrm{AS}\in\mathbb{R} \wedge \mathrm{LS}\in\mathbb{R} \wedge \mathrm{P}\in\mathbb{R} \wedge \mathrm{S}\in\mathbb{R} \wedge \qquad [6.13.2\text{-}1 \text{ repeated}]$$

$$\mathrm{QA}\geq 0 \wedge \mathrm{QL}\geq 0 \wedge \mathrm{AP}\geq 0 \wedge \mathrm{LP}\geq 0 \wedge \mathrm{AS}\geq 0 \wedge \mathrm{LS}\geq 0 \wedge \mathrm{P}\geq 0 \wedge \mathrm{S}\geq 0 \wedge \qquad [6.13.2\text{-}2 \text{ repeated}]$$

$$\mathrm{LP} = (5/3)*\mathrm{AP} \wedge \mathrm{P} = (7/3)*\mathrm{AP} \wedge \qquad [6.13.2\text{-}3 \text{ repeated}]$$

$$\mathrm{LS} = (6/11)*\mathrm{AS} \wedge \mathrm{S} = (19/11)*\mathrm{AS} \wedge \qquad [6.13.2\text{-}4 \text{ repeated}]$$

$$\mathrm{AP} + \mathrm{AS} = \mathrm{QA} \wedge \mathrm{LP} + \mathrm{LS} = \mathrm{QL} \wedge \qquad [6.13.2\text{-}5 \text{ repeated}]$$

$$\mathrm{QA} = 18 \wedge \mathrm{QL} = 19 \qquad [6.13.2\text{-}8]$$

is true.

One way to solve this problem is to transform the mathematical expression above into a form in which every variable appears on the left side of an equation and only a numerical value appears on its right side. In each transformation step, we substitute one subexpression by another subexpression having the same value, thus preserving the value of the entire expression. In some cases, we substitute a variable name by a number.

Line 6.13.2-8 of the mathematical expression above is already in the desired form, so we need not change it. Noting that the value of the variable QA is 18 and that the value of the variable QL is 19, we can substitute these values for QA and QL elsewhere in the body of the mathematical model, obtaining

$$\mathrm{QA}\in\mathbb{R} \wedge \mathrm{QL}\in\mathbb{R} \wedge \mathrm{AP}\in\mathbb{R} \wedge \mathrm{LP}\in\mathbb{R} \wedge \mathrm{AS}\in\mathbb{R} \wedge \mathrm{LS}\in\mathbb{R} \wedge \mathrm{P}\in\mathbb{R} \wedge \mathrm{S}\in\mathbb{R} \wedge \quad \text{[6.13.2-1 repeated]}$$

$$\mathrm{QA}\geq 0 \wedge \mathrm{QL}\geq 0 \wedge \mathrm{AP}\geq 0 \wedge \mathrm{LP}\geq 0 \wedge \mathrm{AS}\geq 0 \wedge \mathrm{LS}\geq 0 \wedge \mathrm{P}\geq 0 \wedge \mathrm{S}\geq 0 \wedge \quad \text{[6.13.2-2 repeated]}$$

$$\mathrm{LP} = (5/3)*\mathrm{AP} \wedge \mathrm{P} = (7/3)*\mathrm{AP} \wedge \quad \text{[6.13.2-3 repeated]}$$

$$\mathrm{LS} = (6/11)*\mathrm{AS} \wedge \mathrm{S} = (19/11)*\mathrm{AS} \wedge \quad \text{[6.13.2-4 repeated]}$$

$$\mathrm{AP} + \mathrm{AS} = 18 \wedge \mathrm{LP} + \mathrm{LS} = 19 \wedge \quad \text{[6.13.2-9]}$$

$$\mathrm{QA} = 18 \wedge \mathrm{QL} = 19 \quad \text{[6.13.2-8 repeated]}$$

If the values of AP and AS were known, the values of all the other variables LP, P, LS, and S would follow directly from lines 6.13.2-3 and 6.13.2-4 of the expression. Therefore, we transform those equations not already in the desired form (variable name = a number or an expression involving only AP and/or AS) so that only AP and AS appear in them. Only the equations in line 6.13.2-9 are not already in this form, so only they need to be changed.

Our goal now is to transform the two equations in line 6.13.2-9 into the forms AP=… and AS=… by suitable manipulations. We begin by substituting equivalent expressions for LP and LS in the second equation in order to transform that equation into a form involving only the variables AP and AS. The result is

$$\mathrm{QA}\in\mathbb{R} \wedge \mathrm{QL}\in\mathbb{R} \wedge \mathrm{AP}\in\mathbb{R} \wedge \mathrm{LP}\in\mathbb{R} \wedge \mathrm{AS}\in\mathbb{R} \wedge \mathrm{LS}\in\mathbb{R} \wedge \mathrm{P}\in\mathbb{R} \wedge \mathrm{S}\in\mathbb{R} \wedge \quad \text{[6.13.2-1 repeated]}$$

$$\mathrm{QA}\geq 0 \wedge \mathrm{QL}\geq 0 \wedge \mathrm{AP}\geq 0 \wedge \mathrm{LP}\geq 0 \wedge \mathrm{AS}\geq 0 \wedge \mathrm{LS}\geq 0 \wedge \mathrm{P}\geq 0 \wedge \mathrm{S}\geq 0 \wedge \quad \text{[6.13.2-2 repeated]}$$

$$\mathrm{LP} = (5/3)*\mathrm{AP} \wedge \mathrm{P} = (7/3) * \mathrm{AP} \wedge \quad \text{[6.13.2-3 repeated]}$$

$$\mathrm{LS} = (6/11)*\mathrm{AS} \wedge \mathrm{S} = (19/11)*\mathrm{AS} \wedge \quad \text{[6.13.2-4 repeated]}$$

$$\mathrm{AP} + \mathrm{AS} = 18 \wedge (5/3)*\mathrm{AP} + (6/11)*\mathrm{AS} = 19 \wedge \quad \text{[6.13.2-10]}$$

$$\mathrm{QA} = 18 \wedge \mathrm{QL} = 19 \quad \text{[6.13.2-8 repeated]}$$

We can change the equalities in line 6.13.2-10 into the forms AP=… and AS=… by subtracting appropriately from each side of each equation to obtain

$QA \in \mathbb{R} \wedge QL \in \mathbb{R} \wedge AP \in \mathbb{R} \wedge LP \in \mathbb{R} \wedge AS \in \mathbb{R} \wedge LS \in \mathbb{R} \wedge P \in \mathbb{R} \wedge S \in \mathbb{R} \wedge$ [6.13.2-1 repeated]

$QA \geq 0 \wedge QL \geq 0 \wedge AP \geq 0 \wedge LP \geq 0 \wedge AS \geq 0 \wedge LS \geq 0 \wedge P \geq 0 \wedge S \geq 0 \wedge$ [6.13.2-2 repeated]

$LP = (5/3)*AP \wedge P = (7/3)*AP \wedge$ [6.13.2-3 repeated]

$LS = (6/11)*AS \wedge S = (19/11)*AS \wedge$ [6.13.2-4 repeated]

$AP = 18-AS \wedge (6/11)*AS = 19 - (5/3)*AP \wedge$ [6.13.2-11]

$QA = 18 \wedge QL = 19$ [6.13.2-8 repeated]

We must now combine the two equations in line 6.13.2-11 to effectively solve for the values of AP and AS. The first equality in line 6.13.2-11 states that the value of AP is the same as the value of the subexpression (18−AS). Therefore, we substitute (18−AS) for the variable AP in the second equality in line 6.13.2-11, obtaining an equality involving only AS

$QA \in \mathbb{R} \wedge QL \in \mathbb{R} \wedge AP \in \mathbb{R} \wedge LP \in \mathbb{R} \wedge AS \in \mathbb{R} \wedge LS \in \mathbb{R} \wedge P \in \mathbb{R} \wedge S \in \mathbb{R} \wedge$ [6.13.2-1 repeated]

$QA \geq 0 \wedge QL \geq 0 \wedge AP \geq 0 \wedge LP \geq 0 \wedge AS \geq 0 \wedge LS \geq 0 \wedge P \geq 0 \wedge S \geq 0 \wedge$ [6.13.2-2 repeated]

$LP = (5/3)*AP \wedge P = (7/3)*AP \wedge$ [6.13.2-3 repeated]

$LS = (6/11)*AS \wedge S = (19/11)*AS \wedge$ [6.13.2-4 repeated]

$AP = 18-AS \wedge (6/11)*AS = 19 - (5/3)*(18-AS) \wedge$ [6.13.2-12]

$QA = 18 \wedge QL = 19$ [6.13.2-8 repeated]

Transforming the second equality in line 6.13.2-12 into the form AS=… yields

$QA \in \mathbb{R} \wedge QL \in \mathbb{R} \wedge AP \in \mathbb{R} \wedge LP \in \mathbb{R} \wedge AS \in \mathbb{R} \wedge LS \in \mathbb{R} \wedge P \in \mathbb{R} \wedge S \in \mathbb{R} \wedge$ [6.13.2-1 repeated]

$QA \geq 0 \wedge QL \geq 0 \wedge AP \geq 0 \wedge LP \geq 0 \wedge AS \geq 0 \wedge LS \geq 0 \wedge P \geq 0 \wedge S \geq 0 \wedge$ [6.13.2-2 repeated]

$LP = (5/3)*AP \wedge P = (7/3)*AP \wedge$ [6.13.2-3 repeated]

$LS = (6/11)*AS \wedge S = (19/11)*AS \wedge$ [6.13.2-4 repeated]

$\mathrm{AP} = 18-\mathrm{AS} \wedge \mathrm{AS} = 363/37 \wedge$ [6.13.2-13]

$\mathrm{QA} = 18 \wedge \mathrm{QL} = 19$ [6.13.2-8 repeated]

The numerical value 363/37 can be substituted for AS everywhere that it appears on the right side of an equation, giving

$\mathrm{QA}\in\mathbb{R} \wedge \mathrm{QL}\in\mathbb{R} \wedge \mathrm{AP}\in\mathbb{R} \wedge \mathrm{LP}\in\mathbb{R} \wedge \mathrm{AS}\in\mathbb{R} \wedge \mathrm{LS}\in\mathbb{R} \wedge \mathrm{P}\in\mathbb{R} \wedge \mathrm{S}\in\mathbb{R} \wedge$

[6.13.2-1 repeated]

$\mathrm{QA}\geq 0 \wedge \mathrm{QL}\geq 0 \wedge \mathrm{AP}\geq 0 \wedge \mathrm{LP}\geq 0 \wedge \mathrm{AS}\geq 0 \wedge \mathrm{LS}\geq 0 \wedge \mathrm{P}\geq 0 \wedge \mathrm{S}\geq 0 \wedge$

[6.13.2-2 repeated]

$\mathrm{LP} = (5/3)*\mathrm{AP} \wedge \mathrm{P} = (7/3)*\mathrm{AP} \wedge$ [6.13.2-3 repeated]

$\mathrm{LS} = (6/11)*(363/37) \wedge \mathrm{S} = (19/11)*(363/37) \wedge$ [6.13.2-14]

$\mathrm{AP} = 18-(363/37) \wedge \mathrm{AS} = 363/37 \wedge$ [6.13.2-15]

$\mathrm{QA} = 18 \wedge \mathrm{QL} = 19$ [6.13.2-8 repeated]

and, performing the indicated arithmetic, we have

$\mathrm{QA}\in\mathbb{R} \wedge \mathrm{QL}\in\mathbb{R} \wedge \mathrm{AP}\in\mathbb{R} \wedge \mathrm{LP}\in\mathbb{R} \wedge \mathrm{AS}\in\mathbb{R} \wedge \mathrm{LS}\in\mathbb{R} \wedge \mathrm{P}\in\mathbb{R} \wedge \mathrm{S}\in\mathbb{R} \wedge$

[6.13.2-1 repeated]

$\mathrm{QA}\geq 0 \wedge \mathrm{QL}\geq 0 \wedge \mathrm{AP}\geq 0 \wedge \mathrm{LP}\geq 0 \wedge \mathrm{AS}\geq 0 \wedge \mathrm{LS}\geq 0 \wedge \mathrm{P}\geq 0 \wedge \mathrm{S}\geq 0 \wedge$

[6.13.2-2 repeated]

$\mathrm{LP} = (5/3)*\mathrm{AP} \wedge \mathrm{P} = (7/3)*\mathrm{AP} \wedge$ [6.13.2-3 repeated]

$\mathrm{LS} = 198/37 \wedge \mathrm{S} = 627/37 \wedge$ [6.13.2-16]

$\mathrm{AP} = 303/37 \wedge \mathrm{AS} = 363/37 \wedge$ [6.13.2-17]

$\mathrm{QA} = 18 \wedge \mathrm{QL} = 19$ [6.13.2-8 repeated]

Finally, substituting the value 303/37 for AP everywhere that it appears on the right side of an equation and performing the arithmetic indicated yields the complete solution desired.

The **solution to the problem** is, therefore,

$\mathrm{QA}\in\mathbb{R} \wedge \mathrm{QL}\in\mathbb{R} \wedge \mathrm{AP}\in\mathbb{R} \wedge \mathrm{LP}\in\mathbb{R} \wedge \mathrm{AS}\in\mathbb{R} \wedge \mathrm{LS}\in\mathbb{R} \wedge \mathrm{P}\in\mathbb{R} \wedge \mathrm{S}\in\mathbb{R} \wedge$

[6.13.2-1 repeated]

$\mathrm{QA}\geq 0 \wedge \mathrm{QL}\geq 0 \wedge \mathrm{AP}\geq 0 \wedge \mathrm{LP}\geq 0 \wedge \mathrm{AS}\geq 0 \wedge \mathrm{LS}\geq 0 \wedge \mathrm{P}\geq 0 \wedge \mathrm{S}\geq 0 \wedge$

[6.13.2-2 repeated]

$LP = 505/37 \wedge P = 707/37 \wedge$ [6.13.2-18]

$LS = 198/37 \wedge S = 627/37 \wedge$ [6.13.2-16 repeated]

$AP = 303/37 \wedge AS = 363/37 \wedge$ [6.13.2-17 repeated]

$QA = 18 \wedge QL = 19$ [6.13.2-8 repeated]

This solution can be translated from the Language of Mathematics into English to obtain

"The 18 units of acrinium and the 19 units of lodus should be used to produce 707/37 (approximately 19.108) units of protana and 627/37 (approximately 16.946) units of satea. To do this, 303/37 (approximately 8.189) units of acrinium and 505/37 (approximately 13.649) units of lodus will be used to produce the protana, while 363/37 (approximately 9.811) units of acrinium and 198/37 (approximately 5.351) units of lodus will be used to produce the satea."

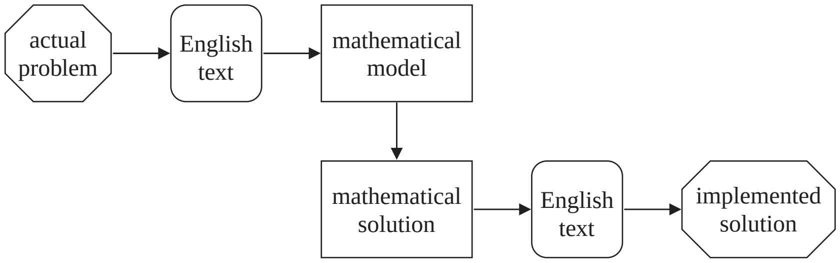
This example is a simple form of a class of industrially important optimization problems. These problems deal with refining and/or mixing components with the goal of maximizing the total value of the various products produced. Refining crude oil into various types of fuels and chemicals and mixing animal feeds are classical examples of this class of optimization problem. Such problems were among the early industrial applications of computers.

In the example in this section, the mathematical problem involved solving a set of linear equations in the same number of variables. Here, the initial form was particularly convenient, and a relatively simple approach led to the solution. This approach is generalized in the mathematical literature, enabling larger problems with less convenient structures to be solved in a standard and straightforward—although often computationally more intensive—way.

7 Translating English to Mathematics

Before translating an English text into the Language of Mathematics one should be clear why a translation is needed at all. An English text is translated into another natural language so that a person not fluent in English can read and understand it. This reason does not apply to translations into the Language of Mathematics. A translation in the Language of Mathematics is useful for a very different purpose: to reason logically about a problem; to analyze it systematically, accurately, and precisely; to find a solution; and to provide a basis for translating the solution back into an English description for those implementing and utilizing the solution in practice.

As pointed out in Chapter 5, the steps in the overall process of going from the original actual problem to a mathematically founded solution are illustrated in the following diagram.



This chapter deals with the second step in this process: translating the English text to a mathematical model. Although the most severe problems regarding accuracy, discrepancies, and errors have their origins in the step translating the English text to the mathematical model, problems can arise in any and all of the steps above. Issues of accuracy, discrepancies, errors, oversights, and other problems in all steps are considered in this chapter, in particular, in Section 7.8.

The first two steps in the process illustrated above are especially prone to problems arising from ambiguity and unrecognized incompleteness and omissions. No reliable method exists to verify that the results of these first two steps are consistent and complete, mainly because the meaning of a natural language text (such as English) is, itself, almost always ambiguous. The meaning of the mathematical model is precisely and unambiguously defined, so it provides a reference for verifying the consistency and completeness of subsequent formulations, provided, of course, that they are themselves defined precisely.

When translating from one natural language to another, the goal is to express the same meaning in the target language as in the source language. When translating from English to the Language of Mathematics, however, this goal is necessarily somewhat different because of fundamental differences between these languages' universes of discourse. These differences are mainly in the areas of ambiguity and abstraction and are discussed in more detail in Section 7.1.

Sometimes a mathematical model is needed only to analyze or investigate certain properties of a system, process, or object, not to solve a particular problem. Then only the English text and the mathematical model shown in the diagram above are necessary; the remaining steps leading to a solution are not needed. The translation from an English text to a mathematical model, the subject of this chapter, remains.

7.1 GENERAL CONSIDERATIONS

The overall process for translating from English to the Language of Mathematics involves the following:

- Understanding thoroughly what the author(s) of the English text *meant*
- Formulating that meaning in:
  - Mathematical expressions in the Language of Mathematics
  - An interpretation linking the values, variables, functions, and, where relevant, subexpressions in the mathematical model with the various parts of the English text

To help the reader to do this, many useful guidelines for translating are given below. Although they are usually applicable, not all are universal laws, and they should not be regarded as instructions for translating the English text mechanistically or automatically. The translator must understand the original text correctly and completely and be able to express that understanding in the Language of Mathematics as adapted by the translator in the interpretation.

Translating written material successfully from any source language into any target language (whether or not the Language of Mathematics is one of these languages) is a multistep (not single) process:

1. Read the text in the source language.
2. Understand (internalize) the meaning of that text in the source language and in the context of the subject of the message.
3. Understand (internalize) that meaning in terms of the target language.

4. Write something in the target language so that a reader of the text in the target language will interpret it to mean the same thing that a reader of the text in the source language will interpret it to mean.

After one believes that the translation is complete, one must verify that the meanings of the texts in the source and target languages are the same:

5. Reread and understand (interpret again) the text in the source language.
6. Read and understand (interpret anew) the translated version in the target language.
7. Compare the meanings of the two.

If any discrepancies are discovered, they must be resolved by repeating some or all of the foregoing activities as appropriate. An important discrepancy arises if in either step 1 or 2, it becomes evident that the original text is inconsistent, incomplete, clearly wrong, or makes no sense. One should always be sensitive to these possibilities, as they arise in English texts more often than one might expect. Often, important information is implied by the context of the English text. Such implied information must be explicitly expressed in a mathematical model.

Note especially that "translating" is *not* a matter of going immediately from activity 1 above to activity 4. Activities 2 and 3—understanding in the two different contexts—is essential. Often, iteration is required to complete activities 2, 3, and 4. An independent check is also needed (activities 5, 6, and 7). If at all possible, one should wait awhile between activities 4 and 5, because as long as the material is fresh in the translator's mind, the brain will often fill in missing detail and subconsciously correct errors present. It is also desirable that a different person perform activities 5, 6, and 7 than the person who performed activities 1, 2, 3, and 4.

The English word *translate* is often used to describe this process, but "translate" suggests an oversimplified view of what actually must take place. The word *interpret* is more descriptive of activities 2 and 3—understanding the meaning of the message before writing it in the target language. In fact, professionals who "translate" from one language to another, especially simultaneously and verbally, are often called "interpreters," not "translators," and their work is often called "interpreting," not "translating."

Attempts to translate material from a source language to a target language without going through the intermediate activity of understanding are typically unsuccessful. The true meaning of the original is not conveyed properly, completely, adequately, and correctly in the target language. Machine translations often exhibit discrepancies attributable to the lack of a real understanding of the meanings of the statements in the two languages.

Another consequence of the observation above that understanding is necessary is that fluency in both the original and target languages is not enough for successful translation. The translator must be able to *understand* the material in the original language in its context. Therefore, if, for example, one is to translate a scientific article in a journal on chemistry from French into English, one must be fluent in both French and English and, in addition, one must have at least a sufficient knowledge of chemistry. Furthermore, knowledge of translating as an activity itself is also necessary for efficient work.

Each language has its own world view, cultural context, style, and idiosyncracies. These differ from language to language. Mathematics and natural languages differ considerably in this regard. Pay close attention to these differences when translating.

Certain terms or phrases in one language often correspond to particular terms or phrases in another language. People who frequently translate between languages often collect a set of such corresponding pairs of terms or phrases and record them in their *translator's glossary*. Appendix G contains the beginnings of such a glossary for English and the Language of Mathematics. Readers should expand upon it in their own areas of expertise and translating activity.

In summary, in order to interpret English into the Language of Mathematics, one must:

- Be fluent in English
- Be fluent in the Language of Mathematics
- Be familiar with mathematics
- Know the subject about which you are interpreting statements from English into the Language of Mathematics
- Be familiar with certain terms and phrases typically corresponding with each other in English and the Language of Mathematics
- Be consciously aware of the special aspects of the process of interpreting from one language into another, especially into the Language of Mathematics

English and other natural languages can express:

- Both static and dynamic views, with the concept of time
- States and actions (having verbs of being and action)

while the conceptual world of the Language of Mathematics is limited to:

- A static view only with no concept of time
- States of being but no actions (having stative verbs only)

These are fundamental conceptual differences between the languages. It is not difficult to bridge these gaps when translating, but translators must pay conscious attention to them.

Some approaches to overcoming difficulties in translating are outlined below. Examples are given later in this chapter and in the examples in Chapter 8.

- Reduce the gaps noted above by reformulating the source text into a form closer to the Language of Mathematics. For example, reformulate clauses with action verbs into clauses with stative verbs only. One technique for this is to convert verbs and verbal phrases of actions to noun phrases (e.g., using participles or gerunds). The noun phrases are candidates for variables or functions in the mathematical model.

- Divide and conquer. Subdivide a complex task into simpler components. Continue hierarchically as necessary. Modularize the problem.
- Understand complex source text by constructing a specific instance of a general problem, drawing a diagram, examining special cases, or enumerating explicitly all possible situations that can arise (e.g., with a table).
- Identify background information in the source text not needed in the mathematical model. This will typically be information relevant in the application domain but:
  - Irrelevant or meaningless in the mathematical domain
  - Outlining the environment of the problem area
  - Explaining the motivation for solving the problem
  - Outside the universe of discourse of the Language of Mathematics, or
  - Not needed to describe or solve the problem
- Identify implicit information (e.g., by context), verify that such information can be assumed, and express it explicitly in the mathematical model.
- Identify and ask the author(s) for missing information.
- Identify false “information” and ask the author(s) for corrections.
- Identify vague and ambiguous statements in the English text and clarify them.
- Identify essential objects in the English text (especially in nouns and noun phrases) and the relationships between them.
- Close the gap between the universes of discourse of English and the Language of Mathematics by formulating the interpretation defining the meanings of the values, variables, functions, and other appropriate parts of the mathematical model in terms of the application domain.

Translating from English to mathematics differs in particular and important ways from translating between two natural languages. When translating from one natural language to another, the goal is to express the same meaning in the target language as in the source language, including the *same degree of ambiguity*. When translating into the Language of Mathematics, however, the intention is to end with a precise, *unambiguous* statement that expresses the intended meaning of the source text. Therefore, ambiguity in the original English text (and ambiguity is present in essentially every natural language text) must be identified, resolved, and eliminated before the mathematical text can be formulated. People not accustomed to working with an unambiguous language such as the Language of Mathematics will find eliminating ambiguity to be an extremely difficult task, simply because they are completely unaccustomed to such a way of thinking. The effort to eliminate ambiguity will often appear to them to be unnecessarily pedantic. They will often be surprised about the ambiguities found in the source text, ambiguities they would otherwise never have noticed. They must remind themselves continually that the Language of Mathematics is used in such situations precisely because it cannot express ambiguity, because it forces one to identify, resolve, and eliminate ambiguity present in the original English text. This, in turn, prevents subsequent errors in reasoning, analysis, and design.

A simple example of an English text containing some of the discrepancies mentioned above is the sentence “A jet airplane flies faster than a propeller airplane.” The basic form “fast” of the comparative adverb “faster” indicates that speed is the characteristic of the two airplanes being compared. The action verb “flies” together with its adverb of comparison “faster” can be eliminated by reformulating this sentence to “The speed of a jet airplane is greater than the speed of a propeller airplane.” The noun phrase “speed of a jet airplane” can be represented by a variable in the mathematical model (e.g., “SpeedJetAirplane”). Similarly, “speed of a propeller airplane” can be represented by another variable (e.g., “SpeedPropellerAirplane”). The corresponding part of the mathematical model is, then,

$$\text{SpeedJetAirplane} > \text{SpeedPropellerAirplane} \qquad [7.1\text{-}1]$$

When one starts to formulate in detail the interpretation of the two variables, a question arises regarding what speed is meant. The actual current speeds of one of each type of airplane are presumably not meant, because those speeds depend on their current use (e.g., whether in flight or parked on the ground), not on the types of their engines. Presumably, some speed in their specification is meant (e.g., “maximum speed,” “cruising speed”). Such additional detail should be included in interpretations of the variables in question. Furthermore, it is not clear from the English text *which* jet airplane(s) and *which* propeller airplane(s) are meant—any? every? all? each? some? Depending on precisely what the English sentence is intended to mean, a quantified expression over all elements of the set of jet airplanes and all elements of the set of propeller airplanes might be a more appropriate translation in the Language of Mathematics.

Another difference between translating between natural languages and from English to mathematics relates to abstraction. This involves generalizing, extracting essential characteristics, and eliminating nonessential aspects. Constructing a mathematical model always involves abstracting characteristics of the actual problem in some way. Abstracting can be done either when writing or reformulating the English text or when constructing the mathematical model from the English text. Part of the latter step will be formulating the interpretation of the functions, variables, and values in the mathematical model in terms of the application domain.

The interpretation itself represents an abstraction. The mathematical model represents the ultimate abstract formulation of the English text. The mathematical model itself includes all essential logical relationships between the values of the various variables, but excludes any and all references to the “outside” world of the application. Those references are contained only in the interpretation of the mathematical model. For example, the expression $x=y*z$ is an abstraction both of the relationship between distance (x), velocity (y), and time (z) of a moving object and of the relationship between voltage (x), current (y), and resistance (z) for an electrical resistor. These two very different things in the practical world can be represented by the same abstract relationship in the mathematical world. In engineering applications, this type of abstraction arises often. The same differential equation can describe either an electrical circuit or a combination of mechanical components, such as a mass, a spring, and a damper (shock absorber, dashpot), or the radioactive decay of a chemical element, or many other applications.

English text often contains explanatory phrases, clauses, or sentences that are redundant as far as the meaning is concerned. They may help a reader to understand the intentions or background, but do not contribute to the actual logical or technical content. Check for such unneeded passages in the English text by deleting each suspicious one and asking the question: "Does deleting or inserting the passage in question change the actual meaning of the text in any essential way?" If the answer is "no," the passage in question can usually be disregarded for the purpose of translating the English text into the Language of Mathematics. Such redundant passages need not be examined for phrases suggesting variables or functions to be included in the mathematical model.

What one can say in the Language of Mathematics is much more restricted than what one can say in any natural language. Natural languages have evolved so that they can all express more or less the same things, but the Language of Mathematics is very different in this regard. Many things that can be expressed in English and other natural languages cannot be expressed directly in the Language of Mathematics at all. When translating from English to the Language of Mathematics, the translator must define anew appropriate mathematical representations of the meanings of the English text. In the example above of jet and propeller airplanes, the notion of "speed" has been represented by each of the two variables. In the Language of Mathematics, they are variables with numerical values, nothing more. Their names, consisting of particular but arbitrary sequences of symbols, are meaningless in the mathematical realm. Whether or not these variables represent anything in the "real" world, and if so what, are questions outside the Language of Mathematics and outside the mathematical model. Their interpretations in terms of speeds of airplanes are completely outside the realm of mathematics; these meanings are exclusively in the minds of the people using mathematics as an aid in some analysis.

7.2 SENTENCES OF THE FORM "… IS (A) …" (SINGULAR FORMS)

At first glance, many beginners tend to translate "is" as "=." Only if the verb "is" in the clause in question means "is equal to" (or sometimes "is the same as") is this correct. Often, "is" means something else and the translation will be quite different.

The verb "is" can have many meanings and uses, among which the following occur commonly in texts upon which mathematical models are based:

- "Is" expresses *membership* in a category or class (i.e., in a set in the Language of Mathematics). In this case, the translation will usually indicate set membership (e.g., "$\ldots \in \ldots$"). An example is "Sarah is a student," which can be translated as "Sarah$\in\mathbb{Students}$."

- "Is" is part of a phrase stating a *relation* (e.g., "is greater than," "is older than," "is heavier than"). In this case, the translation will typically express the relation directly (e.g., with "<," ">"). Functions may be required to convert the identification of the objects in question to suitable numbers; for example, "George is heavier than Sam" can be translated into "weight(George)>weight(Sam)." If the values of the expressions on each side of the relational operator are not numbers, the relation can be expressed as a Boolean function [e.g., "IsOlder(George, Sam)"]), but even here, numerical measurements (e.g., of body weights) are the basis for the relation.
- "Is" precedes a *predicate adjective or noun*, often expressing a characteristic of the object in question. Such a clause can be translated by an appropriate function applied to the subject of the clause, an equals sign (=), and the value associated with the characteristic in question [e.g., "The shirt is green" can be translated by "ColorOf(the shirt)=green"].
- "Is" is followed by a phrase indicating the *state* of the subject of the clause. A clause with "is" in this sense is often translated by a Boolean variable, whose name identifies the subject of the clause and its state. In the example in Section 8.13, the entire sentence "The door is in its fully open position" is translated by the variable named "DoorIsOpen."
- "Is" occurs in sentences providing background information or explanations that are not to be translated into specific expressions in the mathematical model.

Note that the categories of meanings of "is" above are not mutually exclusive. The sentence "The door is in its fully open position" above or, more simply, "The door is open" can also be viewed as a sentence of the predicate adjective type, in which case one might translate it as "Position(Door)=open." The decision to translate such a sentence in one way or another is a question of writing style in the Language of Mathematics. Ease of reading the mathematical model is usually the most important factor influencing the translator's choice. The structure of the mathematical model and the context of the mathematical expressions will affect which alternative is more readable and, hence, influence the choice.

The distinction made above between the several groups of meanings and uses is less important with regard solely to the English text than it is with regard to the mathematical formulation. The Language of Mathematics draws a sharper distinction between these meanings than does English. The clearer distinction makes the statements more precise, and this, in turn, contributes to the unambiguity of the mathematical expressions. Before formulating the mathematical expressions, the translator must clarify precisely which meaning is intended. The Language of Mathematics does not permit the vagueness present in the English text to be carried over to the mathematical translation. This aspect of translating is not unique to the Language of Mathematics, but it is especially important when the target language is mathematics.

Note also that in many sentences in which "is" appears, the verb "is" is not semantically commutative; that is, "Carl is a lawyer" does not mean the same as "A lawyer is Carl" (except possibly in a poetic literary style, but never in normal, colloquial style). This consideration alone eliminates the possibility of translating "is" with "=," because the mathematical function "=" is commutative.

In choosing the form of the expression translated, one must keep in mind the overall needs of the mathematical model. For example, we have at least two choices for translating "Mosche is a physician": "$\mathrm{Mosche}\in\mathbb{Physicians}$" (the membership choice) and "Vocation(Mosche)=physician" (the predicate noun choice). If Mosche is both a physician and a rabbi, a predicate noun translation of "Mosche is a physician and a rabbi" would be "Vocation(Mosche)=physician $\wedge$ Vocation(Mosche)=rabbi." However, this expression can never be true, because a function can have only one value for a given argument and, therefore, at least one of the two terms would always be false. In such cases, either the set of values of the function Vocation must include every possible combination of multiple vocations (often, a clumsy style) or one must choose the membership form of the translation (e.g., "$\mathrm{Mosche}\in\mathbb{Physicians} \wedge \mathrm{Mosche}\in\mathbb{Rabbis}$"). Alternatively, an equivalent relational Boolean function of the form "IsA(name, vocation) can be defined and the sentence "Mosche is a physician and a rabbi" translated as "IsA(Mosche, physician) $\wedge$ IsA(Mosche, rabbi)." The choice between

$$\mathrm{Mosche}\in\mathbb{Physicians} \wedge \mathrm{Mosche}\in\mathbb{Rabbis} \qquad [7.2\text{-}1]$$

and

$$\mathrm{IsA(Mosche,\ physician)} \wedge \mathrm{IsA(Mosche,\ rabbi)} \qquad [7.2\text{-}2]$$

is, again, a matter of style. Choosing the former has the consequence that one must define a set of people for every possible vocation. Using the latter form, a value must be defined for each vocation. Often, a large number of sets, the more abstract entity, will be perceived to lead to a more complex model than the same number of values, the simplest entity in the Language of Mathematics, but one can dispute the validity of this perception. Here the best choice is not generally clear; overall aspects of the mathematical model's structure should determine the translator's selection. In any case, consistency is definitely called for, as mixing the two types of expressions in a mathematical model will tend to confuse the reader and is, therefore, definitely a bad style. In English writing, repetition of the same structure is sometimes considered monotonous and undesirable, and variety is recommended for flavor. In the Language of Mathematics, just the opposite applies: Repetition of the same form calls attention to the fact that the structures are identical, facilitating comparison and understanding.

Equivalent to the definition of the Boolean function IsA in the paragraph above is a corresponding subset of the Cartesian product of the set of persons and the set of professions. The subset represents the same relation as the Boolean function IsA. In the example described above, the ordered pairs (Mosche, physician) and (Mosche, rabbi) would be in the set $\mathbb{PersonProfessionRelation}$. The Boolean function IsA and the set $\mathbb{PersonProfessionRelation}$ are in direct correspondence with each other in the sense of Section 4.1.5. The choice between a Boolean function such as IsA and a set such as $\mathbb{PersonProfessionRelation}$ is a matter of style. The better choice is generally that which most of the intended readers of the mathematical model will find simpler to understand. Many people will perceive the Boolean function model to be simpler than the set model, but mathematically, neither can be argued to be simpler than the other.

Note that the sentence “Mosche is a physician and a rabbi” in the paragraph above is an abbreviated form of “Mosche is a physician and Mosche is a rabbi.” In the paragraph above, we have translated the latter sentence, not the former sentence. From the translation

$$\text{Mosche}\in\mathbb{Physicians} \wedge \text{Mosche}\in\mathbb{Rabbis} \qquad [7.2\text{-}3]$$

one can write the mathematically equivalent expression

$$\text{Mosche}\in\mathbb{Physicians}\cap\mathbb{Rabbis} \qquad [7.2\text{-}4]$$

which is a more direct translation of the original sentence, “Mosche is a physician and a rabbi” or, more emphatically, “Mosche is both a physician and a rabbi.” Sometimes such equivalences are easier to see in the Language of Mathematics than in the English language text. Formal proofs of the equivalence are possible only when the statements are expressed in the Language of Mathematics.

A similar construction in English is “Laslo and Nancy are physicians,” which is a shorter form meaning “Laslo is a physician and Nancy is a physician.” As above, this can be translated into the Language of Mathematics as

$$\text{IsA(Laslo, physician)} \wedge \text{IsA(Nancy, physician)} \qquad [7.2\text{-}5]$$

or as

$$\text{Laslo}\in\mathbb{Physicians} \wedge \text{Nancy}\in\mathbb{Physicians} \qquad [7.2\text{-}6]$$

Sometimes expression 7.2-6 is abbreviated as

$$\text{Laslo, Nancy}\in\mathbb{Physicians} \qquad [7.2\text{-}7]$$

This form in 7.2-7 is best considered to be an idiomatic abbreviation. More formally, one can shorten expression 7.2-6 to

$$\{\text{Laslo, Nancy}\} \subset \mathbb{Physicians} \qquad [7.2\text{-}8]$$

7.3 SENTENCES OF THE FORM “…S ARE …S” (PLURAL FORMS)

Clauses or sentences of the form “…s are …s” appear superficially to be plural forms of the structure “… is (a) …”, which is the subject of Section 7.2. Clauses such as “…s are …s” include references to plural words in English, but the Language of Mathematics does not have correspondingly simple forms for the plural. Plural forms in English and their corresponding expressions in the Language of Mathematics have somewhat different meanings (i.e., they are semantically different constructions).

Often, clauses of the form "…s are …s" have a meaning that differs from the meaning of the corresponding singular form "… is (a) …" in subtle but important ways. The simple grammatical change of the singular to the plural form in English conceals these differences in meaning. The different forms in the Language of Mathematics expose these differences in meaning. Although this may seem at first to complicate matters, it has the advantage of making the differences explicit and eliminating potential ambiguity.

When forming the plural in English, one in effect combines several individual things. The way they are combined may or may not be specified. This can, and often does, lead to ambiguity and misunderstanding. In the Language of Mathematics the function combining them must be specified clearly and unambiguously. The logical functions "and" and "or," the corresponding set functions "intersection" and "union," and the arithmetic functions "sum" and "product" are common ways of combining the individual things in question. The individual things being combined are values or, more generally, elements of sets.

In Section 7.2 various meanings of sentences and clauses of the form "… is (a) …" and their respective translations into English were identified. A distinction was made between the following meanings and uses: *membership* in a category or class, *relation*, *predicate adjective or noun*, and *state*. The translation of plural forms belonging to each of these groups is considered below.

Membership in a Category or Class A plural form of "that girl is a student" is "those girls are students." In Section 7.2, "Sarah is a student" was translated by the expression "Sarah$\in\mathbb{S}\mathbb{t}\mathbb{u}\mathbb{d}\mathbb{e}\mathbb{n}\mathbb{t}\mathbb{s}$," so "that girl is a student" can be translated by the expression "that girl$\in\mathbb{S}\mathbb{t}\mathbb{u}\mathbb{d}\mathbb{e}\mathbb{n}\mathbb{t}\mathbb{s}$."

How, then, can the sentence "those girls are students" be translated? First, the intended meaning must be clarified precisely. Presumably, "those girls are students" means "that girl is a student and that other girl is a student and that other girl next to them is a student and …," leading to a mathematical expression of the form

$$\begin{aligned}&\text{that girl}\in\mathbb{S}\mathbb{t}\mathbb{u}\mathbb{d}\mathbb{e}\mathbb{n}\mathbb{t}\mathbb{s} \wedge \text{that other girl}\in\mathbb{S}\mathbb{t}\mathbb{u}\mathbb{d}\mathbb{e}\mathbb{n}\mathbb{t}\mathbb{s}\\ &\quad\wedge \text{that other girl next to them}\in\mathbb{S}\mathbb{t}\mathbb{u}\mathbb{d}\mathbb{e}\mathbb{n}\mathbb{t}\mathbb{s} \wedge \ldots\end{aligned} \qquad [7.3\text{-}1]$$

The variable number of terms in this expression suggests a quantified expression over the set of "those girls":

$$[\wedge\ g : g\in\mathbb{T}\mathbb{h}\mathbb{o}\mathbb{s}\mathbb{e}\mathbb{G}\mathbb{i}\mathbb{r}\mathbb{l}\mathbb{s} : g\in\mathbb{S}\mathbb{t}\mathbb{u}\mathbb{d}\mathbb{e}\mathbb{n}\mathbb{t}\mathbb{s}] \qquad [7.3\text{-}2]$$

Other formulations of the English sentence leading to mathematical expression 7.3-2 include: "all those girls are students," "each of those girls is a student," and "every one of those girls is a student." The last two sentences lead most directly to the quantified expression 7.3-2.

Note that the plural form "those girls" has led to a quantification. The grammatical number of the verb ("is" or "are") or of the predicate noun "student" or "students" has not affected the mathematical formulation. The quantified expression corresponds to the singular formulation of the English sentence.

It follows from the definition of a subset that the mathematical expression above is equivalent to (i.e., always has the same value as) the following expression:

$$\mathbb{ThoseGirls} \subset \mathbb{Students} \qquad [7.3\text{-}3]$$

In expression 7.3-3 the plural noun phrase “those girls” has been translated by the set $\mathbb{ThoseGirls}$; the plural noun “students,” by the set $\mathbb{Students}$; and the verb “are,” by the set function “subset” ($\subset$). In the case of the singular clause “Sarah is a student,” the name Sarah was translated by the value “Sarah” and the verb “is” by the function “element of” ($\in$). In this sense, a set (e.g., $\mathbb{ThoseGirls}$) can be thought of as the plural of a value (e.g., “Sarah” or “that girl”), and the function “subset” ($\subset$) can be thought of as the plural of the function “element of” ($\in$).

Other examples are “birds are animals,” which can be translated into the Language of Mathematics in the same way described above as “$\mathbb{Birds} \subset \mathbb{Animals}$.” Correspondingly, “animals are birds” would be translated into “$\mathbb{Animals} \subset \mathbb{Birds}$,” both of which are false.

Relation A plural form of a sentence such as “George is heavier than Sam” could be “our boxers are heavier than our sprinters” or “all boxers are heavier than all sprinters.” The latter formulation, especially, could be interpreted in more than one way. Presumably, however, the intended meaning of all of these sentences is that “every boxer is heavier than each sprinter” or, equivalently, “any boxer is heavier than any sprinter” or “each boxer is heavier than each sprinter.” This interpretation leads to a double quantification, one over the boxers and the other over the sprinters:

$$[\wedge\ b, s : b{\in}\mathbb{Boxers} \wedge s{\in}\mathbb{Sprinters} : \text{weight}(b) > \text{weight}(s)] \qquad [7.3\text{-}4]$$

Other equivalent formulations are possible, but each contains two quantifications, one over the boxers and the other over the sprinters. The quantifications can be structured differently (e.g., nested or separated completely), as in an expression corresponding to “the lightest boxer is heavier than the heaviest sprinter,” but both quantifications are needed to express this relationship.

Predicate Adjective or Noun An example of a plural form of a sentence of this type is “The pencils are green.” As in the preceding examples, this sentence can be translated by formulating a mathematical expression for such a sentence in the singular and then embedding that expression in a suitable quantifying structure.

A singular version of this sentence would be “the pencil is green,” which can be translated “ColorOf(the pencil)=green” as in Section 7.2. Assuming that the phrase “the pencils” means all the pencils (i.e., every pencil) in a particular collection (set), quantification over that set is appropriate. The words “all” and “every” both call for the logical “and” function, leading to

$$[\wedge\ p : p{\in}\mathbb{Pencils} : \text{ColorOf}(p){=}\text{green}] \qquad [7.3\text{-}5]$$

This is a typical form for the translation of a sentence with a predicate adjective or noun and a plural subject.

Note that sometimes verbs other than conjugated forms of “to be” can appear in a sentence of this type (e.g., “Emily’s children all have black hair”). This is a more colloquial way of saying “the color of the hair of all Emily’s children is black” or “the color of the hair of every one of Emily’s children is black” or “the color of the hair of each of Emily’s children is black.” The first sentence is in a simpler English style, but the latter sentences express more explicitly and precisely the relationships between Emily, her children, and their hair. The latter sentences, using the verb “is,” highlight the predicate adjective aspect of the color black, rather than the possession of hair suggested by the verb “have,” which is not really the essential relationship here. Furthermore, in the latter versions of the sentence, the directly related aspects are mentioned closer together in the sentence (e.g., color of the hair of a child), corresponding to the functional relationships in the mathematical translation:

$$[\wedge\ c : c \in \mathbb{Emily's Children} : \text{ColorOfHair}(c)=\text{black}] \qquad [7.3\text{-}6]$$

Notice in the example above the use of the verb “to have” in English instead of “to be” to express a relationship that is basically adjectival. Notice also that the words “all,” “every,” and “each” commonly require a quantification with the logical “and” (∧, ∀) function in the translation in the Language of Mathematics.

State Sentences about states containing the plural form “are” appear relatively infrequently. One type of such a sentence is of the form “The states of the system are …, …, .” Such a sentence is an enumeration of the set of states (i.e., defines the set of states) and can therefore best be translated by an expression of the form $\mathbb{States}=\{a, b, c, \ldots\}$.

Another type of such a sentence is of the form “All doors are closed,” which is usually a shorter form of “Door 1 is closed, door 2 is closed, …, and door N is closed.” This type of sentence can be translated by translating each singular clause separately as described in Section 7.2 and combining them in a quantified expression: for example,

$$[\wedge\ i : i \in \mathbb{Z} \wedge\ 1 \le i \le N : \text{DoorIsClosed}(i)] \qquad [7.3\text{-}7]$$

or

$$[\wedge\ i : i \in \mathbb{Z} \wedge\ 1 \le i \le N : \text{PositionOfDoor}(i)=\text{closed}] \qquad [7.3\text{-}8]$$

or

$$[\wedge\ i : i \in \mathbb{Z} \wedge\ 1 \le i \le N : \text{StateOfDoor}(i)=\text{closed}] \qquad [7.3\text{-}9]$$

7.4 PERCENT, PER …, AND OTHER LOW-LEVEL EQUIVALENCES

In English text the term *percent* is often used in a context involving numbers. This term is unnecessary and has, unfortunately, confused many people, but it is in such common use that everyone wishing to apply arithmetic or mathematics in practice must have a good command of this concept and be able to work with it fluently.

The word "percent" (abbreviated %) means "per hundred," "hundredths," or, in more mathematical terminology, "divided by 100"—nothing more, nothing less. When translating from English to the Language of Mathematics, one can simply replace the word "percent" by the partial expression "/100." The prefix "per" itself means "for every" or, equivalently, "divided by."

The preposition "of" frequently appears in phrases with the word percent, and the English preposition "of" can usually be translated as "multiplied by," which is in turn abbreviated by the infix symbol $*$.

The following examples give mathematical translations of English phrases containing "percent":

Thirty percent of the population:	(30/100) $*$ population
Eight is what percentage of 23?:	$8 = (x/100) * 23$
Ten is 4 percent of what?	$10 = (4/100) * x$
Ten is 4 hundredths of what?	$10 = (4/100) * x$
Four is 80% of five:	$4 = (80/100) * 5$

Presumably, the term "percent" was introduced in order to express fractions as whole numbers; that is, instead of writing 1/25 or 4/100, one writes 4%. Although it is unnecessary to do so, people have found this convention convenient, and it has become thoroughly established in talking about and working with numbers and relationships between them.

Similarly, the terms *per mille* (per thousand, abbreviated ‰) and *per million* (often occurring in the phrase "parts per million," abbreviated "ppm," in chemical contexts) can be translated into the Language of Mathematics as /1,000 and /1,000,000, respectively.

As mentioned above, the preposition "of" can often be translated with multiplication (i.e., $*$). The word *per* can often be translated with division, that is, with the infix symbol /, and not only in phrases or word combinations such as percent. A few other such examples of English words and short phrases representing common low-level mathematical functions are the following:

times:	$*$
by (e.g., "4 by 5"):	$*$
to the power (exponentiation):	$\uparrow$ (or ^)
raised to (e.g., "2 raised to the 3rd power"):	$\uparrow$ (or ^)

Other common examples relating to numerical sums and differences and to the logical functions and, or, and implies are contained in many other sections elsewhere in this book.

7.5 MODELING TIME AND DYNAMIC PROCESSES IN THE LANGUAGE OF MATHEMATICS

The Language of Mathematics is made up of expressions, which are, in turn, based on values, variables, and functions. Values can be single entities or any one of a number of well-defined structures, such as sets and sequences. These things are all static; in none of them is there any reference to or any concept of time. In short, the Language of Mathematics is a static language in which time plays no role.

Nonetheless, the Language of Mathematics can be used to model processes in which time does play a role. In fact, many important applications of the Language of Mathematics involve time. Some involve continuous time steps and others, discrete time steps.

7.5.1 Dynamic Processes in Continuous Time

The various subdisciplines of physics and other sciences provide examples in which continuous time is modeled: mechanics, electricity, chemistry, biology, and so on. Many important basic physical laws pertaining to motion and time are formulated as mathematical expressions, such as Newton's second law of motion, $F=m*a$, where F is the force on a mass m and a is its acceleration (i.e., the second derivative of its position with respect to time):

$$F = m * \frac{d^2x}{dt^2} \qquad \text{[Newton's second law of motion, 7.5.1-1]}$$

Example: Dynamic Process in Continuous Time If an object of mass m falls in a force field (e.g., Earth's gravity) applying a constant force F to the mass, the distance x through which it will fall in time t is a solution to equation 7.5.1-1. If at time $t=0$ the mass is at the position $x=0$, is at rest (its velocity is zero), and is released to fall, the solution to equation 7.5.1-1 is

$$x = \frac{F * t^2}{2 * m} \qquad [7.5.1\text{-}2]$$

The velocity v of the mass at time t is the first derivative of x with respect to t and is

$$v = \frac{F * t}{m} \qquad [7.5.1\text{-}3]$$

Thus, the velocity increases linearly with time and the distance fallen increases quadratically with time (i.e., as the square of the time).

Equation 7.5.1-1 neglects the force on the falling mass due to air resistance. To include the effect of this force (drag), which is opposite in direction to the motion of the mass, an appropriate term can be included in equation 7.5.1-1. If this force is proportional to the velocity of the mass, the resulting equation of motion is

$$\frac{d^2x}{dt^2} + \frac{k}{m} * \frac{dx}{dt} = \frac{F}{m} \qquad [7.5.1\text{-}4]$$

where k is the coefficient of the drag force on the falling body. The solution to equation 7.5.1-4 with the same initial conditions as above $\left[x=0 \text{ and the velocity } \left(\frac{dx}{dt}\right) \text{ is 0 at time } t=0\right]$ is

$$x = \frac{F}{k} * t - \frac{m * F}{k^2} * (1 - e^{-k*t/m}) \qquad [7.5.1\text{-}5]$$

where e is 2.71828 …, a mathematical constant, the base of the natural logarithms. This constant appears in solutions of many differential equations.

Equation 7.5.1-4 is an example of a linear, second-order, nonhomogeneous differential equation with constant coefficients. Methods for finding the solution of this and many other types of differential equations are covered extensively and thoroughly in both the theoretical mathematical literature and the engineering-oriented mathematical literature.

Notice in the expressions above that the meanings of distance, force, mass, velocity, and time have been attributed to the variables x, F, m, v, and t by us because of the way we wish to interpret these variables and expressions. None of these meanings is inherent in the mathematical expressions themselves; the values of all of these variables, including t for time, are, mathematically, just real numbers, nothing more. Completely different application environments lead to the same types of differential equations, and for those applications, the corresponding variables and constants are interpreted very differently.

Although we interpret the equations above to apply to dynamic processes, the equations themselves and their solutions are all static relationships between the values of the numerical variables appearing in the equations.

The example above illustrates the typical way in which *continuous time* is represented in mathematical models: by a variable whose value represents time in the application domain. The value of that variable is a real number. The expressions in the mathematical model frequently involve derivatives and differential equations. Sometimes the expressions involve integrals and integral equations.

Other application areas calling for continuous time models include:

- Design and operation of:
  - Vehicle steering systems
  - Engine control systems
  - Heating and cooling systems
  - Chemical reactors
  - Nuclear reactors
  - Systems for generating and distributing electrical power
- Electrical circuits for many purposes
- Mechanical equipment with moving components
- Forecasting continuous processes such as:
  - Weather
  - Demand for electricity
  - Availability of solar, wind, and water power

- Economic processes and business procedures such as:
  - Forecasting gross national product and other economic measures and indices
  - Forecasting sales of goods and services
  - Optimizing inventories
  - Forecasting prices of commodities, bonds, and shares
  - Calculating prices of derivatives in financial markets

7.5.2 Dynamic Processes in Discrete Time Steps

Dynamic processes not involving continuous time but instead discrete steps in time are usually modeled with one or more sequences. Each term in the sequence is interpreted to represent one time step. Successive time steps are represented by successive terms in the sequence. The finite state machine or automaton as defined in Section 4.1.7 is a commonly used structure for such situations.

Example: Dynamic Process in Discrete Time Steps Consider a lock in an inland waterway used to raise and lower ships and boats between two different levels of the waterway. At the upper level of the waterway is a gate that can be opened or closed. Another gate is located at the lower level of the waterway. The level of water in the space between the two gates can be raised or lowered between the two levels of the waterway by allowing water to flow in from the higher level or by allowing water to flow out to the lower level.

The cycle of operation is as follows. The gates at the upper level are open and ships and boats can enter the space between the gates. Then the upper gate is closed. Water is allowed to flow out of the space between the gates (both of which are closed) to the lower level of the waterway. When that level is reached, the lower gates are opened and the boats continue their journey downstream. Then boats from the lower level enter the space between the gates, after which the lower gates are closed. Water is then allowed to flow from the higher level of the waterway into the space between the gates. When the water between the gates reaches the higher level, the upper gates are opened and the boats continue their journey upstream. Then boats from the upper level enter the space between the gates and the cycle is repeated.

In front of each set of gates are red and green signal lights. They are used to indicate to boats outside the lock when they are and are not permitted to enter the lock. The signal lights may be green only if the gate at that level is open; otherwise, they must be red.

A control system coordinates many of the operations of the lock's gates, valves, and signal lights. A human operator monitors the positions and movements of all the boats in the vicinity of the lock, decides when certain operations can and should be performed, and presses corresponding button switches on the control panel. The automatic part of the system controls all operations from the time the operator indicates that the lock should be raised or lowered until the new water level is reached and the gates are open with the signal lights red. The human operator monitors boat movements into and out of the lock at each level. He or she switches the signal lights from red to green and back to red and initiates raising or lowering the level of water in the lock.

Operator's Control Panel On the operator's control panel there are six pushbutton switches:

- To switch on the red signal lights at the lower level
- To switch on the green signal lights at the lower level
- To switch on the red signal lights at the upper level
- To switch on the green signal lights at the upper level
- To initiate raising the level of water in the lock (between the gates) to the upper level
- To initiate lowering the level of water in the lock (between the gates) to the lower level

If the operator presses two or more button switches at the same time, only one is signaled to the controller. The other depressions are disregarded.

There are also several lamps on the operator's control panel indicating which signal lights are red, which are green, and the state of the entire system, in particular whether the level of water between the gates is at the lower level, at the upper level, and if at an intermediate level, whether it is rising or falling.

Interfaces Between the Operator's Control Panel, the Master Controller, and the Subsidiary Controller The mechanized control system consists of the master controller, which is the subject of this example, and a subsidiary controller. The master (higher level, overall) controller communicates with the subsidiary controller by sending output signals to it and receiving input signals from it. Otherwise, the subsidiary controller is not modeled in this example.

The master controller receives the following input signals:

- Signals from the six pushbuttons on the operator's control panel (see above)
- A signal from the subsidiary controller indicating that the water in the lock has reached the upper level, the upper gate is open, and the upper signal lights are red
- A signal from the subsidiary controller indicating that the water in the lock has reached the lower level, the lower gate is open, and the lower signal lights are red

The master controller sends, when appropriate, the following output signals to the subsidiary controller:

- A signal indicating that the level of water in the lock should be raised to the upper level
- A signal indicating that the level of water in the lock should be lowered to the lower level

Safety Conditions For safety reasons, the master controller will respond to certain inputs from the operator's control panel only when certain conditions are met. If they are not met, the master controller will simply ignore the inputs from the operator's control panel as if the corresponding button switch had not been pressed (i.e., the new state will be the same as the old state and no output message will be generated). Superfluous inputs (such as the operator requesting green signal lights at the lower gate when they are already green) will be handled in the same way. The relevant conditions and restrictions are:

- If the operator presses the button to switch on the red signal lights at the lower level, those lights must already be green. (Condition 1)
- If the operator presses the button to switch on the green signal lights at the lower level, those lights must already be red and the lower gate must already be open. (Condition 2)
- If the operator presses the button to switch on the red signal lights at the upper level, those lights must already be green. (Condition 3)
- If the operator presses the button to switch on the green signal lights at the upper level, those lights must already be red and the upper gate must already be open. (Condition 4)
- If the operator presses the button to initiate raising the level of water in the lock to the upper level, the lower gate must be open and the lower signal lights must be red. (Condition 5)
- If the operator presses the button to initiate lowering the level of water in the lock to the lower level, the upper gate must be open and the upper signal lights must be red. (Condition 6)

Master Controller States From the foregoing descriptions of the lock system, it can be deduced that the master controller must distinguish among the following six states:

- The lower gate is open and the lower signal lights are red. The upper gate is closed and the upper signal lights are red.
- The lower gate is open and the lower signal lights are green. The upper gate is closed and the upper signal lights are red.
- The upper gate is open and the upper signal lights are red. The lower gate is closed and the lower signal lights are red.
- The upper gate is open and the upper signal lights are green. The lower gate is closed and the lower signal lights are red.
- The water in the lock is rising, both gates are closed, and all signal lights are red.
- The water in the lock is falling, both gates are closed, and all signal lights are red.

The mathematical model, the detailed functionality of the master controller, and the interpretation of the functions, variables, and values in this model are as follows.

Variables

- input(n): The input message received by the master controller in time step n.
- output(n): The output message sent by the master controller to the subsidiary controller in time step n.
- state(n): The state of the master controller in time step n.
- InputHistory: The sequence of all input messages received by the master controller.
- OutputHistory: The sequence of all output messages sent by the master controller.
- StateHistory: The sequence of all states of the master controller.

Values of input(n)

- SwRedLower: The button to switch on the red signal lights at the lower level is depressed.
- SwGreenLower: The button to switch on the green signal lights at the lower level is depressed.
- SwRedUpper: The button to switch on the red signal lights at the upper level is depressed.
- SwGreenUpper: The button to switch on the green signal lights at the upper level is depressed.
- GoUp: The button to initiate raising the level of water in the lock to the upper level is depressed.
- GoDown: The button to initiate lowering the level of water in the lock to the lower level is depressed.
- WaterUp: This message from the subsidiary controller indicates that the upper water level has been reached, the upper gate is open, and the upper signal lights are red.
- WaterDown: This message from the subsidiary controller indicates that the lower water level has been reached, the lower gate is open, and the lower signal lights are red.

Values of output(n)

- GoUp: This message to the subsidiary controller is a command that the level of water in the lock be raised to the upper level.
- GoDown: This message to the subsidiary controller is a command that the level of water in the lock be lowered to the lower level.
- none: In some time steps the master controller does not output a message. In these cases, the value of output(n) is “none.” See the six conditions listed above.

Values of the State Variable

- LowerGateOpenRed: The lower gate is open and the lower signal lights are red. The upper gate is closed and the upper signal lights are red.

- LowerGateOpenGreen: The lower gate is open and the lower signal lights are green. The upper gate is closed and the upper signal lights are red.
- UpperGateOpenRed: The upper gate is open and the upper signal lights are red. The lower gate is closed and the lower signal lights are red.
- UpperGateOpenGreen: The upper gate is open and the upper signal lights are green. The lower gate is closed and the lower signal lights are red.
- Rising: The water in the lock is rising, both gates are closed, and all signal lights are red.
- Falling: The water in the lock is falling, both gates are closed, and all signal lights are red.

Sets

- $\mathbb{Inputs}$: The set of input messages: {SwRedLower, SwGreenLower, SwRedUpper, SwGreenUpper, GoUp, GoDown, WaterUp, WaterDown}
- $\mathbb{Outputs}$: The set of output messages: {GoUp, GoDown, none}
- $\mathbb{States}$: The set of states: {LowerGateOpenRed, LowerGateOpenGreen, UpperGateOpenRed, UpperGateOpenGreen, Rising, Falling}

Functions

- NextState: NextState is a function from the set $\mathbb{Inputs} \times \mathbb{States}$ to $\mathbb{States}$:

$$\text{NextState} : \mathbb{Inputs} \times \mathbb{States} \rightarrow \mathbb{States} \qquad [7.5.2\text{-}1]$$

- NextOutput: NextOutput is a function from the set $\mathbb{Inputs} \times \mathbb{States}$ to $\mathbb{Outputs}$:

$$\text{NextOutput} : \mathbb{Inputs} \times \mathbb{States} \rightarrow \mathbb{Outputs} \qquad [7.5.2\text{-}2]$$

The following table gives the values of output(n+1) and state(n+1) as functions of state(n) and input(n+1) for all nonnegative integer values of n. The table entries are in the order of the cycle of operations described above.

Comment	state(n)	input(n+1)	output (n+1)	state(n+1)
all boats in lock	UpperGateOpenGreen	SwRedUpper	none	UpperGateOpenRed
start lowering	UpperGateOpenRed	GoDown	GoDown	Falling
at lower level	Falling	WaterDown	none	LowerGateOpenRed
boats have left lock	LowerGateOpenRed	SwGreenLower	none	LowerGateOpenGreen
all boats in lock	LowerGateOpenGreen	SwRedLower	none	LowerGateOpenRed
start raising	LowerGateOpenRed	GoUp	GoUp	Rising
at upper level	Rising	WaterUp	none	UpperGateOpenRed
boats have left lock	UpperGateOpenRed	SwGreenUpper	none	UpperGateOpenGreen

This table is clearly incomplete. It contains only eight data rows. There are six states and eight possible input messages, so a complete table would contain 6∗8 or 48 data rows. The remaining 40 rows must be examined to see if they are all excluded by the safety conditions stated above. If so, it must at least be indicated in the table that state(n+1) is the same as state(n) and that output(n+1) is none.

The following table contains the 40 rows missing from the table above. The comments refer either to the relevant safety condition mentioned above or to the fact that the input from the subsidiary controller is inconsistent with the behavior described above.

Comment	state(n)	input(n+1)	output (n+1)	state(n+1)
Condition 6	Falling	GoDown	none	Falling
Condition 5	Falling	GoUp	none	Falling
Condition 2	Falling	SwGreenLower	none	Falling
Condition 4	Falling	SwGreenUpper	none	Falling
Condition 1	Falling	SwRedLower	none	Falling
Condition 3	Falling	SwRedUpper	none	Falling
Condition 6	LowerGateOpenGreen	GoDown	none	LowerGateOpenGreen
Condition 5	LowerGateOpenGreen	GoUp	none	LowerGateOpenGreen
Condition 2	LowerGateOpenGreen	SwGreenLower	none	LowerGateOpenGreen
Condition 4	LowerGateOpenGreen	SwGreenUpper	none	LowerGateOpenGreen
Condition 3	LowerGateOpenGreen	SwRedUpper	none	LowerGateOpenGreen
Condition 6	LowerGateOpenRed	GoDown	none	LowerGateOpenRed
Condition 4	LowerGateOpenRed	SwGreenUpper	none	LowerGateOpenRed
Condition 1	LowerGateOpenRed	SwRedLower	none	LowerGateOpenRed
Condition 3	LowerGateOpenRed	SwRedUpper	none	LowerGateOpenRed
Condition 6	Rising	GoDown	none	Rising
Condition 5	Rising	GoUp	none	Rising
Condition 2	Rising	SwGreenLower	none	Rising
Condition 4	Rising	SwGreenUpper	none	Rising
Condition 1	Rising	SwRedLower	none	Rising
Condition 3	Rising	SwRedUpper	none	Rising
Condition 6	UpperGateOpenGreen	GoDown	none	UpperGateOpenGreen
Condition 5	UpperGateOpenGreen	GoUp	none	UpperGateOpenGreen
Condition 2	UpperGateOpenGreen	SwGreenLower	none	UpperGateOpenGreen
Condition 4	UpperGateOpenGreen	SwGreenUpper	none	UpperGateOpenGreen
Condition 1	UpperGateOpenGreen	SwRedLower	none	UpperGateOpenGreen
Condition 5	UpperGateOpenRed	GoUp	none	UpperGateOpenRed
Condition 2	UpperGateOpenRed	SwGreenLower	none	UpperGateOpenRed
Condition 1	UpperGateOpenRed	SwRedLower	none	UpperGateOpenRed
Condition 3	UpperGateOpenRed	SwRedUpper	none	UpperGateOpenRed
inconsistent	Falling	WaterUp	none	Falling
inconsistent	LowerGateOpenGreen	WaterDown	none	LowerGateOpenGreen
inconsistent	LowerGateOpenGreen	WaterUp	none	LowerGateOpenGreen
inconsistent	LowerGateOpenRed	WaterDown	none	LowerGateOpenRed

Comment	state(n)	input(n+1)	output (n+1)	state(n+1)
inconsistent	LowerGateOpenRed	WaterUp	none	LowerGateOpenRed
inconsistent	Rising	WaterDown	none	Rising
inconsistent	UpperGateOpenGreen	WaterDown	none	UpperGateOpenGreen
inconsistent	UpperGateOpenGreen	WaterUp	none	UpperGateOpenGreen
inconsistent	UpperGateOpenRed	WaterDown	none	UpperGateOpenRed
inconsistent	UpperGateOpenRed	WaterUp	none	UpperGateOpenRed

For the purposes of this example we assume that in those cases of inconsistent inputs from the subsidiary controller, the output (none) and the next states (the same as the previous states) given in the table are acceptable. However, these inputs indicate faulty behavior which, in an actual application, would have to be examined in more detail: for example, by proving that the design of the subsidiary controller excludes these responses in those states of the master controller. In addition, of course, the reliability of the components of the system must be considered and provision for handling their failures made in the final design of this system.

Under these assumptions, the first table above can be extended to ignore the inappropriate inputs. The resulting table defines the functions NextOutput and NextState:

TABLE 7.5.2-1 Controller Functions for a Lock on an Inland Waterway

Comment	state(n)	input(n+1)	output (n+1)	state(n+1)
all boats in lock	UpperGateOpenGreen	SwRedUpper	none	UpperGateOpenRed
start lowering	UpperGateOpenRed	GoDown	GoDown	Falling
at lower level	Falling	WaterDown	none	LowerGateOpenRed
boats have left lock	LowerGateOpenRed	SwGreenLower	none	LowerGateOpenGreen
all boats in lock	LowerGateOpenGreen	SwRedLower	none	LowerGateOpenRed
start raising	LowerGateOpenRed	GoUp	GoUp	Rising
at upper level	Rising	WaterUp	none	UpperGateOpenRed
boats have left lock	UpperGateOpenRed	SwGreenUpper	none	UpperGateOpenGreen
Conditions 1 to 6 and inconsistencies	all other combinations		none	state(n)

The complete mathematical model for the master controller is, then,

$$\mathbb{Inputs} = \{\text{SwRedLower, SwGreenLower, SwRedUpper, SwGreenUpper, GoUp, GoDown, WaterUp, WaterDown}\}$$
$$\wedge\ \mathbb{Outputs} = \{\text{GoUp, GoDown, none}\}$$

$$
\begin{aligned}
&\land\ \mathbb{States} = \{\text{LowerGateOpenRed, LowerGateOpenGreen,}\\
&\qquad\qquad \text{UpperGateOpenRed, UpperGateOpenGreen, Rising, Falling}\}\\
&\land\ (\text{NextState} : \mathbb{Inputs} \times \mathbb{States} \rightarrow \mathbb{States})\\
&\land\ (\text{NextOutput} : \mathbb{Inputs} \times \mathbb{States} \rightarrow \mathbb{Outputs})\\
&\land\ [\land\ n : n\in\mathbb{Z} \land\ n\geq 0 : \text{input}(n+1)\in\mathbb{Inputs} \land \text{state}(n)\in\mathbb{States}\\
&\qquad\qquad \land\ \text{output}(n+1)\in\mathbb{Outputs}]\\
&\land\ [\land\ n : n\in\mathbb{Z} \land\ n\geq 0 : \text{state}(n+1)=\text{NextState}(\text{input}(n+1), \text{state}(n))\\
&\qquad\qquad \land\ \text{output}(n+1)=\text{NextOutput}(\text{input}(n+1), \text{state}(n))]\\
&\land\ \text{InputHistory} = [\text{input}(1), \text{input}(2), \text{input}(3), \ \ldots]\\
&\land\ \text{StateHistory} = [\text{state}(0), \text{state}(1), \text{state}(2), \ \ldots]\\
&\land\ \text{OutputHistory} = [\text{output}(1), \text{output}(2), \text{output}(3), \ \ldots] \qquad [7.5.2\text{-}3]
\end{aligned}
$$

The values of state(0) and input(n), for all positive integers n, are determined externally (i.e., by this controller's environment). The values of all other variables referred to in the model above are functions of the values of the external variables as stated within the mathematical model above.

The reader should extend the master controller above to output a signal when one of the safety conditions listed above is violated. The output signal should indicate which condition is violated. Assume that this will cause a corresponding indication to be displayed on the operator's panel. Distinguish between safety-relevant violations of the conditions and superfluous inputs from the operator's control panel.

As another exercise, the reader should design the subsidiary controller. Express the design in a similar way as Table 7.5.2-1 for the master controller. Must the subsidiary controller open or close the gates? Switch signal lights on or off? Which, if any, safety interlocks should be provided? How?

Sections 8.11 and 8.13 contain additional examples of dynamic processes in discrete time steps. Section 8.10 contains a simple example of a dynamic process in discrete time steps in which the inherent physical delay in any system provides the time step, so that explicit consideration of the time steps in the mathematical expressions is not needed. Section 8.8 contains an example of one component part of a larger system that would presumably be modeled mathematically by one or more sequences.

A typical structure for modeling a dynamic process evolving in discrete time steps consists of sequences of an input, a change of state, and an output, repeated continually. Depending on the current state and the current input, iterative formulas give the next output and the next state in the corresponding sequences. That is, the next output is a function of the current state and the current input. The next state is also a function of the current state and the current input. This is the structure of the finite state machine, or automaton, in the mathematical theoretical literature and as defined and described in more detail in Section 4.1.7.

To formulate a model based on a finite state machine, begin by identifying its inputs and outputs. Then identify the internal states of the system being modeled.

Finally, develop the iterative equations for the next state and output functions from the relevant parts of the English text.

To identify the states of the finite state machine in the mathematical model, ask the following questions:

- What information is needed to describe the physical state of the system at each step in time?
- What are the results of actions? A state representing the result of an action can often be described by an adjective or by a past participle or a present participle used as an adjective, or by phrases based on these parts of speech. In the example above, the words “red,” “green,” “open,” “closed,” “rising,” and “falling” are used within the descriptions of the several states. In the example in Section 8.13, “stopped” is such a past participle describing a state and used as the value of that state.
- What actions take place over an extended interval of time and should be considered states? The value of a state representing an ongoing action can often be described by a present participle or present participle phrase used as an adjective. In the example above, “rising” and “falling” are such present participles describing states and used as values of the state variables.
- Upon which aspects of the previous terms of the input sequence do or can the subsequent terms of the output sequence depend?

The answers to one or more of these questions will lead to the state variables and their possible values.

In effect, each output is a function of the initial state and the sequence of all previous inputs. Therefore, each state must effectively contain all the information about the previous inputs that is needed to determine the output in response to the next input.

Some examples of other application areas calling for discrete time step models are:

- Design and operation of:
    - Traffic light systems
    - Vehicle movement monitoring and guidance systems
    - Systems for automatically monitoring, opening, and closing doors on trains and buses and in buildings
    - Elevator control systems
    - Electrical communication systems
    - Cryptographic systems
    - Inventory control systems
    - Digital computers
    - Digital controllers

- Forecasting, planning, scheduling, monitoring, and controlling
    - Demand for and sales of products
    - Order processing
    - Allocation of resources and personnel

Some systems require a combination of continuous and discrete time step models. The example in Section 8.13 is such a system.

7.6 QUESTIONS IN TRANSLATIONS FROM ENGLISH TO MATHEMATICS

The most common type of question appearing in a text to be translated into the Language of Mathematics indicates that the value of a particular variable or function of certain arguments is unknown. Determining that value is the goal of the analysis to be performed after the mathematical model has been constructed. The question itself will not be translated, but the translator should ensure that all information in the English text that can be of use in determining the unknown value is included in the mathematical model. If an expression for the variable in question is implied by the context, that expression should be included in the mathematical model. Typically, that expression will involve variables appearing elsewhere in the mathematical model. For example, if the unknown value is the volume of a rectangular solid, an expression of the form Volume=Length$*$Width$*$Height should be included in the mathematical model.

Questions in English are in either one of two forms:

- In the form of a statement in question form (e.g., “Did you go to town yesterday?,” “Are you happy?”), answered by yes or no or true or false
- Introduced by an interrogative word such as what, which, why, when, where, who, whom, or how, answered most simply by a phrase (noun, adjective, or adverb) or by a verb phrase often expressed in a noun form, or possibly in the form of a complete statement repeating much of the question but adding additional information. “How” questions are typically answered by the description of a procedure, often in the form of a command.

The first type of question above is basically a statement, to which a reply is expected either confirming or contradicting the correctness (truth) of the statement. The second type of question above requires a more extensive reply giving additional information (e.g., a clarification or an explanation).

The types of questions posed in English text to be translated into a mathematical model are usually in one of two forms corresponding to the types described above. In their mathematical form, they ask either of the following:

- Is the value of a Boolean expression true or false for the values of the relevant variables?
- For which values of certain variables is the mathematical expression constituting the model true? (Solving a Boolean expression)

In the first case, the truth of a Boolean expression is to be verified or disproved. In the second case, values of dependent variables in the mathematical model are to be determined.

The Language of Mathematics does not contain any notational form for a question. In the case of a question with a yes/no (true/false) answer, the question can be reformulated in the form of a corresponding statement to be verified or disproved. A Boolean variable whose interpretation is that statement is then introduced into the mathematical model. The task is then to determine the value of that variable. This approach is illustrated in Section 8.7, in which the question "Can the board be covered by dominoes?" given certain conditions is to be answered. The answer to the question is interpreted to be the value of the Boolean variable BoardIsCovered. After formulating the mathematical model for the given problem, it can be shown that the value of the variable BoardIsCovered is always false, giving as the answer to the English question, "No, the board cannot be covered by dominoes."

This approach to translating a question of the first type above handles it in the same way as a question of the second type, thereby generalizing the two types of questions in the English text into one type of task in the Language of Mathematics: to find the value(s) of the dependent variables—be they Boolean or non-Boolean—consistent with or implied by the mathematical model (i.e., for which the Boolean value of the mathematical model is true).

In the case of a question whose answer is a non-Boolean value, a non-Boolean variable with the value to be determined is introduced into the mathematical model. Its interpretation is the appropriate answer to the question in the English text. In the mathematical model this variable will be a dependent variable, that is, a variable whose value follows from the values of the independent variables in the model. A simple example is the question: "How long will it take to travel from A to B, a distance of d kilometers, in a vehicle traveling with a velocity of v kilometers per hour?" The answer is interpreted to be the value of the variable t. An appropriate mathematical model is formulated consisting mainly of the expression $v*t=d$, from which the value of t for any given pair of values of d and v can be determined. Several other examples of answering this type of question in a mathematical model are given elsewhere in the book.

In some cases, a question in the English text will be answered by the process involved in the analysis of the completed mathematical model, rather than being given by some mathematical expression following from the model. The question "Is there more than one solution?" is often of this type.

Finally, questions intended to direct attention to the implications of the results of the analysis may appear in the English text. Such questions are intended to provoke subsequent thought or to propose problems going beyond the bounds of the immediate problem, problems that could be the subject of subsequent investigations. Such questions will not be translated into the Language of Mathematics.

During the translation process, translators should ask themselves various questions and answer them carefully. These questions should be formulated in order to:

- Check the correctness of their interpretation of the English text and their translation

- Ensure that nothing was overlooked
- Identify implicit information, thereby making it explicit
- Make explicit any implicit assumption
- Ensure that no unjustified assumption was made (e.g., implicitly)
- Identify irrelevant or redundant information not to be translated
- Identify alternative interpretations of the English text
- Help them interpret the given English text correctly

Neither the questions nor the answers to them will be included in their translation, but the answers to these questions will influence their translation.

7.7 SUMMARY OF GUIDELINES FOR TRANSLATING ENGLISH TO THE LANGUAGE OF MATHEMATICS

To construct a mathematical model based on English text, the translator should begin by reading the English text thoroughly, in detail, critically and questioningly. The translator should understand the author's *intended* message. The translator should distinguish between those parts of the text to be translated into expressions in the mathematical model and those parts of the text giving only background information to facilitate understanding the rest of the text. Then, concentrating on the text to be translated into the mathematical model, the translator should:

- Look for nouns and noun phrases in the English text. They are candidates for non-Boolean variables and functions in the mathematical model.
- Identify the sets of values of these variables and functions. The values will usually be specific instances of the generic category defined by the noun corresponding to the variable or function. Examples are doctor and lawyer as values of the function ProfessionOf(…), and numbers as values of variables for distance, length, height, velocity, force, and mass. The values often appear in the English text as adjectives (especially predicate adjectives) and predicate nouns [e.g., "(The profession of) George is a mechanic"; "The shape of the object is triangular"].
- Identify clauses with a stative verb. They are candidates for Boolean variables or Boolean functions of non-Boolean arguments. Examples are "The value of x is greater than the value of y" and "The button is depressed." These might be the interpretations of the corresponding Boolean variables XIsGreaterThanY and ButtonIsDepressed, respectively, or, alternatively, of the Boolean functions "IsGreater(x, y)," "IsInPosition(button, depressed)," or "StateOfIs(button, depressed)."
- Reformulate clauses containing an action verb. Write them as clauses with a stative verb, that is, by expressing the action with a past or present participle used as an adjective or by referring to the result of the action instead of the

action itself. For example, reformulate “Add x and y to obtain z” as “The value of z is the sum of the values of x and y.”

- Look for conjunctions combining clauses (and, but, or, if … then …, implies, etc.). They will typically become Boolean functions of Boolean arguments in the mathematical expressions. Typical examples are $\wedge$, $\vee$, and $\Rightarrow$.
- Look for conjunctions combining parts of speech other than clauses. They will not translate into Boolean functions of Boolean arguments. In such cases, look for other words to guide the translation. The combination formed by the conjunction can often be translated as a set or as a sequence. See Section 6.2.4 for guidelines on translating English text in which conjunctions combine parts of speech other than clauses.
- Identify nouns or noun phrases in the plural form or as general singular references suggesting a plural meaning. The values of the corresponding (non-Boolean) variables or functions will often be sets or sequences.
- Look for the following words and phrases: all, for all, any, for any, each, every, there exists, there is, there are, for some, at least one. These usually indicate quantified expressions in the mathematical model.
- Be cautious in translating clauses with the verb to be (is, are). Such a clause can have any one of several meanings, each of which calls for a different translation into the Language of Mathematics. Some of these meanings are: membership in a class or category, a relation, introducing a predicate adjective or noun, introducing the description of a state. The meaning and translation into the Language of Mathematics depends on these differences and upon whether the nouns are in singular or plural form. General and useful guidelines exist for each of these various possibilities (see Sections 7.2 and 7.3).
- Those parts of the English text leading to the corresponding variables and functions are candidates for the definitions and descriptions of the variables and functions in question in their interpretations. Abbreviated versions of those parts of the English text are candidates for the names of the variables and functions (see Section 6.2.6).

Although the guidelines above are not absolute laws, they are very strong recommendations. Deviations from them are rare and should be considered and justified very carefully. Translating a noun phrase by a Boolean variable, for example, is likely to lead to logical problems in the mathematical model.

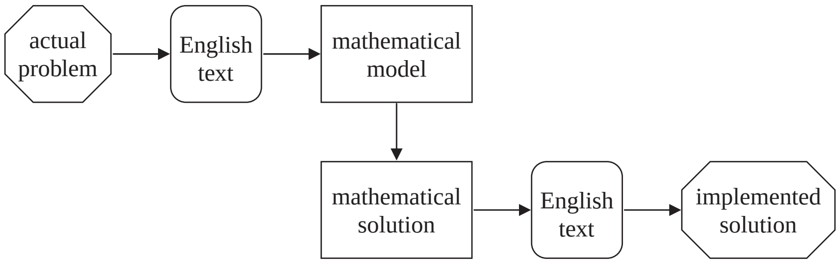
Readability of the mathematical model suggests corresponding naming conventions for variables and functions as well as grammatical conventions for their interpretations. In short, the name of a variable or a function with a Boolean value should be a stative verb phrase, abbreviated if desired, but the stative verb should always be included. The name of a variable or function with a non-Boolean value should be a noun phrase. The definitions and descriptions in their interpretations should be based on the same parts of speech (see Sections 6.13 and 6.2.6).

Because the Language of Mathematics has no action verbs, an action verb should never be the primary element in the name of any variable or function or in its interpretation [i.e., do not name the addition function “Add(…)”; name it “Sum(…)” instead]. Do not define it in the interpretation as “adds …” but, instead, as “is the sum of … .” Action verbs have their place as names of procedures in dynamic languages, such as many computer programming languages, but not in the static Language of Mathematics.

In summary, begin translating an English text into the Language of Mathematics by *understanding* the intended meaning of the text. Next, *distinguish* between statements expressing background information not needed in the mathematical model and statements to be translated into mathematical expressions in the model. In the latter, *nouns and noun phrases* are candidates for variables and functions in the model. The statements themselves should, if necessary, be reformulated into sentences expressing *relationships* between the values of the variables and functions already identified for inclusion in the mathematical model. The result of such reformulation will involve only things expressible in the universe of discourse of the Language of Mathematics. In particular, action verbs will have been eliminated. Either the *results* of the actions are described or the actions are expressed by *adjectives*, often in the form of past or present participles, or by nouns.

7.8 ACCURACY, ERRORS, AND DISCREPANCIES IN MATHEMATICAL MODELS

As pointed out in the introductory comments in Chapters 5 and 7, solving a problem in the real world involves translating or transforming descriptions of the problem and a specification of its solution in several steps:



In the diagram, horizontal arrows represent language translation or interpretation. The vertical arrow represents the derivation of the mathematical formulation of the solution from the mathematical model—a statement of the problem—both in the Language of Mathematics. In this step represented by the vertical arrow, mathematical expressions are manipulated mechanistically according to well-defined and unambiguous rules.

Especially in the last two stages of this process, the question arises—and should be answered explicitly and carefully—if the solution implemented or to be implemented solves the actual problem. In each step, but especially in the language translations represented by the horizontal arrows, errors, oversights, and misinterpretation can lead to an inadequate result. Discrepancies will then propagate in the subsequent steps, leading to a solution of a different problem than the original one. In some cases, the result will be a tolerable loss of accuracy, whereas in others, the result will be an unacceptable and unusable nonsolution.

This section describes typical sources of errors, discrepancies, and loss of accuracy in each of the five steps above:

1. Actual problem to the English text
2. English text to the mathematical model
3. Mathematical model to the mathematical solution
4. Mathematical solution to the English language specification of the solution
5. English specification to the solution actually implemented

The translation processes in steps 1, 4, and, to a somewhat lesser extent, 5 depend strongly on the particular application area and the language and jargon used in it, so will be dealt with here only to a limited extent, but the most important and common general sources of error are highlighted in Sections 7.8.1, 7.8.4, and 7.8.5.

The implemented solution in the general sense in the diagram above consists of both:

- The final embodiment of the solution in the form of mechanisms, devices, processes, procedures, computer software and hardware, and so on
- Associated documentation in a combination of English and the Language of Mathematics for several purposes: the construction of the final embodiment of the solution, its operational use in practice, and informing and training all people affected

Step 2 above, translating English text to the mathematical model, is the primary subject of this book. Section 7.8.2 lists the most common sources of errors in this step and briefly outlines ways of identifying and overcoming them. Details on how to avoid, identify, and overcome them constitute the subject of much of the rest of the book.

Step 3 above, transforming the mathematical model into a mathematical solution, is a step that, in principle, can be performed without error. When performed manually, human error can, of course, arise. Section 7.8.3 discusses this step in general, especially with regard to how errors can arise in it and how they can be avoided, identified, and eliminated. Basic information on how to perform this step is presented in Chapter 5. For more complete details on performing this exclusively abstract mathematical step, the reader should refer to the appropriate mathematical literature.

7.8.1 Errors Translating the Actual Problem into English

In this step, a general awareness of a problem is transformed into an English document. The document describes the context, nature, and key aspects of the problem and states the requirements that a solution must fulfill. In simple cases the "solution" consists of questions to be answered or statements to be verified. In more complex situations the solution can be a full specification of a machine, system, or procedures to be implemented in the application area.

The most important causes of errors and omissions in this step are:

- Inadequate communication among the people living and working in the application area in question and between them and consultants, mathematicians, and others assisting them
- An inadequate, incomplete analysis of the perceived problem

Initially, the actual problem is not articulated or formulated. It is only a general, mental idea, perhaps discussed verbally but only partially, among the parties involved. From this beginning, the problem must be defined consciously and explicitly. The key aspects of the perceived problem must be identified and systematically analyzed. The requirements that a solution must satisfy must then be determined. Finally, the results of these steps are expressed in a written document.

This process often requires communication between people with different areas of knowledge and expertise. To ensure effective communication, the knowledge and experience of adjacent pairs in the communication chain must overlap significantly; otherwise, they will not really understand each other sufficiently. The weakest connection in this chain will limit the overall effectiveness of the communication, just as a chain is no stronger than its weakest link.

The analysis performed in the process outlined above must take all relevant factors into account. If, for example, the task is to design a system or procedures, many aspects of man–machine interaction must be considered—psychological on the human side, technical on the system side. If some quantity is to be optimized, the informational needs of a suitable mathematical model must be identified. Psychological aspects of interpersonal communication must also be taken into account.

Special care must be taken in all parts of the process outlined above, because no basis exists for logically verifying the correctness and completeness of the final English text. Neither the initial perception of the problem nor the final English text is unambiguous, so there is no formal basis for systematically and logically comparing the two against each other; there is no formal, rigorous standard for correctness. Careful, extensive reviewing by the parties involved after some time has elapsed can help to achieve an adequate final document. Review also by independent persons often identifies oversights, omissions, misunderstandings, logical inconsistencies, and errors.

7.8.2 Errors Translating the English Text into a Mathematical Model

In this step, the English text describing the problem and the requirements that its solution must satisfy is translated into a mathematical model together with its accompanying interpretation (as defined in Section 6.13). The mathematical model basically restates—mathematically, precisely, rigorously, rationally, and unambiguously—the context, nature, and key aspects of the problem and the requirements that a solution must fulfill.

The main causes of errors and omissions arising in this step are:

- Ambiguities in the English text
- Inconsistencies (contradictions) in the English text
- Gaps in the English text, that is, information that is needed but not included in the text
- Insufficient familiarity with the problem and the application area on the part of the translator
- Insufficient familiarity with the relevant mathematics and lack of experience in applying mathematics on the part of the translator
- Lack of pedantic attention to detail when reading, interpreting, and understanding the English text

The correctness of the result of this step—the mathematical model—cannot be verified because the English text is not precise and unambiguous. The mathematical model can, however, be checked for internal consistency. Sometimes the entire expression constituting the mathematical model can be shown to be equal to the logical value false, indicating the presence of a contradiction (a logical inconsistency). This can be caused, in turn, either by a logical inconsistency in the original English text, by a misinterpretation of that text, or by a faulty translation. All of these causes are observed in actual practice.

The mathematical model, being written in the Language of Mathematics, is precise and unambiguous. It represents the first documentation of the problem and its possible solutions that can be used as a reference for correctness. Because of this, it is often useful to compare the mathematical model directly not only with the English text but also with the original perceptions of the problem. Interpreting the mathematical model in terms of the application area and discussing the interpretation with the people who will use the solution in their daily work can sometimes lead to the identification of errors, oversights, and omissions in the English text (i.e., to errors in the first step outlined in Section 7.8.1). It will also increase the confidence of all in the mathematical model.

7.8.3 Errors Transforming the Mathematical Model into a Mathematical Solution

The mathematical model is transformed into a mathematical description of a solution by a series of transformations of subexpressions in the mathematical model. This process consists of two parts: (1) deciding what series of transformations is to be applied to which subexpressions and (2) applying those transformations. Deciding on the strategy for transforming the expressions is a creative process in which experience and relevant intuition are helpful. Actually applying the transformations requires knowledge of the allowable transformations and some experience performing them, but it does not require creativity.

The decisions about which transformations to apply where and when can affect the efficiency with which the solution is obtained, but not its correctness (consistency with the original model). Particularly inept decisions regarding which transformations to apply where and when can, in the worst case, lead to the failure to find a solution, which will easily be recognized as such.

Permissible transformations on mathematical expressions are either well known or provable. Therefore, applying allowed transformations correctly can never cause an error in the final result. People can, however, make mistakes in applying transformation rules, and one must exercise considerable care to avoid such mistakes. One should also check the derivation of a solution (e.g., by having someone else verify the manipulations independently). An alternative approach is to give someone else the original mathematical model and the derived solution and ask that person to prove their consistency mathematically. In both of these cases, one should not give any other information to the verifier; that person does not need to know anything about the application area. If a pure mathematician cannot verify the mathematical correctness of the transformation of the original mathematical model into the mathematical statement of the solution, the transformation is not permissible.

The most frequent causes of errors when transforming mathematical expressions are (1) overlooking conditions on the application of the transformation rules and (2) errors in writing and, especially, copying expressions manually from line to line. An example of an error of the first type is dividing a numerical expression by another numerical expression without regard to whether or not the expression for the divisor can be zero (division by zero is not defined and, hence, is not an allowable transformation). Examples of errors of the second type are writing errors: for example, copying “x” in one line as “y” in the next, or copying “x^2” from one line into “x” in the next (a sloppy mistake made by the author once on a physics test).

If a solution involves continuous numerical variables and that solution is to be implemented in practice, it is important that tolerances are included on all parameters in the design. In the physical world, one can never manufacture anything to exactly any particular numerical measure, and similarly, one can never measure anything exactly. For example, if one attempts to manufacture bolts with a diameter of exactly 1 cm and the corresponding nuts to exactly 1 cm, about half the bolts can usually be expected to be slightly wider than 1 cm and half to be slightly narrower than 1 cm, and similarly for the nuts. Typically, then, about half of the combinations of a bolt and a nut will not fit because the bolt is slightly wider than the nut.

Similarly, in computer-based systems, numerical values are often rounded. This can and does lead to apparent errors. For example, if a column of percentage figures, each rounded to a certain number of decimal places, is added, the result calculated is not always 100.

If no tolerances are included in the original mathematical model, one should question their absence. One should seriously consider introducing them both into the original English description of the problem and into the mathematical model.

Numerical parameters that must be integers do not usually need to be associated with tolerances. Discrete items can be counted exactly. The counts can be added exactly, provided that no rounding is applied.

7.8.4 Errors Translating the Mathematical Solution into an English Specification

This translation step is generally less problematic and less error prone than the earlier translation step. Those subexpressions in the mathematical solution that were not present in the original mathematical model are translated one by one into English, using the definitions in the interpretation associated with the mathematical model (see Sections 6.13 and 7.8.2).

The most frequent sources of error are similar to those in the first step of translating the actual problem into an English statement of the problem (see Section 7.8.1). One must take special care to avoid ambiguities in the English specification, for example, by including the mathematical solution or extracts thereof as parts (e.g., appendices) of the English language document specifying implementation of the solution. If the mathematical solution includes a finite state machine, including a table representation of it in the English specification is helpful.

If the solution involves a physical device or system, some overall design decisions of an engineering nature are often made in this step. Corresponding attention must be given to the environment in which the solution to be implemented will be realized and will operate if this was not already included in the mathematical model. Potential timing problems must be foreseen and treated adequately in the English specification.

If the solution to be implemented involves continuous variables (physical systems typically do), design tolerances are essential, as discussed in the latter part of Section 7.8.3.

The mathematical solution is written in the Language of Mathematics and is therefore precise and unambiguous. To the extent that the English specification of the solution is or contains an engineering specification of the solution to be implemented, it will be relatively precise and unambiguous, so it and the mathematical solution can be, at least to a significant extent, formally verifiable against each other. Wherever the English specification of the solution is not formally verifiable against the mathematical solution, errors may have crept in or may creep in in the following step: actual implementation of the solution.

When the English specification of the solution is thought to be finished, it should be compared against all previous results, both English text and mathematical formulations, even though mathematically formal comparisons are not possible. The English specification of the solution should also be compared against the original subjective and not necessarily documented perception of the problem. Any discrepancies found should be eliminated by revising the documents and mathematical formulations appropriately.

7.8.5 Errors Implementing the English Specification of the Solution

The nature of this step depends greatly on the type of solution and the application, so only a few general comments on sources of errors and omissions are within the scope of this book. If the solution involves a physical device or system, implementing the solution is an engineering task, and expertise in the relevant engineering disciplines is needed. If the solution involves a computer-based system, corresponding expertise is required. In any case, knowledge, familiarity, and experience in the application area itself are also needed.

In all but the simplest cases the solution will include a set of procedures to be followed by people in an organization. In such situations, a frequent source of problems and errors in actual operation is insufficient motivation of the people to accept and use the system and inadequate information and training in how to use it. Involving such users of the system in the original conception and planning of the system usually helps considerably to motivate them. Appropriate documentation of the system and training in its use are usually essential for success. The documentation and training must be seen as part of the implementation.

Although the primary basis for implementing the solution is the English specification of the solution, the mathematical solution or parts of it are also often useful in implementing the solution. This can apply even when the solution is a set of procedures being followed by people. For example, tables representing finite state machines describing the behavior of part of the system or certain characteristics of the problem and its solution have been used beneficially in practice for both documentation and training.

8 Examples of Translating English to Mathematics

Each example below begins with an English language statement describing the problem and the requirements that the solution must satisfy. Notice that these statements are typically incomplete (i.e., some necessary information is omitted). Often, but not always, some of this information is suggested or implied.

The English language statement is examined in order to identify the variables and functions needed in the mathematical model. As pointed out in Section 6.13, these variables and functions, together with their meanings in the application domain, constitute the interpretation of the mathematical model. Thus, the interpretation connects the variables and functions in the mathematical model with the English language statement.

The parts of speech of key words in the English text guide the selection of variables and the types of values they take on. A noun or noun phrase will give rise to a variable or function that can take on certain types of non-Boolean values. A clause with a stative verb will lead to a Boolean variable, a Boolean function, or a Boolean expression. An adjective or noun which in effect abbreviates a clause with a predicate adjective or noun can also lead to a Boolean variable. A clause with an action verb (a nonstative verb) will sometimes represent a change of the value of a variable. Clauses with an action verb may describe actions performed by a person, in which case the *result* of the person's action, not the action itself, will be represented by a variable or function.

Finally, the relationships among the values of these variables—the mathematical model—are formulated in mathematical expressions. The mathematical model is developed term by term, expression by expression. The mathematical model represents the information contained in the English language statement, but in the precise, unambiguous Language of Mathematics.

In the process of reformulating the information in the English language text into expressions in the mathematical model, it is almost always necessary to resolve inconsistencies and ambiguities in the English language statement and to fill in missing information. In practice, this is normally done by asking the stakeholders in the final result appropriate questions. In studying the examples that follow, the reader should pay particular attention to how these situations are identified and how appropriate assumptions are made.

The examples below cover a wide range, from simple problems with short mathematical models to moderately complex design tasks with lengthy mathematical models. These examples also relate to a range of application areas, including business, economics, general logic, software, mathematics, reliability, and physical systems. Examples of important applications involving calculus—especially calculus in combination with probability theory—are not included because the knowledge of calculus needed is beyond the scope of this book. Such examples are found in many areas of science, engineering, economics, and business and finance (e.g., for analyzing financial instruments, derivatives, markets).

8.1 STUDENTS WITH THE SAME BIRTHDAY

Consider a class with many students. The instructor bets the students that two or more students have the same birthday. Write a mathematical expression that is true if this condition is met and false otherwise.

We begin by reading and understanding the foregoing text in English. Then we must convert the statement into a mathematical view. Finally, we must formulate the message in mathematics.

Strategy: Understand the meaning of the message.
Tactic: Identify all objects referred to in the message.
Tactic: Distinguish between essential objects and background information.

To identify all objects, one should begin by looking at the nouns and pronouns in the English text. In this case, we have “class,” “students,” “instructor,” and “birthday.” Of these, the condition to be expressed in mathematics clearly involves students and birthday. It is not stated explicitly that this condition involves the class also, but this seems to be strongly suggested by the context set by the first sentence: “Consider a class with many students.” Presumably the phrase “two or more students have the same birthday” means “two or more students *in the class* have the same birthday,” in which case the condition involves the class. If this is not the intended meaning of the task description, the question arises as to which group of students is meant: the entire student body in the school in question, all students in the world, or something else. The condition to be expressed in mathematics does not seem to involve the instructor, who is proposing the bet but who is not part of the condition being betted upon. We conclude, therefore, that the instructor is not one of the objects to be mentioned in the condition. The essential objects for the condition to be written in mathematics are, therefore: class, students, and birthday.

The question also arises as to whether or not the word “many” is essential: that is, whether it is part of the condition to be expressed in mathematics or whether it is general background information not of direct concern to the translator. We assume the latter here. If "many" were significant, this would suggest that we must include a term in the resulting mathematical expression to the effect that the number of students must be greater than some constant (an implicit object), raising in turn the question of what that constant should be. The latter uncertainty in the English text is another reason for assuming that the word "many" is not significant.

Strategy: Understand the meaning of the message.
Tactic: Identify relationships between essential objects referred to in the message.

To identify relationships between the essential objects, we begin by considering all combinations of two or more of these objects. These combinations are:

- class, students
- students, birthday
- class, birthday
- class, students, and birthday

Regarding the relationship between the objects class and students, the text begins "Consider a class with many students." In mathematical vocabulary, the reader would understand "consider a set (a class) of elements (the students)." Renaming "class" as C and a student as S, the relationship becomes, in mathematics, $S \in C$.

Regarding the relationship between the objects students and birthday, we "know" from the cultural context of human society that each student is a person and that each person has a birthday. Furthermore, each person has a unique birthday, so that in the mathematical world view birthday is a function of the person in question (i.e., the student). Renaming "birthday" as bd, the relationship becomes, in mathematics, bd(S).

Regarding the relationship between the objects class and birthday, there appears, again from the cultural context of human society, to be no direct relationship. One could consider the set of birthdays of the students in the class (e.g., $[\cup\ S : S \in C : \{bd(S)\}]$), but this is not referred to explicitly in the English text and from the statement of the task does not seem to be needed.

Regarding the relationship among the objects class, students, and birthday, there does not appear to be any direct relationship among all three of these objects; only the relationships above between pairs of these objects seem to be relevant.

Now we can turn to the condition to be expressed in mathematics. The English version of this condition is, after our addition to it: "Two or more students in the class have the same birthday." Thinking in terms nearer mathematical vocabulary, we can reexpress this condition as "for some student s1 in the class and some *other* student s2 in the class, bd(s1)=bd(s2)" (whereby we do not exclude the possibility that still other students have the same birthday). This sentence is in English syntax, English word order, and mixed English and mathematical vocabulary, so we must finish by reexpressing it once more into purely mathematical syntax, purely mathematical subexpression order, and purely mathematical vocabulary:

$$[\vee\ s1, s2 : s1{\in}C \wedge s2{\in}C \wedge s1{\neq}s2 : bd(s1){=}bd(s2)] \qquad [8.1\text{-}1]$$

Translator's glossary:	
	some $\leftrightarrow \exists, \vee$
	in $\leftrightarrow \in$
	other $\leftrightarrow \neq$
	same $\leftrightarrow =$

At this point the reader should review the mathematical expression above and verify that it is a correct translation of the condition in English into mathematics. Note especially that it properly handles the phrase "*or more*" in the English condition "two or more students in the class have the same birthday."

Potential pitfall: The devil lies in the details Notice especially carefully in the paragraph above that the phrase "two … students in the class" was reexpressed using the word "*other*" in the phrase "for some student s1 … and some *other* student s2." In this way the implicit but clear fact that different students are meant in English by "two" has been made explicit in the mathematical formulation. Without careful attention to such detail we might have left out the essential term "s1≠s2" in the solution above. Without this term, the expression would be wrong and it would be a complicated way of writing the logical constant true. Why would it always be true? If not always true, under what conditions would it be true?

Insufficient attention to detail is a common cause of errors in translating from English to mathematics.

8.2 CRITERION FOR SEARCHING AN ARRAY

In the subsections below, two versions of searching an array are considered: a simple search in which any occurrence of the value desired is identified, and a more specific search in which the first occurrence of the value desired in the array is identified.

8.2.1 Search for Any Occurrence of a Value in an Array

Consider an array D with index values ranging from 1 to n. The subject of this example is part of a specification for a procedure that will find the location in D of a particular given value. The goal of this exercise is to write a mathematical specification of the final state of the search: that is, a relation between the various relevant variables' values when the search is complete. The mathematical expression should indicate where in the array the given value is located.

The first activity in the process of translating from the description in English to an expression in the Language of Mathematics is to read the text in English. Then, that description must be understood in English and in terms of the array and the search.

Note below how the various ambiguities in the English text are identified and resolved in order to obtain the information needed to formulate the unambiguous mathematical expressions.

Strategy: Understand the meaning of the message.
Tactic: Identify missing information.
Tactic: Identify implicit—but not necessarily true—information.
Tactic: Identify special cases, especially ones not mentioned explicitly.

Sometimes, needed information is not given in the English statement of the problem. Often, special cases are not explicitly mentioned in an English description of the task and they are, therefore, overlooked. These discrepancies can lead to an incomplete translation in mathematics and consequent problems that can be difficult and time consuming to resolve.

The first question to be raised is: What does "index values ranging from 1 to n" mean? Inclusive, exclusive? In this case the range is most likely meant to be 1 to n inclusive. Had the wording been "between 1 and n," the meaning would be much less clear. From context, a reader would probably implicitly assume "between 1 and n inclusive," but the precise meaning of the English word "between" would actually imply between 1 and n exclusive. One must be constantly watchful for precisely this type of ambiguity.

To find a "particular given value," that value must be known. Within the context of a mathematical model, it will presumably be given in the form of the value of a particular variable.

Stop. Question: Is this a reasonable presumption? Why or why not? What other alternatives could be used?

The name of the variable will be needed, but it is not given in the English text above. Normally, the person writing the mathematical specification will ask the person who wrote the task description for the missing source of the given value or the name of the variable. We will assume that the answer is "key."

Strategy: Obtain all needed information.
Tactic: Ask the author of the task description.

Stop. Question: Scan the English text again. What variables are mentioned? Does the text give us all possible information about them? If not, what is missing? Do we really need all of the missing information?

The English text above explicitly mentions the array D and the variable n. It says nothing about the type of data values in D: whether they are integers, real numbers, names, elements of some other specified set, or single values or compound structures. In each of these possible cases, additional questions about the properties of the data arise, such as how small and how large the numbers can be, and how long the strings can be. In the case of the variable n, similar questions arise: in particular, whether n may be negative or zero, how large n may be, and whether or not n must be an integer.

Strategy: Obtain all needed information.
Tactic: Identify gaps in the description of the task.
Tactic: Read carefully, thoroughly, and precisely.
Tactic: Identify the data present, and ask questions about related details.
Tactic: Question whether missing information is really needed.
Tactic: Identify implicit "information."
Tactic: Question explicitly whether implied information may be assumed.

Implicit in the English description above of the task is that the value being sought is actually in the array. Sometimes this can be guaranteed, but more often than not, the value being sought cannot be guaranteed to be in the array. In the latter situation, which we assume here, we distinguish between the two cases in which (1) the value being sought is present in the array D and (2) the value being sought is not present in the array D.

Strategy: Divide and conquer.
Tactic: Distinguish between specific cases.

We distinguish between case 1,

$$[\vee\, i : i\in\mathbb{Z} \wedge 1\le i\le n : D(i) = key]$$

[case 1: value of key present in array D, 8.2.1-1]

and case 2;

$$\neg[\vee\, i : i\in\mathbb{Z} \wedge 1\le i\le n : D(i) = key]$$

[case 2: value of key not present in array D, 8.2.1-2]

Expression 8.2.1-2 is mathematically equivalent to

$$[\wedge\, i : i\in\mathbb{Z} \wedge 1\le i\le n : D(i) \neq key]$$

[case (2): value of key not present in array D, 8.2.1-3]

We must now decide what result is desired in each case.

When the value being sought is in array D, the task description above states that its location in D must be identified: that is, that its location must be specified in the mathematical expression. Again, a variable will presumably be needed for this purpose, so we must ask the author of the task description the name of the variable. We assume that the answer is “location”; that is, in case 1, D(location) = key and the value of the variable named “location” will be an integer between 1 and n inclusive.

If the value being sought is not in array D, the task description above says nothing about what the state of the various variables should be on completion of the search. Again, the author of the text should be asked what he or she meant in this case. We assume that the answer is that the value of the variable location should be outside the possible range in the first case above so that case 2 can be easily identified (e.g., the value of location might be n+1). For the sake of simplicity, we require that the value of location be n+1.

Strategy: Divide and conquer.
Tactic: Construct a table.

Combining the answers and assumptions above and presenting them in the form of a table yields

Case (in English):	case (1): key present in array D	case (2): key not present in array D
Case (in mathematics):	$[\vee\, i : i\in\mathbb{Z} \wedge 1\le i\le n : D(i) = key]$	$\neg[\vee\, i : i\in\mathbb{Z} \wedge 1\le i\le n : D(i) = key]$
Expression:	$location\in\mathbb{Z} \wedge 1\le location\le n \wedge D(location) = key$	$location = n+1$

The reader should now reread the English statement of the problem above and compare it with this table. Check word by word, phrase by phrase, and so on, for consistencies, inconsistencies, errors, omissions, and other discrepancies.

The expression desired is given unambiguously by the table above and can either be left in that form or written in an equivalent but more traditional form: for example,

$$\begin{aligned}&([\vee\ i : i\in\mathbb{Z} \wedge 1\leq i\leq n : D(i) = \text{key}] \Rightarrow (\text{location}\in\mathbb{Z} \wedge 1\leq \text{location}\leq n\\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad \wedge D(\text{location}) = \text{key}))\\ &\wedge (\neg[\vee\ i : i\in\mathbb{Z} \wedge 1\leq i\leq n : D(i) = \text{key}] \Rightarrow (\text{location} = n+1)) \qquad [8.2.1\text{-}4]\end{aligned}$$

which can be rewritten in any of several equivalent forms:

$$\begin{aligned}&(\text{location}\in\mathbb{Z}\ \wedge\ 1\leq \text{location}\leq n\ \wedge\ D(\text{location}) = \text{key})\\ &\vee \neg[\vee\ i : i\in\mathbb{Z}\ \wedge\ 1\leq i\leq n : D(i) = \text{key}]\ \wedge\ (\text{location} = n+1) \qquad [8.2.1\text{-}5]\end{aligned}$$

Stop. Mathematical exercise: Prove that expressions 8.2.1-4 and 8.2.1-5 are equivalent. (*Hint*: This is an exercise in manipulating Boolean algebraic expressions; no other analysis or reference to the application domain is needed.) Notice the relationships between the top line of expression 8.2.1-5 and the quantified expression in 8.2.1-4. See also the relevant identities in Section 5.2.4.

Notice that if location=n+1, the value of D(location) and, therefore, the term D(location)=key in the first line of expression 8.2.1-5 is not necessarily defined. This problem and possible solutions are discussed in more detail for this particular type of expression at the end of Section 8.2.2 and for undefined terms in expressions more generally in Section 3.5.3.

The reader should read and understand in detail the final mathematical expressions above and reread and reexamine the original English statement of the problem. Compare them to confirm that they are consistent and that mathematical expressions 8.2.1-4 and 8.2.1-5 above correctly specify all possible results of such a search.

8.2.2 Search for the First Occurrence of a Value in an Array

This task consists of the same requirements as the task in Section 8.2.1 with the added requirement that the element of D found to be equal to key must be the first such element in array D.

Strategy: Divide and conquer.

Tactic: Modularize.

Tactic: Use the results of previous exercises as components of solution to this task.

Translating the text above into mathematics can be done by splitting the task into two parts, one of which has already been solved, and adding the additional requirement. Therefore, we concentrate first on the additional requirement.

The additional requirement is that the element of the array found to be equal to key, that is, the element D(location), be the first such element in D. The question then arises: What is meant by “first”? Presumably this is to be interpreted that no element of D with a lower index value than location is equal to key. Implicit in this reasoning is the assumption that D(location) = key, that there is at least one element of D with the value key. Therefore, this analysis applies only to case 1 in the solution to the search in Section 8.2.1.

The question then arises: What does the additional requirement mean if case 2 applies (i.e., no element of D is equal to key)? The most reasonable interpretation of the phrase

> with the added requirement that the element of D found to be equal to key must be the first such element in D

in the previous task is that the condition “if any” is implied:

> with the added requirement that the element of D found to be equal to key—*if any*—must be the first such element in D

Adding the phrase “if any” to our intrepretation makes it clearer that in case 2 (no element of D is equal to key), the added requirement means nothing (i.e., in this case, the added requirement is redundant, unnecessary, and irrelevant and can, therefore, be disregarded).

Note how the English text has been examined in detail and vagueness and ambiguity identified and reduced to bring it closer to the world view of mathematics (i.e., more precise and less ambiguous).

Strategy: Close the gap between the English text and the Language of Mathematics.

Tactic: Reword the English text to bring it closer to mathematics.

Tactic: Reduce vagueness and ambiguity.

Tactic: Express implicit information explicitly.

We now return to the additional requirement and interpret it within the context of case 1 (at least one element of the array D has the same value as the variable key; see above). D(location) will be the *first* element of D with the value key if:

- D(location) = key (this condition is already part of the solution to the simple search in Section 8.2.1).
- All elements of D with an index less than location are different from (unequal to) key.

Therefore, if we take the solution to the simple search in Section 8.2.1 and impose (in case 1 only), the second bulleted condition above, we will have a solution to our current task.

Thus, we have reduced the current task to the subtask of translating

> "all elements of D with an index less than location are different from (unequal to) key"

into the Language of Mathematics. The word "all" suggests an and-series (universal quantification).

Translator's glossary: each, every, all, any ↔ for all, and (universal quantification, ∀, ∧).

Therefore, we translate the phrase above literally from English into

$$[\wedge\ \mathrm{i} : \mathrm{i}\in\mathbb{Z}\ \wedge\ 1\leq \mathrm{i}\leq \mathrm{n}\ \wedge\ \mathrm{i}<\mathrm{location} : \mathrm{D(i)}\neq\mathrm{key}] \qquad [8.2.2\text{-}1]$$

At this point one should ask: Have special cases, if any, been considered adequately? If the element of D with index 1 is equal to key (i.e., if location=1), there are no elements of D with smaller index values. The mathematical expression is correct for this case, the value of the additional expression being true because the universal quantification is over the empty set.

Combining the solution to the simple search in Section 8.2.1 with this added condition in case 1 only gives

Case (in English):	case (1): key present in array D	case (2): key not present in array D
Case (in mathematics):	$[\vee\ \mathrm{i} : \mathrm{i}\in\mathbb{Z} \wedge 1\leq \mathrm{i}\leq \mathrm{n} : \mathrm{D(i)} = \mathrm{key}]$	$\neg[\vee\ \mathrm{i} : \mathrm{i}\in\mathbb{Z} \wedge 1\leq \mathrm{i}\leq \mathrm{n} : \mathrm{D(i)} = \mathrm{key}]$
Expression:	$\mathrm{location}\in\mathbb{Z} \wedge 1\leq \mathrm{location}\leq \mathrm{n} \wedge \mathrm{D(location)} = \mathrm{key} \wedge [\wedge\ \mathrm{i} : \mathrm{i}\in\mathbb{Z} \wedge 1\leq \mathrm{i}\leq \mathrm{n} \wedge \mathrm{i}<\mathrm{location} : \mathrm{D(i)}\neq\mathrm{key}]$	$\mathrm{location} = \mathrm{n}+1$

which can be simplified to

Case (in English):	case (1): key present in array D	case (2): key not present in array D
Case (in mathematics):	$[\vee\ i : i \in \mathbb{Z} \wedge 1 \le i \le n : D(i) = key]$	$\neg[\vee\ i : i \in \mathbb{Z} \wedge 1 \le i \le n : D(i) = key]$
Expression:	$location \in \mathbb{Z} \wedge 1 \le location \le n \wedge D(location) = key \wedge [\wedge\ i : i \in \mathbb{Z} \wedge 1 \le i < location : D(i) \ne key]$	$location = n+1$

The expression represented by the table above can be expressed in more traditional form as in Section 8.2.1. Among the equivalent expressions (assuming that n is an integer, which seems to follow implicitly from the original statement of the task) is

$$
\begin{aligned}
&location \in \mathbb{Z} \wedge 1 \le location \le n+1\\
&\wedge [\wedge\ i : i \in \mathbb{Z} \wedge 1 \le i < location : D(i) \ne key]\\
&\wedge ((location \le n \wedge D(location) = key) \vee (location = n+1)) \qquad [8.2.2\text{-}2]
\end{aligned}
$$

Stop. Mathematical exercise: Prove that the expressions represented by the two tables and expression 8.2.2-2 are equivalent. (*Hint:* This is an exercise in manipulating Boolean algebraic expressions; no other analysis is needed.) Notice the relationships between the quantified expressions and the value of the variable location. See also the relevant identities in Section 5.2.4.

Notice that if location=n+1, the value of D(location) and, therefore, the term D(location)=key is not necessarily defined. This problem is also mentioned at the end of Section 8.2.1. This problem and possible solutions are discussed in more detail for this particular type of expression and for undefined terms in expressions more generally in Section 3.5.3. Probably the simplest solution here is to consider the function = to be extended such that if either argument is undefined, the value of the function = is false. Note that if the value of D(n+1) is defined, its value has no effect on the value of the expression above.

The reader should read and understand in detail the final mathematical expression above and reread and reexamine the original English statement of the problem. Compare them to confirm that they are consistent and that the mathematical expressions above correctly specify all possible results of such a search.

8.3 SPECIFYING THE INITIAL STATE OF A BOARD GAME

In this section the initial state of a particular board game is to be described mathematically. This description was used as a specification of part of a computer program to play the game against other programs as opponents. The programs were written by different teams in a university course in software development.

One type of error that arose in the early stages of writing this specification in the Language of Mathematics illustrates the confusion that can arise when one is not consciously aware of important linguistic characteristics of the languages one uses. In this case, the description was given in English. The students commonly used computer programming languages in their work. In this exercise they were to write a mathematical model in the Language of Mathematics. English permits one to describe both static and dynamic aspects of the subject matter. The computer programming languages that the students frequently used were strongly oriented to describing dynamic processes, not static characteristics or states. The Language of Mathematics is a language oriented exclusively to describing states and static characteristics of the topic. In their subconscious minds some of the students mixed the various concepts underlying these languages and wrote expressions in mathematical syntax but implicitly with the meaning and structure of a programming language. The result was incorrect in both languages.

Section 8.3.1 presents the process of translating correctly the English description to a specification of the initial state of the game formulated in the Language of Mathematics. Section 8.3.2 shows how the misunderstanding outlined above led to the wrong “solution.”

8.3.1 Initialization of a Game Board: A Correct Solution

Consider a game whose board consists of bowls, some of which contain stones. The players, who sit around a table, are numbered from 1 to NP counterclockwise. In front of each player are NB ordinary bowls placed in a line from left to right and one special bowl to their right. Initially, each ordinary bowl must contain five stones, and each special bowl must be empty.

The goal of this exercise is to write a mathematical expression describing the initial state of the bowls. The number of stones in each bowl will be the value of an element of a one-dimensional array named “StonesInBowl” with index values ranging from 0 to $\mathrm{NP}*(\mathrm{NB}+1)-1$. (Why is this formula for the highest index value correct?)

The first activity in the process of translating from the description in English above to the expression in mathematics is to read the text in English. Then, that description must be understood in English and in terms of the game board.

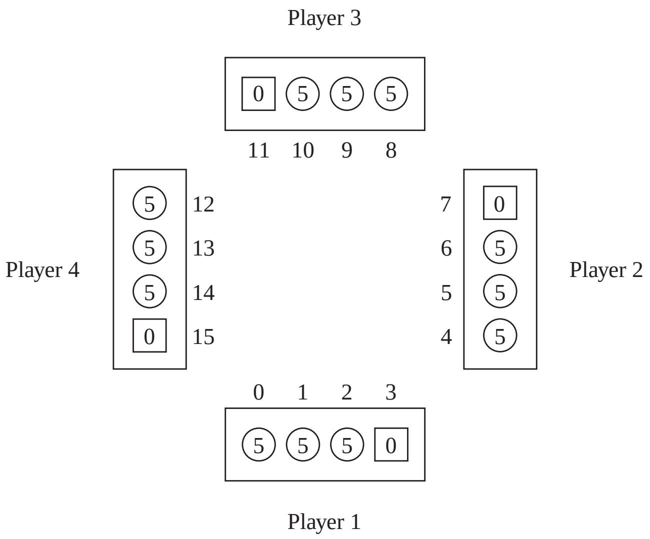
Strategy: Understand the meaning of the message.
Tactic: Draw a diagram.

Because it is more difficult to draw an unambiguous diagram for the general case of an unspecified number of players and of bowls than for a particular case, we assume a specific number of players and a specific number of bowls.

Tactic: Describe a specific instance of the general problem.

To reduce the possibility of confusion later, a different value should be selected for each of the relevant parameters. Avoid especially simple values such as 1. If feasible, select values for the different parameters that have no common divisor.

We select NB=3 and NP=4 and draw the corresponding diagram below. The circles represent the ordinary bowls. The squares represent the special bowls. The number in each bowl indicates the number of stones in the bowl. The numbers in the open space in the middle of the diagram are the indexes of the corresponding elements of the array StonesInBowl.



The reader should now reread the English statement of the problem above and compare it with this diagram. Check word by word, phrase by phrase, and so on, for consistencies, inconsistencies, errors, and other problems.

Next, we must understand the meaning in mathematical terms. The text of the problem statement and the meanings of the various aspects of the diagram (e.g., the circles, the squares) lead to such mathematical expressions as

$$\text{StonesInBowl}(0) = 5 \qquad [8.3.1\text{-}1]$$

$$\text{StonesInBowl}(1) = 5 \qquad [8.3.1\text{-}2]$$

$$\text{StonesInBowl}(2) = 5 \qquad [8.3.1\text{-}3]$$

$$\text{StonesInBowl}(3) = 0 \qquad [8.3.1\text{-}4]$$

…

$$\text{StonesInBowl}(10) = 5 \qquad [8.3.1\text{-}5]$$

$$\text{StonesInBowl}(11) = 0 \qquad [8.3.1\text{-}6]$$

etc.

Each of the expression above, as well as all other comparable terms, must be true [i.e., they must be connected with the logical and function ($\wedge$)].

Translator's glossary: each, every, all, any $\leftrightarrow$ for all, and (universal quantification, $\forall$, $\wedge$).

Because some of the bowls contain 5 stones and others, 0 stones, we will need two quantified subexpressions at the level of the elements of the array "StonesInBowl": one for the ordinary bowls and one for the special bowls.

The initial state of each player's collection of bowls is the same for every player. That is, by considering first the several players (not the bowls), we need only one quantified expression, which will presumably be simpler than the double quantification needed if we would start at the more detailed level of the elements of the array StonesInBowl, as outlined above. Then, within each player's collection of bowls, we would need another—but independent—quantification, leading to a simpler structure. We therefore apply another strategy to subdivide the one structurally complicated problem into two subproblems, each structurally simpler.

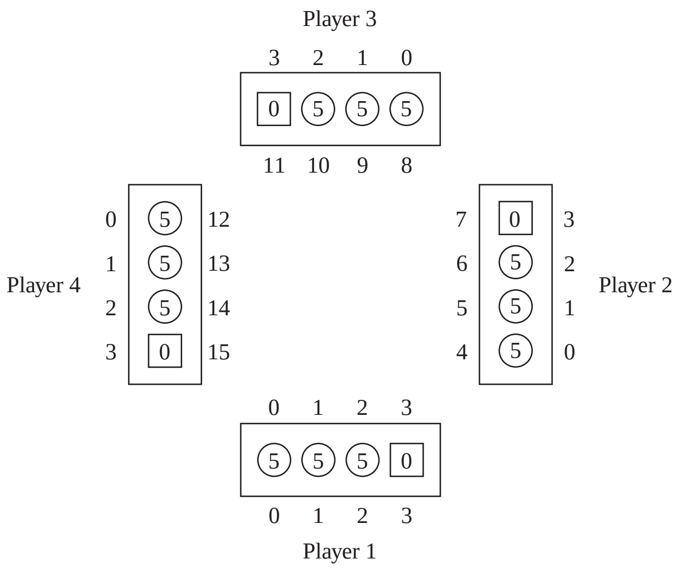
Strategy: Divide and conquer.
Tactic: Modularize. (Subdivide hierarchically.)
Tactic: Introduce an auxiliary mathematical function.

We name the desired mathematical function describing the initial state of all of the bowls "GameBoardInitialized." It is a function of the two parameters NP and NB. It must be true if all bowls are initialized correctly and false otherwise.

We introduce a mathematical function PlayerInitialized(p, NB), which will be true if the NB ordinary bowls and the special bowl in player p's collection of bowls are correctly initialized and false otherwise. GameBoardInitialized(NP, NB) will be true if and only if every player's bowls are initialized correctly, suggesting universal quantification (see the translator's glossary box above) over the players:

$$\text{GameBoardInitialized(NP, NB)} = [\wedge\ p : p\in\mathbb{Z} \wedge 1\le p\le \text{NP} : \text{PlayerInitialized(p, NB)}] \qquad [8.3.1\text{-}7]$$

To write an expression for PlayerInitialized(p, NB), we will need to refer to the individual bowls within each player's collection of bowls. Extending the diagram above to include such new numbers for the bowls in front of each player, we obtain the following diagram. The new numbers are outside the bowl areas.



Again, reread the English statement of the problem above and compare it with this diagram.

Strategy: Understand the meaning of the message.

Tactic: Express it in an intermediate language (e.g., mathematically styled English).

From this diagram we can write the following first draft of an expression for PlayerInitialized(p, 3):

the number of stones in player p's bowl 0 = 5 ∧
the number of stones in player p's bowl 1 = 5 ∧
the number of stones in player p's bowl 2 = 5 ∧
the number of stones in player p's bowl 3 = 0 [8.3.1-8]

The number of stones in each bowl is represented by the value of that bowl's element of the array StonesInBowl. We introduce, therefore, a new auxiliary function that converts the new bowl number b within player p's collection to that bowl's index in the array StonesInBowl. We will call this function "GBIndex" (for "global bowl index"). This function will obviously depend on player number p and bowl number b. Presumably, it will also depend on the game board parameter NB. Conceivably, it could also depend on NP, so we start with the parameter list (p, b, NB, NP) for this function. (It turns out below that the function GBIndex does not depend on NP, so we will drop that parameter later.)

Now we can reformulate expression 8.3.1-8 to obtain the following expression for PlayerInitialized(p, 3):

$$\begin{aligned}&\text{StonesInBowl}(\text{GBIndex}(p, 0, \text{NB}, \text{NP})) = 5 \wedge\\&\text{StonesInBowl}(\text{GBIndex}(p, 1, \text{NB}, \text{NP})) = 5 \wedge\\&\text{StonesInBowl}(\text{GBIndex}(p, 2, \text{NB}, \text{NP})) = 5 \wedge\\&\text{StonesInBowl}(\text{GBIndex}(p, 3, \text{NB}, \text{NP})) = 0\end{aligned}$$

[NP will be dropped later, 8.3.1-9]

or, for general NB,

$$\text{PlayerInitialized}(p, \text{NB})$$

$$=$$

$$[\wedge\ b : b\in\mathbb{Z} \wedge 0\le b\le \text{NB}-1 : \text{StonesInBowl}(\text{GBIndex}(p, b, \text{NB}, \text{NP})) = 5]$$
$$\wedge\ \text{StonesInBowl}(\text{GBIndex}(p, \text{NB}, \text{NB}, \text{NP})) = 0 \qquad [8.3.1\text{-}10]$$

To define the function GBIndex, we refer to the diagram above and note that (1) the value of this function increases by NB+1 whenever p increases by 1 and (2) the value of this function increases by 1 when b increases by 1. An expression for the function GBIndex must, therefore, have the form $p*(\text{NB}+1) + b + K$, where K is an appropriate constant. Note that this expression depends on p, b, and NB, but not on NP, so NP can be dropped from the list of parameters for the function GBIndex. K can be determined by considering the case p=1, b=0 for any NB (and any NP). Note that GBIndex(1, 0, NB) = 0 (see the diagram above):

$$\text{GBIndex}(1, 0, \text{NB}) = 0$$

$$=$$

$$1*(\text{NB}+1) + 0 + K = 0$$

$$=$$

$$K = -(\text{NB}+1)$$

Therefore, because GBIndex(p, b, NB) = p∗(NB+1) + b + K (see the paragraph above),

$$\text{GBIndex}(p, b, NB) = p*(NB + 1) + b - (NB + 1)$$

=

$$\text{GBIndex}(p, b, NB) = (p-1)*(NB + 1) + b \qquad [8.3.1\text{-}11]$$

Bringing together the several intermediate results above, the mathematical specification for the initial state of the board is

$$\begin{aligned}&\text{GameBoardInitialized}(NP, NB)\\&\quad = [\wedge\ p : p\in\mathbb{Z}\ \wedge\ 1\le p\le NP : \text{PlayerInitialized}(p, NB)] \qquad [8.3.1\text{-}7 \text{ repeated}]\end{aligned}$$

where

$$\begin{aligned}&\text{PlayerInitialized}(p, NB)\\&\quad = [\wedge\ b : b\in\mathbb{Z}\ \wedge\ 0\le b\le NB-1 : \text{StonesInBowl}(\text{GBIndex}(p, b, NB)) = 5]\\&\qquad \wedge\ \text{StonesInBowl}(\text{GBIndex}(p, NB, NB)) = 0\\&\hfill [8.3.1\text{-}10 \text{ repeated, NP dropped}]\end{aligned}$$

and where

$$\text{GBIndex}(p, b, NB) = (p-1)*(NB+1) + b \qquad [8.3.1\text{-}11 \text{ repeated}]$$

If desired, the expressions for PlayerInitialized and GBIndex above can be substituted for the function references to obtain a single, closed expression for GameBoardInitialized in terms of StonesInBowl only:

$$\begin{aligned}&\text{GameBoardInitialized}(NP, NB)\\&\quad = [\wedge\ p : p\in\mathbb{Z}\ \wedge\ 1\le p\le NP :\\&\qquad\quad [\wedge\ b : b\in\mathbb{Z}\ \wedge\ 0\le b\le NB-1 : \text{StonesInBowl}((p-1)*(NB+1)+b) = 5]\\&\qquad\quad \wedge\ \text{StonesInBowl}(p*(NB+1)-1) = 0] \qquad [8.3.1\text{-}12]\end{aligned}$$

At this point the reader should reread and understand (interpret again) the English statement of the problem and the translation above in the Language of Mathematics (the mathematical expressions) and compare them. There are various ways of doing this, for example: (1) compare the final mathematical expressions against the English text; (2) draw a diagram from the mathematical expressions and compare it with the English text and our original diagram; (3) select a number of sample cases and calculate the number of stones in the selected bowls by reference to both the English and the mathematics texts independently, and verify that the results are the same (both correctly and incorrectly initialized board states should be included in the selection of sample cases); (4) examine the mathematical expressions for syntactical correctness; (5) examine the mathematical expressions for semantic plausibility; (6) examine the mathematical expressions for compatibility (i.e., that the result of one function is consistent with the use of that result); and so on. Drawing a hierarchical diagram of the composition of the functions in the mathematical expressions aids in the last point.

Persons experienced in interpreting English problem statements into the Language of Mathematics would read the English text at the beginning of this section and would write the equations 8.3.1-7, 8.3.1-10, and 8.3.1-11 more or less directly. Substituting the right-hand side of equation 8.3.1-11 into equation 8.3.1-10 and then substituting the right-hand side of the resulting equation into equation 8.3.1-7 gives equation 8.3.1-12. However, even experienced translators from English into the Language of Mathematics would mentally go through most of the intermediate steps described above. Experience and practice do not enable one to skip intermediate steps in solving such problems; it only enables one to perform some of them mentally and to see in advance which steps are likely to lead in a promising direction and which are not. Less experienced persons are well advised to write down all steps in full detail.

8.3.2 Initialization of a Game Board: A Wrong “Solution”

As mentioned in the introductory paragraph in Section 8.3, students have experienced difficulty in solving the problem of specifying mathematically the initial state of the game board. It had never been explicitly pointed out to them that the languages they were accustomed to using frequently—computer programming languages—and the language they were to use for the specification—the Language of Mathematics—differ from each other linguistically in ways that are critical for the translation process. It had probably never even been pointed out to them before that they were translating from one language to another. They thought that they were doing mathematics.

Common error: Confusing static and dynamic descriptions, confusing a static mathematical expression and a dynamic procedure (e.g., a computer program)

Often, people accustomed to thinking in terms of dynamic procedures will “solve” this problem in a way that reflects the structure of one possible computer program for initializing the game board. In effect, they confuse the desired mathematical expression describing the final state with a step-by-step procedure for achieving that state. Such an erroneous approach is shown below.

Looking at the diagrams in Section 8.3.1, we see that one fairly simple strategy for a dynamic procedure to initialize the board would be first to set all elements of the array StonesInBowl to 5 and afterward, set the elements of the array representing the special bowls to 0. This thinking might lead one to write the following expression for GameBoardInitialized(NP, NB):

$$[\wedge\ i : i{\in}\mathbb{Z}\ \wedge\ 0{\le}i{\le}NP{*}(NB{+}1){-}1 : StonesInBowl(i) = 5]\ \wedge$$

[all bowls initialized]

$$[\wedge\ p : p{\in}\mathbb{Z}\ \wedge\ 1{\le}p{\le}NP : StonesInBowl(p{*}(NB{+}1){-}1) = 0]$$

[special bowls initialized]

[8.3.2-1]

Stop, question, and think: What is wrong with the expression above as a solution to the problem? Why does it not describe correctly the initial state of the game board?
What is wrong with this approach in general?
Answer these questions before proceeding.

In a correct initial state, every special bowl contains 0 stones and every ordinary bowl contains 5 stones. The first line in the expression above states that all bowls—ordinary and special—contain 5 stones each. The second line, which is anded with the first line, states that every special bowl contains 0 stones. Thus, the expression above states that every special bowl contains 5 stones and that every special bowl contains 0 stones. The two statements contradict each other (i.e., they cannot both be true). Expressed differently but equivalently, the expression above is always false for all positive NP and NB, even for a correctly initialized board, so the expression is not a solution to the problem. The expression above is an unnecessarily complicated way of writing the logical constant false.

Specific cases further illustrate this error. Consider, for example, the case NP=4, NB=3 (as in the diagrams in Section 8.3.1), p=1, and b=3 (the first player's special bowl). The index for this bowl in the array StonesInBowl is GBIndex(p, b, NB) = (p−1)∗(NB+1)+b = 3. Therefore, the number of stones in this bowl is StonesInBowl(3). The first line in expression 8.3.2-1 contains the term StonesInBowl(3)=5, and the second line contains the expression StonesInBowl(3) = 0. Thus, the entire expression is

$$\ldots \wedge\ StonesInBowl(3) = 5\ \wedge \ldots \wedge\ StonesInBowl(3) = 0\ \wedge \ldots$$

which is always false.

The members of the team that wrote the expression above thought in terms of a two-part program. The first part would initialize all bowls to 5. Afterward, the second part would initialize only the special bowls to 0. The first line in the expression above describes the board state (the values of the elements of the array StonesInBowl) after the execution of the first part of the program but before the execution of the second part. The second line of the expression describes part of the state (the values of the elements of the array StonesInBowl representing the special bowls only) after the execution of the second part of the program.

The first and second lines of expression 8.3.2-1 are intended to describe different states, but the mathematical expression as written does not do this. It combines the descriptions of the different states as if they were different parts of the same state. The result is a contradiction, an error.

8.4 PRICE DISCOUNTS

For various reasons, suppliers often give reduced prices to customers. The reductions of the price are called by various names, such as discounts, rebates, or in slang, "… % off." Discounts are also often expressed in the form of "free" additional quantities, such as "buy 2, get 1 free," or a new size, such as a new 1.5-liter bottle of a drink for the price of the old 1-liter bottle.

Several different discount structures are presented in the following sections.

8.4.1 Flat Discounts

The flat discount has the simplest structure of all forms of discount. The seller offers a certain percentage off the normal list price, independent of the quantity sold.

The variables in this model of a flat discount and their interpretation are:

$\mathbb{P}$:	the set of possible normal list prices.
ListPr:	the normal list price per unit purchased
DPct:	the discount in percent (%) of the normal list price
Discount:	the monetary amount of the discount per unit
NetPr:	the net price per unit (i.e. the price per unit after deduction of the discount)
Quan:	the quantity purchased
TotCost:	the discounted price for the quantity Quan

The set $\mathbb{P}$ is often the set of all positive numbers expressible exactly with two decimal digits in the fractional part (i.e., 3.67). In some cases three or more decimal digits in the fractional part are allowed.

Translating the sentence "The seller offers a certain percentage off the normal list price, independent of the quantity sold" leads directly to the mathematical expression

$$\text{Discount} = \text{ListPr} * \text{DPct}/100 \qquad [8.4.1\text{-}1]$$

See Section 7.4 for the translation of "percentage" and "percent."

Clearly implied is the fact that the net price to be paid is the normal list price less the discount, or in the Language of Mathematics,

$$\text{NetPr} = \text{ListPr} - \text{Discount} \qquad [8.4.1\text{-}2]$$

Combining expressions 8.4.1-1 and 8.4.1-2 gives the following expression for the value of the net price per unit:

$$\text{NetPr} = \text{ListPr} * (1 - \text{DPct}/100) \qquad [8.4.1\text{-}3]$$

Note the factor (1−DPct/100) in expression 8.4.1-3. It and variants of it recur frequently in expressions for the net price in terms of the list price in all mathematical models of price discounts.

A complete mathematical model for the flat discount, including the total cost for the quantity purchased, is, then,

$$\begin{aligned}&\text{ListPr} \in \mathbb{P} \wedge \text{DPct} \in \mathbb{R} \wedge 0 \leq \text{DPct} < 100 \wedge \text{Quan} \in \mathbb{N} \wedge \\ &\text{NetPr} = \text{ListPr} * (1 - \text{DPct}/100) \wedge \text{TotCost} = \text{Quan} * \text{NetPr} \wedge \\ &\text{Discount} = \text{ListPr} * \text{DPct}/100 \qquad [8.4.1\text{-}4]\end{aligned}$$

In this model the net price does not depend on the quantity purchased, hence the name *flat discount*. The total cost does, of course, depend on the quantity purchased.

The mathematical model above does not round net price, total cost, or discount to an allowed monetary amount. In practice, at least the total cost is rounded to an element of the set $\mathbb{M}$ of monetary amounts allowed. The set $\mathbb{M}$ can, but need not, be the same set as $\mathbb{P}$. For the sake of generality and simplicity, rounding will not be included in this model.

The mathematical model and interpretation above form a basis for the models in Sections 8.4.2 and 8.4.3, in which the discount rates vary with quantity.

8.4.2 Discount Rates Depending on Quantity

Often, the supplier offers different discount rates depending on the quantity ordered. Several different relationships between the discount rate and the quantity are possible and are found in practice.

Simple Range Dependency A common relationship between the discount rate and the quantity ordered is shown in the following example, in which the discount rate applicable to the entire order depends on the quantity ordered. If the quantity is in one range, a certain discount rate applies; if the quantity is in a different range, a different discount rate applies; and so on.

TABLE 8.4.2-1 Discount Rate as Function of Quantity

Quantity (Inclusive)	Discount Rate (%)	Price Factor (1 − Discount Rate/100)
1 to 100	3	0.97
101 to 200	7	0.93
201 or more	15	0.85

The mathematical model for this quantity discount structure is the model in Section 8.4.1 with the addition of the following expression:

$$\begin{aligned}&(1 \le \text{Quan} \le 100 \Rightarrow \text{DPct}=3) \wedge (101 \le \text{Quan} \le 200 \Rightarrow \text{DPct}=7) \wedge \\ &\quad (201 \le \text{Quan} \Rightarrow \text{DPct}=15) \wedge \end{aligned} \qquad [8.4.2\text{-}1]$$

Note that this discount structure can lead to a decrease in the total cost when the quantity ordered is increased across a quantity range border. For example, the total cost of an order for 100 units is 100∗ListPr∗0.97 monetary units, or 97∗ListPr. The total cost of an order for 101 units is 101∗ListPr∗0.93, or 93.93∗ListPr, less than the total cost of an order for 100 units.

Progressive Discount Rates If the supplier wants to avoid such reductions in total cost across quantity range borders, one convenient way to do so is to offer discount rates applying differently to the different quantity ranges. For example, one discount rate applies to the first 100 units, a different rate applies to the second 100 units, and a third rate applies to all units over 200. Using the rates given in Table 8.4.2-1, the total cost function would then be

$$\begin{aligned}&(1 \le \text{Quan} \le 100 \Rightarrow \text{TotalCost} = \text{Quan} * \text{ListPr} * 0.97) \wedge \\ &(101 \le \text{Quan} \le 200 \Rightarrow \text{TotalCost} = 100 * \text{ListPr} * 0.97 \\ &\qquad\qquad + (\text{Quan} - 100) * \text{ListPr} * 0.93) \wedge \\ &(201 \le \text{Quan} \Rightarrow \text{TotalCost} = 100 * \text{ListPr} * 0.97 + 100 * \text{ListPr} * 0.93 \wedge \\ &\qquad\qquad + (\text{Quan} - 200) * \text{ListPr} * 0.85) \wedge \end{aligned} \qquad [8.4.2\text{-}2]$$

The mathematical model for this progressive discount structure is based on the mathematical model given in expression 8.4.1-4. In full, it is as follows:

$$\begin{aligned}&\text{ListPr} \in \mathbb{P} \wedge \text{Quan} \in \mathbb{N} \wedge \\ &(1 \le \text{Quan} \le 100 \Rightarrow \text{TotalCost} = \text{Quan} * \text{ListPr} * 0.97) \wedge \\ &(101 \le \text{Quan} \le 200 \Rightarrow \text{TotalCost} = 100 * \text{ListPr} * 0.97 \\ &\qquad\qquad + (\text{Quan} - 100) * \text{ListPr} * 0.93) \wedge \\ &(201 \le \text{Quan} \Rightarrow \text{TotalCost} = 100 * \text{ListPr} * 0.97 + 100 * \text{ListPr} * 0.93 \\ &\qquad\qquad + (\text{Quan} - 200) * \text{ListPr} * 0.85) \wedge \\ &\text{NetPr} = \text{TotalCost}/\text{Quan} \wedge \text{Discount} = \text{ListPr} - \text{NetPr} \wedge \\ &\text{DPct} = 100 * \text{Discount}/\text{ListPr}\end{aligned} \qquad [8.4.2\text{-}3]$$

Other discount functions are also possible and are sometimes used in practice. Defining the discount rate as a continuous function of the quantity is a general approach, but such a definition is not easily expressed in a commonly understood form. All methods amount, in effect, to defining the discount rate to be a function of the quantity ordered; the differences are in the shape of the function and how it is formulated. A flat discount rate and a tabular definition of a varying discount rate as in this section are probably the most widely and easily understood forms for defining a discount structure.

8.4.3 Buy 2, Get 1 Free

The “buy x, get y free” form of defining a quantity discount is of psychological advantage only. It tries to delude the purchaser into believing that he or she is getting something for nothing, which, of course, is not the case. However, many people fall into the trap subconsciously. A more accurate description of this type of discount would be “get x+y for the list price of x.”

Mathematically and logically, the main difficulty with such a “definition” is interpreting precisely what is meant. The basic case is clear: “buy 2 and get 1 free” means that if you buy 3, you pay for 2 (i.e., the discount rate is 33.33… %). But if you order 2, or 4, or 7, or 8, etc., just what is the price to be paid?

Probably the most common interpretation of “buy 2, get 1 free,” and presumably the intended one, is that if you purchase a multiple of 3 units, the cost will be that multiple times the normal list price for 2. If you purchase only 1 or 2 units more than a multiple of 3, you pay the discounted price for the multiple of 3 units and the normal list price for each unit exceeding those in that multiple of 3. That is, if you purchase 7 units, and the normal list price per unit is 5 m.u. (monetary units), you pay 10 m.u. for the first group of 3, another 10 m.u. for the second group of 3, and another 5 m.u. for the seventh unit. The total price is 25 m.u.

Translated into a mathematical expression, the total cost is

$$\begin{aligned}\text{TotalCost} = {} & 2{*}\text{ListPr}{*}\text{integer}(\text{Quan}/3)\\ & + \text{ListPr}{*}(\text{Quan}-3{*}\text{integer}(\text{Quan}/3)) \qquad [8.4.3\text{-}1]\end{aligned}$$

where the value of the function integer is the argument if it is an integer, and is the next-lower integer if the argument is not an integer; that is, integer(5)=5, integer(4.999)=4, integer(5.2)=5, and so on. Expressed more mathematically, integer(z) is the largest integer such that integer(z)≤z.

In expression 8.4.3-1, integer(Quan/3) is the number of multiples of 3 in the quantity Quan, 3∗integer(Quan/3) is the number of units in that multiple of 3, and 2∗ListPr is the price to be paid for each multiple of 3. The value of the subexpression (Quan−3∗integer(Quan/3)) is the number of units purchased beyond the last multiple of 3, and ListPr∗(Quan−3∗integer(Quan/3)) is the cost of those units beyond the last multiple of 3.

Generalizing expression 8.4.3-1 for the “buy x, get y free” offer (i.e., “get x+y for the price of x”), the total cost for Quan units is

$$\begin{aligned}\text{TotalCost} = {} & x{*}\text{ListPr}{*}\text{integer}(\text{Quan}/(x+y)) \quad [\text{cost of the multiples of } (x+y)]\\ & + \text{ListPr}{*}(\text{Quan}-(x+y){*}\text{integer}(\text{Quan}/(x+y)))\\ & \qquad [\text{cost of the rest, } 8.4.3\text{-}2]\end{aligned}$$

One can view this discount scheme as offering two separate and separately priced products: (1) a package of x+y units at a price of x∗ListPr for each package and (2) single units at a price of ListPr for each unit. Expression 8.4.3-2 gives the price of the least expensive combination of the two products containing a total of Quan units.

The complete mathematical model for the “buy x, get y free” discount scheme is as follows. Notice how this mathematical model is based on expression 8.4.2-3 for the mathematical model in Section 8.4.2 and the total cost function in expression 8.4.3-2 in this section.

$$\begin{aligned}
&\text{ListPr} \in \mathbb{P} \wedge \text{Quan} \in \mathbb{N} \wedge x \in \mathbb{N} \wedge y \in \mathbb{N} \wedge \\
&\text{TotalCost} = x*\text{ListPr}*\text{integer}(\text{Quan}/(x+y)) \\
&\qquad\qquad + \text{ListPr}*(\text{Quan}-(x+y)*\text{integer}(\text{Quan}/(x+y))) \wedge \\
&\text{NetPr} = \text{TotalCost}/\text{Quan} \wedge \text{Discount} = \text{ListPr} - \text{NetPr} \wedge \\
&\text{DPct} = 100*\text{Discount}/\text{ListPr} \qquad [8.4.3\text{-}3]
\end{aligned}$$

8.5 MODEL OF A VERY SMALL ECONOMY

In this small economy there are four types of jobs (labeled J1, J2, J3, and J4) and people with 3 types of skills (labeled S1, S2, and S3). People with skill S1 are capable of performing jobs J1 and J2, but none other. People with skill S2 can perform jobs J2 and J3 only. People with skill S3 can perform jobs J3 and J4 only. The nature of the jobs is such that for every person performing job J1, three people are required for job J2. For every person performing job J2, two people are required for job J3. For every person performing job J3, three people are required for job J4. In this remote tribal group there are 84 people in total in the workforce, 4 people with skill S1, 6 people with skill S2, and 74 people with skill S3. Any one person can split his or her time between the jobs for which he or she is skilled. How many people will be unemployed?

To answer the question, a mathematical model of this economy must be formulated based on the given English description above. Implicit in the description above is the notion that a particular number of people with a particular skill are performing a particular job. This must be appended to the description above. These numbers are not given, but are to be determined. They will determine the value of the variable representing the answer to the question—the number of unemployed people.

The most frequent nouns in the description above are “person,” “people,” “skill,” and “job.” Three types of phrases modifying these nouns occur: “with skill S …,” “performing job J … ” and “required for job J … .” The nouns “skill” and “job” appear mainly in phrases related to the noun “person” or its plural, “people.” The numbers of people in the several categories also appear in the description. Therefore, most, perhaps all, of the variables in the model will be the numbers of each category of people.

The Variables and Their Interpretation From the English text above we deduce the variables in the model and their interpretations as given below. The numbers of people with each type of skill are given as constants in the description above, but they will also be assigned variable names so that the model can be generalized easily.

NSJ(*s*, *j*):	the number of people with skill s (s=1, 2 or 3) performing job j (j=1, 2, 3 or 4)
NS(*s*):	the number of people with skill s (s=1, 2 or 3) in the tribal group
NJ(*j*):	the number of people performing job j (j=1, 2, 3 or 4)
NTP:	the total number of people in the work force
NE:	the number of employed people
NU:	the number of unemployed people

Note that the sentence "People with skill S1 are capable of performing jobs J1 and J2, but none other" represents a condition formulated more mathematically as "the number of people with skill S1 performing job J3 is zero and the number of people with skill S1 performing job J4 is zero." Other sentences of this form are to be interpreted correspondingly. Note that all numbers must be nonnegative, but they need not be integers. Not stated but implied in the description of the economy are that:

- The number of people with a particular skill performing some job cannot be greater than the number of people with that skill in the tribal group.
- The total number of people with any skill performing any job cannot be greater than the total number of people in the workforce.
- The number of people employed is the number of people performing some job.
- The number of unemployed people is the total number of people in the workforce less the number of people employed.

The **mathematical model** is formulated by translating the sentences in the English description above, arranging them into a clear order, and inserting terms defining the sets for the values of the variables. The resulting mathematical model is

$$[\wedge\ s, j : s\in\{1, 2, 3\} \wedge j\in\{1, 2, 3, 4\} : NSJ(s, j)\in\mathbb{R} \wedge NSJ(s, j)\geqslant 0] \wedge \qquad [8.5\text{-}1]$$

$$[\wedge\ s : s\in\{1, 2, 3\} : NS(s)\in\mathbb{R} \wedge NS(s)\geqslant 0] \wedge \qquad [8.5\text{-}2]$$

$$NS(1) = 4 \wedge NS(2) = 6 \wedge NS(3) = 74 \wedge \qquad [8.5\text{-}3]$$

$$NTP = [+\ s : s\in\{1, 2, 3\} : NS(s)] \wedge \qquad [8.5\text{-}4]$$

$$[\wedge\ j : j\in\{1, 2, 3, 4\} : NJ(j) = [+\ s : s\in\{1, 2, 3\} : NSJ(s, j)]] \wedge \qquad [8.5\text{-}5]$$

$$NSJ(1, 3)=0 \wedge NSJ(1, 4)=0 \wedge \qquad [8.5\text{-}6]$$

$$NSJ(2, 1)=0 \wedge NSJ(2, 4)=0 \wedge \qquad [8.5\text{-}7]$$

$$NSJ(3, 1)=0 \wedge NSJ(3, 2)=0 \wedge \qquad [8.5\text{-}8]$$

$$NJ(2) = 3*NJ(1) \wedge NJ(3) = 2*NJ(2) \wedge NJ(4) = 3*NJ(3) \wedge \qquad [8.5\text{-}9]$$

$$[\wedge\ s : s\in\{1, 2, 3\} : [+\ j : j\in\{1, 2, 3, 4\} : NSJ(s, j)] \leq NS(s)] \wedge \qquad [8.5\text{-}10]$$

$$[+\ s, j : s\in\{1, 2, 3\} \wedge j\in\{1, 2, 3, 4\} : NSJ(s, j)] \leq NTP \wedge \qquad [8.5\text{-}11]$$

$$\text{NE} = [+\ \text{j} : \text{j}\in\{1, 2, 3, 4\} : \text{NJ(j)}]\ \wedge \qquad [8.5\text{-}12]$$

$$\text{NU} = \text{NTP} - \text{NE} \qquad [8.5\text{-}13]$$

Why are the terms NJ(j)$\in\mathbb{R}$, NJ(j)$\geq$0, NTP$\in\mathbb{R}$, and NTP$\geq$0 not needed in the mathematical model above?

The mathematical model above can be formulated in various equivalent ways, some of which are perhaps simpler for some purposes. For example, the subexpression 8.5-9 is equivalent to

$$\text{NJ}(2) = 3*\text{NJ}(1)\ \wedge\ \text{NJ}(3) = 6*\text{NJ}(1)\ \wedge\ \text{NJ}(4) = 18*\text{NJ}(1)\wedge \qquad [8.5\text{-}14]$$

and subexpressions 8.5-5 through 8.5-8 inclusive imply that

$$\begin{aligned}&\text{NJ}(1) = \text{NSJ}(1, 1)\ \wedge\\ &\text{NJ}(2) = \text{NSJ}(1, 2) + \text{NSJ}(2, 2)\ \wedge\\ &\text{NJ}(3) = \text{NSJ}(2, 3) + \text{NSJ}(3, 3)\ \wedge\\ &\text{NJ}(4) = \text{NSJ}(3, 4)\end{aligned} \qquad [8.5\text{-}15]$$

Thus, the variables NJ(j) are functions of the variables NSJ(s, j). The numerical values of the variables NS(s) are given. The values of all other variables in the model are determined by the values of NSJ(s, j) and NS(s). The values of NSJ(s, j) and NS(s) are the independent variables in this model. All other variables are dependent variables. The values of the variables NSJ(s, j) are subject to upper bounds imposed by the values of NS(s) (see subexpression 8.5-10).

The number of people unemployed (variable NU)—one of the dependent variables—depends on the values of NSJ(s, j), which are not given in the statement of the problem. Presumably they are to be determined, but no criterion is given. The simplest valid assignment of values for the NSJ(s, j) is 0 for all, in which case NE would be 0 and NU, 84, for an employment rate of 0% and an unemployment rate of 100%. For a short while, most people in this tribal group would be happy and content, until they all starved to death. Presumably, the intended question is not “How many people will be unemployed?” but “What is the minimum possible number of people who will be unemployed?” To answer this question, a (or the) combination of values of the variables NSJ(s, j) must be determined that minimizes the value of the variable NU.

The relationships between the relevant variables are illustrated in Table 8.5-1.

TABLE 8.5-1 Small Economy: Relationships Between Skills and Jobs

	Job j				Upper Bound
Skill s	1	2	3	4	NS(s)
1	NSJ(1, 1)	NSJ(1, 2)	0	0	4
2	0	NSJ(2, 2)	NSJ(2, 3)	0	6
3	0	0	NSJ(3, 3)	NSJ(3, 4)	74
	NSJ(1, 1) = NJ(1)	3∗NSJ(1, 1) = NJ(2)	6∗NSJ(1, 1) = NJ(3)	18∗NSJ(1, 1) = NJ(4)	84 NTP

Notice that the entire distribution of people across job types is determined by the value of one variable, either NJ(1) or NSJ(1, 1), whose values are equal, provided that the distribution can be met subject to the availability of the types of skill required. The availability of the three types of skills is shown in the rightmost column of Table 8.5-1.

As a example, consider the distribution of people given in the problem statement across the job types. If 84 people are to be assigned to the four job types, the value of NSJ(1, 1) must be 84/(1+3+6+18)) or 84/28, which is 3. Replacing the expressions in the cells in Table 8.5-1 by the values of those expressions gives Table 8.5-2.

TABLE 8.5-2 Small Economy: Attempted Full Employment

	Job j				Upper Bound
Skill s	1	2	3	4	NS(s)
1	3	1	0	0	4
2	0	8*		0	6
3	0	0			74
NJ(j) =	3	9	18	54	84 NTP

*Condition is not met; eight people with skill S2 are needed, but only six are available. Full employment is not possible.

Three of the four persons with skill S1 are assigned to job J1, leaving one person with skill S1 to be assigned to job J2. Eight more people are needed for job J2, but only six with the only remaining suitable skill S2 are available. Therefore, the 84 people with the skills given in the problem statement cannot be assigned to the four job types. Full employment cannot be achieved.

The question then becomes: How many people with the given skills can be assigned to the four job types? Job types J1 and J2 require skill types S1 and S2, which only 10 people in the tribal group have. Referring to the bottom cells in the columns for jobs J1 and J2 in Table 8.5-1, a new value of NSJ(1, 1) can be calculated so that NSJ(1, 1)+3∗NSJ(1, 1)=10 [instead of 12 (3+9) as in Table 8.5-2]. NSJ(1, 1) is, then, 10/4 or 2.5. Recalculating from Table 8.5-1 accordingly, we obtain Table 8.5-3 for our next attempt to assign people to jobs.

TABLE 8.5-3 Small Economy: Partial but Maximum Employment

	Job j				Actually Assigned,
Skill s	1	2	3	4	(Upper Bound NS(s))
1	2.5	1.5	0	0	4 (4)
2	0	6	0	0	6 (6)
3	0	0	15	45	60 (74)
NJ(j) =	2.5	7.5	15	45	70 (84)

Any greater employment would require more than 10 people with skills S1 or S2, but only 10 are available, so 70 is the greatest employment possible with the given skills in the workforce. The unemployment is 14, or 16.67% of the workforce. To achieve a greater employment than 70, more people with skills S1 or S2 and correspondingly fewer with skills S3 are needed.

The skill distribution required to enable full employment can be determined by repeating the calculation in Table 8.5-2, but without regard to the actual availability of the skill types. This is done in Table 8.5-4.

TABLE 8.5-4 Small Economy: Skills Required for Full Employment

	Job j				Skill Requirements for
Skill s	1	2	3	4	Full Employment, (NS(s))
1	3	1	0	0	4 (4)
2	0	8	0	0	8* (6)
3	0	0	18	54	72* (74)
NJ(j) =	3	9	18	54	84 (84)

*Reskilling two people from skill S3 to S2 will enable full employment.

Table 8.5-4 shows that all 84 people could be employed if two of the people with skill S3 were reskilled to S2. More generally, two people with skill S3 must be reskilled to S1 and/or S2 to achieve full employment. From the relations in the tables above, one can conclude that the precise condition for full employment of the 84 people in the workforce is that between three and 12 people inclusive have skill S1 and that between 54 and 72 people inclusive have skill S3.

This example illustrates that in an economy with specialization of tasks, the availability of the different types of skills must match the needs of the various tasks. If there is a mismatch, there will be unemployment. The greater the mismatch, the greater the unemployment. Just as a chain is no stronger than its weakest link, the capacity of an economy with linked activities is limited by the skill types in shortest supply.

Mathematically, the problem in this example is to minimize a function of several variables subject to constraints on the values of the variables. This is a classical optimization problem in mathematics. When all functions of the variables are linear, it is called *linear programming*. The subject of linear programming is beyond the scope of this book. An extensive mathematical literature on solving linear programming problems in general exists and has existed for many decades. Some commercial applications of linear programming involve mathematical models with thousands of variables and constraints. The techniques used to solve the problem in this example relied on the simple and specific structure of skill–job interactions and on the small size of the problem; for larger, general problems, they would be inefficient and even inadequate.

8.6 A LOGICAL PUZZLE

Many logical puzzles consist of several types of objects related in several ways. Some of the relationships are given and the remaining relationships are to be deduced by logically analyzing the information given. Usually, some of the information needed to solve the puzzle is implied, not stated explicitly. The following puzzle is of this type.

8.6.1 English Statement of the Puzzle

Six passengers on a flight from London to Singapore live in London, Paris, Madrid, Rome, Cairo, and Bombay. The following facts about these passengers are given:

- Arnold and the person from London are lawyers.
- Bill and the person from London disembarked from the flight at Rome.
- Charlotte and the person from Madrid are professors.
- The person from Bombay is older than Charlotte.
- Charlotte and the person from Cairo disembarked at Doha.
- Emily and the person from Paris are dentists.
- The person from Cairo is older than Arnold.
- Bill and Fred participated in Olympic Games, but the person from Madrid never did.

Match the names, professions, and home cities of these six people. How many solutions does this puzzle have?

8.6.2 Restatement of the Puzzle

Often, it is desirable to restate the problem in a more convenient form or structure. It will often simplify formulating the mathematical model if sentences conveying similar information are grouped together. Especially the types of information contained in the questions to be answered in the solution to the puzzle should be used in grouping the sentences.

The eight statements above giving various types of information refer to (1) names of the passengers, (2) home cities, (3) professions, (4) ages (“older than”), (5) cities where disembarked, and (6) participation in Olympic Games. Note that all of these are expressed in the form of noun phrases. All eight sentences refer to names of passengers and to home cities, so these types of information provide no criteria for grouping the sentences. Three sentences refer to professions. Two other sentences refer to ages of passengers. Still another two sentences refer to the city where passengers disembarked. The one remaining sentence refers to participation in Olympic Games. Therefore, the four types of information—profession, age, city of disembarkation, and participation in Olympic Games—provide convenient criteria for regrouping the eight sentences.

The questions to be answered by the solution to the puzzle refer to names, home cities, and professions. As concluded above, names and home cities are not useful for grouping the eight sentences, but references to professions do give us a criterion for grouping the sentences.

One way to resequence the eight statements above by these categories is as follows:

1. Arnold and the person from London are lawyers.
2. Emily and the person from Paris are dentists.
3. Charlotte and the person from Madrid are professors.
4. The person from Cairo is older than Arnold.
5. The person from Bombay is older than Charlotte.
6. Bill and the person from London disembarked from the flight at Rome.
7. Charlotte and the person from Cairo disembarked at Doha.
8. Bill and Fred participated in Olympic Games, but the person from Madrid never did.

8.6.3 General Assumptions

In puzzles of this type, certain assumptions are usually implied, even though they do not strictly follow from the text and, therefore, should be stated explicitly. In this puzzle, it is stated that there are six passengers and six cities they are from. In such a case, it is usually implied that each of the six passengers lives in only one city and that in each of the six cities named, only one of these passengers lives; that is, there is a one-to-one correspondence between the passengers and the cities in which they live.

When a phrase appears to refer to two people, it is usually implied that two different people are meant; for example, “Arnold and the person from London” will be interpreted to mean two people, Arnold and some other person who is from London. Together with the assumption that the people and the cities are in one-to-one correspondence, this implies that Arnold is not the passenger from London (i.e., that Arnold does not live in London).

If six different professions had been mentioned in the statement of the puzzle, one could usually assume a one-to-one correspondence between the passengers and their professions. In this puzzle, only three professions are mentioned, so this assumption is not justified. Here, more than one passenger can have the same profession. One can, however, assume that each passenger has only one profession—another “fact” not strictly following from the text but usually implied in statements of this type.

One should keep in mind the distinction between clearly stated facts and likely implications (assumptions). If finding the solution becomes very problematic or impossible, it may be necessary to revise the assumptions based on such typical implications of the English text.

The presence of the definite article “the” or of the indefinite article “a” or “an” can sometimes help in deciding which such implications are probably meant or not intended.

8.6.4 The Values, Variables, and Functions in the Mathematical Model

We begin translating the English statement of the puzzle into a model in the Language of Mathematics by identifying the values mentioned in the English statements, the types of those values and the relationships between them. The different types of values will suggest relevant sets of values to define in the mathematical model. The relationships between the values will suggest functions in the mathematical model.

Six values in the statement of the puzzle refer to cities in which the passengers live: Bombay, Cairo, London, Madrid, Paris, and Rome.

Five values in the statement of the puzzle are the names of passengers: Arnold, Bill, Charlotte, Emily, and Fred. The sixth passenger is not named in the statement of the puzzle. We will arbitrarily call him “Zeke” here. “Zeke” can be viewed mathematically as a value. Alternatively, “Zeke” can be viewed mathematically as the name of a variable whose value is unknown.

It was noted in Section 8.6.3 that the names of the passengers and the cities in which they live are in one-to-one correspondence. This correspondence can be represented by one of two functions:

- A function whose argument is a city and whose value is the name of the passenger living in that city
- A function whose argument is the name of a passenger and whose value is the city in which that passenger lives

When deciding which of these two alternatives to choose, one should look at the phrases in which the names and the home cities are mentioned. If one of these two types of values is always given, it will be simpler if that type of value is the argument, not the value, of the function selected. In the eight statements given above, all phrases connecting a name and a home city are of the form “person from London” (i.e., the person from a given city). No phrase is of the form “the place from which Arnold comes” or any other reference to an unstated city in which a named person lives. It will simply matter if the function is selected whose argument is the given value—a city—and whose value is the unknown—the name of the passenger living in that city. The value of the function will be the name of the person from the given city, so a suitable name of the function is “PersonFrom.”

Three values in the statement of the puzzle are the professions lawyer, dentist, and professor. In Section 8.6.3 it was stated that each passenger has only one profession but that more than one passenger can have the same profession. This relationship can be represented by a function whose argument is the name of the passenger and whose value is the profession practiced by that passenger. The relationship cannot be represented by the reverse, that is, cannot be represented by a function whose argument is a profession and whose value is the name of the person practicing that profession. Why not?

A suitable name for the function whose argument is the name of a passenger and whose value is the profession of that passenger is “ProfessionOf.”

One simple way of translating the phrases “… older than …” is to define a function “Age” whose argument is the name of a passenger and whose value is the age of that person. The “older than” relation can then be expressed as “Age(…)>Age(…).” Other possibilities include defining a Boolean function “IsOlderThan” with two arguments (the names of the two people whose ages are being compared) and a Boolean value, true if the first person named is older than the second, false otherwise.

The relationship between passengers and cities of disembarkation is clearly not one to one, because statement 6 states that two different people disembark at Rome (see also Section 8.6.3). Even though it is not stated explicitly, it seems reasonable to assume that any one person can disembark at only one city. This assumption excludes the possibility that someone disembarks at one city and then later takes another flight and disembarks at another city. Under this assumption, the relationship between a passenger and that passenger’s city of disembarkation can be represented by a function whose argument is the name of the passenger and whose value is the city where the passenger disembarked. A suitable name for this function is “CityWhereDisembarked.”

An experienced translator from English to the Language of Mathematics will, however, probably ask her/himself at this point whether or not a function such as “CityWhereDisembarked” is needed at all. This function is relevant to statements 6 and 7 only. The only information of use in these two statements appears to be that the four references to passengers are to four different people, and this information can be expressed without any reference to the cities of disembarkation. We will, however, use the function “CityWhereDisembarked” in the first translation into a mathematical model and will see how the first mathematical formulation of statements 6 and 7 can be used to deduce mathematically that the four references in question are to four different passengers. These comments apply also to the function “Age” above, but the situation there is complicated somewhat by the possibility that Arnold might be the person from Bombay or Charlotte might be the person from Cairo.

The last type of information to be considered is participation in Olympic Games. This type of information is present in statement 8 only and is not connected directly or indirectly with the information in any other statement. Thus, the comments in the paragraph above apply here, too. To illustrate the other way of handling such a situation, we will not define a function corresponding to participation in Olympic Games, but will translate statement 8 into a mathematical expression expressing only that the references to people refer to different people: in particular, that neither Bill nor Fred is the person from Madrid.

The functions above are not the only way of representing the relationships between names of passengers, cities of residence, their professions, their ages, the cities of their disembarkation, and their participation in Olympic Games. One could, for example, define for each profession a set whose elements are the names of the passengers practicing that profession. Because the elements are not known initially, but constitute part of the solution to the puzzle, this approach would presumably lead to a more complicated mathematical model. Similarly, one could define for each city a set whose (only) element is the name of the passenger living in that city. Still other alternatives are possible. One should try to select that alternative which leads to a mathematical model with a simple structure and which can be easily manipulated into a form giving a solution to the problem.

8.6.5 The Interpretation of Values, Variables, Functions, and Sets

Extracting the conclusions of the analysis and comments above leads to the following interpretation of the values, variables, functions, and sets in the mathematical model corresponding to the statement of the puzzle:

Arnold, *Bill*, *Charlotte*, *Emily*, and *Fred* are values and are also the names of five of the six passengers.

Zeke is the name of a variable whose value is the name of the sixth passenger.

$\mathbb{NamesPass}$ is the set of names of passengers:

$$\mathbb{NamesPass} = \{\text{Arnold, Bill, Charlotte, Emily, Fred, Zeke}\} \qquad [8.6.5\text{-}1]$$

Bombay, *Cairo*, *London*, *Madrid*, *Paris*, and *Rome* are values and are also the names of the cities in which the passengers live.

$\mathbb{HomeCities}$ is the set of cities in which the passengers live:

$$\mathbb{HomeCities} = \{\text{Bombay, Cairo, London, Madrid, Paris, Rome}\} \qquad [8.6.5\text{-}2]$$

PersonFrom is the function that maps a city to the name of the passenger who lives in that city:

$$\text{PersonFrom} : \mathbb{HomeCities} \to \mathbb{NamesPass} \qquad [8.6.5\text{-}3]$$

lawyer, *dentist*, and *professor* are values and are also the professions of the passengers.

$\mathbb{Professions}$ is the set of professions of the passengers:

$$\mathbb{Professions} = \{\text{lawyer},\ \text{dentist},\ \text{professor}\} \qquad [8.6.5\text{-}4]$$

ProfessionOf is the function that maps the name of a passenger to the profession which that passenger practices:

$$\text{ProfessionOf} : \mathbb{NamesPass} \to \mathbb{Professions} \qquad [8.6.5\text{-}5]$$

Age is the function that maps the name of a passenger to the age in years of that passenger:

$$\text{Age} : \mathbb{NamesPass} \to \mathbb{R} \qquad [8.6.5\text{-}6]$$

Doha and *Rome* are values and are also the cities in which some passengers disembark.

$\mathbb{DisembarkationCities}$ is the set of cities at which passengers disembarked:

$$\mathbb{DisembarkationCities} = \{\text{Doha, Rome}\} \qquad [8.6.5\text{-}7]$$

CityWhereDisembarked is the function that maps the name of a passenger to the city at which that passenger disembarked:

$$\text{CityWhereDisembarked} : \mathbb{NamesPass} \rightarrow \mathbb{DisembarkationCities} \qquad [8.6.5\text{-}8]$$

8.6.6 The Mathematical Model

The mathematical model consists of two parts: (1) expressions specifying the sets and functions of the model and (2) expressions relating the elements of the sets to one another. The latter expressions will contain references to the functions.

The first part of the mathematical model is a copy of the appropriate parts of the interpretation above, combined with the logical "and" function:

$$\mathbb{NamesPass} = \{\text{Arnold, Bill, Charlotte, Emily, Fred, Zeke}\} \wedge \qquad [8.6.5\text{-}1 \text{ repeated}]$$

$$\mathbb{HomeCities} = \{\text{Bombay, Cairo, London, Madrid, Paris, Rome}\} \wedge \qquad [8.6.5\text{-}2 \text{ repeated}]$$

$$\mathbb{Professions} = \{\text{lawyer, dentist, professor}\} \wedge \qquad [8.6.5\text{-}4 \text{ repeated}]$$

$$\mathbb{DisembarkationCities} = \{\text{Doha, Rome}\} \wedge \qquad [8.6.5\text{-}7 \text{ repeated}]$$

$$(\text{PersonFrom} : \mathbb{HomeCities} \rightarrow \mathbb{NamesPass}) \wedge \qquad [8.6.5\text{-}3 \text{ repeated}]$$

$$(\text{ProfessionOf} : \mathbb{NamesPass} \rightarrow \mathbb{Professions}) \wedge \qquad [8.6.5\text{-}5 \text{ repeated}]$$

$$(\text{Age} : \mathbb{NamesPass} \rightarrow \mathbb{R}) \wedge \qquad [8.6.5\text{-}6 \text{ repeated}]$$

$$(\text{CityWhereDisembarked} : \mathbb{NamesPass} \rightarrow \mathbb{DisembarkationCities}) \qquad [8.6.5\text{-}8 \text{ repeated}]$$

The second part of the mathematical model is a translation of the eight sentences in the restatement of the puzzle (see Section 8.6.2). This translation is most easily constructed sentence by sentence, translated independently of each other.

Sentence 1:

1. Arnold and the person from London are lawyers.

can be rephrased as "Arnold is a lawyer, the person from London is a lawyer, and Arnold is not the person from London." This can be written in the Language of Mathematics as

$$\begin{aligned}&\text{ProfessionOf(Arnold)}{=}\text{lawyer} \wedge \\ &\text{ProfessionOf(PersonFrom(London))}{=}\text{lawyer} \wedge \\ &\text{Arnold}{\neq}\text{PersonFrom(London)}\end{aligned} \qquad [8.6.6\text{-}1]$$

Similarly, sentence 2

2. Emily and the person from Paris are dentists.

can be rephrased as "Emily is a dentist, the person from Paris is a dentist, and Emily is not the person from Paris." In the Language of Mathematics this becomes

$$\begin{aligned}&\text{ProfessionOf(Emily)}=\text{dentist} \wedge\\ &\text{ProfessionOf(PersonFrom(Paris))}=\text{dentist} \wedge\\ &\text{Emily}\neq\text{PersonFrom(Paris)}\end{aligned} \qquad [8.6.6\text{-}2]$$

Sentence 3:

3. Charlotte and the person from Madrid are professors.

is translated in the same way to

$$\begin{aligned}&\text{ProfessionOf(Charlotte)}=\text{professor} \wedge\\ &\text{ProfessionOf(PersonFrom(Madrid))}=\text{professor} \wedge\\ &\text{Charlotte}\neq\text{PersonFrom(Madrid)}\end{aligned} \qquad [8.6.6\text{-}3]$$

Sentence 4 is

4. The person from Cairo is older than Arnold.

can be rephrased to "the age of the person from Cairo is greater than the age of Arnold" and, in turn, translated to

$$\text{Age(PersonFrom(Cairo))}>\text{Age(Arnold)} \qquad [8.6.6\text{-}4]$$

Sentence 5:

5. The person from Bombay is older than Charlotte.

can be translated in the same way to

$$\text{Age(PersonFrom(Bombay))}>\text{Age(Charlotte)} \qquad [8.6.6\text{-}5]$$

Sentence 6 is

6. Bill and the person from London disembarked from the flight at Rome.

which can be translated in the same way as sentences 1, 2, and 3 to obtain

$$\begin{aligned}&\text{CityWhereDisembarked(Bill)}=\text{Rome} \wedge\\ &\text{CityWhereDisembarked(PersonFrom(London))}=\text{Rome} \wedge\\ &\text{Bill}\neq\text{PersonFrom(London)}\end{aligned} \qquad [8.6.6\text{-}6]$$

Sentence 7:

7. Charlotte and the person from Cairo disembarked at Doha.

can be translated in the same way to obtain

$$\begin{aligned}&\text{CityWhereDisembarked(Charlotte)}=\text{Doha} \wedge\\ &\text{CityWhereDisembarked(PersonFrom(Cairo))}=\text{Doha} \wedge\\ &\text{Charlotte}\neq\text{PersonFrom(Cairo)}\end{aligned} \qquad [8.6.6\text{-}7]$$

Sentence 8:

8. Bill and Fred participated in Olympic Games, but the person from Madrid never did.

as pointed out earlier states in this context only that neither Bill nor Fred is the person from Madrid. In other words, Bill is not the person from Madrid, and Fred is not the person from Madrid. This can be expressed in the Language of Mathematics directly as

$$\text{Bill}\neq\text{PersonFrom(Madrid)} \wedge \text{Fred}\neq\text{PersonFrom(Madrid)} \qquad [8.6.6\text{-}8]$$

This completes the translation of the eight sentences in the statement of the puzzle. The eight parts must be combined with the logical "and" function to form the single mathematical expression corresponding to all eight English sentences.

The complete mathematical model is the logical "and" combination of the two parts above: the definitions of the sets from Section 8.6.5 and the functions and the translation above of the eight English sentences from this section:

$$\mathbb{NamesPass} = \{\text{Arnold, Bill, Charlotte, Emily, Fred, Zeke}\} \wedge \qquad [8.6.5\text{-}1 \text{ repeated}]$$

$$\mathbb{HomeCities} = \{\text{Bombay, Cairo, London, Madrid, Paris, Rome}\} \wedge \qquad [8.6.5\text{-}2 \text{ repeated}]$$

$$\mathbb{Professions} = \{\text{lawyer, dentist, professor}\} \wedge \qquad [8.6.5\text{-}4 \text{ repeated}]$$

$$\mathbb{DisembarkationCities} = \{\text{Doha, Rome}\} \wedge \qquad [8.6.5\text{-}7 \text{ repeated}]$$

$$(\text{PersonFrom} : \mathbb{HomeCities} \to \mathbb{NamesPass}) \wedge \qquad [8.6.5\text{-}3 \text{ repeated}]$$

$$(\text{ProfessionOf} : \mathbb{NamesPass} \to \mathbb{Professions}) \wedge \qquad [8.6.5\text{-}5 \text{ repeated}]$$

$$(\text{Age} : \mathbb{NamesPass} \to \mathbb{R}) \wedge \qquad [8.6.5\text{-}6 \text{ repeated}]$$

$$(\text{CityWhereDisembarked} : \mathbb{NamesPass} \to \mathbb{DisembarkationCities}) \wedge \qquad [8.6.5\text{-}8 \text{ repeated}]$$

$$\text{ProfessionOf(Arnold)}=\text{lawyer} \wedge$$

$$\text{ProfessionOf(PersonFrom(London))}=\text{lawyer} \wedge$$

$$\text{Arnold}\neq\text{PersonFrom(London)} \wedge \qquad [8.6.6\text{-}1 \text{ repeated}]$$

$$\text{ProfessionOf(Emily)}=\text{dentist} \wedge$$

$$\text{ProfessionOf(PersonFrom(Paris))}=\text{dentist} \wedge$$

$$\text{Emily}\neq\text{PersonFrom(Paris)} \wedge \qquad [8.6.6\text{-}2 \text{ repeated}]$$

$$\text{ProfessionOf(Charlotte)}=\text{professor} \wedge$$

$$\text{ProfessionOf(PersonFrom(Madrid))}=\text{professor} \wedge$$

$$\text{Charlotte}\neq\text{PersonFrom(Madrid)} \wedge \qquad [8.6.6\text{-}3 \text{ repeated}]$$

$$\text{Age(PersonFrom(Cairo))} > \text{Age(Arnold)} \wedge \qquad \text{[8.6.6-4 repeated]}$$
$$\text{Age(PersonFrom(Bombay))} > \text{Age(Charlotte)} \wedge \qquad \text{[8.6.6-5 repeated]}$$
$$\text{CityWhereDisembarked(Bill)} = \text{Rome} \wedge$$
$$\text{CityWhereDisembarked(PersonFrom(London))} = \text{Rome} \wedge$$
$$\text{Bill} \neq \text{PersonFrom(London)} \wedge \qquad \text{[8.6.6-6 repeated]}$$
$$\text{CityWhereDisembarked(Charlotte)} = \text{Doha} \wedge$$
$$\text{CityWhereDisembarked(PersonFrom(Cairo))} = \text{Doha} \wedge$$
$$\text{Charlotte} \neq \text{PersonFrom(Cairo)} \wedge \qquad \text{[8.6.6-7 repeated]}$$
$$\text{Bill} \neq \text{PersonFrom(Madrid)} \wedge \text{Fred} \neq \text{PersonFrom(Madrid)} \qquad \text{[8.6.6-8 repeated]}$$

Note that this expression is compact and terse. Having the English statement of the puzzle and the interpretation of the values, variables, and functions as a context, the mathematical model above is decipherable, but without that context, it would not be clear what it represents or means. It would not even be clear that it represents a puzzle. In fact, it could represent any one of several quite different things. On the other hand, this model can be transformed by a sequence of well-defined mechanistic steps to a solution of the puzzle, if one exists. The transformation is completely independent of the interpretation of the model; it depends only on the expressions in the model. If a solution does not exist, the inconsistency of the expressions, and hence of the original English statements from which they came, can be proved. That cannot be done with the English statement of the puzzle alone. This is a major advantage of the mathematical model.

8.6.7 Solving the Puzzle

The task is to “match the names, professions, and home cities of these six people” (see Section 8.6.1). Expressed in the terminology of the mathematical model at the end of Section 8.6.6, the task is to determine the functions PersonFrom and ProfessionOf. The latter part of the mathematical model above gives explicit information about these two functions, giving the values of the functions for some arguments and giving values that the functions do not take on for some arguments. Additional terms of these types can be deduced from other terms in the mathematical model above. For example, from the term Age(PersonFrom(Cairo))>Age(Arnold) one can conclude that Age(PersonFrom(Cairo))≠Age(Arnold) and, in turn, that PersonFrom(Cairo)≠Arnold. Why? (*Hint:* Reread the definition of a function in Section 3.3.) Also, the two terms ProfessionOf(Arnold)=lawyer and ProfessionOf(PersonFrom(Paris))=dentist in expressions 8.6.6-1 and 8.6.6-2, respectively, imply that Arnold≠PersonFrom(Paris). Applying this property of a function throughout expressions 8.6.6-1 through 8.6.6-8 leads to the following additional inequalities:

$$\begin{aligned}
&\text{Arnold}\neq\text{PersonFrom(Paris)} \wedge \text{Arnold}\neq\text{PersonFrom(Madrid)} \wedge \\
&\text{Emily}\neq\text{PersonFrom(London)} \wedge \text{Emily}\neq\text{PersonFrom(Madrid)} \wedge \\
&\text{Charlotte}\neq\text{PersonFrom(London)} \wedge \text{Charlotte}\neq\text{PersonFrom(Paris)} \wedge \\
&\text{Arnold}\neq\text{PersonFrom(Cairo)} \wedge \\
&\text{Charlotte}\neq\text{PersonFrom(Bombay)} \wedge \\
&\text{Bill}\neq\text{PersonFrom(Cairo)} \wedge \\
&\text{Charlotte}\neq\text{PersonFrom(London)} \qquad [8.6.7\text{-}1]
\end{aligned}$$

Expression 8.6.7-1 can be appended to the complete mathematical model at the end of Section 8.6.6 with the logical “and” function. This is allowed because the mathematical model implies the additional expression above and one of the general identities of Boolean algebra (expression 5.2.4-19) states that

$$(X \Rightarrow Y) \Rightarrow (X = (X \wedge Y)) \qquad [5.2.4\text{-}19 \text{ repeated}]$$

which in English means “if the truth of X implies the truth of Y, then X and X∧Y always have the same value.”

Thus, “and”ing the additional expression to the mathematical model introduces no new information, but it does introduce terms that will be useful in determining the function PersonFrom. For example, among the terms in the mathematical model and in the additional expression 8.6.7-1 are terms stating that the person from Madrid is neither Arnold, Bill, Charlotte, Emily, nor Fred. The person from Madrid must be one of the six passengers, and the only one left is Zeke. So the mathematical model above also implies that Zeke=PersonFrom(Madrid).

Continuing this type of analysis leads to a complete determination of the functions PersonFrom and ProfessionOf, but a more systematic way of structuring the analysis is desirable. Representing the functions PersonFrom and ProfessionOf in tabular form gives a systematic overall view of the process of deriving the two functions from the mathematical model and, in particular, identifies clearly and visually what still needs to be determined at each step of finding the solution.

The tables below illustrate the function PersonFrom in the upper part of each table and the function ProfessionOf in the lower part of each table. As pointed out earlier, the names of passengers and their home cities are in one-to-one correspondence, so the functional relationship goes in both ways; given a name, the home city is determined uniquely, and given a home city, the name is determined uniquely. In mathematical terminology, one says that the function PersonFrom has an inverse. This is not the case with the function ProfessionOf. Given a name, the profession is determined uniquely, but given a profession, the name is not determined uniquely. Statements 8.6.6-1, 8.6.6-2, and 8.6.6-3 in the mathematical model state that each profession has two practitioners.

In the tables below, information about each function is represented by symbols in the cells. Each cell corresponds to a value of the argument of the function and to a value of the function. Where it is known that a particular value of the function does not correspond to a particular value of the argument, the symbol × is entered in the corresponding cell. Where the value of the function is known for a particular value of the argument, the symbol ⮐̂ or ↵ is entered in the corresponding cell, the symbol ⮐̂ for the "two-way" function PersonFrom and the symbol ↵ for the function ProfessionOf.

Entering the appropriate symbols in the tables for the terms

$$\text{ProfessionOf(Arnold)=lawyer} \wedge \text{Arnold}\neq\text{PersonFrom(London)} \wedge$$
$$\text{Arnold}\neq\text{PersonFrom(Paris)} \wedge \text{Arnold}\neq\text{PersonFrom(Madrid)}$$

from the mathematical model and the additional expression 8.6.7-1 gives the following table:

	Function PersonFrom					
Home City	Arnold	Bill	Charlotte	Emily	Fred	Zeke
Bombay						
Cairo						
London	×					
Madrid	×					
Paris	×					
Rome						

	Function ProfessionOf					
Profession	Arnold	Bill	Charlotte	Emily	Fred	Zeke
lawyer	↵					
dentist						
professor						

Because ProfessionOf(Arnold)=lawyer, the value of the function ProfessionOf for the argument Arnold cannot be anything else. Therefore, the cells for all other values (dentist, professor) of the function ProfessionOf and the argument Arnold may be eliminated by entering the symbol × in them. The table then becomes

	Function PersonFrom					
Home city	Arnold	Bill	Charlotte	Emily	Fred	Zeke
Bombay						
Cairo						
London	×					
Madrid	×					
Paris	×					
Rome						

	Function ProfessionOf					
Profession	Arnold	Bill	Charlotte	Emily	Fred	Zeke
lawyer	↵					
dentist	×					
professor	×					

This is a general rule: Whenever the value of a function is known for a given argument, an × can be inserted in all cells corresponding to other values of the function and the same argument. In the case of the two-way function PersonFrom, this rule applies both ways; that is, when the symbol ↵ for a function value is inserted a cell, the symbol × may be inserted in all other cells in the same row and all other cells in the same column.

If one scans the mathematical model and the additional expression for all terms of the form value=function(argument) or value≠function(argument) and fills in the corresponding cells as described above, the following table results:

	Function PersonFrom					
Home city	Arnold	Bill	Charlotte	Emily	Fred	Zeke
Bombay			×			
Cairo	×	×	×			
London	×	×	×	×		
Madrid	×	×	×	×	×	
Paris	×		×	×		
Rome						

	Function ProfessionOf					
Profession	Arnold	Bill	Charlotte	Emily	Fred	Zeke
lawyer	↵		×	×		
dentist	×		×	↵		
professor	×		↵	×		

For the home city Madrid, the cells for five of the six passengers have been crossed out, so the only remaining one, Zeke, must be from Madrid. Similarly, for the passenger Charlotte, the cells for five of the six home cities have been crossed out, so the only remaining one, Rome, must be Charlotte’s home. Inserting the symbol ↵ in the cell for Zeke and Madrid and in the cell for Charlotte and Rome, and crossing out the other cells in the column for Zeke and in the row for Rome, one obtains the following table:

	Function PersonFrom					
Home city	Arnold	Bill	Charlotte	Emily	Fred	Zeke
Bombay			×			×
Cairo	×	×	×			×
London	×	×	×	×		×
Madrid	×	×	×	×	×	⮥
Paris	×		×	×		×
Rome	×	×	⮥	×	×	×

	Function ProfessionOf					
Profession	Arnold	Bill	Charlotte	Emily	Fred	Zeke
lawyer	↵		×	×		
dentist	×		×	↵		
professor	×		↵	×		

From this table one can see that Fred must be from London. Filling in the cell for Fred and London and crossing out the other cells in that column, it then follows that Bill must be from Paris; Arnold, from Bombay; and Emily, from Cairo. The table then becomes

	Function PersonFrom					
Home city	Arnold	Bill	Charlotte	Emily	Fred	Zeke
Bombay	⮥	×	×	×	×	×
Cairo	×	×	×	⮥	×	×
London	×	×	×	×	⮥	×
Madrid	×	×	×	×	×	⮥
Paris	×	⮥	×	×	×	×
Rome	×	×	⮥	×	×	×

	Function ProfessionOf					
Profession	Arnold	Bill	Charlotte	Emily	Fred	Zeke
lawyer	↵		×	×		
dentist	×		×	↵		
professor	×		↵	×		

The function PersonFrom has been determined completely. To complete the table for the function ProfessionOf, refer to the terms of the form ProfessionOf (PersonFrom(…))=… in statements 8.6.6-1, 8.6.6-2, and 8.6.6-3. The person from London, Fred, is a lawyer. The person from Paris, Bill, is a dentist. The person from Madrid, Zeke, is a professor. This completes the table for the function ProfessionOf:

	Function PersonFrom					
Home city	Arnold	Bill	Charlotte	Emily	Fred	Zeke
Bombay	⮤	×	×	×	×	×
Cairo	×	×	×	⮤	×	×
London	×	×	×	×	⮤	×
Madrid	×	×	×	×	×	⮤
Paris	×	⮤	×	×	×	×
Rome	×	×	⮤	×	×	×

	Function ProfessionOf					
Profession	Arnold	Bill	Charlotte	Emily	Fred	Zeke
lawyer	↵	×	×	×	↵	×
dentist	×	↵	×	↵	×	×
professor	×	×	↵	×	×	↵

That is, the solution to the puzzle is:

- Arnold is from Bombay and is a lawyer.
- Bill is from Paris and is a dentist.
- Charlotte is from Rome and is a professor.
- Emily is from Cairo and is a dentist.
- Fred is from London and is a lawyer.
- Zeke (the person whose name is not given in the statement of the puzzle) is from Madrid and is a professor.

During the process of determining the two functions, no alternative choice arose. Therefore, the solution above is the only one.

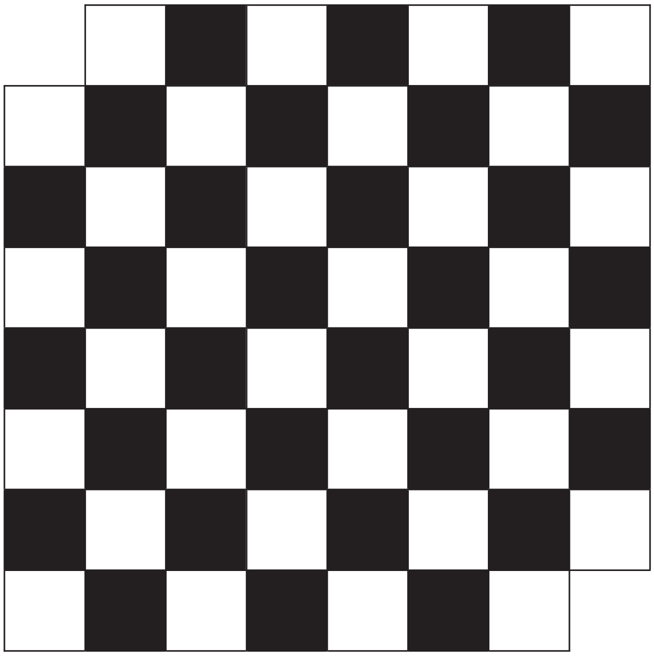
The sequence of steps in the analysis leading to this solution is not, however, unique. If, for example, one had not identified the inequalities in the additional expression implied by the mathematical model, it would have been necessary to complete the tables for the functions by examining interactions between the functions. One such situation might have been that at one point in the analysis, both Fred and Charlotte were still candidates for being from London, but the mathematical model states that the person from London is a lawyer and that Charlotte is a professor, from which one would conclude that Fred (and not Charlotte) is from London and that Fred is a lawyer. Such steps in the analysis are more difficult to identify and to examine and one is more likely to make errors in them.

Finally, one should verify that the solution above satisfies the statement of the puzzle. If it did not, that would indicate that an error had been made somewhere in the process of determining the solution to the puzzle. The solution above does, in fact, satisfy the statement of the puzzle.

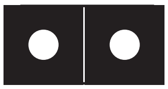
8.7 COVERING A MODIFIED CHESS BOARD WITH DOMINOES

This problem is posed as a logical puzzle, but it has direct application to planning the laying of certain kinds of tiles in a given pattern. The solution technique uses the concept of an *invariant*, which is useful in proving a wide variety of theorems in mathematics.

Given is a chess board consisting of 64 squares arranged in an 8 by 8 pattern. The squares are colored white and black alternately, so that squares with a common border are colored differently. Two diagonally opposite corner squares are cut out and removed, leaving 62 squares remaining on the modified chess board.



In addition, dominoes are available, each of which can be placed onto two adjacent squares of the chess board:



The question is: Can the chess board with the two missing corner squares be completely covered with dominoes, whereby no two dominoes may overlap?

At first glance it appears possible, because each domino covers two squares and the number of squares to be covered is even.

As dominoes are placed one by one on the board, the state of coverage changes. That state can be defined in detail by indicating which squares are covered and which are not after each domino is laid down.

By examining the domino and the chess board, it is clear that each domino laid down must cover one white square and one black square. It is not possible to lay a domino down so that it covers two white squares or two black squares. The number of uncovered white squares will, therefore, always be equal to the number of white squares on the board less the number of dominoes laid down, and correspondingly for the black squares. This suggests a coarser and simpler indication of the state of coverage of the board at each step of laying down the dominoes: the number of uncovered white squares and the number of uncovered black squares.

Variables and Their Interpretation The noun phrases in the text above lead to the following non-Boolean variables in the mathematical model and their interpretation. The condition that the board is completely covered—the answer to the question above—leads to the last variable listed below.

NWhite:	the number of white squares not covered by dominoes
NBlack:	the number of black squares not covered by dominoes
NDominoes:	the number of dominoes placed on the board
NWhiteOnBoard:	the number of white squares on the board
NBlackOnBoard:	the number of black squares on the board
BoardIsCovered:	All squares on the board are covered by dominoes.

Mathematical Model Translating the essential points in the English text above leads to the following mathematical model.

$$\begin{aligned}&\mathrm{NWhite}\in\mathbb{Z} \wedge \mathrm{NBlack}\in\mathbb{Z} \wedge \mathrm{NDominoes}\in\mathbb{Z} \wedge\\ &\mathrm{NWhiteOnBoard}\in\mathbb{Z} \wedge \mathrm{NBlackOnBoard}\in\mathbb{Z} \wedge\\ &\mathrm{BoardIsCovered}\in\mathbb{B} \wedge\\ &\mathrm{NWhiteOnBoard}=32 \wedge \mathrm{NBlackOnBoard}=30 \wedge\\ &\mathrm{NWhite}=\mathrm{NWhiteOnBoard}-\mathrm{NDominoes} \wedge\\ &\mathrm{NBlack}=\mathrm{NBlackOnBoard}-\mathrm{NDominoes} \wedge\\ &\mathrm{BoardIsCovered} = (\mathrm{NWhite}=0 \wedge \mathrm{NBlack}=0)\end{aligned} \qquad [8.7\text{-}1]$$

The mathematical model in expression 8.7-1 implies that

$$(\mathrm{NWhite}-\mathrm{NBlack}=2) \wedge \mathrm{BoardIsCovered} = (\mathrm{NWhite}-\mathrm{NBlack}=0) \qquad [8.7\text{-}2]$$

which, in turn, implies that

$$\text{BoardIsCovered} = \text{false} \qquad [8.7\text{-}3]$$

independent of the number of dominoes on the board. Therefore, it is impossible to cover the board described above by dominoes.

The expression NWhite−NBlack=2 above is always true, regardless of the values of the other variables. Such an expression is called an *invariant* in mathematics. Invariants are often helpful in proving a theorem.

The reader should consider under what conditions it is impossible and possible to cover a chess board with squares cut out, as well as similar boards of other dimensions. An invariant of the type above is useful in identifying configurations of a board that cannot be covered with dominoes. It must still be proved that other configurations can be covered, but in essentially all such cases, this is easily done.

8.8 VALIDITY OF A PLAY IN A CARD GAME

This example deals with the logic of one part of a card game: determining if a proposed play satisfies the rules of the game. The game is played with a normal deck of 52 different cards. Each card is marked with a "number" (2 to 9, J, Q, K, or A) and with a suit (clubs ♣, diamonds ♢, hearts ♡, or spades ♠). Cards of the suits clubs and spades are black in color and the cards of the other suits (diamonds and hearts) are red.

Cards are dealt to the several players, each player receiving the same number of cards. Cards left over, if any, are placed face up in a single pile in the center of the table. Only the top card in the pile shows. Then each player, one after the other in turn, plays a card from his or her hand onto the top of the center pile, face up, or skips his or her turn, according to the rules of play below.

8.8.1 The Rules of Play

The player whose turn it is must follow suit, that is, play a card of the same suit as the top card on the center pile, if possible. If the player's hand contains no card of that suit, any card of the other color may be played. If neither type of card is in the player's hand, that player skips a turn. If at the beginning of the game the center pile is empty, the first player may play any card from his or her hand. Our goal is to write a mathematical expression whose value is true if a planned play is valid and false if not.

8.8.2 Translating the Rules of Play

To translate the English text above into the mathematical expression desired, one should begin by searching the text for noun phrases, distinguishing between content of a general, explanatory nature, and content essential for answering the question (whether the planned play is valid or not). The results of the search and selection of essential noun phrases should be carefully considered with regard to their relationships. Intermediate English formulations of the text, closer to the style of mathematics, is often helpful in this process.

8.8.3 Identifying the Noun Phrases in the English Text

The noun phrases in the English text above and initial comments on them are:

- player (We are concerned with only one player in the context of the English text for the rules of play.)
- turn (This is a general term in the context of a game with more than one player. Only one turn is the subject of the English text for the rules of play.)
- suit (This noun refers to the suit of a card.)
- card of the same suit [This noun phrase refers to the card to be played. The sentence in which this noun phrase appears indicates that the suits of the card to be played and the top card on the center pile (see below) must be equal.]
- top card on the center pile (This noun phrase refers to a specific card.)
- hand (This is a common term in card games and refers to the collection of cards that a player holds in his or her hand.)
- card of that suit [This noun phrase refers to a card in the player's hand (i.e., a potential card to be played) and the suit of other cards referred to specifically.]
- card of the other color (This noun phrase refers to the card to be played and the color of other cards.)
- color (This noun refers to the color of a card. See also "card of the other color" above.)
- neither type of card (The sentence in which this noun phrase appears refers to the condition that the player whose turn it is does not have a card in his or her hand that is a valid card to play.)
- beginning of the game (This noun phrase refers to one of the several states of play that may apply to the player whose turn it is.)
- center pile (This noun refers to a set of cards described in the rules of the game. See "top card on the center pile" above.)
- first player (This is the same player as referred to throughout the English text. The word "first" indicates that in this case, the player is the first to play, see "beginning of the game" above.)
- any card from his or her hand (See above.)

There are at least two major ambiguities in the English text above. (1) If, at the beginning of the game, the center pile is empty, the text above states that the player *may* play any card from his or her hand, but it does not state that the player *must* play a card. If that first player would choose not to play a card, the next player would not be the first player and the center pile would still be empty, a situation not covered at all by the rules of play. We will, therefore, assume that the first player *must* play a card if the center pile is empty. In this case, the player has a free choice of which card from his or her hand to play. (2) The rules of play do not indicate what happens if the player whose turn it is has no more cards left in his or her hand. We will assume that the game is then over, that is, that we are not to consider that situation. The "rules of play" paragraph above contains additional ambiguities, but they are resolved in the preceding general description of the game.

Examining the noun phrases above, we note that three types of cards are to be considered. These types of cards and the names we assign to the corresponding variables are as follows:

- The card to be played: CardToPlay
- A card in the player's hand: c (This variable will occur only in quantified expressions.)
- The top card on the center pile: TopCard

In the cases of CardToPlay and TopCard, it must be remembered that the corresponding set of values must include the situations in which no card is relevant, that is, (1) when the player cannot play any card from his or her hand and therefore must skip a turn and (2) when the center pile is empty. These situations will be represented by the value "none" for CardToPlay or for TopCard as appropriate.

Every reference above to a player is to the same player, the player whose turn it is. Because this player is always the same, we will not need to refer to the player by a variable. Similarly, the English text refers to one turn only, so we will not need to refer to it by a variable.

The only relevant aspect of the center pile is the top card showing (if any). Therefore, the noun phrase "center pile" does not need to be referred to separately in the mathematical model; references to TopCard will suffice.

The remaining nouns are hand, suit, and color. The player's hand will be represented by the variable named "hand," the value of that variable being a set of cards. We define "suit" and "color" to be functions of a card.

8.8.4 Developing the Mathematical Model

We translate the rules of play above into a mathematical expression sentence by sentence as follows.

The player whose turn it is must follow suit: that is, play a card of the same suit as the top card on the center pile, if possible. In other words, if the player's hand contains a card of the same suit as the top card of the center pile, a card of that suit must be played.

$$[\cup\ c : c{\in}\text{hand} \wedge \text{suit}(c){=}\text{suit}(\text{TopCard}) : c]{\neq}\emptyset \qquad \text{[card of same suit in hand]}$$

$$\Rightarrow\ \text{CardToPlay}{\in}[\cup\ c : c{\in}\text{hand} \wedge \text{suit}(c){=}\text{suit}(\text{TopCard}) : c] \qquad [8.8.4\text{-}1]$$

If the player's hand contains no card of that suit, any card of the other color may be played. That is, if the player's hand contains no card of that suit but does contain a card of the other color, a card of the other color must be played.

$$[\cup\ c : c{\in}\text{hand} \wedge \text{suit}(c){=}\text{suit}(\text{TopCard}) : c]{=}\emptyset \qquad \text{[card of same suit not in hand]}$$

$$\wedge\ [\cup\ c : c{\in}\text{hand} \wedge \text{color}(c){\neq}\text{color}(\text{TopCard}) : c]{\neq}\emptyset \qquad \text{[card of other color in hand]}$$

$$\Rightarrow\ \text{CardToPlay}{\in}[\cup\ c : c{\in}\text{hand} \wedge \text{color}(c){\neq}\text{color}(\text{TopCard}) : c] \qquad \text{[8.8.4-2]}$$

If neither type of card is in the player's hand, that player skips a turn.

$$[\cup\ c : c{\in}\text{hand} \wedge \text{suit}(c){=}\text{suit}(\text{TopCard}) : c]{=}\emptyset \qquad \text{[card of same suit not in hand]}$$

$$\wedge\ [\cup\ c : c{\in}\text{hand} \wedge \text{color}(c){\neq}\text{color}(\text{TopCard}) : c]{=}\emptyset \qquad \text{[card of other color not in hand]}$$

$$\Rightarrow\ \text{CardToPlay}{=}\text{none} \qquad \text{[skip a turn (i.e., play no card), 8.8.4-3]}$$

If at the beginning of the game the center pile is empty, the first player may play any card from his or her hand.

$$\text{TopCard}{=}\text{none} \qquad \text{[center pile empty]}$$

$$\Rightarrow\ \text{CardToPlay}{\in}\text{hand} \qquad \text{[play any card from hand, 8.8.4-4]}$$

At the latest, it becomes clear at this point that another condition is missing from the three expressions 8.8.4-1, 8.8.4-2, and 8.8.4-3, specifically that the center pile is not empty. This can be seen in two ways: (1) The terms suit(TopCard) are obviously inappropriate if there is no center pile and, therefore, no top card, and (2) without this condition, the antecedents in the four expressions would not be mutually exclusive.

The first sentence of the English text for the rules of play above is meaningful only if there is a top card on the center pile. In other words, that sentence implies that there is a nonempty center pile. This implicit information must be stated explicitly in the mathematical model.

Therefore, the additional condition TopCard≠none must be anded to the antecedents of the implications in expressions 8.8.4-1, 8.8.4-2, and 8.8.4-3. Including additional terms to specify the sets for the values of the variables and to define the functions "suit" and "color" leads to the following expression for the validity of the card to play, where Deck is the set of the 52 cards in the game:

$$\text{hand}{\subset}\text{Deck} \wedge \text{TopCard}{\in}(\text{Deck}{\cup}\{\text{none}\}) \wedge \text{CardToPlay}{\in}(\text{Deck}{\cup}\{\text{none}\}) \wedge$$

$$(\text{suit} : \text{Deck} \rightarrow \{\clubsuit, \diamondsuit, \heartsuit, \spadesuit\}) \wedge (\text{color} : \text{Deck} \rightarrow \{\text{red}, \text{black}\})$$

$\wedge$

(TopCard$\neq$none [center pile present]

$\wedge$ [$\cup$ c : c$\in$hand $\wedge$ suit(c)=suit(TopCard) : c]$\neq\emptyset$ [card of same suit in hand]

$\Rightarrow$ CardToPlay$\in$[$\cup$ c : c$\in$hand $\wedge$ suit(c)=suit(TopCard) : c])

$\wedge$

(TopCard$\neq$none [center pile present]

$\wedge$ [$\cup$ c : c$\in$hand $\wedge$ suit(c)=suit(TopCard) : c]=$\emptyset$
[card of same suit not in hand]

$\wedge$ [$\cup$ c : c$\in$hand $\wedge$ color(c)$\neq$color(TopCard) : c]$\neq\emptyset$
[card of other color in hand]

$\Rightarrow$ CardToPlay$\in$[$\cup$ c : c$\in$hand $\wedge$ color(c)$\neq$color(TopCard) : c])

$\wedge$

(TopCard$\neq$none [center pile present]

$\wedge$ [$\cup$ c : c$\in$hand $\wedge$ suit(c)=suit(TopCard) : c]=$\emptyset$
[card of same suit not in hand]

$\wedge$ [$\cup$ c : c$\in$hand $\wedge$ color(c)$\neq$color(TopCard) : c]=$\emptyset$
[card of other color not in hand]

$\Rightarrow$ CardToPlay=none) [skip a turn, i.e. play no card]

$\wedge$

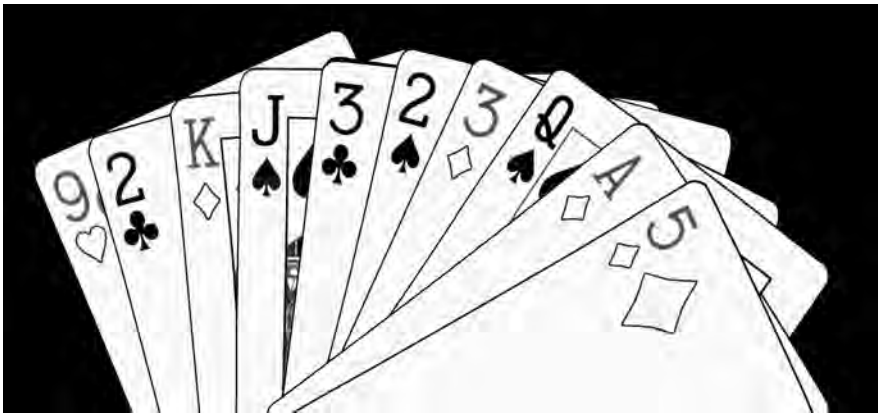
(TopCard=none [center pile empty]

$\Rightarrow$ CardToPlay$\in$hand) [play any card from hand]
[8.8.4-5]

Expression 8.8.4-5 is the complete mathematical model for the validity of the card to be played. For any value of CardToPlay, the value of the expression above is true if CardToPlay is a valid card to play, and false, otherwise.

Note that in the three cases above beginning with the condition TopCard$\neq$none, the references to suit(TopCard) and color(TopCard) will be undefined when the value of TopCard is actually none. In these cases, the values of the function references suit(TopCard) and color(TopCard) have no effect on the value of the entire expression 8.8.4-5. This type of situation is discussed in Section 3.5.3. Probably the simplest solution to this formal problem is to extend the definitions of the functions "suit" and "color" to include the value "none" in the domain and the range of each and to define these functions so that suit(none)=none and color(none)=none:

(suit : (Deck$\cup${none}) $\rightarrow$ {$\clubsuit$, $\diamondsuit$, $\heartsuit$, $\spadesuit$, none}) $\wedge$
(color : (Deck$\cup${none}) $\rightarrow$ {red, black, none}) [8.8.4-6]



hand = {(9,♡), (2,♣), (K, ♢), (J, ♠), (3, ♣), (2,♠), (3, ♢), (Q, ♠), (A, ♢), (5, ♢)}

As exercises for the reader, determine the values of the following quantified expressions, that is, determine which of the cards above are elements of the following sets:

$$[\cup\ c : c{\in}hand\ \wedge\ suit(c){=}suit((8, ♠)) : c] \quad [8.8.4\text{-}7]$$

$$[\cup\ c : c{\in}hand\ \wedge\ color(c){\neq}color((Q,♢)) : c] \quad [8.8.4\text{-}8]$$

$$[\cup\ c : c{\in}hand\ \wedge\ suit(c){\neq}suit((8,♠)) : c] \quad [8.8.4\text{-}9]$$

$$[\cup\ c : c{\in}hand\ \wedge\ color(c){=}color((Q,♢)) : c] \quad [8.8.4\text{-}10]$$

$$[\cup\ c : c{\in}hand\ \wedge\ suit(c){=}♡ : c] \quad [8.8.4\text{-}11]$$

$$[\cup\ c : c{\in}hand\ \wedge\ color(c){\neq}color((Q,♡)) : c] \quad [8.8.4\text{-}12]$$

$$[\cup\ c : c{\in}hand\ \wedge\ suit(c){\neq}♣ : c] \quad [8.8.4\text{-}13]$$

$$[\cup\ c : c{\in}hand\ \wedge\ color(c){=}color((Q,♣)) : c] \quad [8.8.4\text{-}14]$$

Are any of the sets above equal to one another? If so, which? If not, why not?

8.9 THE LOGICAL PARADOX OF THE BARBER OF SEVILLE

This example is one of several well known and essentially equivalent logical paradoxes. Perhaps the best known of the other formulations are the liar's paradox and Russell's paradox. The same structure appears in mathematical proofs that certain things cannot exist. One of these is a theorem which states that no algorithm can exist that can determine whether or not any given algorithm terminates. So the idea behind this and similar paradoxes is useful not only as an entertaining example or exercise; it has important implications in mathematics and its practical applications.

8.9.1 English Statement of the Paradox

Figaro, the Barber of Seville, shaves those men of Seville, and only those men, who do not shave themselves. Does Figaro shave himself?

To translate the sentence and the question above into the Language of Mathematics, we begin by identifying the key nouns and noun phrases. They are, clearly, "Figaro" and "men of Seville." The pronouns refer to these, so need not be considered separately. The noun phrase "Barber of Seville" is synonymous with Figaro, so also need not be considered separately.

Although the sentence above mentions "men" only in the plural, some of these references are, at least implicitly, to individuals. Actually, every reference to "men" can be interpreted as a reference to a single man. The sentence can be reworded in a form closer to a mathematical formulation as "Figaro shaves each man of Seville who does not shave himself, and no other" or even "Figaro shaves each and every man of Seville if and only if he does not shave himself."

But wait. Is this the only interpretation? If not, is it the best? If one man (not Figaro) of Seville shaves another man of Seville, does that exclude Figaro from shaving him? That is, does Figaro shave only those who no one else shaves? One possibility is that some subset of the men of Seville shave each other, forming a subset of the men of Sevilla who do "shave themselves," but each shaving a different member of that subset. Are we to interpret the original English sentence so that Figaro does or does not shave those men? Also, to whom does the pronoun "he" refer—the man of Seville in question or Figaro? Here, we note these ambiguities in the original English statement of the problem and proceed as if the statement is to be interpreted as stated at the end of the preceding paragraph: that is, that "Figaro shaves each and every man of Seville if and only if he (i.e., that man of Seville) does not shave himself."

All of the formulations above suggest a quantification over the men of Seville. The noun phrase "men of Seville" will correspond to a variable whose value is a set, the set of all men of Seville.

Other variables in our model will correspond to Figaro and to each man of Seville.

The verb "shave" or "shaves" is clearly an important term in the sentence above. At first, it might appear to be a verb of action, for which no directly corresponding variable or function can exist in a mathematical model. The meaning of the English sentence in the first paragraph above is not, however, that the shaving took place, is taking place, or will take place at any particular time in the future (i.e., the meaning is not related to any time). Instead, "shaves" refers to the relationship between the barber and a customer or a noncustomer. The original sentence could be reformulated to express the stative nature of this relationship more clearly: "Figaro is the shaver (barber) of any man of Seville if and only if that man is not his own shaver (barber)." The English style of this sentence is perhaps not as good as the original sentence, but it expresses more precisely and explicitly the implicit meaning. It is in a style and form closer to the ultimate mathematical expression and, hence, closer to a form that can be translated more directly into the Language of Mathematics.

In short, "shaves" is, in the original English statement of this paradox, a stative verb. It can, therefore, be represented by a Boolean variable, expression, or function. Because this relationship pertains to many different pairs of men, it will be represented by a Boolean function in our mathematical model.

8.9.2 Mathematical Model

Therefore, the interpretation of the elements of our mathematical model is:

- Figaro: a value representing Figaro, the Barber of Seville
- MenOfSeville: a variable, the value of which is the set of the men of Seville
- shaves(person1, person2): a Boolean function corresponding to the English clause "person1 shaves person2." The value of shaves(person1, person2) is true if person1 does, in fact, shave person 2, and false if person1 does not shave person2.

We begin by writing a mathematical expression corresponding to whether or not Figaro shaves a particular man of Seville. Below we write on the left parts of the reformulated statement of the paradox, and on the right, the corresponding parts of the mathematical expression

Figaro shaves each and every man m of Seville	shaves(Figaro, m)
if and only if	$=$
he (i.e., that man m of Seville) does not shave himself	$\neg$shaves(m, m)

The corresponding expression is, then,

$$\text{shaves(Figaro, m)} = \neg\text{shaves(m, m)} \qquad [8.9.2\text{-}1]$$

The expression above pertains to an individual man of Seville. To have it apply to all men of Seville, it must be embedded in an appropriate quantified expression:

$$[\wedge\ \text{m} : \text{m}\in\text{MenOfSeville} : \text{shaves(Figaro, m)} = \neg\text{shaves(m, m)}] \qquad [8.9.2\text{-}2]$$

This mathematical expression is a translation of the original English sentence "Figaro, the Barber of Seville, shaves those men of Seville, and only those men, who do not shave themselves" as interpreted and understood in Section 8.9.1.

Now to the question "Does Figaro shave himself?" This is equivalent to asking the question "Is the value of shaves(Figaro, Figaro) true?" If, as the statement of the paradox seems to imply, Figaro is a man of Seville (he is clearly the Barber of Seville), then he is one of the elements of the set MenOfSeville, to which expression 8.9.2-1 applies. Then

$$\text{shaves(Figaro, Figaro)} = \neg\text{shaves(Figaro, Figaro)} \qquad [8.9.2\text{-}3]$$

The question then becomes "which value of shaves(Figaro, Figaro) is consistent with expression 8.9.2-3," that is, makes expression 8.9.2-3 true? However, expression 8.9.2-3 is always false, never true. The assumption that shaves(Figaro, Figaro) is true contradicts expression 8.9.2-3. The assumption that shaves(Figaro, Figaro) is false also contradicts expression 8.9.2-3. The original English statement of the problem is self-contradictory with regard to the question posed and hence is inadequate as a basis for answering that question—at least if Figaro is an element of the set MenOfSeville. If Figaro is not to be viewed as a man of Seville, the contradiction disappears, but so does the term m=Figaro in the quantified expression and with it, any possible information on the value of the function shaves(Figaro, Figaro). In the latter case, both statements “Figaro shaves himself” (shaves(Figaro, Figaro)=true) and “Figaro does not shave himself” (shaves(Figaro, Figaro)=false) are possible solutions.

In short, the English formulation of the problem is inadequate as a basis for answering the question. Depending on how we interpret it precisely, it contains either self-contradictory or no information on whether Figaro shaves himself.

Viewed purely mathematically, the problem (question) as stated and interpreted has no solution. This situation is by no means rare in mathematics; many Boolean expressions exist which have no solution (i.e., are never true). They are false for all values of the variables appearing in them. Section 5.3 contains other examples of Boolean expressions having no solution.

Few people will recognize at first reading the inadequacy of the English statement in Section 8.9.1 for answering the question until they translate the English statement into the Language of Mathematics. Also, few people will recognize the ambiguities in the original English statement; only when one attempts to translate the sentence into the Language of Mathematics do the detailed questions of interpretation arise so clearly.

8.10 CONTROLLING THE WATER LEVEL IN A RESERVOIR: SIMPLE ON/OFF CONTROL

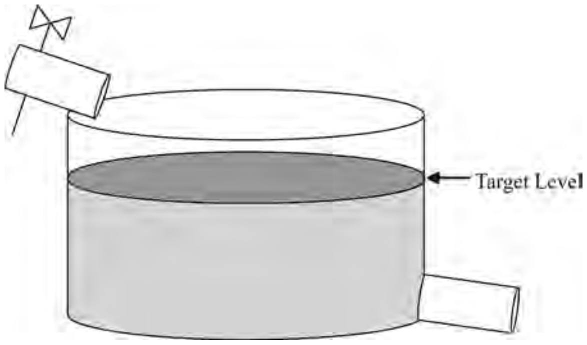
Several different ways of controlling the water level in a reservoir can be specified and implemented. Each has certain advantages and disadvantages. Below, a simple on/off mechanism is presented. In Section 8.11, a more advanced control system with improved operational characteristics is presented.

Notice that these examples deal only with the control of the flow into the reservoir. The English texts say nothing about either the flow out of the reservoir, the rate of flow into the reservoir when the flow is on, or how these determine the actual water level, so these aspects of the reservoir will not be included in the mathematical models in these examples.

These examples illustrate that a mathematical model can be restricted to those aspects of direct concern in a system; it is not necessary that it include all characteristics and properties of the system and its environment. Expressed differently, a mathematical model can, and usually does, abstract the essentials of the entire environment of the problem being investigated. In these examples, only the on/off control of flow into the reservoir needed to prevent overflowing the reservoir is the issue. The factors determining the flow out and the question of whether or not the flow rate in is sufficient to satisfy the demand for flow out of the reservoir are not of concern in these examples. They could be dealt with by other mathemtical models if desired.

8.10.1 English Statement of the Requirements

When the water level in the reservoir is below a certain level (the target level), the incoming water supply should be turned on. When the water level in the reservoir is above this level, the incoming water supply should be turned off.



8.10.2 The Mathematical Variables and Their Interpretation

The key noun phrases appearing in the English text are associated with mathematical variables as given below.

w:	Water level in the reservoir
s:	Incoming water supply (on or off)
t:	The target level of water in the reservoir, measured in the same units as w

8.10.3 The Mathematical Model

The English language specification does not state what should be done if the water level is at exactly the target level. Presumably, such precision is not of practical importance, but an appropriate assumption must be built into the mathematical model for completeness.

The mathematical model can then be written either in tabular form:

Condition	Value of s
$w \le t$	on
$w > t$	off

or as an expression:

$$\begin{aligned}&w \in \mathbb{R} \wedge t \in \mathbb{R} \wedge s \in \{\text{off}, \text{on}\} \wedge \\ &(w \le t \wedge s = \text{on} \vee w > t \wedge s = \text{off})\end{aligned} \qquad [8.10.3\text{-}1]$$

In expression 8.10.3-1 the first line defines the range of values for every variable. The second line expresses the relationships between the values of the variables as stated in the table. The expression can be written in several equivalent forms. The reader should identify some of them. Mathematically, this model defines a function s of arguments w and t: nothing more, nothing less.

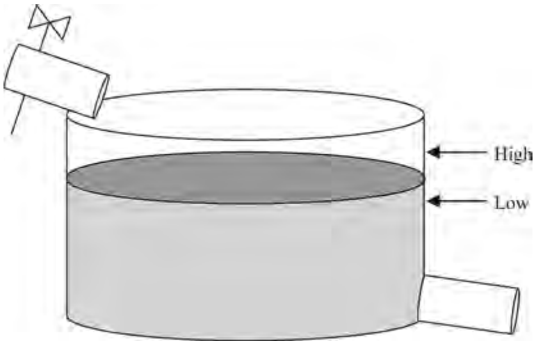
8.10.4 Shortcomings of the Simple On/Off Control

In practice, the accuracy to which the water level w can be measured is limited. Measurements are not exactly repeatable; that is, successive measurements can differ slightly. In addition, waves on the surface of the water can cause successive measurements to differ. When the water level is close to the target level, these factors can cause the valve controlling the incoming water supply to chatter, that is, to be turned on and off in rapid succession. This, in turn, can cause unnecessary and excessive wear and tear on the mechanical components. To avoid this problem, a suitable time delay (hysteresis) must be introduced into the control loop turning the water supply on and off. One of several ways of introducing such a delay is illustrated in the control mechanism described below.

8.11 CONTROLLING THE WATER LEVEL IN A RESERVOIR: TWO-LEVEL ON/OFF CONTROL

8.11.1 English Statement of the Requirements

When the water level in the reservoir is below a certain level, the incoming water supply should be turned on. When the water level in the reservoir is above a certain higher level, the incoming water supply should be turned off. When the water level is between these two levels, the incoming water supply should be left unchanged (i.e., off if it was off, on if it was on).



8.11.2 Interpretation

The several noun phrases appearing in the English text are associated with mathematical variables as given below. The passage of time is modeled with a nonnegative integer, called k, the idea being that at each time step k the water level is measured and the incoming water flow set on or off accordingly. The value of k increases with time. The English text suggests by default that the values of the lower and upper threshold levels do not vary with time, so they do not need to be subscripted by k.

w(k):	Water level in reservoir at time point k
s(k):	Incoming water supply (on or off) at time point k
Low:	Lowest desired water level in reservoir, measured in the same units as w(k)
High:	Highest desired water level in reservoir, measured in the same units as w(k)

8.11.3 The Mathematical Model

The English language specification does not state what should be done if the water level is at exactly the lower or the higher threshold level. Presumably, such precision is not of practical importance, but an appropriate assumption must be built into the mathematical model for completeness. Furthermore, the specification also fails to state whether the water supply is initially on or off. We assume that it is initially off.

The length of time between measurements of the water level is not mentioned in the English text; presumably, it is short enough so that the reservoir cannot overflow. It is omitted from this mathematical model. The mathematical model can then be written either in tabular form:

	s(k)=	
Condition	k=0	k>0
$w(k) \leq \text{Low}$	off	on
$\text{Low} < w(k) < \text{High}$	off	s(k–1)
$\text{High} \leq w(k)$	off	off

or as an expression:

$$\begin{aligned}
&\text{High}\in\mathbb{R} \wedge \text{Low}\in\mathbb{R} \wedge \text{Low}<\text{High} \wedge \\
&[\wedge\ k : k\in\mathbb{Z} \wedge k\geqslant 0 : s(k)\in\{\text{off}, \text{on}\} \wedge w(k)\in\mathbb{R}] \wedge \\
&\qquad (k=0 \wedge s(k)=\text{off} \vee \\
&\qquad k>0 \wedge w(k) \leq \text{Low} \wedge s(k)=\text{on} \vee \\
&\qquad k>0 \wedge \text{Low} < w(k) < \text{High} \wedge s(k)=s(k-1) \vee \\
&\qquad k>0 \wedge \text{High} \leq w(k) \wedge s(k)=\text{off}) \qquad [8.11.3\text{-}1]
\end{aligned}$$

In expression 8.11.3-1 the first two lines define the range of values for every variable. The third through last lines express the relationships between the values of the various variables as stated in the table. The expression can be written in several other equivalent forms. The reader should identify some of them.

Mathematically, expression 8.11.3-1 defines the sequence

$$[s(0), s(1), s(2), s(3), \ \ldots] \qquad [8.11.3\text{-}2]$$

as a function of the input sequence

$$[w(1), w(2), w(3), \ \ldots] \qquad [8.11.3\text{-}3]$$

8.12 RELIABLE COMBINATIONS OF LESS RELIABLE COMPONENTS

Large systems consisting of many components are common in today’s world. Furthermore, the number of components in such systems is increasing. With increasing numbers of components of constant reliability, the overall reliability of our systems would decrease unless design techniques were employed in order to construct more reliable systems of less reliable components. Fortunately, such design techniques exist and have been well known and used by design engineers for a long time. In this section, simple designs employing redundancy to increase the overall reliability are presented and quantitatively analyzed. Probability theory (see Section 4.6) provides the mathematical basis for such an analysis.

An especially simple example of redundancy to increase reliability is the overhead projector used for many types of lectures. Of the many components in such a projector, the lamp has the shortest lifetime. Therefore, it limits the overall reliability of the projector. Most such projectors now contain two lamps, so that when one fails, the other can be switched on, bridging the gap in time until the other lamp is replaced.

8.12.1 A Door Closure Sensor

More illustrative of engineering designs using redundancy to improve reliability is a subsystem for sensing whether a semiautomatic door to a building or on a train is open or closed. A single sensor can consist of mechanical, electrical, electronic, and/or optical parts, any or all of which can fail. Normally, the sensor will correctly sense the door to be open when it is open and closed when it is closed. Although unlikely, it is also possible that the door is

- sensed to be open despite the fact that it is actually closed, or
- sensed to be closed despite the fact that it is actually open.

The latter fault can represent a particularly unsafe situation. For example, a train’s control system will typically include an interlock to prevent the train from being started if any door is open. If the sensor on a door indicates that the door is closed when it is actually open, the train can be started into motion with a door open, an obviously unsafe mode of operation. Although such a fault can never be excluded with absolute certainty, very low failure rates can be specified for such events. Standard engineering practice typically requires very low failure probabilities for safety critical functions—failure probabilities that are often lower than any single component can achieve.

The first fault above (a closed door being sensed as open) will prevent the train from proceeding and hence will hinder operation of the train and cause unnecessary delays, but it will not lead to unsafe operation. It represents a “fail safe” mode.

Consider a single door sensor whose probabilities of correct and faulty operation are given in Table 8.12.1-1. Notice that the probabilities of correct and faulty operation are different when the door is open and when it is closed. The nondeterministic behavior of the sensor is modeled here by two different probability functions. One is applicable when the door is actually closed, and the other, when the door is actually open. The actual position of the door is not considered to be a random variable and, therefore, is not modeled probabilistically.

TABLE 8.12.1-1 Probabilities of Failure for a Single Door Sensor

Actual Position of Door	Door Sensed as	Probability	Sensor Function
closed	closed	0.999	correct
closed	open	0.001	safe failure, hinders operation
open	closed	0.0005	unsafe failure
open	open	0.9995	correct

These probabilities of faulty operation are atypically high for the purposes of this numerical example. In an actual safety critical application, more reliable components would be used. However, as will be seen below, even with such poor sensors one can achieve much higher system reliability with appropriate redundancy.

Notice that Table 8.12.1-1 in effect includes both of the following:

- The interpretation of the (unnamed) variables in the mathematical model
- The mathematical model, including the values of the unnamed variables

Note also that use of the word “probability” implicitly includes by reference the relevant parts of the mathematical model of a probabilistic process as defined in Section 4.6.1. Thus, that model is part of the environment of the mathematical expressions represented by Table 8.12.1-1. The general mathematical model of a probabilistic process is often left implicit in documentation of the analysis of an application, although at least the sample space and the events (the subsets of the sample space) of interest should be clearly identified. For completeness, Section 8.12.3 contains an explicit formulation of the entire mathematical model for the door sensors considered in this example, including the case of the single sensor represented in Table 8.12.1-1 and the cases of the several redundant door sensor systems presented in Section 8.12.2.

8.12.2 Increased Reliability with Additional Redundant Door Sensors

One way of increasing the reliability of sensing the position of the door is to use two sensors operating statistically independently of each other. That is, they would interact physically with the door independently of each other, so that the probability that one sensor fails is the same regardless of whether or not the other sensor fails. Then the probability of a joint event is the product of the two individual events (see Section 4.6.3).

The output signals (open or closed) of the two sensors are combined into a single signal (open or closed) representing the presumed or inferred position of the door. The simplest way to minimize the probability that the door is considered to be closed when it is actually open would be to generate a combined signal that always indicates open, but that would result in an inoperative system. The next best way to minimize the probability of the unsafe failure is to combine the two signals from the door sensors into a composite signal, implying that the door is closed only when both individual sensors sense that the door is closed. If either sensor senses that the door is open, the combined signal indicates that the door is presumed to be open. This composition of the two door sensors' signals is illustrated in Table 8.12.2-1 which shows all possible combinations of door position and sensor outputs.

TABLE 8.12.2-1 Probabilities of Failure for a Dual Door Sensor System

Actual Position of Door	Output of Sensor 1	Output of Sensor 2	Probability	Inferred Position of Door	Probability	Sensor Function
closed	closed	closed	0.998001	closed	0.998001	correct
closed	closed	open	0.000999	open	0.001999	safe failure, hinders operation
closed	open	closed	0.000999	open		
closed	open	open	0.000001	open		
open	closed	closed	0.00000025	closed	0.00000025	unsafe failure
open	closed	open	0.00049975	open	0.99999975	correct
open	open	closed	0.00049975	open		
open	open	open	0.99900025	open		

Notice that the probability of an unsafe failure has been reduced from 0.0005 to 0.00000025, a reduction by a factor of 2000. The cost of this improvement is a second door sensor and a component to combine the signals from the two sensors. In addition, the probability of correct operation when the door is closed is reduced from 0.999 to 0.998001. Although this reduction is small, this low a probability of correctly sensing a closed door would presumably hinder the operation of the train system unacceptably. With additional redundancy, this probability can be increased.

One way of adding more redundancy is to use three sensors whose signals are combined by a majority vote. That is, the door is considered to be:

- Closed if at least two sensors indicate a closed door
- Open if at least two sensors indicate an open door.

Table 8.12.2-2 shows all possible combinations of door position and sensor outputs as well as the corresponding probabilities for this combination of sensors.

TABLE 8.12.2-2 Probabilities of Failure for a Triple Door Sensor System

Actual Position of Door	Output of Sensor 1	2	3	Probability	Inferred Position of Door	Probability	Sensor Function
closed	closed	closed	closed	0.997002999	closed	0.999997002	correct
closed	closed	closed	open	0.000998001	closed		
closed	closed	open	closed	0.000998001	closed		
closed	open	closed	closed	0.000998001	closed		
closed	closed	open	open	0.000000999	open	0.000002998	safe failure, hinders operation
closed	open	closed	open	0.000000999	open		
closed	open	open	closed	0.000000999	open		
closed	open	open	open	0.000000001	open		
open	closed	closed	closed	0.000000000125	closed	0.000000749750	unsafe failure
open	closed	closed	open	0.000000249875	closed		
open	closed	open	closed	0.000000249875	closed		
open	open	closed	closed	0.000000249875	closed		
open	closed	open	open	0.000499500125	open	0.999999250250	correct
open	open	closed	open	0.000499500125	open		
open	open	open	closed	0.000499500125	open		
open	open	open	open	0.998500749875	open		

Table 8.12.2-3 compares the three sensor structures described above. Notice in particular the probabilities of correct operation, safe failure, and unsafe failure.

TABLE 8.12.2-3 Comparison of the Probabilities of Failure for the Three Systems

Actual Position of Door	Inferred Position of Door	Probability with … Sensors 1	2	3	Sensor Function
closed	closed	0.999	0.998001	0.999997002	correct
closed	open	0.001	0.001999	0.000002998	safe failure, hinders operation
open	closed	0.0005	0.00000025	0.000000749750	unsafe failure
open	open	0.9995	0.99999975	0.999999250250	correct

With two sensors the probability of sensing the door as closed when it is really open is reduced greatly compared to a single sensor, but the probability of failure is still too high for a safety critical commercial or public application. Furthermore, this improvement is achieved at the expense of a reduction of operational reliability. The dual sensor system is still not satisfactory. With three sensors the operational reliability is increased significantly, but this improvement is accompanied by an increase in the probability of an unsafe failure.

The probability of an unsafe failure must be reduced considerably, and an additional decrease in the probability of a safe failure (an increase in the probability of correct operation) is desirable. One way of achieving such an improvement is to incorporate four sensors in the door-sensing system. To reduce the probability of an unsafe failure, that is, the probability of inferring an open door to be closed, the door will be considered open if two or more sensors sense the door as being open (i.e., also in an equal vote of 2 to 2). This system is represented in Table 8.12.2-4.

TABLE 8.12.2-4 Probabilities of Failure for a Quadruple Door Sensor System

Actual Position of Door	Outputs of the 4 Sensors	Probability	Inferred Position of Door	Probability	Sensor Function
closed	4 closed	0.996005996001	closed		
closed	3 closed, 1 open	0.003988011996	closed	0.999994007997	correct
closed	2 closed, 2 open	0.000005988006	open		safe failure, hinders operation
closed	1 closed, 3 open	0.000000003996	open	0.000005992003	
closed	4 open	0.000000000001	open		
open	4 closed	0.0000000000000625	closed	0.0000000004998125	unsafe failure
open	3 closed, 1 open	0.0000000004997500	closed		
open	2 closed, 2 open	0.0000014985003750	open		
open	1 closed, 3 open	0.0019970014997500	open	0.9999999995001875	correct
open	4 open	0.9980014995000625	open		

The probabilities in Table 8.12.2-4 were calculated in the same way as in the previous tables. However, they are presented in a more concise format in Table 8.12.2-4. In particular, the several lines with the same number of closed and open sensor outputs have been combined into only one line.

The probability of an unsafe failure in the four-sensor system represented by Table 8.12.2-4 is less than 10^{-9}, a figure often mentioned in connection with requirements for safety-critical systems. Thus, this configuration of four sensors, each of relatively low reliability, can satisfy stringent reliability requirements. This example shows one way of achieving high reliability with components of low reliability and how to analyze such redundant system designs.

The probability of a safe failure, although perhaps acceptable, is higher than might be desired. In an actual situation, the operational consequences of this failure rate would be analyzed and compared with the costs of achieving a lower probability of safe failure. One of the several alternatives to consider would be a more reliable individual sensor.

In the beginning of this section it was pointed out that the individual sensors are assumed to fail independently of one another. In any system of this nature there are, however, common mode failures (e.g., the failure of a common power supply, large-scale physical damage affecting more than one sensor at one time). Also, a possible failure of the component combining the outputs of the several individual sensors has been neglected in this example. In the real engineering world, these and many other factors affecting the reliability of components in a system and of the system as a whole are included in failure analyses.

In engineering practice, one often distinguishes between failure, fault, and malfunction, and between permanent and intermittent failures, faults, and malfunctions. For reasons of simplicity, no such distinction has been made in this example.

8.12.3 The Complete Mathematical Model for the Redundant Door Sensing Systems

In Sections 8.12.1 and 8.12.2 the mathematical models were expressed in tabular notation, together with much of the interpretation of the (unnamed) variables, values, and functions in the model. In this section, the complete mathematical model together with the interpretation of all of its parts is stated completely and explicitly. The mathematical model is presented in the form of an infix expression.

The expressions involving probabilities in the mathematical model below follow from definitions in Section 4.6 on probability theory (see especially Sections 4.6.1 and 4.6.3).

Interpretation The several values, variables, and functions in the mathematical model below have the following meanings in the application domain.

- **N:** the number of individual sensors used in the door-sensing system
- **SensedOpen, SensedClosed:** the possible values output from each sensor. Each sensor outputs the value SensedOpen if it senses the door as being open, and SensedClosed if it senses the door as being closed.
- **S:** the set {SensedOpen, SensedClosed}. S^N is the set of all sequences of N elements of the set S (i.e., S^N is the set of all possible sequences of the outputs of the N sensors). S^N is the sample space of the probability space being modeled (see Section 4.6.1).
- **SeqIntOpen:** the set of those sequences of sensor outputs that are interpreted as indicating that the door is open. SeqIntOpen is a subset of S^N.
- **SeqIntClosed:** the set of those sequences of sensor outputs that are interpreted as indicating that the door is closed. SeqIntClosed is a subset of S^N. The sets SeqIntOpen and SeqIntClosed are mutually exclusive; that is, they have no element in common. Together, they constitute the entire sample space S^N.

- **Events:** the subsets of S^N corresponding to random events that are of interest in this application and for which probabilities are defined. These events are the elements (as singleton sets) of S, the elements (as singleton sets) of S^N, the subsets SeqIntOpen and SeqIntClosed of S^N, the empty set Ø, and the entire set S^N.
- **PrDO:** the probability function of an event when the door is actually open. The outputs from the N sensors are statistically independent.
- **PrDC:** the probability function of an event when the door is actually closed. The outputs from the N sensors are statistically independent.

Mathematical Model The complete mathematical model consists of the expression 8.12.3-1. In it, the value of the function reference termseq(i, seq) is defined to be the ith term of the sequence seq.

$$\begin{aligned}
&\mathrm{N}\in\mathbb{Z} \wedge 1\le \mathrm{N} \wedge \mathbb{P} = [\cup\, \mathrm{p} : \mathrm{p}\in\mathbb{R} \wedge 0\le \mathrm{p}\le 1 : \{\mathrm{p}\}] \wedge \qquad \text{[header, 8.12.3-1]}\\
&(\mathrm{termseq} : \mathbb{Z}\times \mathrm{S}^{\mathrm{N}} \to \mathrm{S}) \wedge\\
&[\wedge\, \mathrm{seq} : \mathrm{seq}\in \mathrm{S}^{\mathrm{N}} : \mathrm{seq} = [\&\, \mathrm{i} : \mathrm{i}\in\mathbb{Z} \wedge 1\le \mathrm{i}\le \mathrm{N} : \mathrm{termseq(i, seq)}]] \wedge\\
&\mathrm{S} = \{\mathrm{SensedOpen, SensedClosed}\} \wedge\\
&\mathrm{Events} = \{\{[\mathrm{SensedOpen}]\},\ \{[\mathrm{SensedClosed}]\},\\
&\qquad\qquad \mathrm{SeqIntOpen, SeqIntClosed}, \emptyset, \mathrm{S}^{\mathrm{N}}\}\\
&\qquad \cup [\cup\, \mathrm{seq} : \mathrm{seq}\in \mathrm{S}^{\mathrm{N}} : \{\{\mathrm{seq}\}\}] \wedge \qquad \text{[see comments below]}\\
&(\mathrm{PrDO} : \mathrm{Events} \to \mathbb{P}) \wedge (\mathrm{PrDC} : \mathrm{Events} \to \mathbb{P}) \wedge\\
&(\mathrm{N}=1 \Rightarrow \mathrm{SeqIntClosed} = \{[\mathrm{SensedClosed}]\}) \wedge\\
&\qquad\qquad\qquad \text{[definition of SeqIntClosed]}\\
&(\mathrm{N}=2 \Rightarrow \mathrm{SeqIntClosed} = \{[\mathrm{SensedClosed, SensedClosed}]\}) \wedge\\
&(\mathrm{N}=3 \Rightarrow \mathrm{SeqIntClosed}\\
&\qquad = \{[\mathrm{SensedClosed, SensedClosed, SensedClosed}],\\
&\qquad\quad [\mathrm{SensedOpen, SensedClosed, SensedClosed}],\\
&\qquad\quad [\mathrm{SensedClosed, SensedOpen, SensedClosed}],\\
&\qquad\quad [\mathrm{SensedClosed, SensedClosed, SensedOpen}]\}) \wedge\\
&(\mathrm{N}=4 \Rightarrow \mathrm{SeqIntClosed}\\
&\qquad = \{[\mathrm{SensedClosed, SensedClosed, SensedClosed, SensedClosed}],\\
&\qquad\quad [\mathrm{SensedOpen, SensedClosed, SensedClosed, SensedClosed}],\\
&\qquad\quad [\mathrm{SensedClosed, SensedOpen, SensedClosed, SensedClosed}],\\
&\qquad\quad [\mathrm{SensedClosed, SensedClosed, SensedOpen, SensedClosed}],\\
&\qquad\quad [\mathrm{SensedClosed, SensedClosed, SensedClosed, SensedOpen}]\}) \wedge
\end{aligned}$$

$$
\begin{aligned}
&\text{SeqIntOpen} = \text{S}^{\text{N}} \setminus \text{SeqIntClosed} \wedge && \text{[definition of SeqIntOpen]}\\
&\text{PrDO}(\emptyset){=}0 \wedge \text{PrDO}(\text{S}^{\text{N}}){=}1 \wedge && \text{[definition of PrDO]}\\
&\text{PrDO}([\text{SensedOpen}]) = 0.9995 \wedge \text{PrDO}([\text{SensedClosed}]) = 0.0005 \wedge &&\\
&[\wedge\ \text{seq} : \text{seq}{\in}\text{S}^{\text{N}} &&\\
&\qquad : \text{PrDO}(\text{seq}) = [*\ \text{i} : \text{i}{\in}\mathbb{Z} \wedge 1{\le}\text{i}{\le}\text{N} : \text{PrDO}(\text{termseq}(\text{i}, \text{seq}))]] \wedge &&\\
&\text{PrDO}(\text{SeqIntClosed}) = [+\ \text{seq} : \text{seq}{\in}\text{SeqIntClosed} : \text{PrDO}(\text{seq})] \wedge &&\\
&\text{PrDO}(\text{SeqIntOpen}) = 1 - \text{PrDO}(\text{SeqIntClosed}) \wedge &&\\
&\text{PrDC}(\emptyset){=}0 \wedge \text{PrDC}(\text{S}^{\text{N}}){=}1 \wedge && \text{[definition of PrDC]}\\
&\text{PrDC}([\text{SensedOpen}]) = 0.001 \wedge \text{PrDC}([\text{SensedClosed}]) = 0.999 \wedge &&\\
&[\wedge\ \text{seq} : \text{seq}{\in}\text{S}^{\text{N}} &&\\
&\qquad : \text{PrDC}(\text{seq}) = [*\ \text{i} : \text{i}{\in}\mathbb{Z} \wedge 1{\le}\text{i}{\le}\text{N} : \text{PrDC}(\text{termseq}(\text{i}, \text{seq}))]] \wedge &&\\
&\text{PrDC}(\text{SeqIntClosed}) = [+\ \text{seq} : \text{seq}{\in}\text{SeqIntClosed} : \text{PrDC}(\text{seq})] \wedge &&\\
&\text{PrDC}(\text{SeqIntOpen}) = 1 - \text{PrDC}(\text{SeqIntClosed}) &&
\end{aligned}
$$

In the definitions of the functions PrDO and PrDC, the probability of each sequence of outputs from the N sensors is the product of the probabilities of the individual output values of the sensors, because the outputs of the door sensors are statistically independent. The probability of the event SeqIntClosed is the sum of the probabilities of the sequences of the sensor outputs in the set SeqIntClosed, because the sequences are all different from one another (mutually exclusive). It follows from the mathematical model in expression 8.12.3-1 that the same is true for SeqIntOpen.

In the definition of the set variable Events in the mathematical model in expression 8.12.3-1, each element of the set Events is, itself, a set. Singleton sets are also explicitly expressed as sets. If one would follow the convention of not distinguishing between a value, a singleton set of that value, and a singleton set of a singleton sequence of that value, the expression defining the set variable Events could be simplified to

$$\text{Events} = \{\text{SensedOpen}, \text{SensedClosed}, \text{SeqIntOpen}, \text{SeqIntClosed}, \emptyset, \text{S}^{\text{N}}\} \cup \text{S}^{\text{N}} \qquad [8.12.3\text{-}2]$$

The last term, $\cup\ \text{S}^{\text{N}}$, would still be necessary because it (and only it) brings the individual elements of S^{N} into the set Events. Without that last term, the *set* S^{N} would be an element of the set Events, but the *elements* of the set S^{N}—the individual sequences of sensor outputs—would not be elements of the set Events. Those individual sequences must be elements of the set Events, because we need to be able to refer to the probability of each such sequence alone, and the probability functions PrDO and PrDC can be applied only to an argument that is an element of the set Events.

8.13 SHOPPING MALL DOOR CONTROLLER

In this example, the requirements for the control mechanism for an automatic door to a shopping mall is first stated in English. This description is then transformed as described in the beginning of Chapter 8 into a mathematical model written in the Language of Mathematics. This mathematical model will constitute an engineering specification of the logic in the door controller.

The English description of the door controller will be prefaced here by a description of:

- The door and its operation as seen from the persons' standpoint
- The main physical devices associated with the door

The persons' view of the door will provide the background for a description of the requirements of the door controller, but itself will contribute only indirectly to the mathematical model of the door controller. The description of the physical devices associated with the door and especially the description of their relationships with the door constitute the specification of the controller's immediate environment (i.e., the specification of the controller's interface with the door and the physical devices associated with the door). Both categories of time are dealt with in this example: discrete time steps and continuous time.

8.13.1 Persons' View of the Door

On and near the door are motion and proximity detectors which detect people and objects in the door area. When someone approaches from either direction, the door opens and the person walks through. After a short delay, the door closes if no one is in the vicinity of the door. If, while the door is closing, someone approaches the door, it opens again.

8.13.2 Physical Devices Associated with the Door

The following devices are associated with the door:

- Motion and proximity detectors detect people and objects near the door, both inside and outside the door, and passing through the door. The several detectors operate as a single subsystem to indicate whether or not a person, animal, or object is in the vicinity of the door or is blocking the door.
- The door is equipped with a sensor to indicate whether or not the door is physically fully open.
- The door is equipped with another, independent sensor, to indicate whether or not the door is physically fully closed.

- The door has a motor to open or to close the door. The signal from the door controller to the motor controls power to the motor: power on to open the door, power on to close the door, or power off to hold the door still.

8.13.3 The Door Controller's Inputs and Outputs

The descriptions above imply that the door controller will have the following inputs and outputs:

- Motion and proximity detection (input from detectors on and near the door)
- Door open sensor (input from sensor on door)
- Door closed sensor (input from sensor on door)
- Motor power (output to motor on door)

8.13.4 Required Responses of the Door Controller

Specific responses to the various input signals and states of the door system depend on various conditions as follows. These requirements give the rules for calculating the output and state variables for one point in time based on the values of the variables at the preceding point in time, that is, for stepping the values from one point in time to the next in succession.

- If the motion and proximity detectors detect someone in the vicinity of the door, the door controller should open the door, unless it is already open.
- If the motor is on (under power), its direction should not be reversed immediately. Motor power should first be turned off, then turned on in the reverse direction. Any brief off period is sufficient.
- If the door is opening and the door sensors indicate that it has reached its open position, power to the motor should be turned off.
- If the door is closing and the door sensors indicate that it has reached its closed position, power to the motor should be turned off.
- If the door has been open and no person has been detected in the vicinity of the door for a given period of time (called the closure delay time), the door controller should close the door (see Section 8.13.1).
- If the door is closing and the motion and proximity detectors detect someone in the vicinity of the door, the controller should open the door (see Section 8.13.1).

8.13.5 Method of Operation of the Controller

From the descriptions above it is clear that the door system and the controller are dynamic in nature; that is, various aspects of them change with time. Characteristic of any dynamic system is that at any point in time the system is in a particular state (condition, situation) and that the state changes with time. Similarly, the values of input and output variables will also typically change with time. Variables whose values represent the state, inputs, and outputs are, therefore, typically represented as subscripted (indexed or array) variables, with the subscript representing specific points in time. Alternatively, these variables can be viewed as functions of time, but these two views are semantically equivalent (see Sections 4.1.2 and 4.1.4).

The state of the controller should be recalculated (updated) frequently in relation to the time frame of the operation of the door: that is, in relation to the time it takes people to approach and leave the door area, the time it takes the door's motor to start and stop, and the time it takes the door to open and close. These times are typically on the order of a second or longer. The controller should, therefore, recalculate the state much more frequently (i.e., in a small fraction of a second). Such a mode of operation would approximate sufficiently well a controller mechanism continuously updating the state (e.g., one implemented with analog electronic circuitry). One feasible implementation of the controller would use a small digital computer, which would be able to calculate each new state in milliseconds or even microseconds.

If the controller is implemented with a digital computer, a convenient way to implement the logic for closing the door is to maintain, as part of the state of the controller, the time at which the door was most recently not in a "closable state." By "closable" state we mean that the door is open and no person is in the vicinity of the door. The length of time the door has been continuously in a closable state is, then, the current time minus the time at which the door was most recently not in a closable state. When this difference in time is greater than the closure delay time, the condition for closing the door is met. The design of a controller implemented with analog electronic circuitry can also be based on this logical consideration.

In this example, we assume that the controller will be implemented with a small digital computer.

8.13.6 The Variables

We begin identifying and defining the mathematical variables by considering the inputs and outputs listed above and the types of values they assume.

The motion and proximity detectors together deliver a single signal indicating whether or not someone or something is in the vicinity of the door (i.e., this input has one of two values, so can be viewed as a Boolean variable). The information this variable delivers is best described by a clause with a stative verb (e.g., "A person is near the door)", with the value being either true or false. We abbreviate this clause to form the variable's name "PersonIsNear."

The next input listed above is provided by the door-open sensor. This variable also takes on one of two values, indicating that the door is either open or is not open. The stative clause "the door is open" can be abbreviated to the variable's name "DoorIsOpen," with the value being either true or false.

Similarly, the signal provided by the door-closed sensor can be modeled with a variable named "DoorIsClosed," with its value being either true or false.

The output of the controller sets the power to the motor to one of three values: power on to open the door, power on to close the door, or power off to hold the door still. This can be described by the noun phrase "motor power" and abbreviated to the variable name "MotorPower," with the value being either "on to close," "on to open," or "off."

Implementing the requirement to close the door after a certain time delay requires two variables and a constant parameter. The two variables are the current time and the time at which the door was most recently not in a closable state (see the discussion above on closing the door). The parameter is the closure delay time. Such parameters are normally modeled with a variable whose value is fixed and does not change during normal operation. Thus, we need three variables whose names abbreviate the noun phrases "current time," "time at which the door was most recently not in a closable state," and "closure delay time." We select the names "CurTime," "LastNotClosableTime," and "ClosureDelay." The value of each will be a number representing time expressed in an appropriate time unit (e.g., millisecond, second). For our description and mathematical model it is essential that all three times be expressed in the same unit, but the unit selected is itself not important.

Additional suggestions for a variable or variables can be extracted from Section 8.13.4. References to the "states of the door system" are: open, opening, closed, and closing, suggesting a state variable taking on those values. The two variables DoorIsOpen and DoorIsClosed distinguish between the states open and closed, so the needed values of the state variable, which we name simply "State," can be reduced to opening and closing. Because the door can also be stopped, this value should be included in the set of values of the variable "State."

An alternative approach to identifying the state variables in the finite state machine in the mathematical model is to consider the possible *positions* of the door and the possible *movements* of the door, independent of one another. Physically, the possible positions are open, closed, and between open and closed. The values of the two variables DoorIsOpen and DoorIsClosed distinguish between these three possible positions of the door, as mentioned in the paragraph above. Physically, the possible movements of the door are opening, closing, and not moving (stationary, stopped). This leads to a state variable whose possible values are opening, closing, and stopped, the same design conclusion as that reached in the paragraph above.

At this point it is appropriate to consider all possible combinations of the values of the variables identified so far. Among these combinations are those in which the door-open input variable DoorIsOpen and the door-closed input variable DoorIsClosed are both true. This combination is illogical—the door cannot be both open and closed at the same time—but because the two sensors are independent, it is physically possible. If these two variables are both true, one or both sensors are faulty. This suggests another possible value for the state variable: "fault." In this case, the controller cannot tell what position the door is in: open, closed, or in between. In general, one should always consider the need for one or more fault states when designing a dynamic system. Often, considering failure situations is a more time-consuming part of the design task than the normal operation, but it is essential for a sufficiently reliable and safe design. It is a critical aspect of engineering responsibility. Mathematical models are helpful in fulfilling this responsibility, too.

In summary, the variables identified are:

- **Input variables:** Note that of the variables above, *PersonIsNear*, *DoorIsOpen*, *DoorIsClosed*, and *CurTime* are input variables whose values are determined by the controller's environment.
- **Output variable:** The variable *MotorPower* is an output variable whose value is determined by the controller and used as input by the motor subsystem.
- **State variables:** The variables *State* and *LastNotClosableTime* are state variables whose values are both determined and used later by the controller.
- **Constant parameter:** The variable *ClosureDelay* is a parameter, that is, a variable whose value does not change during operation of the door system. Its value is used by the controller only to calculate the values of other variables.

8.13.7 Interpretation of the Variables

In summary, the variables in the mathematical model and their interpretation in the context of the door and its controller are listed below. Each variable whose value can change during operation of the door is a subscripted (indexed, array) variable. The subscript (index) n is the point of time at which the value applies. The parameter ClosureDelay is a mathematical variable with a fixed value that does not change during operation of the door, so ClosureDelay is an ordinary (not subscripted) variable.

- **PersonIsNear(n):** indicates whether or not the motion and proximity detectors detect the presence of a person or object near the door at the nth point in time. PersonIsNear(n) $\in$ {false, true}.
- **DoorIsOpen(n):** indicates whether or not the door is in its fully open position at the nth point in time. DoorIsOpen(n) $\in$ {false, true}.
- **DoorIsClosed(n):** indicates whether or not the door is in its fully closed position at the nth point in time. DoorIsClosed(n) $\in$ {false, true}.
- **CurTime(n):** the current clock time at the nth point in time. This time is maintained by a real-time clock in the device implementing the door controller. CurTime(n) $\in \mathbb{R}$.
- **MotorPower(n):** a command signal from the controller instructing the motor circuitry at the nth point in time either to apply power to the motor to close the door or to open the door or not to apply power to the motor (to stop moving the door). MotorPower(n) $\in$ {on to close, on to open, off}.
- **State(n):** the state of motion or fault of the door system at the nth point in time. State(n) $\in$ {closing, opening, stopped, fault}.
- **LastNotClosableTime(n):** at the nth point in time, the time at which the door was most recently not in a closable state. The door is in a closable state in this sense if the door is open and no person is in the vicinity of the door. The door is

not in a closable state, therefore, if the door is not open or if a person is in the vicinity of the door. The value of LastNotClosableTime(n) must be in the same time unit as CurTime(n) above. LastNotClosableTime(n) $\in \mathbb{R}$.

- **ClosureDelay:** the time that should elapse in a closable state before closure of the door is initiated. The value of ClosureDelay must be in the same time unit as CurTime(n) above. ClosureDelay $\in \mathbb{R}$.

8.13.8 The Mathematical Model

The mathematical model of the controller consists of an expression that restricts appropriately the values of the variables listed above. More precisely, it is an expression that defines (1) the value of each state variable at a point in time in terms of the values of the variables at the preceding point in time and (2) the value of each output variable at a point in time in terms of the values of the variables at the preceding or the current point in time. The values of the input variables are determined by the operational environment of the controller, so their values are not restricted by the mathematical model of the controller.

The coarsest restriction on the values of the variables relates to the sets of values they may assume. Expressed mathematically, this restriction is

$$\begin{aligned}
&\text{State(n)} \in \{\text{closing, opening, stopped, fault}\} \wedge \\
&\text{PersonIsNear(n)} \in \{\text{false, true}\} \wedge \\
&\text{DoorIsOpen(n)} \in \{\text{false, true}\} \wedge \text{DoorIsClosed(n)} \in \{\text{false, true}\} \wedge \\
&\text{MotorPower(n)} \in \{\text{on to close, on to open, off}\} \wedge \\
&\text{CurTime(n)} \in \mathbb{R} \wedge \text{LastNotClosableTime(n)} \in \mathbb{R} \wedge \\
&\text{ClosureDelay} \in \mathbb{R} \wedge \text{ClosureDelay} > 0 \qquad [8.13.8\text{-}1]
\end{aligned}$$

This restriction applies for all integer values of n, so expression 8.13.8-1 should be extended accordingly to obtain

$$\begin{aligned}
&[\wedge\ \text{n} : \text{n} \in \mathbb{Z} \wedge \text{n} \geq 0 : \\
&\qquad \text{State(n)} \in \{\text{closing, opening, stopped, fault}\} \wedge \\
&\qquad \text{PersonIsNear(n)} \in \{\text{false, true}\} \wedge \\
&\qquad \text{DoorIsOpen(n)} \in \{\text{false, true}\} \wedge \text{DoorIsClosed(n)} \in \{\text{false, true}\} \wedge \\
&\qquad \text{MotorPower(n)} \in \{\text{on to close, on to open, off}\} \wedge \\
&\qquad \text{CurTime(n)} \in \mathbb{R} \wedge \text{LastNotClosableTime(n)} \in \mathbb{R}] \wedge \\
&\text{ClosureDelay} \in \mathbb{R} \wedge \text{ClosureDelay} > 0 \qquad [8.13.8\text{-}2]
\end{aligned}$$

The values of the state variables are determined only by the controller, not by its environment, so their values at the initial point in time must also be stated. An appropriate initial condition would seem to be in a stopped state with the motor off and with no accumulated waiting time for the purpose of initiating closing the door.

These considerations lead to the following expression:

$$\begin{aligned}
&[\wedge\ n : n\in\mathbb{Z} \wedge n\geq 0 : \\
&\qquad \text{State}(n) \in \{\text{closing, opening, stopped, fault}\} \wedge \\
&\qquad \text{PersonIsNear}(n) \in \{\text{false, true}\} \wedge \\
&\qquad \text{DoorIsOpen}(n) \in \{\text{false, true}\} \wedge \text{DoorIsClosed}(n) \in \{\text{false, true}\} \wedge \\
&\qquad \text{MotorPower}(n) \in \{\text{on to close, on to open, off}\} \wedge \\
&\qquad \text{CurTime}(n) \in \mathbb{R} \wedge \text{LastNotClosableTime}(n) \in \mathbb{R}] \wedge \\
&\text{ClosureDelay} \in \mathbb{R} \wedge \text{ClosureDelay} > 0 \wedge \\
&\text{State}(0)=\text{stopped} \wedge \text{MotorPower}(0)=\text{off} \wedge \\
&\text{LastNotClosableTime}(0)=\text{CurTime}(0) \qquad [8.13.8\text{-}3]
\end{aligned}$$

The value of the parameter ClosureDelay is still unspecified. For the purposes of this example we leave it undefined but note that it is required for a final and complete specification of the door controller. Often, such systems, especially when implemented using a digital computer, allow the values of parameters of this type to be set to suit the nature of each individual installation.

8.13.9 The Controller Function

To mathematical expression 8.13.8-3 we must now append terms defining the values of the state variables (State and LastNotClosableTime) and of the output variable MotorPower at the next point in time. More precisely, we must define State(n+1), LastNotClosableTime(n+1), and MotorPower(n+1) as functions of the values of the variables State(n), PersonIsNear(n), DoorIsOpen(n), DoorIsClosed(n), CurTime(n), LastNotClosableTime(n), and ClosureDelay. These functions we call collectively the *controller function*.

The last three time variables occur in the relevant expressions only in the subexpression

$$(\text{CurTime}(n) - \text{LastNotClosableTime}(n)) > \text{ClosureDelay} \qquad [8.13.9\text{-}1]$$

for the condition for closing the door, see the discussion on closing the door in Section 8.13.5.

Thus, the functions defining State(n+1), LastNotClosableTime(n+1), and MotorPower(n+1) will depend on the following arguments, which take on the following numbers of different values:

- State(n) ∈ {closing, opening, stopped, fault}, four values
- PersonIsNear(n) ∈ {false, true}, two values
- DoorIsOpen(n) ∈ {false, true}, two values
- DoorIsClosed(n) ∈ {false, true}, two values
- [(CurTime(n)−LastNotClosableTime(n)) > ClosureDelay] ∈ {false, true}, two values

The number of combinations of values for these arguments is $4{*}2{*}2{*}2{*}2 = 64$. In principle each combination must be reviewed and checked for correctness and safety of operation before the design of the controller can be considered complete. Tabular notation is typically the notational form of choice for functions depending on arguments having so many combinations of values. A table with one or more rows for headers and one row for each combination of values of the arguments and with one column for each argument and one column for each function whose value is being defined would be a convenient format.

A table defining the functions will, therefore, have a header row and 64 rows, one for each combination of values for the arguments. Some reduction is possible if, for example, for a value of fault for the variable State, the values of the functions do not depend on the other arguments. Then $3{*}2{*}2{*}2{*}2+1 = 49$ rows are needed for the combinations of the values of the arguments. The table would contain five columns for the arguments and three columns for the functions being defined, for a total of eight columns.

The requirements that the controller function must satisfy are given in the English description in Section 8.13.4. Each part of that paragraph will be used to fill in one or more of the rows in the table described above, which defines the controller function mathematically. After the table is filled in by examining each part of the paragraph describing the requirements, it can be checked for (1) consistency, that is, that no two rows contain the same values for the arguments but a different value for a result, and (2) completeness, that is, that every possible combination of the values of the arguments appears in some row of the table. This is a major advantage of tables as a notational form for complex expressions and requirements. Implementing a system from an English description of the requirements can and often does lead to undesired and inexplicable behavior because of unnoticed gaps in the requirements. Situations no one thought of can arise in practice, with unspecified and undefined behavior resulting. With an English text, it is essentially impossible to tell whether or not the statements are consistent and complete; using the Language of Mathematics, it can be determined systematically and reliably whether or not the specification is consistent and complete.

8.13.10 Constructing the Controller Function Table

There are two different systematic approaches to constructing a table such as our controller function table. In the first approach, one constructs a table with all possible combinations of values for the arguments and then, for each row, one by one, examines the relevant requirements in the English description and fills in the results. In the second approach, for each requirement in the English description, one fills in the corresponding values of the arguments and the results in the table. We choose the second approach because it probably illustrates the principles and method of construction of the table more clearly. In practice, either approach can be used. Sometimes it is even appropriate to have two separate designers (or design teams), each using a different approach, construct the table and afterward, compare their tables and resolve the differences.

Taking the second approach mentioned above, we examine each of the requirements in Section 8.13.4, one by one. For each requirement, we insert the appropriate entries in the table. As mentioned above, the table will have the form

INTERMEDIATE TABLE 8.13.10-1 Form of the Controller Function Table

Arguments					Results		
State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)

The argument "ClosureDelayIsExceeded(n)" in the header of the table is a function defined to be the value of the expression

$$(\text{CurTime}(n) - \text{LastNotClosableTime}(n)) > \text{ClosureDelay} \qquad [8.13.10\text{-}1]$$

for all integers $n \geq 0$.

We begin with the first requirement in Section 8.13.4:

- If the motion and proximity detectors detect someone in the vicinity of the door, the door controller should open the door, unless it is already open.

We then insert the appropriate values into the appropriate rows and columns into Table 8.13.10-2.

INTERMEDIATE TABLE 8.13.10-2 Partial Controller Function Table Entries

Arguments					Results		
State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
—	true	false	—	—	opening		on to open

Cells for arguments whose values are irrelevant are marked with an "—." These are sometimes called "don't care" cases. The results in the three columns to the right do not depend on the values of the arguments marked "—." In principle, one could enter all possible combinations of values for the "don't care" arguments, leading to a number of rows being filled in, but this is not necessary at this stage. Later requirements may lead to the need to expand the table entries in this way.

Cells are left blank for results that are not determined by the requirement being examined. Other requirements will, presumably, lead to blank cells being filled in later.

We now go to the next requirement:

- If the motor is on (under power), its direction should not be reversed immediately. Motor power should first be turned off, then turned on in the reverse direction. Any brief off period is sufficient.

This requirement applies to many situations, one of which could arise in the case of the requirement already considered above. This requirement will presumably apply to situations arising later and should, therefore, be kept constantly in mind. Because of this requirement, we must expand the table already constructed above to distinguish between situations in which the motor is on (the state being either "opening" or "closing") and in which the motor is off (the state being "stopped"). If the state is "opening," this requirement permits leaving the state unchanged and motor power on to open. If the state is "closing," this requirement requires setting the state to "stopped" and setting the motor power to "off." If the state is "stopped," this requirement permits turning on motor power. Expanding the one row in Table 8.13.10-2 and correcting the results to satisfy this new requirement, we obtain Table 8.13.10-3, which corrects and replaces Table 8.13.10-2.

INTERMEDIATE TABLE 8.13.10-3 Partial Controller Function Table Entries

Arguments					Results		
State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
closing	true	false	—	—	stopped		off
opening	true	false	—	—	opening		on to open
stopped	true	false	—	—	opening		on to open

If, because of this change, motor power is turned off and the new state becomes "stopped," the last line in Table 8.13.10-3 will, in the next time step, set motor power "on to open" and set the state to "opening," achieving the same effect as Table 8.13.10-2 would have achieved, but with the effect of avoiding an immediate change of direction of the motor (unless, of course, some other event arises in the meantime, causing some other reaction).

The next requirement is:

- If the door is opening and the door sensors indicate that it has reached its open position, power to the motor should be turned off.

INTERMEDIATE TABLE 8.13.10-4 Partial Controller Function Table Entries

Arguments					Results		
State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
opening	—	true	—	—	stopped		off

The next requirement deals with the corresponding situation on closing:

- If the door is closing and the door sensors indicate that it has reached its closed position, power to the motor should be turned off.

INTERMEDIATE TABLE 8.13.10-5 Partial Controller Function Table Entries

Arguments					Results		
State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
closing	—	—	true	—	stopped		off

The next requirement defines when the door is to be closed automatically.

- If the door has been open and no person has been detected in the vicinity of the door for a given period of time (called the closure delay time), the door controller should close the door.

INTERMEDIATE TABLE 8.13.10-6 Partial Controller Function Table Entries

Arguments					Results		
State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
—	—	—	—	true	closing	CurTime(n)	on to close

When the closure delay is exceeded, the door will have been open for at least the closure delay and the motor power will be off. Therefore, the requirement that the motor direction not be changed immediately does not apply.

Upon starting the motor to close, the door is no longer in a closable state, so measuring the delay for closing the door automatically must be reset. This is achieved by setting the variable LastNotClosableTime(n+1) to the value of CurTime(n), as in Table 8.13.10-6.

Implied by this requirement is also the need to set the value of the variable LastNotClosableTime(n+1) in various other situations. This is represented in Table 8.13.10-7 below. [See Sections 8.13.1 and 8.13.5 and the interpretation of the variable "LastNotClosableTime(n)" in Section 8.13.7 for discussions of the logic for closing the door automatically]. The first nonheader row in Table 8.13.10-7 represents a closable state, and the other rows represent nonclosable states. The entry "…(n)" in a cell of a table means the value of the same variable mentioned in the header but for time step n, in this case, LastNotClosableTime(n).

INTERMEDIATE TABLE 8.13.10-7 Partial Controller Function Table Entries

Arguments					Results		
State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
—	false	true	—	—		…(n)	
—	—	false	—	—		CurTime(n)	
—	true	—	—	—		CurTime(n)	

The next requirement covers the situation in which the door is closing and someone approaches the door. Note that another requirement above prohibits changing the motor direction immediately, so the controller should first turn motor power off and then, in a second step, turn motor power on to open.

- If the door is closing and the motion and proximity detectors detect someone in the vicinity of the door, the controller should open the door. (See Section 8.13.1).

INTERMEDIATE TABLE 8.13.10-8 Partial Controller Function Table Entries

Arguments					Results		
State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
closing	true	—	—	—	stopped		off
stopped	true	—	—	—	opening		on to open

This completes translating the requirements in the bulleted list in Section 8.13.4.

The next step is to consolidate the rows in the several tables above into a single table. Collecting the nonheader rows in Tables 8.13.10-3 through 8.13.10-8 inclusive, we obtain Table 8.13.10-9.

INTERMEDIATE TABLE 8.13.10-9 Partial Controller Function Table Entries

Arguments					Results		
State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
closing	true	false	—	—	stopped		off
opening	true	false	—	—	opening		on to open
stopped	true	false	—	—	opening		on to open
opening	—	true	—	—	stopped		off
closing	—	—	true	—	stopped		off
—	—	—	—	true	closing	CurTime(n)	on to close
—	false	true	—	—		…(n)	
—	—	false	—	—		CurTime(n)	
—	true	—	—	—		CurTime(n)	
closing	true	—	—	—	stopped		off
stopped	true	—	—	—	opening		on to open

The state “fault” was discussed briefly earlier but was not included in the requirements. Clearly, if the variables DoorIsOpen(n) and DoorIsClosed(n) both have the value “true,” something is faulty in the door system, and this is an obvious condition in which the value of State(n+1) should become “fault.” The description of the door and its control system says nothing about how the system should behave if this state arises, so at least in this regard the English statement of the requirements is incomplete. At this point in the construction of the controller function table, we detect this fault state and turn motor power off to avoid damaging some hardware component.

Adding the detection of the fault condition mentioned above, we obtain Table8.13.10-10.

INTERMEDIATE TABLE 8.13.10-10 Partial Controller Function Table Entries

Arguments					Results		
State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
closing	true	false	—	—	stopped		off
opening	true	false	—	—	opening		on to open
stopped	true	false	—	—	opening		on to open
opening	—	true	—	—	stopped		off
closing	—	—	true	—	stopped		off
—	—	—	—	true	closing	CurTime(n)	on to close
—	false	true	—	—		…(n)	
—	—	false	—	—		CurTime(n)	
—	true	—	—	—		CurTime(n)	
closing	true	—	—	—	stopped		off
stopped	true	—	—	—	opening		on to open
—	—	true	true	—	fault		off

At this point it is desirable to check the table for consistency and completeness. Because of the many entries “—” it is clear that there is overlap of the conditions covered by some rows; if those rows lead to different values for the results, inconsistency is present. With the many entries “—” it is not easy to see whether all possible combinations of arguments are covered by the table. Therefore, each “—” entry in Table 8.13.10-10 should be expanded. As calculated in Section 8.13.9, the fully expanded table will have 64 nonheader rows. Below, we expand the table fully and represent it in four parts, one for each value of the variable “State(n).”

We begin by expanding the "—" entries in the first column. Each row containing a "—" in the first column will be replaced by four copies, and then the four possible values of the variable "State(n)" will replace the four "—" entries. In the resulting Table 8.13.10-11, a box encloses each group of four rows expanded from a single row.

INTERMEDIATE TABLE 8.13.10-11 Partial Controller Function Table Entries

Arguments					Results		
State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
closing	true	false	—	—	stopped		off
opening	true	false	—	—	opening		on to open
stopped	true	false	—	—	opening		on to open
opening	—	true	—	—	stopped		off
closing	—	—	true	—	stopped		off
closing	—	—	—	true	closing	CurTime(n)	on to close
opening	—	—	—	true	closing	CurTime(n)	on to close
stopped	—	—	—	true	closing	CurTime(n)	on to close
fault	—	—	—	true	closing	CurTime(n)	on to close
closing	false	true	—	—		…(n)	
opening	false	true	—	—		…(n)	
stopped	false	true	—	—		…(n)	
fault	false	true	—	—		…(n)	
closing	—	false	—	—		CurTime(n)	
opening	—	false	—	—		CurTime(n)	
stopped	—	false	—	—		CurTime(n)	
fault	—	false	—	—		CurTime(n)	
closing	true	—	—	—		CurTime(n)	
opening	true	—	—	—		CurTime(n)	
stopped	true	—	—	—		CurTime(n)	
fault	true	—	—	—		CurTime(n)	
closing	true	—	—	—	stopped		off
stopped	true	—	—	—	opening		on to open
closing	—	true	true	—	fault		off
opening	—	true	true	—	fault		off
stopped	—	true	true	—	fault		off
fault	—	true	true	—	fault		off

The sequence of the rows in the table is of no logical significance. Therefore, the rows in Table 8.13.10-11 can be sorted for viewing convenience. The result is Table 8.13.10-12.

INTERMEDIATE TABLE 8.13.10-12 Partial Controller Function Table Entries

Arguments					Results		
State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
closing	—	—	—	true	closing	CurTime(n)	on to close
closing	—	—	true	—	stopped		off
closing	—	false	—	—		CurTime(n)	
closing	—	true	true	—	fault		off
closing	false	true	—	—		…(n)	
closing	true	—	—	—		CurTime(n)	
closing	true	—	—	—	stopped		off
closing	true	false	—	—	stopped		off
fault	—	—	—	true	closing	CurTime(n)	on to close
fault	—	false	—	—		CurTime(n)	
fault	—	true	true	—	fault		off
fault	false	true	—	—		…(n)	
fault	true	—	—	—		CurTime(n)	
opening	—	—	—	true	closing	CurTime(n)	on to close
opening	—	false	—	—		CurTime(n)	
opening	—	true	—	—	stopped		off
opening	—	true	true	—	fault		off
opening	false	true	—	—		…(n)	
opening	true	—	—	—		CurTime(n)	
opening	true	false	—	—	opening		on to open
stopped	—	—	—	true	closing	CurTime(n)	on to close
stopped	—	false	—	—		CurTime(n)	
stopped	—	true	true	—	fault		off
stopped	false	true	—	—		…(n)	
stopped	true	—	—	—		CurTime(n)	
stopped	true	—	—	—	opening		on to open
stopped	true	false	—	—	opening		on to open

In Table 8.13.10-12, notice the pairs of rows surrounded by boxes. In each pair the arguments are the same and the results complement each other; that is, one row defines the value of LastTimeNotClosable(n+1) and the other row defines the values for State(n+1) and MotorPower(n+1). The two rows in each pair can be combined into one row. For convenience, we also divide the table into four parts, one part for each value of State(n). The resulting four tables, 8.13.10-13 through 8.13.10-16, are shown below.

INTERMEDIATE TABLE 8.13.10-13 Partial Controller Function Table Entries

Arguments					Results		
State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
fault	—	—	—	true	closing	CurTime(n)	on to close
fault	—	false	—	—		CurTime(n)	
fault	—	true	true	—	fault		off
fault	false	true	—	—		…(n)	
fault	true	—	—	—		CurTime(n)	

As pointed out earlier, the English text is incomplete with regard to handling fault conditions. The client must be consulted to complete this part of the specification and, if appropriate, even modify entries in the Fault Table 8.13.10-13. After completing the other parts of the controller function table we will return to the fault table, assume certain decisions regarding handling faults, and complete the fault table.

The Closing Table 8.13.10-14 is shown below after combining the two rows with identical arguments and complementary results.

INTERMEDIATE TABLE 8.13.10-14 Partial Controller Function Table Entries

Arguments					Results		
State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
closing	—	—	—	true	closing	CurTime(n)	on to close
closing	—	—	true	—	stopped		off
closing	—	false	—	—		CurTime(n)	
closing	—	true	true	—	fault		off
closing	false	true	—	—		…(n)	
closing	true	—	—	—	stopped	CurTime(n)	off
closing	true	false	—	—	stopped		off

The Opening Table 8.13.10-15 is shown below.

INTERMEDIATE TABLE 8.13.10-15 Partial Controller Function Table Entries

Arguments					Results		
State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
opening	—	—	—	true	closing	CurTime(n)	on to close
opening	—	false	—	—		CurTime(n)	
opening	—	true	—	—	stopped		off
opening	—	true	true	—	fault		off
opening	false	true	—	—		…(n)	
opening	true	—	—	—		CurTime(n)	
opening	true	false	—	—	opening		on to open

The Stopped Table 8.13.10-16 is shown below after combining the two rows with identical arguments and complementary results.

INTERMEDIATE TABLE 8.13.10-16 Partial Controller Function Table Entries

Arguments					Results		
State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
stopped	—	—	—	true	closing	CurTime(n)	on to close
stopped	—	false	—	—		CurTime(n)	
stopped	—	true	true	—	fault		off
stopped	false	true	—	—		…(n)	
stopped	true	—	—	—	opening	CurTime(n)	on to open
stopped	true	false	—	—	opening		on to open

The three Tables 8.13.10-14 through 8.13.10-16, for the states closing, opening, and stopped, will now be expanded and sorted to check for consistency and completeness. We begin with the Closing Table 8.13.10-14. Note that the second and third rows overlap and are complementary, so they can be expanded in order to combine them. The same applies to the last two rows.

In each of these pairs of rows, the nonmatched "—" entries are expanded. Each row containing a "—" to be expanded is replaced by two copies, and then the two values false and true replace the two "—" entries (cf. the expansion of rows in Table 8.13.10-10). Finally, the resulting table is sorted, yielding Table 8.13.10-17.

INTERMEDIATE TABLE 8.13.10-17 Partial Controller Function Table Entries

Arguments					Results		
State (n)	PersonIsNear (n)	DoorIsOpen (n)	DoorIsClosed (n)	ClosureDelayIsExceeded (n)	State (n+1)	LastNotClosableTime (n+1)	MotorPower (n+1)
closing	—	—	—	true	closing	CurTime(n)	on to close
closing	—	false	false	—		CurTime(n)	
closing	—	false	true	—	stopped		off
closing	—	false	true	—		CurTime(n)	
closing	—	true	true	—	stopped		off
closing	—	true	true	—	fault		off
closing	false	true	—	—		…(n)	
closing	true	false	—	—	stopped	CurTime(n)	off
closing	true	false	—	—	stopped		off
closing	true	true	—	—	stopped	CurTime(n)	off

Notice that the third and fourth rows in Table 8.13.10-17, which are boxed, have the same arguments and that the results complement each other. Therefore, these two rows can be combined into one row.

Notice also that the eighth and ninth rows, which are also boxed, have the same arguments and the same or complementary results. These rows can be combined into one row.

Applying these two modifications to Table 8.13.10-17 leads to Table 8.13.10-18.

INTERMEDIATE TABLE 8.13.10-18 Partial Controller Function Table Entries

Arguments					Results		
State (n)	PersonIsNear (n)	DoorIsOpen (n)	DoorIsClosed (n)	ClosureDelayIsExceeded (n)	State (n+1)	LastNotClosableTime (n+1)	MotorPower (n+1)
closing	—	—	—	true	closing	CurTime(n)	on to close
closing	—	false	false	—		CurTime(n)	
closing	—	false	true	—	stopped	CurTime(n)	off
closing	—	true	true	—	stopped		off
closing	—	true	true	—	fault		off
closing	false	true	—	—		…(n)	
closing	true	false	—	—	stopped	CurTime(n)	off
closing	true	true	—	—	stopped	CurTime(n)	off

Notice the boxed fourth and fifth rows in Table 8.13.10-18. Their arguments are the same but their results are different. They are, therefore, inconsistent and contradictory. Because a true value for both variables DoorIsOpen and DoorIsClosed clearly indicates a fault in the detectors sensing the door position, we keep the row with the result "fault" and delete the other row.

The arguments in the first and last rows in Table 8.13.10-18 overlap, but these rows give different results. Some inconsistency is present. The interaction between the two requirements regarding closing the door and opening the door automatically if a person approaches the door while it is closing has not been considered completely and in sufficient detail. To examine these rows in full detail, the "—" entries in the "PersonIsNear" and "DoorIsOpen" columns of the first row and the "—" entry in the "ClosureDelayIsExceeded" column of the last row will be expanded.

Both of the anomalies mentioned above highlight the inadequacy of working from English descriptions of the requirements. They also highlight the value of a mathematical analysis in identifying and correcting the following:

- Ambiguities
- Oversights in interpreting English text
- Inconsistencies

The modifications described above lead, after sorting, to Table 8.13.10-19.

INTERMEDIATE TABLE 8.13.10-19 Partial Controller Function Table Entries

Arguments					Results		
State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
closing	—	false	false	—		CurTime(n)	
closing	—	false	true	—	stopped	CurTime(n)	off
closing	—	true	true	—	fault		off
closing	false	false	—	true	closing	CurTime(n)	on to close
closing	false	true	—	—		…(n)	
closing	false	true	—	true	closing	CurTime(n)	on to close
closing	true	false	—	—	stopped	CurTime(n)	off
closing	true	false	—	true	closing	CurTime(n)	on to close
closing	true	true	—	false	stopped	CurTime(n)	off
closing	true	true	—	true	stopped	CurTime(n)	off
closing	true	true	—	true	closing	CurTime(n)	on to close

The last two boxed rows in Table 8.13.10-19 specify different results for the same argument and are, therefore, inconsistent. When the requirement to close the door automatically was originally translated into a row in the table, only the condition that the closure delay was exceeded was used as the decision criterion. It was overlooked that a person might approach the door in the same time step that the closure delay was exceeded when the door was still open but starting to close. In this case, two different requirements apply simultaneously. Normally in such a case the client should be asked which requirement has priority over the other, but in this case the need to open the door for the approaching person can be assumed to take precedence over closing the door. Then the last row in Table 8.13.10-19 is incorrect and can be deleted from the table.

The result is Table 8.13.10-20.

INTERMEDIATE TABLE 8.13.10-20 Partial Controller Function Table Entries

Arguments					Results		
State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
closing	—	false	false	—		CurTime(n)	
closing	—	false	true	—	stopped	CurTime(n)	off
closing	—	true	true	—	fault		off
closing	false	false	—	true	closing	CurTime(n)	on to close
closing	false	true	—	—		…(n)	
closing	false	true	—	true	closing	CurTime(n)	on to close
closing	true	false	—	—	stopped	CurTime(n)	off
closing	true	false	—	true	closing	CurTime(n)	on to close
closing	true	true	—	false	stopped	CurTime(n)	off
closing	true	true	—	true	stopped	CurTime(n)	off

In Table 8.13.10-20, other overlapping rows still contain inconsistencies, and two of the rows give incomplete results. To resolve the remaining deficiencies, we expand every "—" entry still present, sort the table, and insert row numbers. The result is Table 8.13.10-21.

INTERMEDIATE TABLE 8.13.10-21 Partial Controller Function Table Entries

	Arguments					Results		
Row	State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
1	closing	false	false	false	false		CurTime(n)	
2	closing	false	false	false	true	closing	CurTime(n)	on to close
3	closing	false	false	false	true		CurTime(n)	
4	closing	false	false	true	false	stopped	CurTime(n)	off
5	closing	false	false	true	true	closing	CurTime(n)	on to close
6	closing	false	false	true	true	stopped	CurTime(n)	off
7	closing	false	true	false	false		…(n)	
8	closing	false	true	false	true		…(n)	
9	closing	false	true	false	true	closing	CurTime(n)	on to close
10	closing	false	true	true	false		…(n)	
11	closing	false	true	true	false	fault		off
12	closing	false	true	true	true		…(n)	
13	closing	false	true	true	true	closing	CurTime(n)	on to close
14	closing	false	true	true	true	fault		off
15	closing	true	false	false	false	stopped	CurTime(n)	off
16	closing	true	false	false	false		CurTime(n)	
17	closing	true	false	false	true	stopped	CurTime(n)	off
18	closing	true	false	false	true	closing	CurTime(n)	on to close
19	closing	true	false	false	true		CurTime(n)	
20	closing	true	false	true	false	stopped	CurTime(n)	off
21	closing	true	false	true	false	stopped	CurTime(n)	off
22	closing	true	false	true	true	stopped	CurTime(n)	off
23	closing	true	false	true	true	closing	CurTime(n)	on to close
24	closing	true	false	true	true	stopped	CurTime(n)	off
25	closing	true	true	false	false	stopped	CurTime(n)	off
26	closing	true	true	false	true	stopped	CurTime(n)	off
27	closing	true	true	true	false	stopped	CurTime(n)	off
28	closing	true	true	true	false	fault		off
29	closing	true	true	true	true	stopped	CurTime(n)	off
30	closing	true	true	true	true	fault		off

Row 1 is the only row with its arguments, but its results are incomplete. The English requirements do not cover this situation in which the door is closing, is neither open nor closed, and the criteria for closing the door automatically are not met. Presumably the door should continue to close, so the next state should remain "closing" and the motor power should be maintained “on to close.” See the more extensive discussion below regarding row 1 of Table 8.13.10-30 and the corresponding situation in which the door is opening and the other arguments are all false.

Rows 2 and 3 have the same arguments and their results are consistent, although row 3 is incomplete. Row 2 should be retained and row 3 deleted.

Row 4 is the only row with its arguments and its results complete. It should be retained.

Rows 5 and 6 have the same arguments but give different results. This is another case for which two of the original English requirements are relevant: the conditions for automatically closing the door are fulfilled, but the door was closing anyway and had just reached the closed position. Row 6 is appropriate in order to stop the door movement because it is in the closed position. Row 5 should be deleted. It might turn out that the combination State(n)=closing and ClosureDelayIsExceeded(n)=true is an unreachable state, that is, can never arise, in which case this row would be irrelevant. At this stage of the design it is inappropriate to make such an assumption. Also, a malfunction of the system might lead to this state. It should be included in the analysis and design.

Row 7 is the only row with its arguments and its results incomplete. The English requirements do not explicitly cover this situation in which the door is closing, the door is still open, no person is near the door, and the condition for closing the door automatically is not fulfilled. This situation can arise shortly after the controller has initiated closing the door automatically but the door has as yet moved so little that it is still in the open position as detected by the door sensor. Presumably the door should continue to close, so State(n+1) should have the value “closing” and MotorPower(n+1) should have the value “on to close.”

Rows 8 and 9 have the same arguments but give different results. In addition, the results in row 8 are incomplete. The different results arise from the coarse form of the English requirements for closing the door automatically and the interaction between this requirement and the similarly coarse English formulation of the criterion for calculating the value of the state variable LastNotClosableTime(n+1). The English formulations of these requirements do not distinguish between different values of seemingly irrelevant input variables and in the process of expanding the corresponding rows, the different values for LastNotClosableTime(n+1) arise. It might turn out that rows 8 and 9 represent an unreachable state, but for the reasons given in the paragraph about rows 5 and 6 above, one of them should be included in the table. The question remains, “if the door is closing, no one is near the door, the door is still sensed as being open, and the condition for closing the door automatically is fulfilled, what should the values of the results be?” As in similar situations discussed previously, the controller should continue closing the door. To effectively reset the timer for closing the door automatically, the value of LastNotClosableTime(n+1) should be the current time CurTime(n). Viewed differently but equivalently, the door is, in a sense, no longer “logically” open, so this situation should be handled in the same way as in row 2. Row 8 should be deleted and row 9 should be retained.

Rows 10 and 11 have the same arguments and complementary results. These two rows can be combined into one row.

Rows 12, 13, and 14 have the same arguments but give different results. In addition, the results in rows 12 and 14 are incomplete. Both input variables DoorIsOpen(n) and DoorIsClosed(n) have the value true, indicating a fault condition. These rows should be combined with, presumably, the same results as for rows 10 and 11.

Rows 15 and 16 have the same arguments and their results are consistent, although row 16 is incomplete. Row 15 should be retained and row 16 deleted.

Rows 17, 18, and 19 have the same arguments. Rows 17 and 18 give different results. The results in row 19 are incomplete but consistent with both rows 17 and 18, so row 19 can be deleted. To choose between row 17 and row 18, consider the current situation: The door is closing, a person is near the door, the door is neither open nor closed, and the condition for closing the door automatically is fulfilled. Clearly, the door should be opened to permit the person to pass. This excludes row 18, which must be deleted. Because of the requirement that the motor direction may not be reversed immediately, the controller must first stop the door movement. Thus row 17 gives the appropriate result. (In the following time step the controller will open the door.) Compare the discussion about rows 5 and 6 above.

Rows 20 and 21 are identical. Either one can be deleted.

Rows 22, 23, and 24 have the same arguments. Rows 22 and 24 give the same results, but row 23 gives different results. In this situation, the door is closing, a person is near the door, the door is closed (has just reached the closed position) and the condition for closing the door automatically is (has just become) fulfilled. Clearly, the door should be opened to permit the person to pass, but because of the requirement that the motor direction may not be reversed immediately, the controller must first stop the door movement. Thus the identical rows 22 and 24 give the appropriate result. Either row 22 or row 24 can be deleted, and row 23 must be deleted. This situation is comparable to other situations discussed above.

Row 25 is the only row with its arguments and its results are complete. It should be retained.

Row 26 is the only row with its arguments and its results are complete. It should be retained.

Rows 27 and 28 have the same arguments but give different results. Both input variables DoorIsOpen(n) and DoorIsClosed(n) have the value true, indicating a fault condition. Row 27 gives CurTime(n) as the value of the variable LastNotClosableTime(n+1), while row 28 has no entry for this value, so we will take CurTime(n). Furthermore, accepting this value effectively disables automatic door closing for the next door closure delay, which seems to be the safer option in this fault condition. Both rows 27 and 28 give off as the value of MotorPower(n+1). As mentioned earlier, the subject of the fault state must be reexamined because the English requirements for handling faults are incomplete.

Rows 29 and 30 have the same arguments but give different results. Both input variables DoorIsOpen(n) and DoorIsClosed(n) have the value true, indicating a fault condition. See the comments above regarding rows 27 and 28.

After applying the modifications described above, the final table for the state “closing” is Table 8.13.10-22.

INTERMEDIATE TABLE 8.13.10-22 Partial Controller Function Table Entries

	Arguments					Results		
Row	State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
1	closing	false	false	false	false	closing	CurTime(n)	on to close
2	closing	false	false	false	true	closing	CurTime(n)	on to close
4	closing	false	false	true	false	stopped	CurTime(n)	off
6	closing	false	false	true	true	stopped	CurTime(n)	off
7	closing	false	true	false	false	closing	…(n)	on to close
9	closing	false	true	false	true	closing	CurTime(n)	on to close
11	closing	false	true	true	false	fault	…(n)	off
14	closing	false	true	true	true	fault	…(n)	off
15	closing	true	false	false	false	stopped	CurTime(n)	off
17	closing	true	false	false	true	stopped	CurTime(n)	off
21	closing	true	false	true	false	stopped	CurTime(n)	off
24	closing	true	false	true	true	stopped	CurTime(n)	off
25	closing	true	true	false	false	stopped	CurTime(n)	off
26	closing	true	true	false	true	stopped	CurTime(n)	off
28	closing	true	true	true	false	fault	CurTime(n)	off
30	closing	true	true	true	true	fault	CurTime(n)	off

Table 8.13.10-22 above contains 16 data rows. It contains all possible combinations of values for the arguments PersonIsNear(n), DoorIsOpen(n), DoorIsClosed(n), and ClosureDelayIsExceeded(n). It is, therefore, complete for the state “closing.”

The reader should examine carefully the structure of Table 8.13.10-22. In the column for PersonIsNear(n), the first eight data rows contain the value false and the last eight rows, the value true. In the next column, for DoorIsOpen(n), the first four data rows contain the value false; the next four rows, the value true; the next four rows, the value false; and the last four rows, the value true; that is, the values false and true alternate in groups of four each. In the column for DoorIsClosed(n), the values false and true alternate in groups of two each. In the last argument column, for ClosureDelayIsExceeded(n), the values false and true alternate. This pattern makes it particularly easy to verify visually that all combinations of values for these arguments are included in the table. If any combination of values for these arguments were missing, it would be clearly visible which was missing. If any combination of values were present more than once, this situation would also be clearly visible.

Next, Table 8.13.10-15 for the state “opening” will be fully expanded and modified as necessary, as was done for the closing table above. Here, however, the table entries will be expanded column by column. This procedure is more systematic than the one followed above but will still avoid the extremely long table that would result if the table were initially expanded fully on all columns at once.

The Opening Table 8.13.10-15 is repeated here as Table 8.13.10-23.

INTERMEDIATE TABLE 8.13.10-23 Partial Controller Function Table Entries

Arguments					Results		
State (n)	PersonIsNear (n)	DoorIsOpen (n)	DoorIsClosed (n)	ClosureDelayIsExceeded (n)	State (n+1)	LastNotClosableTime (n+1)	MotorPower (n+1)
opening	—	—	—	true	closing	CurTime(n)	on to close
opening	—	false	—	—		CurTime(n)	
opening	—	true	—	—	stopped		off
opening	—	true	true	—	fault		off
opening	false	true	—	—		…(n)	
opening	true	—	—	—		CurTime(n)	
opening	true	false	—	—	opening		on to open

We begin by expanding the entries with "—" in the column for the argument PersonIsNear(n). Sorting the table and adding a column for row numbers results in Table 8.13.10-24.

INTERMEDIATE TABLE 8.13.10-24 Partial Controller Function Table Entries

	Arguments					Results		
Row	State (n)	PersonIsNear (n)	DoorIsOpen (n)	DoorIsClosed (n)	ClosureDelayIsExceeded (n)	State (n+1)	LastNotClosableTime (n+1)	MotorPower (n+1)
1	opening	false	—	—	true	closing	CurTime(n)	on to close
2	opening	false	false	—	—		CurTime(n)	
3	opening	false	true	—	—	stopped		off
4	opening	false	true	—	—		…(n)	
5	opening	false	true	true	—	fault		off
6	opening	true	—	—	true	closing	CurTime(n)	on to close
7	opening	true	—	—	—		CurTime(n)	
8	opening	true	false	—	—		CurTime(n)	
9	opening	true	false	—	—	opening		on to open
10	opening	true	true	—	—	stopped		off
11	opening	true	true	true	—	fault		off

In Table 8.13.10-24, rows 3 and 4 have the same arguments and complementary results. These two rows can be combined.

Rows 8 and 9 have the same arguments and complementary results. These two rows can be combined.

The result is Table 8.13.10-25.

INTERMEDIATE TABLE 8.13.10-25 Partial Controller Function Table Entries

	Arguments					Results		
Row	State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
1	opening	false	—	—	true	closing	CurTime(n)	on to close
2	opening	false	false	—	—		CurTime(n)	
3	opening	false	true	—	—	stopped	…(n)	off
5	opening	false	true	true	—	fault		off
6	opening	true	—	—	true	closing	CurTime(n)	on to close
7	opening	true	—	—	—		CurTime(n)	
9	opening	true	false	—	—	opening	CurTime(n)	on to open
10	opening	true	true	—	—	stopped		off
11	opening	true	true	true	—	fault		off

Next, the "—" entries in the row for the argument DoorIsOpen(n) are expanded. After sorting and reassigning the row numbers we obtain Table 8.13.10-26.

INTERMEDIATE TABLE 8.13.10-26 Partial Controller Function Table Entries

	Arguments					Results		
Row	State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
1	opening	false	false	—	—		CurTime(n)	
2	opening	false	false	—	true	closing	CurTime(n)	on to close
3	opening	false	true	—	—	stopped	…(n)	off
4	opening	false	true	—	true	closing	CurTime(n)	on to close
5	opening	false	true	true	—	fault		off
6	opening	true	false	—	—		CurTime(n)	
7	opening	true	false	—	—	opening	CurTime(n)	on to open
8	opening	true	false	—	true	closing	CurTime(n)	on to close
9	opening	true	true	—	—		CurTime(n)	
10	opening	true	true	—	—	stopped		off
11	opening	true	true	—	true	closing	CurTime(n)	on to close
12	opening	true	true	true	—	fault		off

In Table 8.13.10-26, rows 6 and 7 have the same arguments. The results in row 6 are incomplete but consistent with the results in row 7. Row 6 can be deleted and row 7 retained.

Rows 9 and 10 have the same arguments and complementary results. These rows can be combined. The result is Table 8.13.10-27.

INTERMEDIATE TABLE 8.13.10-27 Partial Controller Function Table Entries

	Arguments					Results		
Row	State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
1	opening	false	false	—	—		CurTime(n)	
2	opening	false	false	—	true	closing	CurTime(n)	on to close
3	opening	false	true	—	—	stopped	…(n)	off
4	opening	false	true	—	true	closing	CurTime(n)	on to close
5	opening	false	true	true	—	fault		off
7	opening	true	false	—	—	opening	CurTime(n)	on to open
8	opening	true	false	—	true	closing	CurTime(n)	on to close
10	opening	true	true	—	—	stopped	CurTime(n)	off
11	opening	true	true	—	true	closing	CurTime(n)	on to close
12	opening	true	true	true	—	fault		off

Next, we repeat the process above by expanding the "—" entries in the column for DoorIsClosed(n). After sorting and reassigning the row numbers, the result is Table 8.13.10-28.

INTERMEDIATE TABLE 8.13.10-28 Partial Controller Function Table Entries

	Arguments					Results		
Row	State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
1	opening	false	false	false	—		CurTime(n)	
2	opening	false	false	false	true	closing	CurTime(n)	on to close
3	opening	false	false	true	—		CurTime(n)	
4	opening	false	false	true	true	closing	CurTime(n)	on to close
5	opening	false	true	false	—	stopped	…(n)	off
6	opening	false	true	false	true	closing	CurTime(n)	on to close
7	opening	false	true	true	—	stopped	…(n)	off
8	opening	false	true	true	—	fault		off
9	opening	false	true	true	true	closing	CurTime(n)	on to close
10	opening	true	false	false	—	opening	CurTime(n)	on to open
11	opening	true	false	false	true	closing	CurTime(n)	on to close
12	opening	true	false	true	—	opening	CurTime(n)	on to open
13	opening	true	false	true	true	closing	CurTime(n)	on to close
14	opening	true	true	false	—	stopped	CurTime(n)	off
15	opening	true	true	false	true	closing	CurTime(n)	on to close
16	opening	true	true	true	—	stopped	CurTime(n)	off
17	opening	true	true	true	—	fault		off
18	opening	true	true	true	true	closing	CurTime(n)	on to close

In Table 8.13.10-28, rows 7 and 8 have the same arguments but give different results. In this situation, both the arguments DoorIsOpen(n) and DoorIsClosed(n) have the value true, indicating a fault in the subsystem sensing the position of the door. Rows 7 and 8 can be combined, taking the values from row 8 where the two rows conflict.

Rows 16 and 17 have the same arguments but give different results. In this situation, both the arguments DoorIsOpen(n) and DoorIsClosed(n) have the value true, indicating a fault in the subsystem sensing the position of the door. As in the case of rows 7 and 8 above, rows 16 and 17 can be combined, taking the values from row 17, where the two rows conflict.

The result is Table 8.13.10-29.

INTERMEDIATE TABLE 8.13.10-29 Partial Controller Function Table Entries

	Arguments					Results		
Row	State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
1	opening	false	false	false	—		CurTime(n)	
2	opening	false	false	false	true	closing	CurTime(n)	on to close
3	opening	false	false	true	—		CurTime(n)	
4	opening	false	false	true	true	closing	CurTime(n)	on to close
5	opening	false	true	false	—	stopped	…(n)	off
6	opening	false	true	false	true	closing	CurTime(n)	on to close
8	opening	false	true	true	—	fault	…(n)	off
9	opening	false	true	true	true	closing	CurTime(n)	on to close
10	opening	true	false	false	—	opening	CurTime(n)	on to open
11	opening	true	false	false	true	closing	CurTime(n)	on to close
12	opening	true	false	true	—	opening	CurTime(n)	on to open
13	opening	true	false	true	true	closing	CurTime(n)	on to close
14	opening	true	true	false	—	stopped	CurTime(n)	off
15	opening	true	true	false	true	closing	CurTime(n)	on to close
17	opening	true	true	true	—	fault	CurTime(n)	off
18	opening	true	true	true	true	closing	CurTime(n)	on to close

Next, we expand the "—" entries in the last column of the arguments. After sorting and reassigning the row numbers, we obtain Table 8.13.10-30.

INTERMEDIATE TABLE 8.13.10-30 Partial Controller Function Table Entries

	Arguments					Results		
Row	State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
1	opening	false	false	false	false		CurTime(n)	
2	opening	false	false	false	true		CurTime(n)	
3	opening	false	false	false	true	closing	CurTime(n)	on to close
4	opening	false	false	true	false		CurTime(n)	
5	opening	false	false	true	true		CurTime(n)	
6	opening	false	false	true	true	closing	CurTime(n)	on to close
7	opening	false	true	false	false	stopped	…(n)	off
8	opening	false	true	false	true	stopped	…(n)	off
9	opening	false	true	false	true	closing	CurTime(n)	on to close
10	opening	false	true	true	false	fault	…(n)	off
11	opening	false	true	true	true	fault	…(n)	off
12	opening	false	true	true	true	closing	CurTime(n)	on to close
13	opening	true	false	false	false	opening	CurTime(n)	on to open
14	opening	true	false	false	true	opening	CurTime(n)	on to open
15	opening	true	false	false	true	closing	CurTime(n)	on to close
16	opening	true	false	true	false	opening	CurTime(n)	on to open
17	opening	true	false	true	true	opening	CurTime(n)	on to open
18	opening	true	false	true	true	closing	CurTime(n)	on to close
19	opening	true	true	false	false	stopped	CurTime(n)	off
20	opening	true	true	false	true	stopped	CurTime(n)	off
21	opening	true	true	false	true	closing	CurTime(n)	on to close
22	opening	true	true	true	false	fault	CurTime(n)	off
23	opening	true	true	true	true	fault	CurTime(n)	off
24	opening	true	true	true	true	closing	CurTime(n)	on to close

In Table 8.13.10-30, row 1 is the only row with its arguments, but its results are incomplete. The English requirements were incomplete in the sense that they did not state that if already opening, the door should continue opening unless some stated requirement gives a reason to stop opening the door. This is an example of incomplete English statements of requirements, arising in this case at least partially because of the linguistic difference between English, which permits expressing action and changes, and the Language of Mathematics, which does not permit expressing action and changes, but only static relationships. Therefore, the results of row 1 should be completed to continue opening the door. Compare the situation in

Table 8.13.10-21, in which the door is closing and all other arguments have the value false.

Rows 2 and 3 have the same arguments. The results in row 2 are incomplete but consistent with the results in row 3. Row 2 can be deleted. However, row 3 calls for the motor direction to be reversed immediately, in violation of the requirement that this not happen. When that requirement was translated into a row in the table, it was argued that the door must still be open; otherwise, the condition for closing the door automatically would not be satisfied. Presumably this state is unreachable, but in case some fault results in it arising, the motor power should not be reversed. The next state should be either opening or stopped, not closing. The door could be left in its opening state. In any event, this row in the table should be noted for a detailed analysis after the entire table is complete. The desired behavior of the system in this and other comparable situations (see below) should be discussed with the client for whom the door controller is being designed.

It has been assumed implicitly that the time between controller steps is much smaller than the closure delay. One should analyze the behavior of the completed table to determine whether a violation of this implicit assumption could enable the situation in row 3 to arise. In any case, this assumption should be written explicitly into the specifications for the door controller—both the English and mathematical versions.

Here we replace the results in row 3 to continue opening the door.

Row 4 is the only row with its arguments, but its results are incomplete. In this situation, the door is opening (has just started to open) but is still in a position sensed as closed. This is another example of the incomplete English requirements as discussed with regard to row 1. The results of row 4 should be completed so that the door continues to open.

Rows 5 and 6 have the same arguments. The results in row 5 are incomplete but consistent with the results in row 6. Row 5 can be deleted. However, row 6 calls for the motor direction to be reversed immediately, in violation of the requirement that this not be done. This state is presumably unreachable (see the discussion about rows 2 and 3 above). The door could be left in its opening state or stopped. If stopped, the door might travel beyond the positions sensed as closed, and left neither closed nor open. If this alternative is chosen, the stopped table should be reviewed to ensure that the door is never left stopped and partially open (i.e., neither open nor closed). Here we choose to continue to open the door, but note that this situation should be further analyzed and discussed with the client.

Row 7 is the only row with its arguments and is complete.

Rows 8 and 9 have the same arguments but give different results. Row 9 calls for reversing the motor direction immediately, violating one of the requirements. Row 9 can be deleted. Row 8 represents the situation in which the door is opening, has just reached the open position and the condition for closing the door automatically is satisfied. This is unrealistic and presumably represents an unreachable state. In any case, stopping door movement is appropriate because the door has reached its open position. Automatic closure will be initiated in the next time step because the condition for closing the door automatically will still be satisfied.

Row 10 is the only row with its arguments. It represents a fault condition. The English requirements are incomplete regarding the fault state. Row 10 should be retained as is.

Rows 11 and 12 have the same arguments but give different results. This situation clearly represents a fault condition, so row 11 should be retained and row 12 deleted.

Row 13 is the only row with its arguments and is complete.

Rows 14 and 15 have the same arguments but give different results. The condition for closing the door automatically is fulfilled even though the door is not in its open position. As in similar situations discussed above, this state is presumably unreachable. Row 15 calls for reversing the motor direction immediately in violation of one of the English requirements, so row 15 should be deleted. Because a person is near the door, row 14 should be retained so that the door will continue opening.

Row 16 is the only row with its arguments and is complete.

Rows 17 and 18 have the same arguments but give different results. This situation is similar to that of rows 14 and 15 above. For the same reasons as given in the discussion of rows 14 and 15, row 17 should be retained. Row 18 should be deleted.

Row 19 is the only row with its arguments and is complete.

Rows 20 and 21 have the same arguments but give different results. This state, in which the condition for closing the door automatically is fulfilled although the door is still opening, is presumably unreachable. Row 21 calls for reversing the motor direction immediately in violation of one of the English requirements, so row 21 should be deleted. Row 20, which stops door movement, should be retained, because the door has just reached its open position and because a person is near the door.

Row 22 is the only row with its arguments. It represents a fault condition. The English requirements are incomplete regarding the fault state. Row 22 should be retained as is.

Rows 23 and 24 have the same arguments but give different results. This situation is similar to that of rows 11 and 12 above and represents a fault condition. Row 23 should be retained and row 24, deleted.

After applying the modifications described above, the resulting final table for the state “opening” is Table 8.13.10-31.

INTERMEDIATE TABLE 8.13.10-31 Partial Controller Function Table Entries

	Arguments					Results		
Row	State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
1	opening	false	false	false	false	opening	CurTime(n)	on to open
3	opening	false	false	false	true	opening	CurTime(n)	on to open
4	opening	false	false	true	false	opening	CurTime(n)	on to open
6	opening	false	false	true	true	opening	CurTime(n)	on to open
7	opening	false	true	false	false	stopped	…(n)	off
8	opening	false	true	false	true	stopped	…(n)	off
10	opening	false	true	true	false	fault	…(n)	off
11	opening	false	true	true	true	fault	…(n)	off
13	opening	true	false	false	false	opening	CurTime(n)	on to open
14	opening	true	false	false	true	opening	CurTime(n)	on to open
16	opening	true	false	true	false	opening	CurTime(n)	on to open
17	opening	true	false	true	true	opening	CurTime(n)	on to open
19	opening	true	true	false	false	stopped	CurTime(n)	off
20	opening	true	true	false	true	stopped	CurTime(n)	off
22	opening	true	true	true	false	fault	CurTime(n)	off
23	opening	true	true	true	true	fault	CurTime(n)	off

Table 8.13.10-31 contains 16 data rows. It contains all possible combinations of values for the arguments PersonIsNear(n), DoorIsOpen(n), DoorIsClosed(n), and ClosureDelayIsExceeded(n). It is, therefore, complete for the state “opening.” Note that it has the same structure as the Final Table 8.13.10-22 for the state “closing” (see above).

Next, Table 8.13.10-16 for the state “stopped” will be fully expanded and modified as necessary in the same way as for the opening table above. The table entries will be expanded column by column. In the case of the opening table above, the entries were expanded beginning with the leftmost column, proceeding column by column to the right. This is not necessary; the columns can be expanded in any order.

Stopped Table 8.13.10-16 is repeated as Table 8.13.10-32.

INTERMEDIATE TABLE 8.13.10-32 Partial Controller Function Table Entries

Arguments					Results		
State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
stopped	—	—	—	true	closing	CurTime(n)	on to close
stopped	—	false	—	—		CurTime(n)	
stopped	—	true	true	—	fault		off
stopped	false	true	—	—		…(n)	
stopped	true	—	—	—	opening	CurTime(n)	on to open
stopped	true	false	—	—	opening		on to open

We begin by expanding the entries with "—" in the column for the argument DoorIsOpen(n). Sorting the table and adding a column for row numbers yields Table 8.13.10-33.

INTERMEDIATE TABLE 8.13.10-33 Partial Controller Function Table Entries

	Arguments					Results		
Row	State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
1	stopped	—	false	—	—		CurTime(n)	
2	stopped	—	false	—	true	closing	CurTime(n)	on to close
3	stopped	—	true	—	true	closing	CurTime(n)	on to close
4	stopped	—	true	true	—	fault		off
5	stopped	false	true	—	—		…(n)	
6	stopped	true	false	—	—	opening	CurTime(n)	on to open
7	stopped	true	false	—	—	opening		on to open
8	stopped	true	true	—	—	opening	CurTime(n)	on to open

In Table 8.13.10-33, rows 6 and 7 have the same arguments and complementary results. Row 6, with its complete results, should be retained. Row 7, with its incomplete results, can be deleted.

The result is Table 8.13.10-34.

INTERMEDIATE TABLE 8.13.10-34 Partial Controller Function Table Entries

	Arguments					Results		
Row	State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
1	stopped	—	false	—	—		CurTime(n)	
2	stopped	—	false	—	true	closing	CurTime(n)	on to close
3	stopped	—	true	—	true	closing	CurTime(n)	on to close
4	stopped	—	true	true	—	fault		off
5	stopped	false	true	—	—		…(n)	
6	stopped	true	false	—	—	opening	CurTime(n)	on to open
8	stopped	true	true	—	—	opening	CurTime(n)	on to open

Next, we expand the "—" entries in the column for PersonIsNear(n). After sorting and reassigning the row numbers, the result is Table 8.13.10-35.

INTERMEDIATE TABLE 8.13.10-35 Partial Controller Function Table Entries

	Arguments					Results		
Row	State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
1	stopped	false	false	—	—		CurTime(n)	
2	stopped	false	false	—	true	closing	CurTime(n)	on to close
3	stopped	false	true	—	—		…(n)	
4	stopped	false	true	—	true	closing	CurTime(n)	on to close
5	stopped	false	true	true	—	fault		off
6	stopped	true	false	—	—		CurTime(n)	
7	stopped	true	false	—	—	opening	CurTime(n)	on to open
8	stopped	true	false	—	true	closing	CurTime(n)	on to close
9	stopped	true	true	—	—	opening	CurTime(n)	on to open
10	stopped	true	true	—	true	closing	CurTime(n)	on to close
11	stopped	true	true	true	—	fault		off

In Table 8.13.10-35 rows 6 and 7 have the same arguments and consistent results. Row 6, with incomplete results, can be deleted. Row 7, with complete results, should be retained.

The result is Table 8.13.10-36.

INTERMEDIATE TABLE 8.13.10-36 Partial Controller Function Table Entries

	Arguments					Results		
Row	State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
1	stopped	false	false	—	—		CurTime(n)	
2	stopped	false	false	—	true	closing	CurTime(n)	on to close
3	stopped	false	true	—	—		…(n)	
4	stopped	false	true	—	true	closing	CurTime(n)	on to close
5	stopped	false	true	true	—	fault		off
7	stopped	true	false	—	—	opening	CurTime(n)	on to open
8	stopped	true	false	—	true	closing	CurTime(n)	on to close
9	stopped	true	true	—	—	opening	CurTime(n)	on to open
10	stopped	true	true	—	true	closing	CurTime(n)	on to close
11	stopped	true	true	true	—	fault		off

Next, we expand the "—" entries in the column for ClosureDelayIsExceeded(n). After sorting and reassigning the row numbers, we obtain Table 8.13.10-37.

INTERMEDIATE TABLE 8.13.10-37 Partial Controller Function Table Entries

	Arguments					Results		
Row	State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
1	stopped	false	false	—	false		CurTime(n)	
2	stopped	false	false	—	true		CurTime(n)	
3	stopped	false	false	—	true	closing	CurTime(n)	on to close
4	stopped	false	true	—	false		…(n)	
5	stopped	false	true	—	true		…(n)	
6	stopped	false	true	—	true	closing	CurTime(n)	on to close
7	stopped	false	true	true	false	fault		off
8	stopped	false	true	true	true	fault		off
9	stopped	true	false	—	false	opening	CurTime(n)	on to open
10	stopped	true	false	—	true	opening	CurTime(n)	on to open
11	stopped	true	false	—	true	closing	CurTime(n)	on to close
12	stopped	true	true	—	false	opening	CurTime(n)	on to open
13	stopped	true	true	—	true	opening	CurTime(n)	on to open
14	stopped	true	true	—	true	closing	CurTime(n)	on to close
15	stopped	true	true	true	false	fault		off
16	stopped	true	true	true	true	fault		off

In Table 8.13.10-37, rows 2 and 3 have the same arguments and consistent results. The results in row 3 are complete, so row 3 should be retained. Row 2 can be deleted.

Row 4 is the only row with its arguments. Its results are incomplete. The door is stopped in the open position, no person is near the door, and the closure delay is not yet exceeded. The door should remain stopped. This is another example of an implied requirement: In the absence of any reason to do otherwise, the current status should be continued. Therefore, the door should remain in the stopped state unless the door is sensed as also being closed, a fault condition that will be noticed later after the "—" entries in the DoorIsClosed(n) column are expanded.

Rows 5 and 6 have the same arguments but different results. Because the closure delay is exceeded, the door should be closed, as called for in row 6. Row 6 should be retained. Row 5 can be deleted.

Row 7 is the only row with its arguments. Its results are incomplete because the English requirements for a fault condition are incomplete. Row 7 should be retained.

Row 8 is the only row with its arguments. As in the case of row 7, the results in row 8 are incomplete. Row 8 should be retained.

Row 9 is the only row with its arguments. Its results are complete. Row 9 should be retained.

Rows 10 and 11 have the same arguments but different results. In this example, two different English requirements contradict each other, one to open the door for a person approaching the door, and the other to close the door automatically when the closure delay has elapsed. Presumably the first of these two requirements should have priority, so row 10 should be retained. Row 11 should be deleted.

Row 12 is the only row with its arguments. Its results are complete. However, note that it calls for opening the door when the door is already in its open position. The door should be left in the open position in the state "stopped." This error arose above when the last English requirement was translated into the initial table entries. The English requirement called for opening the door when it was closing and a person approached the door. Because of the other requirement that the motor direction may not be reversed immediately, table entries were formulated that first stopped the door and then opened it, with the door position as a "don't care" condition. One cannot say that the translation is wrong, because nowhere do the English requirements state that the motor may not be turned on to open the door when it is already open. Such "obvious" but unstated comments are a common cause of misunderstanding in communication in general. They can and do give rise to critical, costly, and even dangerous errors in designing technical artifacts of all kinds. Translating into the exacting and very detailed Language of Mathematics and then reviewing the translation helps considerably to identify such oversights and then to correct the resulting errors.

Rows 13 and 14 have the same arguments but different results. These two rows are similar to rows 10 and 11 and result from two different English requirements. Presumably the door should be left open for the person approaching the door and not closed because the closure time has elapsed. Row 13 should be modified to leave the door stopped and open (see the discussion of row 12 above). Row 14 should be deleted.

Row 15 is the only row with its arguments. Its results are incomplete because the English requirements for a fault condition are incomplete. Row 15 should be retained.

Row 16 is the only row with its arguments. Its results are incomplete because the English requirements for a fault condition are incomplete. Row 16 should be retained.

The result is Table 8.13.10-38.

INTERMEDIATE TABLE 8.13.10-38 Partial Controller Function Table Entries

	Arguments					Results		
Row	State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
1	stopped	false	false	—	false		CurTime(n)	
3	stopped	false	false	—	true	closing	CurTime(n)	on to close
4	stopped	false	true	—	false	stopped	…(n)	off
6	stopped	false	true	—	true	closing	CurTime(n)	on to close
7	stopped	false	true	true	false	fault		off
8	stopped	false	true	true	true	fault		off
9	stopped	true	false	—	false	opening	CurTime(n)	on to open
10	stopped	true	false	—	true	opening	CurTime(n)	on to open
12	stopped	true	true	—	false	stopped	CurTime(n)	off
13	stopped	true	true	—	true	stopped	CurTime(n)	off
15	stopped	true	true	true	false	fault		off
16	stopped	true	true	true	true	fault		off

Next, we expand the "—" entries in the column for DoorIsClosed(n). After sorting and reassigning the row numbers, we obtain Table 8.13.10-39.

INTERMEDIATE TABLE 8.13.10-39 Partial Controller Function Table Entries

	Arguments					Results		
Row	State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
1	stopped	false	false	false	false		CurTime(n)	
2	stopped	false	false	false	true	closing	CurTime(n)	on to close
3	stopped	false	false	true	false		CurTime(n)	
4	stopped	false	false	true	true	closing	CurTime(n)	on to close
5	stopped	false	true	false	false	stopped	…(n)	off
6	stopped	false	true	false	true	closing	CurTime(n)	on to close
7	stopped	false	true	true	false	stopped	…(n)	off
8	stopped	false	true	true	false	fault		off
9	stopped	false	true	true	true	closing	CurTime(n)	on to close
10	stopped	false	true	true	true	fault		off
11	stopped	true	false	false	false	opening	CurTime(n)	on to open
12	stopped	true	false	false	true	opening	CurTime(n)	on to open
13	stopped	true	false	true	false	opening	CurTime(n)	on to open
14	stopped	true	false	true	true	opening	CurTime(n)	on to open
15	stopped	true	true	false	false	stopped	CurTime(n)	off
16	stopped	true	true	false	true	stopped	CurTime(n)	off
17	stopped	true	true	true	false	stopped	CurTime(n)	off
18	stopped	true	true	true	false	fault		off
19	stopped	true	true	true	true	stopped	CurTime(n)	off
20	stopped	true	true	true	true	fault		off

In Table 8.13.10-39, row 1 is the only row with its arguments, but its results are incomplete. The door is stopped between its open and closed positions, no person is near the door and the condition for closing the door automatically is not met. The English requirements do not state what should be done in this situation. This situation would seem to be unusual, but it could conceivably be reached. For example, if the door is stopped before reversing the motor direction for a person near the door but in the meantime the person has left, what should the door controller do? In principle any of the three alternatives is feasible: leave the door stopped, open it, or close it. Here we choose to close the door, but note that this question must be raised with the client.

Row 2 is the only row with its arguments. Its results are complete. Row 2 should be retained.

Row 3 is the only row with its arguments, but its results are incomplete. The door is closed and stopped. Because no person is near the door, there is no reason to open it. This is another example of a situation in which the status quo should presumably be maintained: that is, of a presumed and implied requirement that in the absence of a reason to do otherwise, the current situation should be maintained.

Row 4 is the only row with its arguments. Its results are complete. However, the next state is inappropriate. The door is sensed as being closed, so the controller should not turn motor power on to close the door. The door should remain in the stopped state. This row resulted from the translation of the coarse condition for closing the door, in which only the closure delay was considered, and the other conditions neglected (see Table 8.13.10-6; see also the discussion about rows 5 and 6 in Table 8.13.10-21).

Row 5 is the only row with its arguments. Its results are complete. Row 5 should be retained.

Row 6 is the only row with its arguments. Its results are complete. Row 6 should be retained.

Rows 7 and 8 have the same arguments but give different results. The door is sensed as being both open and closed, indicating a fault. Rows 7 and 8 can be combined, taking results from row 8 where they are inconsistent.

Rows 9 and 10 have the same arguments but give different results. The door is sensed as being both open and closed, indicating a fault. Rows 9 and 10 can be combined, taking results from row 10 where they are inconsistent.

Row 11 is the only row with its arguments. Its results are complete. Row 11 should be retained.

Row 12 is the only row with its arguments. Its results are complete. Row 12 should be retained.

Row 13 is the only row with its arguments. Its results are complete. Row 13 should be retained.

Row 14 is the only row with its arguments. Its results are complete. Row 14 should be retained.

Row 15 is the only row with its arguments. Its results are complete. Row 15 should be retained.

Row 16 is the only row with its arguments. Its results are complete. Row 16 should be retained.

Rows 17 and 18 have the same arguments but give different results. The door is sensed as being both open and closed, indicating a fault. Rows 17 and 18 can be combined, taking results from row 18 where they are inconsistent.

Rows 19 and 20 have the same arguments but give different results. The door is sensed as being both open and closed, indicating a fault. Rows 19 and 20 can be combined, taking results from row 20 where they are inconsistent.

The result is Table 8.13.10-40.

INTERMEDIATE TABLE 8.13.10-40 Partial Controller Function Table Entries

	Arguments					Results		
Row	State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
1	stopped	false	false	false	false	closing	CurTime(n)	on to close
2	stopped	false	false	false	true	closing	CurTime(n)	on to close
3	stopped	false	false	true	false	stopped	CurTime(n)	off
4	stopped	false	false	true	true	stopped	CurTime(n)	off
5	stopped	false	true	false	false	stopped	…(n)	off
6	stopped	false	true	false	true	closing	CurTime(n)	on to close
8	stopped	false	true	true	false	fault	…(n)	off
10	stopped	false	true	true	true	fault	CurTime(n)	off
11	stopped	true	false	false	false	opening	CurTime(n)	on to open
12	stopped	true	false	false	true	opening	CurTime(n)	on to open
13	stopped	true	false	true	false	opening	CurTime(n)	on to open
14	stopped	true	false	true	true	opening	CurTime(n)	on to open
15	stopped	true	true	false	false	stopped	CurTime(n)	off
16	stopped	true	true	false	true	stopped	CurTime(n)	off
18	stopped	true	true	true	false	fault	CurTime(n)	off
20	stopped	true	true	true	true	fault	CurTime(n)	off

Table 8.13.10-40 contains 16 data rows. It contains all possible combinations of values for the arguments PersonIsNear(n), DoorIsOpen(n), DoorIsClosed(n), and ClosureDelayIsExceeded(n). It is, therefore, complete for the state “stopped.” Note that it has the same structure as the Final Tables 8.13.10-22 and 8.13.10-31 for the states “closing” and “opening,” respectively.

Table 8.13.10-13 for the state “fault” must still be expanded and completed in a similar way. The English description of the desired reactions to the various inputs is, as already identified and stated, incomplete. One fundamental decision to be made by the designers and clients is whether or not the controller should attempt to recover from a fault condition, that is, what the controller should do when in the fault state the values of the position variables DoorIsOpen(n) and DoorIsClosed(n) are consistent (not both true). One possibility is to remain in the fault state, while the other possibility is to continue in one of the other states opening, closing, or stopped. For the purposes of completing this example, we choose the latter strategy (i.e., to recover from the fault state when possible). In a real situation this may or may not be the best choice. In any event, some indication should be made to the maintenance group, indicating that a fault has occurred and the type of fault.

Table 8.13.10-13 for the state “fault” is repeated as Table 8.13.10-41. It will be expanded and completed in the same manner as the other tables above.

INTERMEDIATE TABLE 8.13.10-41 Partial Controller Function Table Entries

Arguments					Results		
State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
fault	—	—	—	true	closing	CurTime(n)	on to close
fault	—	false	—	—		CurTime(n)	
fault	—	true	true	—	fault		off
fault	false	true	—	—		…(n)	
fault	true	—	—	—		CurTime(n)	

We begin by expanding the entries with "—" in the column for the argument DoorIsOpen(n). Sorting the table and adding a column for row numbers yields Table 8.13.10-42.

INTERMEDIATE TABLE 8.13.10-42 Partial Controller Function Table Entries

	Arguments					Results		
Row	State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
1	fault	—	false	—	—		CurTime(n)	
2	fault	—	false	—	true	closing	CurTime(n)	on to close
3	fault	—	true	—	true	closing	CurTime(n)	on to close
4	fault	—	true	true	—	fault		off
5	fault	false	true	—	—		…(n)	
6	fault	true	false	—	—		CurTime(n)	
7	fault	true	true	—	—		CurTime(n)	

In Table 8.13.10-42, every row has different arguments, so no rows can be combined, changed, or deleted.

Next, we expand the "—" entries in the column for PersonIsNear(n). After sorting and reassigning the row numbers, the result is Table 8.13.10-43.

INTERMEDIATE TABLE 8.13.10-43 Partial Controller Function Table Entries

	Arguments					Results		
Row	State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
1	fault	false	false	—	—		CurTime(n)	
2	fault	false	false	—	true	closing	CurTime(n)	on to close
3	fault	false	true	—	—		…(n)	
4	fault	false	true	—	true	closing	CurTime(n)	on to close
5	fault	false	true	true	—	fault		off
6	fault	true	false	—	—		CurTime(n)	
7	fault	true	false	—	—		CurTime(n)	
8	fault	true	false	—	true	closing	CurTime(n)	on to close
9	fault	true	true	—	—		CurTime(n)	
10	fault	true	true	—	true	closing	CurTime(n)	on to close
11	fault	true	true	true	—	fault		off

Rows 6 and 7 are identical, so row 7 can be deleted.
The result is Table 8.13.10-44.

INTERMEDIATE TABLE 8.13.10-44 Partial Controller Function Table Entries

	Arguments					Results		
Row	State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
1	fault	false	false	—	—		CurTime(n)	
2	fault	false	false	—	true	closing	CurTime(n)	on to close
3	fault	false	true	—	—		…(n)	
4	fault	false	true	—	true	closing	CurTime(n)	on to close
5	fault	false	true	true	—	fault		off
6	fault	true	false	—	—		CurTime(n)	
8	fault	true	false	—	true	closing	CurTime(n)	on to close
9	fault	true	true	—	—		CurTime(n)	
10	fault	true	true	—	true	closing	CurTime(n)	on to close
11	fault	true	true	true	—	fault		off

Next, we expand the "—" entries in the column for ClosureDelayIsExceeded(n). After sorting and reassigning the row numbers, the result is Table 8.13.10-45.

INTERMEDIATE TABLE 8.13.10-45 Partial Controller Function Table Entries

	Arguments					Results		
Row	State (n)	PersonIsNear (n)	DoorIsOpen (n)	DoorIsClosed (n)	ClosureDelayIsExceeded (n)	State (n+1)	LastNotClosableTime (n+1)	MotorPower (n+1)
1	fault	false	false	—	false		CurTime(n)	
2	fault	false	false	—	true		CurTime(n)	
3	fault	false	false	—	true	closing	CurTime(n)	on to close
4	fault	false	true	—	false		…(n)	
5	fault	false	true	—	true		…(n)	
6	fault	false	true	—	true	closing	CurTime(n)	on to close
7	fault	false	true	true	false	fault		off
8	fault	false	true	true	true	fault		off
9	fault	true	false	—	false		CurTime(n)	
10	fault	true	false	—	true		CurTime(n)	
11	fault	true	false	—	true	closing	CurTime(n)	on to close
12	fault	true	true	—	false		CurTime(n)	
13	fault	true	true	—	true		CurTime(n)	
14	fault	true	true	—	true	closing	CurTime(n)	on to close
15	fault	true	true	true	false	fault		off
16	fault	true	true	true	true	fault		off

In Table 8.13.10-45, rows 2 and 3 have the same arguments and compatible results. Row 2 can be deleted.

Rows 5 and 6 have the same arguments but different results for LastNotClosableTime(n+1). In this situation, no one is near the door, the door is open, and the closure delay is exceeded. In the fault state, motor power is off, so closing the door can be initiated. (That motor power is off in the fault state can be seen from the results columns in all the tables; this is also a theorem about the behavior of the controller that can easily be proved after the entire table has been constructed.) Then, the door is no longer in a closable state, so LastNotClosableTime(n+1) should be CurTime(n). Therefore, row 6 should be retained and row 5, deleted. This is a situation in which the English requirements are inconsistent: Because the door is open and no one is near the door, the door is closable, and one requirement implies that LastNotClosableTime(n+1) should be LastNotClosableTime(n). However, the closure delay is exceeded, so another requirement implies that LastNotClosableTime(n+1) should be CurTime(n).

Rows 10 and 11 have the same arguments and compatible results. Row 10 can be deleted.

Rows 13 and 14 have the same arguments and compatible results. Row 13 can be deleted.

The result is Table 8.13.10-46.

INTERMEDIATE TABLE 8.13.10-46 Partial Controller Function Table Entries

	Arguments					Results		
Row	State (n)	PersonIsNear (n)	DoorIsOpen (n)	DoorIsClosed (n)	ClosureDelayIsExceeded (n)	State (n+1)	LastNotClosableTime (n+1)	MotorPower (n+1)
1	fault	false	false	—	false		CurTime(n)	
3	fault	false	false	—	true	closing	CurTime(n)	on to close
4	fault	false	true	—	false		…(n)	
6	fault	false	true	—	true	closing	CurTime(n)	on to close
7	fault	false	true	true	false	fault		off
8	fault	false	true	true	true	fault		off
9	fault	true	false	—	false		CurTime(n)	
11	fault	true	false	—	true	closing	CurTime(n)	on to close
12	fault	true	true	—	false		CurTime(n)	
14	fault	true	true	—	true	closing	CurTime(n)	on to close
15	fault	true	true	true	false	fault		off
16	fault	true	true	true	true	fault		off

Finally, we expand the “—” entries in the column for DoorIsClosed(n). After sorting and reassigning the row numbers, we obtain Table 8.13.10-47.

INTERMEDIATE TABLE 8.13.10-47 Partial Controller Function Table Entries

	Arguments					Results		
Row	State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
1	fault	false	false	false	false		CurTime(n)	
2	fault	false	false	false	true	closing	CurTime(n)	on to close
3	fault	false	false	true	false		CurTime(n)	
4	fault	false	false	true	true	closing	CurTime(n)	on to close
5	fault	false	true	false	false		…(n)	
6	fault	false	true	false	true	closing	CurTime(n)	on to close
7	fault	false	true	true	false		…(n)	
8	fault	false	true	true	false	fault		off
9	fault	false	true	true	true	closing	CurTime(n)	on to close
10	fault	false	true	true	true	fault		off
11	fault	true	false	false	false		CurTime(n)	
12	fault	true	false	false	true	closing	CurTime(n)	on to close
13	fault	true	false	true	false		CurTime(n)	
14	fault	true	false	true	true	closing	CurTime(n)	on to close
15	fault	true	true	false	false		CurTime(n)	
16	fault	true	true	false	true	closing	CurTime(n)	on to close
17	fault	true	true	true	false		CurTime(n)	
18	fault	true	true	true	false	fault		off
19	fault	true	true	true	true	closing	CurTime(n)	on to close
20	fault	true	true	true	true	fault		off

Rows 7 and 8 have the same arguments and compatible results. They can be combined.

Rows 9 and 10 have the same arguments but different results. The door is sensed as being both open and closed, so State(n+1) should be fault. CurTime(n) can be taken as the value of LastNotClosableTime(n+1).

Rows 17 and 18 have the same arguments and compatible results. They can be combined.

Rows 19 and 20 have the same arguments but different results. The door is sensed as being both open and closed, so State(n+1) should be fault. CurTime(n) can be taken as the value of LastNotClosableTime(n+1).

The result is Table 8.13.10-48.

INTERMEDIATE TABLE 8.13.10-48 Partial Controller Function Table Entries

	Arguments					Results		
Row	State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
1	fault	false	false	false	false		CurTime(n)	
2	fault	false	false	false	true	closing	CurTime(n)	on to close
3	fault	false	false	true	false		CurTime(n)	
4	fault	false	false	true	true	closing	CurTime(n)	on to close
5	fault	false	true	false	false		…(n)	
6	fault	false	true	false	true	closing	CurTime(n)	on to close
8	fault	false	true	true	false	fault	…(n)	off
10	fault	false	true	true	true	fault	CurTime(n)	off
11	fault	true	false	false	false		CurTime(n)	
12	fault	true	false	false	true	closing	CurTime(n)	on to close
13	fault	true	false	true	false		CurTime(n)	
14	fault	true	false	true	true	closing	CurTime(n)	on to close
15	fault	true	true	false	false		CurTime(n)	
16	fault	true	true	false	true	closing	CurTime(n)	on to close
18	fault	true	true	true	false	fault	CurTime(n)	off
20	fault	true	true	true	true	fault	CurTime(n)	off

Table 8.13.10-48 contains 16 data rows, one for every possible combination of values for the arguments PersonIsNear(n), DoorIsOpen(n), DoorIsClosed(n), and ClosureDelayIsExceeded(n). It is, therefore, structurally complete. Note that it has the same structure as the final tables for the states "opening," "closing," and "stopped" (see above).

Several results cells are empty in Table 8.13.10-48 and they must be filled. Also, all but two values of LastNotClosableTime(n+1) are CurTime(n); the other two (rows 5 and 8) should be examined for correctness. These apparent discrepancies are at least partially due to the fact that the English requirements did not explicitly consider the fault state.

In row 1, the results are incomplete. Recovery from the fault state is possible, because the door is sensed as neither open nor closed. The closure delay is not yet exceeded, so there is no reason to close the door. No one is near the door, so there is no reason to open the door. The stopped state with motor power off appears to be the most appropriate decision.

In row 3, the results are incomplete. Recovery from the fault state is possible, because the door is sensed as closed and not open. No one is near the door, so there is no reason to open the door. The stopped state with motor power off appears to be the most appropriate decision.

In row 11, the results are incomplete. Recovery from the fault state is possible, because the door is sensed as neither open nor closed. Someone is near the door, so the door should be opened.

In row 13, the results are incomplete. Recovery from the fault state is possible, because the door is sensed as closed and not open. Someone is near the door, so the door should be opened.

In row 15, the results are incomplete. Recovery from the fault state is possible, because the door is sensed as open and not closed. Someone is near the door, so the door should be left open. The next state, State(n+1), should be stopped and MotorPower(n+1) should be off.

Regarding closing the door automatically, a question arises as to whether the fault state can represent a "closable" condition. The English statement of the requirements says nothing about this.

In row 5, recovery from the fault state is possible, because the door is sensed as open and not closed. The closure delay is not yet exceeded, so the door should remain open (i.e., in the stopped state with motor power off). The time to automatic door closure can continue uninterrupted; that is, LastNotClosableTime(n+1) can be LastNotClosableTime(n).

In row 8, recovery from the fault state is not possible. It is not clear whether or not the countdown to automatic door closure should continue. It is not unreasonable to allow it to do so; that is, that the value of LastNotClosableTime(n+1) can be LastNotClosableTime(n).

The final table for the fault state is, then, Table 8.13.10-49.

INTERMEDIATE TABLE 8.13.10-49 Partial Controller Function Table Entries

	Arguments					Results		
Row	State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
1	fault	false	false	false	false	stopped	CurTime(n)	off
2	fault	false	false	false	true	closing	CurTime(n)	on to close
3	fault	false	false	true	false	stopped	CurTime(n)	off
4	fault	false	false	true	true	closing	CurTime(n)	on to close
5	fault	false	true	false	false	stopped	…(n)	off
6	fault	false	true	false	true	closing	CurTime(n)	on to close
8	fault	false	true	true	false	fault	…(n)	off
10	fault	false	true	true	true	fault	CurTime(n)	off
11	fault	true	false	false	false	opening	CurTime(n)	on to open
12	fault	true	false	false	true	closing	CurTime(n)	on to close
13	fault	true	false	true	false	opening	CurTime(n)	on to open
14	fault	true	false	true	true	closing	CurTime(n)	on to close
15	fault	true	true	false	false	stopped	CurTime(n)	off
16	fault	true	true	false	true	closing	CurTime(n)	on to close
18	fault	true	true	true	false	fault	CurTime(n)	off
20	fault	true	true	true	true	fault	CurTime(n)	off

Table 8.13.10-49 is the complete table for the fault state, permitting recovery when the door sensors return compatible signals.

8.13.11 The Complete Controller Function Table

The following tables are the final Intermediate Tables 8.13.10-31, 8.13.10-22, 8.13.10-40, and 8.13.10-49 for the four states opening, closing, stopped, and fault, respectively, as constructed in Section 8.13.10. The rows have been renumbered in the tables below. These four tables actually form one logical table, but they have been left in four parts for ease of reading. Together, they define the values of the functions State(n+1), LastNotClosableTime(n+1), and MotorPower(n+1) for all combinations of values of the arguments State(n), PersonIsNear(n), DoorIsOpen(n), DoorIsClosed(n), and ClosureDelayIsExceeded(n).

Table 8.13.11-1 is the final table for the opening state.

TABLE 8.13.11-1 Shopping Mall Door Controller Functions for the Opening State

	Arguments					Results		
Row	State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
1	opening	false	false	false	false	opening	CurTime(n)	on to open
2	opening	false	false	false	true	opening	CurTime(n)	on to open
3	opening	false	false	true	false	opening	CurTime(n)	on to open
4	opening	false	false	true	true	opening	CurTime(n)	on to open
5	opening	false	true	false	false	stopped	…(n)	off
6	opening	false	true	false	true	stopped	…(n)	off
7	opening	false	true	true	false	fault	…(n)	off
8	opening	false	true	true	true	fault	…(n)	off
9	opening	true	false	false	false	opening	CurTime(n)	on to open
10	opening	true	false	false	true	opening	CurTime(n)	on to open
11	opening	true	false	true	false	opening	CurTime(n)	on to open
12	opening	true	false	true	true	opening	CurTime(n)	on to open
13	opening	true	true	false	false	stopped	CurTime(n)	off
14	opening	true	true	false	true	stopped	CurTime(n)	off
15	opening	true	true	true	false	fault	CurTime(n)	off
16	opening	true	true	true	true	fault	CurTime(n)	off

Table 8.13.11-2 is the final table for the closing state.

TABLE 8.13.11-2 Shopping Mall Door Controller Functions for the Closing State

	Arguments					Results		
Row	State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
17	closing	false	false	false	false	closing	CurTime(n)	on to close
18	closing	false	false	false	true	closing	CurTime(n)	on to close
19	closing	false	false	true	false	stopped	CurTime(n)	off
20	closing	false	false	true	true	stopped	CurTime(n)	off
21	closing	false	true	false	false	closing	…(n)	on to close
22	closing	false	true	false	true	closing	CurTime(n)	on to close
23	closing	false	true	true	false	fault	…(n)	off
24	closing	false	true	true	true	fault	…(n)	off
25	closing	true	false	false	false	stopped	CurTime(n)	off
26	closing	true	false	false	true	stopped	CurTime(n)	off
27	closing	true	false	true	false	stopped	CurTime(n)	off
28	closing	true	false	true	true	stopped	CurTime(n)	off
29	closing	true	true	false	false	stopped	CurTime(n)	off
30	closing	true	true	false	true	stopped	CurTime(n)	off
31	closing	true	true	true	false	fault	CurTime(n)	off
32	closing	true	true	true	true	fault	CurTime(n)	off

Table 8.13.11-3 is the final table for the stopped state.

TABLE 8.13.11-3 Shopping Mall Door Controller Functions for the Stopped State

	Arguments					Results		
Row	State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
33	stopped	false	false	false	false	closing	CurTime(n)	on to close
34	stopped	false	false	false	true	closing	CurTime(n)	on to close
35	stopped	false	false	true	false	stopped	CurTime(n)	off
36	stopped	false	false	true	true	stopped	CurTime(n)	off
37	stopped	false	true	false	false	stopped	…(n)	off
38	stopped	false	true	false	true	closing	CurTime(n)	on to close
39	stopped	false	true	true	false	fault	…(n)	off
40	stopped	false	true	true	true	fault	CurTime(n)	off
41	stopped	true	false	false	false	opening	CurTime(n)	on to open
42	stopped	true	false	false	true	opening	CurTime(n)	on to open
43	stopped	true	false	true	false	opening	CurTime(n)	on to open
44	stopped	true	false	true	true	opening	CurTime(n)	on to open
45	stopped	true	true	false	false	stopped	CurTime(n)	off
46	stopped	true	true	false	true	stopped	CurTime(n)	off
47	stopped	true	true	true	false	fault	CurTime(n)	off
48	stopped	true	true	true	true	fault	CurTime(n)	off

Table 8.13.11-4 is the final table for the fault state.

TABLE 8.13.11-4 Shopping Mall Door Controller Functions for the Fault State

	Arguments					Results		
Row	State (n)	Person IsNear (n)	Door IsOpen (n)	Door IsClosed (n)	Closure DelayIs Exceeded (n)	State (n+1)	LastNot Closable Time (n+1)	Motor Power (n+1)
49	fault	false	false	false	false	stopped	CurTime(n)	off
50	fault	false	false	false	true	closing	CurTime(n)	on to close
51	fault	false	false	true	false	stopped	CurTime(n)	off
52	fault	false	false	true	true	closing	CurTime(n)	on to close
53	fault	false	true	false	false	stopped	…(n)	off
54	fault	false	true	false	true	closing	CurTime(n)	on to close
55	fault	false	true	true	false	fault	…(n)	off
56	fault	false	true	true	true	fault	CurTime(n)	off
57	fault	true	false	false	false	opening	CurTime(n)	on to open
58	fault	true	false	false	true	closing	CurTime(n)	on to close
59	fault	true	false	true	false	opening	CurTime(n)	on to open
60	fault	true	false	true	true	closing	CurTime(n)	on to close
61	fault	true	true	false	false	stopped	CurTime(n)	off
62	fault	true	true	false	true	closing	CurTime(n)	on to close
63	fault	true	true	true	false	fault	CurTime(n)	off
64	fault	true	true	true	true	fault	CurTime(n)	off

The function ClosureDelayIsExceeded(n) was defined in Section 8.13.10 as

$$(\mathrm{CurTime}(n) - \mathrm{LastNotClosableTime}(n)) > \mathrm{ClosureDelay} \qquad [8.13.11\text{-}1]$$

for all integers n≥0.

Tables 8.13.11-1 through 8.13.11-4 define iteratively the values of the result variables at time point n+1 in terms of the arguments at time point n. Expressed more mathematically, they define the n+1st terms in the result sequences as functions of the nth terms in the argument sequences, for all integers n≥0. The 0th terms in the result sequences (State(0), LastNotClosableTime(0), and MotorPower(0)) were defined near the end of Section 8.13.8 as

$$\mathrm{State}(0)=\mathrm{stopped} \wedge \mathrm{LastNotClosableTime}(0)=\mathrm{CurTime}(0) \\ \wedge \mathrm{MotorPower}(0)=\mathrm{off} \qquad [8.13.11\text{-}2]$$

thus completing the definitions of all terms in the result sequences.

In an actual practical application the tables above would be reviewed, row by row, and each row would be compared with each item listed in the original English specification in Section 8.13.4. Discrepancies, ambiguities, and gaps discovered would lead to corresponding analyses, reconsideration, and where appropriate, correction. Such a review is left as an exercise for the reader. (*Hint:* There is at least one inconsistency or error to be found in Tables 8.13.11-1 through 8.13.11-4. Find all of them). Where the English specification is inconsistent or ambiguous in certain cases, apply the priorities and conventions adopted in Section 8.13.10. Investigate the source of each inconsistency or error found.

At one point in the construction of the fault table, it was stated that in the fault state, motor power is always off. This can be stated as a theorem, for example in the form

$$[\wedge\ k : k\in\mathbb{Z} \wedge k\geqslant 0 : (\text{State}(k)=\text{fault}) \Rightarrow (\text{MotorPower}(k)=\text{off})] \qquad [8.13.11\text{-}3]$$

Initially, MotorPower(0)=off, so the theorem is trivially true for k=0. By examining every data row in the four tables defining the controller function, one can see that in every row in which State(n+1)=fault, MotorPower(n+1)=off; that is, for every $k>0$, (State(k)=fault) $\Rightarrow$ (MotorPower(k)=off), verifying the theorem.

In the same way it can be shown that, for all integer $k\geq 0$:

- Whenever State(k)=stopped, MotorPower(k)=off.
- Whenever State(k)=opening, MotorPower(k)=on to open.
- Whenever State(k)=closing, MotorPower(k)=on to close.

Thus, MotorPower(k) is a function of State(k). One could have eliminated the column for MotorPower in the tables and defined MotorPower(k) as this function of State(k). Experience with this type of problem could have given the foresight needed to make such simplifications at the beginning. Lacking the necessary experience and insight, one must go through the detailed work.

In a similar way, other theorems of interest about the behavior of the shopping mall door controller can be formulated and proved. Some of these theorems express certain aspects of the English specifications, for example, that if the door is open at time step k, the state at time step k+1 will not be opening, and correspondingly for a closed door.

Notice the different orientation of the English statements and the corresponding model in the Language of Mathematics. The English requirements are oriented to changes in the movement of the door: when it should stop, when it should start to open, when it should start to close and so on. The expressions in the mathematical model, on the other hand, are oriented to static states. Although those states can and sometimes do differ from one term in the sequences to the next, they also sometimes stay unchanged, and those steps in the sequences in which the states do not change are often not mentioned in the English requirement. Unless changes are stated explicitly in the English requirement, no change is usually meant implicitly. This gave rise to gaps in some of the table entries, gaps that we had to fill in later.

Despite the fact that the requirements were formulated carefully in the English statements of the requirements for the door controller, both ambiguities and gaps were discovered in the process of constructing the tables in the mathematical model. These ambiguities and gaps, including

- implied, “obvious” but missing statements, requirements, facts, and so on
- the need to apply priorities to the English requirements that could, in particular circumstances, contradict each other

were discovered and resolved during the construction of the mathematical model, in this case in tabular notation. Some of the English requirements overlapped, but the overlaps were not clearly or immediately obvious, and some of the overlaps contradicted themselves. Statements needed to establish priorities to resolve the conflicts were missing. In a practical situation, all these ambiguities, gaps, and so on, would be discussed with the clients and their approval or modifications obtained.

Like all mathematical functions, the functions defined in the tables above are static objects. In an actual application (e.g., involving a computer to determine the next state), the arguments may change while the computer is searching the table for the argument values. This can lead to erroneous results. The arguments being searched for must remain unchanged while the computer is evaluating the function for those arguments. There are various ways of ensuring this, but attention must be consciously paid to this aspect of implementing the design with a computer. This is an example of the possible effects of the differences between the static nature of the mathematical solution (deriving from the static nature of the Language of Mathematics) and the dynamic nature of a computer process and of many computer programming languages.

The temporal history of the door controller system is reflected in the static sequences representing the state variables, input signals, and output signals. The terms of the static sequences in the mathematical domain represent the time steps in the application domain. These sequences can be recorded in a log of the actual dynamic behavior of the system as a whole. Such information can be used for analyzing aspects of the demands placed on the door controller (e.g., patterns of arrival of people at the door), failures, and other characteristics of the actual operation of the door controller and its environment.

This example of a shopping mall door controller began with a specification of the desired behavior of the controller written in English. In a series of steps, this English specification was translated into a mathematical model in the Language of Mathematics in which the several controller functions were specified mathematically: in this case, in the form of tables. In the process of translating from English to the Language of Mathematics and then completing all entries in the tables, a number of ambiguities, gaps in the information provided, and conflicts in the information provided were discovered. All these are typical of practical applications. In an actual design project, these discrepancies would be resolved in discussions with all parties involved, including safety experts. These discussions and negotiations typically represent a significant part of the time and effort required to complete the design of such a system. They constitute a normal part of the process of defining the requirements that the system to be designed and implemented must satisfy during its later operation.

PART D
Conclusion

9 Summary

9.1 TRANSFORMING ENGLISH TO MATHEMATICS: A LANGUAGE—NOT A MATHEMATICAL—PROBLEM

Mathematical notation can and should be viewed as a language. Transforming an English text describing a problem and the requirements of its solution into mathematical expressions can be beneficially viewed as a translation problem, a language problem. Only the derivation of the solution from the mathematical expressions is a mathematical problem.

Translating from English to the Language of Mathematics is, however, typically considered to be a mathematical problem. This complicates unnecessarily the application of mathematics in practice by hiding many important characteristics of the translation process and clues for doing it in practice. Applying mathematics in practice is significantly easier if the first translation step is viewed as the language problem that it is.

This book's linguistic view of mathematical notation – the Language of Mathematics – adds a new and different dimension to our insight into mathematics. It has strong implications for the following:

- How to apply mathematics in practice
- How to teach mathematics
- How to teach how to apply mathematics at all levels

It can help those who have difficulty applying mathematics effectively. It can make mathematics more accessible to people who have until now eschewed mathematics, having been unable to relate it to the world in which they think, work, and live. It can make mathematics accessible to those who otherwise would be—or have been—turned off by it. Even mathematicians and persons with considerable successful experience in applying mathematics will find that it makes their work easier. It can enable nonmathematically oriented people to work effectively in teams applying mathematics to practical problems.

Presenting word problems as language problems as early as in primary schools will draw students' conscious attention to key issues involved in solving the problems and make learning this material easier. It will give them a broader and deeper basic understanding of mathematics, link mathematics to their previous knowledge of language, and provide them with a better foundation upon which specific skills in applying mathematics can be developed later. Instead of learning mathematics as something different, new, and unrelated to their previous experience and knowledge, they will learn mathematics as an extension of their already accumulated experience with and knowledge of language.

9.2 ADVANTAGES OF THE LANGUAGE OF MATHEMATICS FOR REASONING AND ANALYZING

One of the major advantages of translating an English text into the Language of Mathematics is that the resulting mathematical expressions can be transformed mechanistically, without changing their meanings, in order to analyze the problem and to derive solutions. These mechanistic transformations must satisfy mathematical rules only; the application domain need not be considered during this phase of solving the problem. The mathematical model serves as an interface connecting the application and the mathematical manipulations and, at the same time, making each independent of the other.

Another important advantage of translating an English text into the Language of Mathematics is that ambiguity is absent from the mathematical formulation. Ambiguity in the English text must be identified and resolved in the process of constructing the mathematical model. This eliminates misunderstanding that would otherwise arise from ambiguity.

Because of the advantages noted above, the Language of Mathematics is the only suitable language for solving a logically intricate problem. Trying to solve such a problem using English or any other natural language is prone to errors and omissions and, in the end, is more difficult. Such an attempt will be inefficient and usually ineffective.

The insight into many things gained through mathematics and its language surprises and astonishes again and again.

9.3 COMPARISON OF KEY CHARACTERISTICS OF ENGLISH AND THE LANGUAGE OF MATHEMATICS

The Language of Mathematics is a very limited and simple language. Its only elements are values, variables, and various combinations and structures thereof. Expressions and functions are among the common combinations. Common structures are sets and sequences. Although the Language of Mathematics is a simple language, it can be adapted to a great variety of application areas. What one can do with it can be as complex as needed, but the language itself is simple.

The concept of time is absent from the Language of Mathematics itself. Time in the application domain can be modeled mathematically. For modeling discrete time steps, the sequence or a similar structure is typically used. For modeling continuous time, a variable representing time is introduced. In any case, however, the concept of time remains exclusively in the application domain.

The concept of cause and effect is absent from the Language of Mathematics. The Language of Mathematics expresses relations, but a causality direction is absent. Where needed and appropriate in the application domain, cause-and-effect relationships must be introduced in the interpretation of the components of the mathematical model.

Syntactically, a natural language such as English can be viewed as a collection (set) of sentences constructed according to certain rules. The meaning of each sentence derives from the meanings of the individual words in the sentence related to one another in ways corresponding to the syntactical structure of the sentence. Many different meanings can be expressed by the available words, so that a very great variety of meanings can be expressed by the sentences in the language. Any single sentence can almost always be interpreted in different ways (i.e., to have any one of several different meanings).

Correspondingly, the Language of Mathematics can be viewed as a collection (set) of expressions constructed according to certain rules. Each expression can be written in any of several different notational forms. Each expression in the Language of Mathematics is a combination of one or more values, variables, functions, or other expressions. The meanings directly associated with these building blocks are very limited; in fact, the meaning is always a value. In particular, any given expression can be interpreted to have only one meaning: its value. That is, every expression is unambiguous within the Language of Mathematics. Any interpretation in the application world associated with a particular value, variable, function, or expression must be defined explicitly outside the Language of Mathematics itself.

English and the Language of Mathematics complement each other in ways that are particularly important for practical applications. The Language of Mathematics is unambiguous, but it is capable of expressing only a rather limited range of meanings directly. On the other hand, English is ambiguous, but it is capable of expressing a much wider range of meanings. When richness of expression is desired, a natural language such as English is the language of choice. When ambiguity must be avoided, such as when reasoning logically, the Language of Mathematics is clearly the language of choice. The Language of Mathematics enables one to reason logically by manipulating mathematical expressions mechanistically. This, in turn, makes it feasible to solve much more complex problems than can be solved employing arguments formulated only in natural language.

For reasoning, analyzing things, and solving problems, the primary strengths of the Language of Mathematics (its unambiguity and precision) offset the weaknesses of English (its ambiguity and vagueness). A comparative strength of English (its vast universe of discourse) can offset the corresponding weakness of the Language of Mathematics (its very narrow and limited universe of discourse).

Among the similarities between English and the Language of Mathematics are the following:

- Each uses a set of defined symbols as the basis for composing sentences and expressions.
- Each has well-defined rules of syntax (grammar) for combining symbols into meaningful (acceptable, correct) sentences and expressions.

- Each is a medium for expressing and communicating, verbally or visually, facts, opinions, thoughts, ideas, and so on, between different people at one time or between one or more people at different times.
- Each is a medium for thinking, reasoning, and analyzing.

Among the differences between English and the Language of Mathematics are the following:

- Their universes of discourse are very different. The universe of discourse of the Language of Mathematics consists of values, variables, functions, and expressions built up of values, variables, and functions; it is very limited. The universe of discourse of English encompasses much, much more, including most concrete and abstract aspects of the human environment and experience; it is comparatively unlimited.
- Only very simple statements in English can be formulated unambiguously; almost every statement in English is ambiguous in some way. Every expression in the Language of Mathematics is unambiguous. Therefore, expressions in the Language of Mathematics are suitable for rational logical reasoning, while statements in English are not. For reasoning, English statements can be employed effectively only for analyses of limited complexity and length.
- Correspondingly, vague or imprecise statements cannot be formulated in the Language of Mathematics. They can be formulated in English. Statements about uncertainty in the application domain can be formulated in the Language of Mathematics, but they are still unambiguous, precise statements.
- The Language of Mathematics does not include the concept of time, whereas English does. Time can be modeled mathematically, but the concept of time is in the external interpretation of a mathematical model, not within the realm of mathematics itself.

9.4 TRANSLATING FROM ENGLISH TO THE LANGUAGE OF MATHEMATICS: INTERPRETATION

Because of its limited universe of discourse, the Language of Mathematics is, for practical applications, only a *template language*. This language must be adapted to each specific application area or problem. This is achieved by an *interpretation*. The interpretation assigns meanings in the application area to the variables, values, and various combinations and structures thereof appearing in the mathematical expressions representing the original English statements. The interpretation must be written in English, supplemented appropriately by application terminology; it cannot be written in the Language of Mathematics. The interpretation is not part of the mathematical model, which, mathematically, stands on its own and is independent of the interpretation and the application. The interpretation serves as a link, a bridge, between the mathematical model and the application problem.

When translating from English to the Language of Mathematics, bridging the gap between their two very different universes of discourse represents the main problem. In principle, a different "bridge" must be constructed for each type of problem to which mathematics is to be applied. That bridge is the interpretation of the components (values, variables, functions, and expressions) of the mathematical model to the various elements of the application domain. Thus, for any particular application, the Language of Mathematics is an incomplete language, a template language, which must be adapted and extended to the particular application domain in question.

Formulating an appropriate interpretation is a step generally required when translating from English to the Language of Mathematics. Such a particular interpretation is not normally required when translating from English to some other natural language. This additional task required when translating English to the Language of Mathematics adds to the complexity of translating. It changes the character of translating somewhat. The translator must design an extension of the Language of Mathematics in order to satisfy the needs of the application.

In practice, the interpretation is too often formulated casually and superficially or even not documented at all. This usually leads to confusion, misunderstanding, and errors.

The interpretation is important and necessary from the application point of view, but from a purely mathematical standpoint, the interpretation is unnecessary. The expressions in the mathematical model can be transformed and manipulated without regard to the interpretation, the application problem, and its solution. Only the rules of mathematics restrict how the mathematical expressions may be transformed. The world of the mathematical model is independent of any interpretation of it in an application area; in fact, it is independent of the very existence of such an interpretation. This gives rise to another advantage of solving problems with mathematics: Whenever different applications lead to the same mathematical model, any solution found for the mathematical model is a solution for all such applications. Many such examples exist in the real world; for example, radioactive decay, certain electrical circuits, particular mechanical systems, and certain business processes are represented by the same differential equations and, hence, have the same solution. Similarly, many random processes or parts thereof in different application areas share the same probabilistic models and, therefore, the same characteristics and solutions.

9.5 TRANSLATING FROM ENGLISH TO THE LANGUAGE OF MATHEMATICS: APPROACH AND STRATEGY

To translate an English text into the Language of Mathematics, begin by understanding the intended meaning of the text. Next, distinguish between statements expressing background information not needed in the mathematical model and statements to be translated into mathematical expressions in the model. In the latter, nouns and noun phrases are candidates for non-Boolean variables and functions in the model. Clauses with verbs of state or being are candidates for Boolean variables. Clauses with action verbs should be reformulated to statements about the results of the actions or to statements expressing the actions as nouns or adjectives (e.g., participles). Statements should, where appropriate, be reformulated into sentences expressing relations between the values of the variables and functions already identified for inclusion in the mathematical model. The result will be sentences expressing only things in the universe of discourse of the Language of Mathematics. In particular, action verbs will have been eliminated.

Variables, functions, and expressions with Boolean values (“true” or “false”) correspond to sentences and clauses in English. Variables, functions, and expressions with other values generally correspond to nouns, noun phrases, or adjectives, often as predicate adjectives. The Boolean functions “and,” “or,” “implies,” and so on, correspond to conjunctions combining clauses.

Values of non-Boolean variables and functions are instances of the generic descriptions in the interpretation of the variables or functions in question. Such values are usually nouns or adjectives, often in the form of present or past participles or predicate adjectives, or phrases thereof.

All verbs in the Language of Mathematics are implicit and stative (i.e., are timeless verbs of state or being). There are no action verbs in the Language of Mathematics. Therefore, when translating a sentence in English containing an action verb, one will find it helpful to reformulate the English sentence so that only a verb of state remains. To the extent that the sense of action is to be retained, express it with an adjective (e.g., a present or past participle) in a noun phrase or with a predicate adjective with a stative verb.

Following the guidelines in this book, grammatical aspects of the English text and their relationships with linguistic aspects of the Language of Mathematics are used to identify the variables, values, functions, and expressions in the mathematical model. They are also used to formulate the interpretation of the mathematical model—the bridge between the mathematical model and the application domain. In this way, one can translate an English text into a mathematical model relying primarily on rational, linguistic considerations and only minimally on intuition.

APPENDIX A
Representing Numbers

In every system for representing numbers, a number is represented by a sequence of symbols. Each system is characterized by (1) the set of symbols used and (2) whether or not the value of any particular symbol depends on its position in the sequence. Most number systems used in human societies, even the earliest known, are positional systems; that is, the value of a symbol depends on its position in the sequence. The Roman number system, the ancient Egyptian number system, and an ancient Greek number system are probably the best known nonpositional systems.

Positional Decimal System Our common system for representing numbers uses the 10 symbols 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. It is a positional system in which the rightmost symbol represents its basic value. The symbol second from the right represents the value 10 times the symbol's basic value; the symbol third from the right represents the value 100 times the symbol's basic value; and so on. The nth symbol from the right represents the value $10\uparrow(n-1)$ times the symbol's basic value. The value represented by a sequence of symbols is the sum of the values represented by the symbols (i.e., the sequence 2361 represents two thousand three hundred and sixty-one). The sequence 4502 means four thousand plus five hundred plus no tens plus two, or four thousand five hundred and two. For ease of reading, a comma is often inserted every third symbol (i.e., 2361 is sometimes written 2,361), but such commas do not affect the value represented by the sequence of symbols.

The positional value scheme is continued to the right into the fractional part of the number. The integral and fractional parts are separated by a period (.). The first position after the separator represents tenths; the second position, hundredths; and so on.

The convention for the thousands separator and the separator between the integral and fractional parts of the number varies from country to country. The convention in the United States and some other countries is, as described in the paragraphs above, the comma (,) for the thousands separator and the period (.) for the separator between the integral and fractional parts. In other countries, their roles are reversed. Thus, two thousand three hundred and sixty-one and four tenths would be written as 2,361.4 in the United States and 2.361,4 in Germany. In still other countries, an apostrophe

(’) as the thousands separator was until recently the convention and is sometimes still seen. A blank space is also sometimes used as the thousands separator.

Positional Number Systems in General The positional value of a symbol need not be a power of 10 as described above. The positional value can be a power of any integer greater than or equal to 2. This integer is called the *radix* of the positional number system in question. The most common number system in current human use has the radix 10. Within computers, the radix 2 system is the most common. Earlier societies used systems with radices 10, 20, and 60. Remnants of the radix 20 system can still be seen in some natural languages, for example in the French word quatre-vingts (four twenties) for 80. Different kinds of remnants of a radix 20 system can be found in other languages, including several European languages. Currently common units of measure for time and angles are remnants of the radix 60 system (60 seconds in a minute, 60 minutes in a degree and in an hour).

In positional number systems in general, the value represented by the sequence

$$s_n\, s_{n-1} \ldots s_1\, s_0 \,.\, s_{-1} \ldots s_{-(m-1)}\, s_{-m}$$

of symbols and the separator (.) between the integral and fractional parts of the number is

$$\sum_{i=-m}^{n} \text{value}(s_i) * R^i$$

where R is the radix of the number system and the basic value of each symbol s_i is value(s_i), that is, the function “value” maps the set of symbols onto the set of integers {0, 1, …, R−1}. There are R different symbols, including the symbol for zero. From this definition it can be proved that for any given radix, the sequence of symbols representing any given number is unique.

Earlier, the Mayas in Central America used a positional number system based on the radix 20. The radix 60 system was used in ancient Mesopotamia. In both of these systems, the “symbol” in each position was composed of repetitions of two other symbols, representing 1 and 10 in the Mesopotamian number system and 1 and 5 in the Mayan number system (and sometimes a third symbol for zero).

For purposes of calculating, the radix 60 system has the advantage that any number can be divided by 2, 3, 5, or any product of these integers and the result can be represented exactly in the system. This is not the case with systems with radix 10 or 20, in which division by 3 (or any number divisible by 3) can result in an unending representation. A radix 30 system would offer the same advantage as the radix 60 system, but the author is not aware of any society ever having used a radix 30 number system. The disadvantage of the radix 60 system is the large number of basic symbols required. Currently, human society has, in effect, decided that the base 10 system is the best compromise.

The most commonly encountered positional number systems and their English names are shown in Table A-1. The value of each radix is expressed as a decimal number.

TABLE A-1 Radixes and Names of Selected Positional Number Systems

Radix	English Name
2	binary
8	octal
10	decimal
12	duodecimal
16	hexadecimal
20	vigesimal
60	sexagesimal

Nonpositional Roman Number System This is the most widely known nonpositional system, but even in it, the relative positions of two symbols can affect the value of a symbol to a limited extent. The symbols in the Roman number system and their values in our decimal system are I (1), V (5), X (10), L (50), C (100), D (500), and M (1000). A bar over a symbol represents 1000 times the value of the symbol.

A number is written as a sequence of these symbols, repeated as necessary. If a smaller symbol is followed immediately by a larger one, the value of the smaller symbol is the negative of its normal value. Only certain pairs of a smaller symbol followed by a larger one are permitted (e.g., I may precede V or X but not L or a higher-valued symbol; V may not precede any larger symbol). Otherwise, symbols with larger values are written before those with smaller values. For example:

1 is represented by I.

3 is represented by III.

4 is represented by IV. On clocks, 4 was often represented by IIII.

5 is represented by V.

8 is represented by VIII.

9 is represented by IX.

80 is represented by LXXX.

90 is represented by XC.

2399 is represented by MMCCCXCIX (not MMCCCIC).

Adding Roman numbers is fairly straightforward, but it can become tedious. One must distinguish between negative and positive values of individual symbols and accumulate unnecessarily long sequences of each symbol into symbols with higher values. Multiplication and division are more complicated.

Note that the Roman number system does not have an inherently unique representation for every number; see the numbers 4 and 2399 above. Convention regulated which representations were to be used and which not.

Using the symbols above in the way described and illustrated above, the numbers that can be represented are limited, even employing bars over symbols. Further extension of the set of numbers to be represented requires additional symbols—without limit—or allowing unlimited sequences of the same symbol. Both approaches are impractical. This comment applies to nonpositional number systems in general, not just the Roman number system. Probably for this reason, nonpositional number systems have not stood the test of time and are no longer in widespread use.

Nonpositional Ancient Egyptian Number System In this system, a different symbol is used for each of the values 1, 10, 100, 1000, 10000, 100000, and 1000000. A number was written by repeating each symbol up to nine times as required, writing the higher-valued symbols first and the lower-valued symbols last. A fractional part was represented as the sum of unit fractions (fractions having the numerator 1).

Mixed Positional Number Systems Mixed systems are still in use for specific areas. A common one is the combination of decimal and sexagesimal systems for measuring angles and time. The minutes and seconds of each are counted sexagesimally. Each of the positions hours/degrees, minutes, and seconds are, internally, written today with decimal numbers. Until the early 1970s a mixed system with radices 10, 12, and 20 was used in the British monetary system. In it, there were 12 pence in a shilling and 20 shillings in a pound, so that the sequence 1/2/6 represented one pound, two shillings, and six pence, or two hundred and seventy pence (270 pence in pure decimal representation). Within each field, the decimal number system was used; for example, 425/19/11 represented four hundred and twenty-five pounds, nineteen shillings, and eleven pence.

Scientific Notation All of the number systems described above are intended to express a number exactly. In many scientific, engineering, and technical applications, it is not necessary to express a number exactly; limited accuracy (e.g., several decimal digits) is satisfactory for practical purposes. In the case of many physical constants, the exact value has never been determined. For such numbers, especially large ones, scientific notation is often used. The number is expressed as the product of a number (usually with a fractional part) and a power of 10. For example, $2.556{*}10^7$ is 25,560,000. It also represents, for example, 25,563,489 with four decimal digits of accuracy. The number of electrons in a coulomb (a unit of electrical charge) is $6.2415{*}10^{18}$, expressed to five decimal digits of accuracy. This physical constant is known with greater accuracy (at least 15 decimal digits), but as with most physical constants, the exact value is not known. The speed of light in a vacuum is approximately $2.99776{*}10^8$ meters per second.

A frequently observed convention for scientific notation is that the first part of the number is at least 1 and less than 10 (i.e., it has exactly one nonzero digit to the left of the decimal point).

In a particular case of scientific notation often called **engineering notation**, the exponent of 10 is a multiple of 3.

Summary Number systems represent a number by a sequence of symbols. Each symbol is assigned a certain numerical value. Any number system capable of representing arbitrarily large numbers must do one of the following:

- Provide for the set of symbols used to be infinite or extended indefinitely
- Permit symbols in a fixed, limited set to be repeated indefinitely
- Allow the sequence to be of arbitrary length and assign a value to each symbol dependent on its position in the sequence as well as on the symbol itself

Only the third alternative is practical when arbitrarily large numbers are to be represented. A few of the other, nonpositional number systems do appear in the historical record. They were applied usefully in practice, but only for a limited range of numbers. Their practical upper limits were a few thousand or a few million.

Positional number systems have stood the test of time in practice. Systems with a radix of 10, 20, and 60 are well known among current and ancient societies, the radix 10 system being currently the most widespread system for human use. Systems based on a radix of 2 are commonly implemented in computers. The closely related systems with a radix of 8 or 16 are frequently used by people when working with internal aspects of computing systems.

APPENDIX B
Symbols in the Language of Mathematics

The meanings of the individual special symbols used in the Language of Mathematics and appearing in this book are given below. These symbols are used to represent functions, values, sets, and structures and to write expressions.

TABLE B-1 Mathematical Symbols and Their Meanings

Symbol	Meaning
$+$	sum (addition)
$-$	difference (subtraction)
$*$	product (multiplication)
$\cdot$	In mathematics, also this symbol is used to represent multiplication. In this book this symbol is not used.
.	sometimes used as an infix symbol for the application of a function to its argument; that is, one can write f.x instead of f(x). See Section 3.4.9.
/	quotient (division)
$\div$	In mathematics this symbol is also used to represent division (sometimes restricted to integer division). In this book this symbol is not used.
$\uparrow$ (or ˆ)	exponentiation, raise to a power
x^2	Exponentiation is also represented by a superscript.
$\times$	Cartesian product (product of sets). See Section 4.1.1. In mathematics, this symbol is sometimes used for multiplication (see the symbol $*$ above) also. In this book, these two different functions are represented by the two different symbols to avoid confusion.
$=$	equals (is equal to)
$\neq$	does not equal, is not equal to
$<$	is less than
$>$	is greater than, exceeds
$\leq$	is less than or equal to. See Section 4.1.1.
$\geq$	is greater than or equal to. See Section 4.1.1.
$\in$	is an element of the set. See Section 4.1.1.

TABLE B-1 (*Continued*)

Symbol	Meaning
$\wedge$	conjunction (logical and). See the last part of Section 3.3.
$\vee$	disjunction (logical or). See the last part of Section 3.3.
$\neg$	not, logical negation. See the last part of Section 3.3.
$\Rightarrow$	implies, logical implication. See the last part of Section 3.3.
$\forall$	for all (logical and series). See Section 3.4.8.
$\exists$	there exists (logical or series). See Section 3.4.8.
$\cap$	set intersection. See Section 4.1.1.
$\cup$	set union. See Section 4.1.1.
$\setminus$	set difference. See Section 4.1.1.
$\subset$	is a subset of. See Section 4.1.1.
$\subseteq$	is a subset of (or equal to). See Section 4.1.1.
$\subsetneq$	is a proper subset of (is a subset of but not equal to)
&	concatenation. See the end of Section 4.1.3.
$\emptyset$	the empty set. See Section 4.1.1.
∞	infinity. See Section 4.2.
$\mathbb{B}$	the set of Boolean values (i.e., the set consisting of the values false and true). See Section 3.1.
$\mathbb{C}$	the set of complex numbers. See Appendix C.
$\mathbb{N}_0$, $\mathbb{N}_1$	the sets of natural numbers including and excluding 0, respectively. See Appendix C.
Perm	permutation. See the end of Section 4.1.3.
$\mathbb{Q}$	the set of rational numbers. See Appendix C.
$\mathbb{R}$	the set of real numbers. See Appendix C.
$\mathbb{Z}$	the set of integers, negative, 0, and positive. See Appendix C.

Structures formed by symbols appearing in this book have the following meanings.

TABLE B-2 Structures of Mathematical Symbols and Their Meanings

Structure	Meaning
(…)	1. Parentheses enclose a subexpression to be evaluated independently of its environment. See Section 3.4.2. 2. Parentheses enclose the arguments of a function [e.g., f(x), g(y, z)]. See Sections 3.3 and 3.4.1.
{…, …, …}	set of values. See Section 4.1.1.
{… | …}	set of values. See Section 4.1.1.
[[…, …, …]	sequence of values. See Section 4.1.3.
[… : … : …]	series and quantification (repeated application of a function). See Section 3.4.8.
|…|	absolute value of the expression inside the vertical bars [e.g., $\lvert -2\rvert = 2$ and $\lvert 2\rvert = 2$. |…| is sometimes written as abs(…)].

(*continued*)

TABLE B-2 (*Continued*)

Structure	Meaning
$\underset{i=1}{\overset{n}{\ldots}}\ x(i)$	series and quantification (repeated application of a function). See Section 3.4.8.
Σ	sum, used as the function symbol in the series notation above. See Section 3.4.8.
Π	product, used as the function symbol in the series notation above. See Section 3.4.8.
$f : \mathbb{X} \rightarrow \mathbb{Y}$	f is a function whose domain is the set $\mathbb{X}$ and whose range is the set $\mathbb{Y}$. See Sections 3.3, 3.4.9, and 4.1.1.

APPENDIX C
Sets of Numbers

In Section 4.1.1 on sets, several types of numbers were introduced:

- The *natural numbers* (i.e., the positive whole numbers)
- The *natural numbers with zero*
- The *integers* (positive, negative, and zero)
- The *rational numbers* (the quotients of any integer divided by a nonzero integer)
- The *real numbers*

In this appendix still another kind of number will be introduced, the *complex numbers*. This name is very unfortunate, as it frightens many people learning mathematics, suggesting to them that there is something particularly difficult and complicated about these numbers. There isn't, and the unsuccessful search for the expected complexity confuses them further.

Each of the sets of numbers in the list above is incomplete in some way. Adding elements to complete the set leads to the next set in the list.

Consider the set of natural numbers (e.g., $\{1, 2, 3, \ldots\}$) and the addition operation $+$. The sum of every pair of natural numbers is also a natural number, so this set is complete with respect to addition. It does not, however, contain an element which, when added to a natural number, gives the same natural number (e.g., there is no natural number x such that x+3=3). Such a "neutral" number would be useful in many cases and is called an *identity* in mathematical terminology. This set's identity element is named "zero," also written "0," and is inserted into the set to form the set of *natural numbers with zero*, $\{0, 1, 2, 3, \ldots\}$. This set is complete with respect to addition and has an identity element.

The set of natural numbers with zero is complete with respect to addition but not with respect to subtraction. The difference between a larger and a smaller natural number is in this set, but the difference between a smaller and a larger number is not. For example, the value of the difference 2–4 is not a number in this set. Negative numbers are defined as such differences and they, in turn, are inserted into the set of natural numbers with zero to form the set of *integers* $\{\ldots -3, -2, -1, 0, 1, 2, 3, \ldots\}$.

The set of integers is complete with respect to addition, subtraction, and multiplication, but not with respect to division. Some quotients, such as 35/5, which is 7, are elements of this set, but many are not (e.g., fractional values such as 34/5). They, in turn, are inserted into this set to form the set of *rational numbers* (i.e., the set of all quotients of an integer and a nonzero integer). Zero is excluded as a divisor because division by 0 is not defined mathematically (e.g., the value of 3/0 is not defined).

The set of rational numbers is complete with respect to addition, subtraction, multiplication, and division (except division by 0). It turns out, however, that some interesting values that we would like to consider numbers are not in the set of rational numbers. The square roots of many numbers, such as 2, 3, 5, 6, 7, and 101, are examples. The ratio of the circumference of a circle to its diameter, usually called "π," and e, the base of the natural logarithms, are other examples, because neither can be expressed as a rational number (i.e., as the ratio of two integers). These values are called *irrational numbers*. The existence of such numbers was recognized at least as early as classical Greek times; some pairs of lengths (numbers) were known not to be *commensurable* (i.e., were not integral multiples of any length). Irrational numbers have the property that they can be approximated arbitrarily closely by rational numbers (because rational numbers arbitrarily close to one another always exist). In other words, a sequence of rational numbers can be constructed which become and remain arbitrarily close to any particular irrational number (e.g., $\sqrt{2}$, π, e). Such a sequence is said to *converge* to the number in question (see Section 4.4). Therefore, the limits of all convergent sequences of rational numbers (as representatives of the irrational numbers) are appended to the set of rational numbers to form the set of *real numbers* (i.e., the set of rational and irrational numbers).

One sequence of rational numbers that converges to $\sqrt{2}$ is

$$[1, 1.4, 1.41, 1.414, 1.4142, 1.41421, 1.414213, 1.4142135, 1.41421356, \ldots]$$

Another sequence of rational numbers that converges to $\sqrt{2}$ is the sequence $[t_1, t_2, t_3, \ldots]$, where $t_1=1$ and, for all $i\geq 1$, $t_{i+1}=(t_i+2/t_i)/2$ (e.g., [1, 3/2, 17/12, 577/408, …]). Each term in this sequence is the average of the previous term and 2 divided by that previous term. One of these two (rational) numbers is smaller than the square root of 2 and the other is larger by a similar amount, so each term in the sequence is much closer to the square root of 2 than the preceding term. The convergence is rapid, as one can see from the fact that the fourth term in the sequence above is $577/408 = 1.414215\ldots$, which is the square root of 2 within six digits of accuracy, although the first term in the sequence (1) is a very poor approximation to $\sqrt{2}$.

The set of real numbers is complete with respect to addition, subtraction, multiplication, division, and limits of convergent sequences. It also contains all roots of nonnegative numbers. It does not contain square (and some other) roots of negative numbers. To include these numbers, the set of *complex numbers* is defined as the set of all ordered pairs of real numbers. The first number in a pair is called the *real part* and the second number in the pair is called the *imaginary part*. Again, this terminology is unfortunate, as it confuses many learners. There is nothing imaginary at all about the "imaginary" part of a complex number; it is no less real than the real part. Often, the pair (a, b) representing a complex number is written as a+b∗i, where the special constant i (for “imaginary”) represents the square root of –1 (i.e., $i^2 = -1$). Addition, subtraction, multiplication, and division are defined on such pairs so that square roots of negative numbers are present in the set of complex numbers. In particular, multiplication is defined as

$$(a+b*i)*(c+d*i) = a*c + a*d*i + b*c*i + b*d*i^2 = a*c-b*d+(a*d+b*c)*i$$

Square roots of negative numbers are, in this representation, “imaginary” numbers, that is, complex numbers with the real part equal to 0 and the “imaginary” part equal to the square root of the positive value of the number whose square root is desired [e.g., $(3*i)^2 = 9*i^2 = -9$, so $\sqrt{-9} = \pm 3*i$].

In summary, the several different sets of numbers are:

- Natural numbers: $\mathbb{N}_1 = \{1, 2, 3, \ldots\}$
- Natural numbers with zero: $\mathbb{N}_0 = \{0, 1, 2, 3, \ldots\}$
- Integers: $\mathbb{Z} = \{\ldots -3, -2, -1, 0, 1, 2, 3, \ldots\}$
- Rational numbers: $\mathbb{Q} = [\cup\ a, b : a\in\mathbb{Z} \wedge b\in\mathbb{N}_1 : \{a/b\}]$
- Real numbers: $\mathbb{R}$ (see the definition above)
- Complex numbers: $\mathbb{C} = \mathbb{R} \times \mathbb{R}$

Each of these sets is a subset of those built upon it:

$$\mathbb{N}_1 \subset \mathbb{N}_0 \subset \mathbb{Z} \subset \mathbb{Q} \subset \mathbb{R} \subset \mathbb{C}$$

under the convention that $\mathbb{R} \subset \mathbb{R}\times\{0\}$ (which is clearly a subset of $\mathbb{C}$). In any case, the set $\mathbb{R}$ is isomorphic to the set $\mathbb{R}\times\{0\}$; that is, there is a one-to-one correspondence between the elements of $\mathbb{R}$ and the elements of $\mathbb{R}\times\{0\}$, and the corresponding arithmetic functions on $\mathbb{R}$ and on $\mathbb{R}\times\{0\}$ have the same properties.

APPENDIX D Special Structures in Mathematics

An important characteristic of the Language of Mathematics is that it can be extended to include additional structures. Each new structure can be based on structures defined previously. Several such structures, including finite state machines (finite automata) and the basic model for probability theory, were defined in Sections 4.1.7 and 4.6.1.

This appendix defines a few additional examples of structures that have, in the course of the history of mathematics, been found to be useful. In addition, ordinary users of the Language of Mathematics can define additional structures that they deem useful for their own particular purposes and applications.

In the following examples of building new structures on structures defined previously, a definition will first be given in the combination of English and the Language of Mathematics as typically used in mathematical texts. Then, in some cases as examples, a definition in purely mathematical terminology is given. The latter is rarely, if ever, seen in the mathematical literature.

Group A *group* consists of a nonempty set S and a function, for which the infix symbol $\circ$ is used here, with the properties G1 through G4.

G1. Closure: For every a and b $\in$ S, $a \circ b \in S$.

G2. Associativity: For every a, b and c $\in$ S, $(a \circ b) \circ c = a \circ (b \circ c)$.

G3. Right identity: There is at least one element $e \in S$ such that for every $a \in S$, $a \circ e = a$.

G4. Right inverse: For every $a \in S$ there is at least one element $a' \in S$ such that $a \circ a' = e$.

A definition purely in the Language of Mathematics can be formulated as follows. (The comments in brackets on the right margin are not part of the mathematical expression; see Section 5.2.2.)

$$\text{IsGroup}(S, \circ) = \text{IsSet}(S) \wedge S \neq \emptyset \wedge (\circ\colon S \times S \rightarrow S) \wedge$$

[specification of S and $\circ$, D-1]

The Language of Mathematics: Utilizing Math in Practice, First Edition. Robert Laurence Baber.

$$[\wedge\ a, b : a{\in}S \wedge b{\in}S : a{\circ}b{\in}S] \wedge \qquad [G1]$$
$$[\wedge\ a, b, c : a{\in}S \wedge b{\in}S \wedge c{\in}S : (a{\circ}b){\circ}c{=}a{\circ}(b{\circ}c)] \wedge \qquad [G2]$$
$$[\vee\ e : e{\in}S : [\wedge\ a : a{\in}S : a{\circ}e{=}a]] \wedge \qquad [G3]$$
$$[\wedge\ a : a{\in}S : [\vee\ a' : a'{\in}S : a{\circ}a'{=}e]] \qquad [G4]$$

Many conclusions can be drawn from the group axioms (properties) above. Among them are: e is unique, e is also a left identity, the right inverse is unique and is also a left inverse, and e is its own inverse.

($\mathbb{Z}$, +) and ($\mathbb{R}$, +) are groups with the identity 0. The inverse of any nonzero element x is −x. The inverse of 0 is 0. ($\mathbb{R}\backslash\{0\}$, ∗) is a group with the identity 1. The inverse of any element x is 1/x.

The set S of a group need not be a set of numbers. In fact, there are many other types of sets associated with an appropriate function to form a group.

($\mathbb{Z}\backslash\{0\}$, ∗) is not a group because it does not satisfy property G4.

Semigroup A *semigroup* consists of a nonempty set S and a function with the properties G1 through G3. A semigroup does not have inverses, whereas a group does.

Many theorems can be proved about a semigroup. Theorems that are true for groups but do not involve an inverse are also true for semigroups.

Among the important semigroups are ($\mathbb{B}, \wedge$) and ($\mathbb{B}, \vee$). The fact that they are not groups has an important consequence: "cancellation" (see Section 5.1) cannot be used to deduce a=b from, for example, a∧c=b∧c. (A counterexample is a=false, b=true, and c=false.)

Note that the identity of the semigroup ($\mathbb{B}$, ∧) is true and that the identity of the semigroup ($\mathbb{B}$, ∨) is false.

Commutative Group A *commutative group,* also known as an *Abelian group,* is a group which exhibits the additional property called G5.

G5. Commutativity: The function ∘ is commutative, that is, for every a and b ∈ S, a∘b=b∘a.

A definition written exclusively in the Language of Mathematics is as follows:

$$\text{IsCommutativeGroup}(S, \circ\,) = \text{IsGroup}(S, \circ\,) \wedge \qquad [D\text{-}2]$$
$$[\wedge\ a, b : a{\in}S \wedge b{\in}S : a{\circ}b = b{\circ}a] \qquad [G5]$$

Field A field consists of a nonempty set S and two functions ∘ and •, where (S, ∘) is a commutative group (with identity element labelled 0), (S, •) is a commutative semigroup, ($S\backslash\{0\}$, •) is a (commutative) group, and the function • distributes over ∘ (defined in detail in line "Distrib." below). The condition that (S, •) is a semigroup ensures that the properties of closure, associativity, and the existence of an identity apply on (S, •) including the 0 element (only an inverse of 0 with respect to the function • is missing). More formally,

$$\begin{aligned}\text{IsField}(S, \circ, \bullet) = {} & \text{IsCommutativeGroup}(S, \circ) \wedge \qquad \text{[D-3]}\\ & \text{IsSemiGroup}(S, \bullet) \wedge [\wedge\ a, b : a{\in}S \wedge b{\in}S : a{\bullet}b = b{\bullet}a] \wedge\\ & \text{IsGroup}(S\backslash\{\text{IdentityOfGroup}(S, \circ)\}, \bullet) \wedge\\ & [\wedge\ a, b, c : a{\in}S \wedge b{\in}S \wedge c{\in}S : a{\bullet}(b{\circ}c) = (a{\bullet}b) \circ (a{\bullet}c)]\\ & \qquad \text{[Distrib.]}\end{aligned}$$

Real numbers, addition, and multiplication ($\mathbb{R}$, +, ∗) constitute an important example of a field. Many properties of the structure consisting of the real numbers, addition, and multiplication follow from the fact that ($\mathbb{R}$, +, ∗) is a field as defined above.

Exercise for the Reader In the definition above, the function IdentityOfGroup appears. Write a mathematical expression defining this function. What is its range? (*Hint*: Use the expression in line G3 above as a basis for developing your expression defining this function.)

The examples above illustrate the idea of combining already defined mathematical objects to define another object. This pattern repeats in mathematics in various areas. Anyone applying mathematics can extend mathematics in this way as desired and when useful.

APPENDIX E
Mathematical Logic

Practical Application When translating English statements involving logic into the Language of Mathematics, one generally proceeds as follows. First, the lowest-level logical statements are either defined as variables with Boolean values or as Boolean-valued expressions containing non-Boolean terms (arguments). These lowest-level logical statements are then combined with the appropriate logical functions: primarily $\wedge$ (and), $\vee$ (or), $\neg$ (not), and $\Rightarrow$ (implies). Quantified structures based on $\wedge$ and $\vee$ ("for all" and "there exists") are formed as needed.

When forming compound expressions, one must be careful to translate the *meaning* behind the English statements and not blindly transliterate individual words. A common pitfall in this regard is the use of the word "and" in English. Although the word "and" in English often corresponds to the logical "and" function ($\wedge$), it can and sometimes does correspond instead to the logical "or" function ($\vee$). For example, the English sentence "select all people living in Paris and London" usually means that all the people living *either* in Paris *or* in London are to be selected, not all those who live simultaneously *both* in Paris *and* in London. If "city" is the name of a variable and "Paris" and "London" are values, the selection criterion for "all people living in Paris and London" will usually be

$$\text{city}=\text{Paris} \vee \text{city}=\text{London}$$

not the expression

$$\text{city}=\text{Paris} \wedge \text{city}=\text{London}$$

Usually, the latter expression is an unnecessarily complicated way of writing the logical constant "false." The reader should answer the question "Why?"

In summary, when translating, one must always:

- First, read the text in the source language
- Second, *understand* what that text means
- Finally, *restate that meaning* in the target language

Skipping the second step—understanding—and instead translating literally from the source to the target language will too often lead to an incorrect translation (i.e., to a text in the target language that does not have the same meaning as the text in the source language). This danger is greatest for English texts containing statements involving logic and reasoning.

General Comments on Mathematical Logic There are at least two significantly different views of logic in mathematics and at least two corresponding ways of teaching mathematical logic. One approach I will call *application-pull* and the other, *research-push*. These terms are not strictly and precisely descriptive, because each approach has both a research-based and an application-oriented component. However, each approach seems to me to emphasize strongly one or the other aspect, so these two terms do distinguish well between the two approaches.

The application-pull approach views mathematical logic as that part of mathematics which deals with Boolean-valued expressions as already defined and used many times throughout this book. Otherwise, logic is nothing fundamentally new or different. One must, of course, ensure that the mathematical model being used adequately represents the application situation being modeled, but this requirement must be fulfilled in any case, regardless of whether the model involves logical, numerical, or other types of values, variables, and expressions. In this approach, the point of departure in teaching and using mathematics for problems in logic is the *application*. The necessary theory follows and plays a secondary role.

The research-push approach basically follows, at least coarsely, the historical development of a major research stream in mathematical logic. Steps in this development—and in corresponding courses for students—involve definitions of particular types of logical expressions and concepts, such as "propositions," "predicates," "first-order logic," and "higher-order logic." A fairly complex mathematical structure is constructed (see Appendix D). Upon this mathematical structure, a theory of logic and logical expressions is developed. Special symbols are used for some of the logical functions [e.g., $\supset$ for logical implication (not the subset relation)]. For some students these symbols can be confusing at first, because they are used for completely different functions in other parts of mathematics. Other symbols are unique to logic and represent new things to be learned. In this approach, the point of departure in teaching and using mathematics for problems in logic is the *theory*. Its application follows and plays a secondary role.

The research-push approach to logic begins by reformulating certain types of English statements in a more rigid form. Notational conventions relying ever more heavily on symbols are introduced until a mathematical language for logic has been constructed. In effect, the mathematical language for logic is constructed as a by-product of developing an interpretation from that language into English. The interpretation is built first, and the mathematical language for logic follows. The resulting mathematical language is similar to the mathematical language for numerical expressions, but there are significant differences in notation and the structure of expressions. For example, the conceptual similarity between

- the logical "for all" and "there exists" expressions for *quantification* and
- the *series* expressions for sums, products, set union, maximum, and so on,

is not made clear. As a result, the student must learn and the practitioner must apply two apparently different and unrelated concepts instead of one unified concept.

The application-pull approach is essentially the reverse. Standard mathematical language for numerical expressions is extended slightly to allow the truth values (false, true) needed for logic. Corresponding functions (and, or, negation, implication) are introduced, but the basic structure and notation of the mathematical language for numerical expressions are retained. The result is an extended Language of Mathematics which includes functions appropriate for logic. For each individual application, an appropriate interpretation from the Language of Mathematics to English is formulated. Certain common classes of applications correspond to the applications which formed the starting point for the research-push approach outlined in the paragraph above.

Among the disadvantages of the research-push approach for learning and practice are the following: A substantial part of the mathematical structure mentioned above is built up before one arrives at expressions generally useful for practical applications. New ideas, concepts, and terminology are introduced, complicating the learning process. Many of the concepts and distinctions introduced and defined in that structure are neither needed nor helpful for most practical applications.

The way in which any particular field of knowledge has been developed is often—probably even usually—not the best way to introduce and teach the topic to students. The development often involved a number of dead-end paths, some inefficient, circuitous routes, unnecessary concepts, and so on. For teaching purposes, the material should be consolidated and reorganized based on didactical–psychological considerations. Mathematical logic is no exception.

In my qualitative observation, students who have been taught logic by the research-push approach have more difficulty in applying logic to practical problems and situations than do students who have learned mathematical logic by the application-pull approach. Unfortunately, in my observation, research-push is the more commonly used teaching approach in mathematics, computer science, and other science courses. In engineering courses, on the other hand, again in my own experience and observation, the application-pull approach is typically used for teaching—to the extent that logic is taught at all to engineering students.

My own experience in learning, applying, and teaching logic has, admittedly, influenced (and perhaps even biased or prejudiced) my opinion on which approach is easier to learn and which should be the basis for teaching logic to different types of students. If the students' goals are to apply mathematics in general and mathematical logic in particular to practical problems, they are best served by learning via the application-pull approach. They will proceed more directly and quickly to the required knowledge and ability to use the material. They will not spend time learning those theoretical concepts that are of little or no relevance in applying mathematical logic in practice.

If, on the other hand, the students' goals are to pursue mathematical research, the research-push approach might serve them better. They will learn terms and concepts appearing frequently in the historical and research literature on mathematical logic and related subdisciplines of mathematics. Without knowledge of these special terms and concepts, that research literature will be inaccessible to them.

The application-pull view of logic is a component part of the material presented elsewhere in this book, so no addition to the Language of Mathematics is needed to cover it. The research-push view of logic requires developing the additional mathematical structure mentioned above and the theory built upon it, but most of the language elements required to do this are already included in the Language of Mathematics as presented in this book.

The historical literature on logic contains at least two main streams of thought and presentation. The approach taken in the body of this book is the more recent one. It can be contained completely within the Language of Mathematics. An older approach, dating back to at least classical Greek times, uses only natural language. Mathematical devices such as variables or other named references to values and mathematical expressions as introduced in Chapter 3 are not employed. In short, the more classical approach makes little use of the features of the Language of Mathematics presented in this book.

APPENDIX F
Waves and the Wave Equation

In this appendix part of a mathematical model for physical systems in which vibrations and waves can occur is presented. Insight into the nature of such vibrations can often be gained only with the help of the mathematical model. For example, it would not at all be obvious to a person observing the vibration of a string on a musical instrument that waves are traveling along the string in opposite directions in such a way that they always cancel at certain points along the string: in particular, at the two fixed ends of the string. Yet that view follows directly from the mathematical model of a vibrating string and is, in fact, the simplest way of explaining the string's vibration mathematically. Only with the help of this view can the frequencies of vibration be calculated from physical properties of the string, its tension, and the positions at which it is fixed.

Such a mathematical model led in the latter half of the nineteenth century to the discovery of radio waves and electromagnetic radiation in general. James Clerk Maxwell consolidated the then-current knowledge of electricity and magnetism into Maxwell's equations, an integrated mathematical model of electrical and magnetic physics. Theoreticians noticed that self-propagating waves followed mathematically from that model. The fact that no such waves had ever been observed led critics of the theory to point out how ridiculous and impractical the mathematical model was. A short time later, Hertz, knowing from the mathematical model what to look for and how to look for it, observed the predicted electromagnetic waves in his laboratory. In the early twentieth century, Marconi began using electromagnetic waves commercially to communicate via the atmosphere across the Atlantic Ocean. This was an early and major example of theoretical predictions, based on a mathematical model, leading to practical observations and applications in the electrical field. Later, it was recognized that light consists of these electromagnetic waves.

In the physical world waves of various kinds often occur. Some examples follow.

- Electromagnetic waves travel on a transmission line consisting of parallel wires or on a coaxial cable.
- The vibrations of strings in musical instruments consist of waves moving in opposite directions and interacting with each other.

- Sound waves are generated in air by all types of musical instruments: woodwind, brass, string, percussion, and so on.
- Sonic waves generated by ships' and submarines' engines propagate through the sea and can be detected by other ships and submarines.
- The sufficiently fast movement of a boat in water generates waves that propagate on the surface of the water.
- Electromagnetic waves propagate in three-dimensional space.
- The sound generated by the engine of an automobile or of an airplane propagates through the air.
- A supersonic airplane generates sound waves (often called *shock waves*) which propagate in the atmosphere in a different manner than do sound waves generated by a subsonic airplane.
- Verbal communication takes place via sound waves generated by the speaker's voice traveling through the air and sensed by the listener's ears.

In the case of waves along a one-dimensional lossless medium (such as an ideal electrical transmission line or a string on a musical instrument), the following equation describes or approximates the physical phenomenon involved:

$$\frac{\partial^2 f(x, t)}{\partial t^2} = v^2 * \frac{\partial^2 f(x, t)}{\partial x^2} \qquad \text{[the wave equation, F-1]}$$

where f is the amplitude of the wave at position x along the medium at time t. The constant v in equation F-1 depends on various physical properties of the medium, such as the diameter and the separation of the wires in a transmission line and physical electrical and magnetic properties of the material between and around the wires, or a string's tension and mass per unit length. The value of the function f represents the electric or magnetic field on a transmission line, the distance of the string from its resting position, and so on. Many physical systems giving rise to a wave on a one-dimensional medium lead to (or are approximated by) equation F-1, which, therefore, is called the *wave equation*.

The general solution to equation F-1 is any function of the quantity (x–v∗t) or of the quantity (x+v∗t) for which the derivatives in the formula exist or any sum of such functions, for example:

$$f(x, t) = g1(x-v*t) + g2(x+v*t) \qquad \text{[F-2]}$$

That such functions are solutions to the wave equation F-1 can be shown by deriving the partial derivatives

$$\frac{\partial^2 g1(x-v*t)}{\partial t^2}, \frac{\partial^2 g1(x-v*t)}{\partial x^2}, \frac{\partial^2 g2(x+v*t)}{\partial t^2}, \quad \text{and} \quad \frac{\partial^2 g2(x+v*t)}{\partial x^2} \qquad \text{[F-3]}$$

of the presumed solutions and showing that they satisfy the wave equation. The derivation involves a “chain” rule which is beyond the scope of this book, so the details are not shown here. The interested reader can find such derivations in introductory books on differential calculus dealing with partial derivatives.

Note that a function g1(x–v∗t) corresponds to a wave traveling to the right with velocity v and that a function g2(x+v∗t) corresponds to a wave traveling to the left with velocity v. *Explanatory hint*: If both x and t increase together so that the value of (x–v∗t) remains constant, the value of g1 remains constant (i.e., the entire graph of g1 moves to the right). Correspondingly, if x decreases as t increases in such a way that (x+v∗t) remains constant, g2 remains constant (i.e., the entire graph of g2 moves to the left).

In practice, the values of f are often restrained by boundary conditions at two positions, thereby constraining the solutions in significant ways. In the case of a vibrating string in a violin, harp, piano, or other stringed instrument constrained to f=0 at positions x=0 and x=L for all times t, the constraint f=0 at x=0 leads to the requirement that for all times t,

$$g1(-v*t) = -g2(v*t) \qquad \text{[F-4]}$$

or, generally,

$$g1(-a) = -g2(a) \qquad \text{[F-5]}$$

for all real values of the argument a. If the function g1 is known, the function g2 is determined by reflecting the known function about both axes (x=0 and g1=0). If the function g2 is known, the function g1 is determined in the same way.

The constraint f=0 at x=L for all times t leads to the requirement that for all t,

$$g1(L-v*t) = -g2(L+v*t) \qquad \text{[F-6]}$$

Together, these constraints (expressions F-5 and F-6) lead to the requirements that

$$g1(L-v*t) = g1(-L-v*t) \qquad \text{[F-7]}$$

and

$$g2(L+v*t) = g2(-L+v*t) \qquad \text{[F-8]}$$

In other words, the functions g1 and g2 must be periodic with period 2∗L. Expressed perhaps more clearly but equivalently,

$$g1(2*L+a) = g1(a) \qquad \text{[F-9]}$$

and

$$g2(2*L+a) = g2(a) \qquad [F-10]$$

for all real values of a.

The general solution to the one-dimensional wave equation above with boundary conditions f(0, t)=0 and f(L, t)=0 at all times t is, therefore,

$$f(x, t) = g1(x-v*t) + g2(x+v*t) \qquad [F-11]$$

which is equal to

$$f(x, t) = g1(x-v*t) - g1(-x-v*t) \qquad [F-12]$$

where g1 is any function with period 2∗L and for which the several derivatives exist.

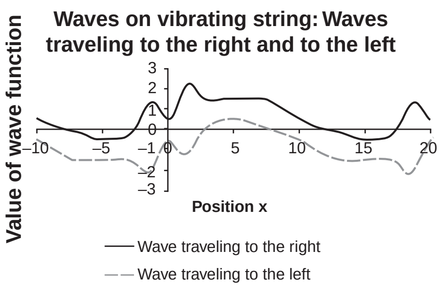
Thus, two waves with period 2∗L, each a mirror image of the other, pass each other in opposite directions traveling at the same speed so that they cancel each other at positions x=0 and x=L. The two waves add together at each point along the transmission medium. The resulting vibration of the string is called a *standing wave* because its apparent position does not move. The various points on the string vibrate rapidly up and down in place. Normally, one will see only a blur of the string as it vibrates tens or hundreds of times per second.

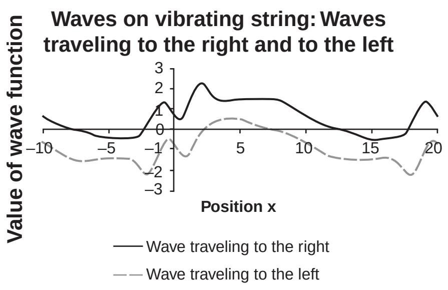
The following graphs illustrate a string fixed at positions x=0 and x=10 over which two waves are traveling, one to the right and one to the left. Each graph shows the values of the wave functions or the displacement of the string as functions of position along the string at one point in time. Conceptually, the wave functions are not limited to the length of the string, but extend indefinitely both to the right and to the left. The string itself is, of course, limited to its actual physical length.

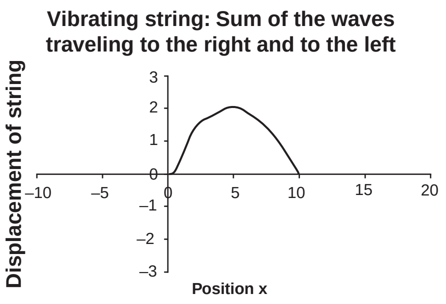
In this example, the length L of the string is 10, so the wavelength WL of the wave functions is 2∗L, or 20.

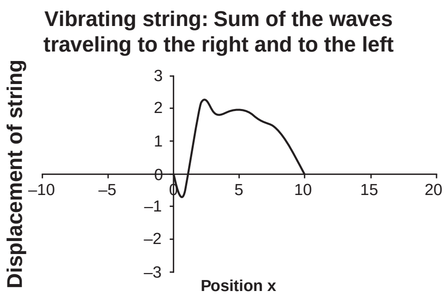
In each vertical pair of graphs below, the upper graph shows—at one point in time—the two traveling waves and the lower graph shows the displacement of the string at the same point in time. Over the length of the string, the displacement of the string (the function f above) is equal to the sum of the two wave functions g1 and g2.

The first vertical pair of graphs below shows the waves and the displacement of the string at one particular point in time. The second vertical pair of graphs shows the waves and the displacement of the string at a slightly later time. Notice in the upper two graphs that the entire graph of each wave moves slightly to the right or to the left.

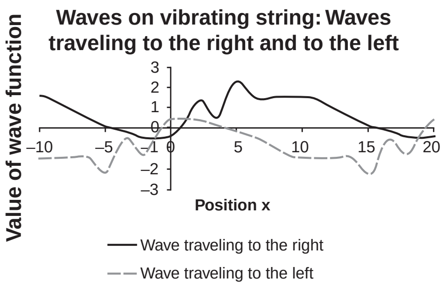


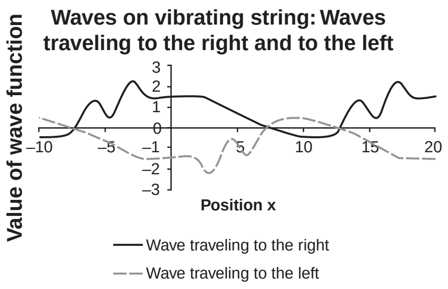


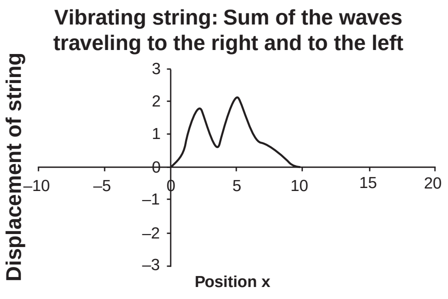


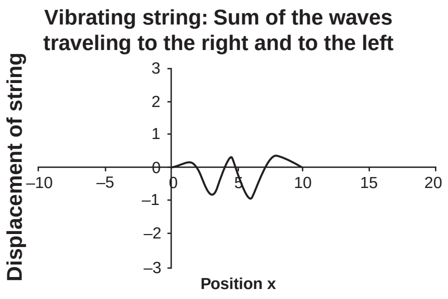


The following two pairs of graphs show the waves and the displacement of the string at two points in time significantly later in the cycle.









A wave traveling with velocity v travels a distance v∗t in time t. If the wave repeats every WL units of distance (but not in a shorter distance), its *wavelength* is defined to be WL. In the example of a vibrating string above, the solutions to the wave equation must repeat every 2∗L units of distance (i.e., their wavelengths are all 2∗L or a submultiple thereof). The time period (the time between repetitions of the wave at a particular position) is, therefore, 2∗L/v or a submultiple thereof. The frequency (number of repeated cycles per unit time) is the reciprocal of this quantity [i.e., v/(2∗L) or a multiple thereof]. As a result of the constraints on the string, it can vibrate only at a frequency of v/(2∗L) cycles per unit of time or a multiple thereof. Such multiples are called *harmonics*. Any particular string can vibrate only at its *fundamental frequency* (also sometimes called the *resonant frequency*) and/or harmonics thereof. It is physically impossible for it to vibrate at any other frequency. Thus, the frequencies at which the string on a musical instrument vibrates depends only on the length L and the velocity v, which, in turn, depends only on the mass per unit length of the string and its tension.

There are comparable equations for the propagation of waves in two and three dimensions and for lossy transmission media. The problem of signal distortion in a lossy electrical transmission line was the subject of both experimental and theoretical investigation in the late nineteenth century. Extensive physical experiments failed to identify the cause of the problem and its solution. Heaviside's theoretical investigations led to the result that if the transmission line's four parameters were related in a certain way, the signal would not be distorted, only attenuated. When solving a difficult problem, this experience was another example of how mathematics can sometimes be more efficient than experimentation. The key lay in translating the problem into the Language of Mathematics and then finding the solution to the resulting mathematical model.

The topic of waves has many implications of importance, such as vibrations and oscillations, some of which can destroy (and have destroyed) physical structures such as bridges and buildings. A particularly notable example was the first Tacoma Narrows Bridge, which collapsed in 1940. Wind across the bridge caused the bridge to twist, at first slightly, but at a resonant frequency of the bridge. The oscillations grew in amplitude and finally twisted the bridge apart, dumping automobiles on the bridge into the water below. In several movies of the episode one can see the twisting oscillations, which, mathematically, correspond to the standing wave on a vibrating string, as in the example above. The period of the oscillation was a few seconds, so the twisting of the roadbed could be clearly seen.

The paragraphs above deal with classical waves, such as waves on a vibrating string, pressure waves in gases, liquids and solids, electromagnetic waves, and waves on the surface of a liquid. Another type of wave arises in quantum mechanics, a subfield of physics. The solution to **Schrödinger's equation** is a complex-valued function whose complex conjugate is a probability density function for the state of a physical system as it varies with time. Both the mathematics and the physics associated with quantum wave mechanics is beyond the scope of this book. Detailed information on this subject can be found in the literature on quantum mechanics, Erwin Schrödinger, quantum wave function, and so on.

APPENDIX G
Glossary: English to the Language of Mathematics

Table G-1 contains a short glossary of English words, phrases, and parts of speech and their common translations in the Language of Mathematics.

TABLE G-1 Glossary from English to the Language of Mathematics

English	Language of Mathematics
adjective or adjective phrase (if not part of a noun phrase)	usually a value
adverb (if not part of a clause or an adjective phrase – rare)	value
all, for all	and series or quantification (e.g., $[\wedge \ldots : \ldots : \ldots]$), also $\forall$
and, but	logically: $\wedge$ or other functions (see Section 6.2.4) numerically (and only, not but): $+$
any, for any	usually, and series or universal quantification (e.g., $[\wedge \ldots : \ldots : \ldots]$, also $\forall$, but occasionally, or series or existential quantification (e.g., $[\vee \ldots : \ldots : \ldots]$), also $\exists$
are, is	See Sections 7.2 and 7.3.
at least one	or series or quantification (e.g., $[\vee \ldots : \ldots : \ldots]$), also $\exists$
average	See probability theory, Section 4.6.
bound, upper bound, lower bound	See convergence, limits, and bounds, Section 4.4.
but, and	$\wedge$
clause with action verb	Reformulate as clause with stative verb or express action with a past or present participle used as an adjective.
clause with stative verb	Boolean variable or Boolean function with non-Boolean arguments
collection, set	$\{\ldots, \ldots, \ldots\}$ (see Section 4.1.1)
converge(s)	See convergence, limits, and bounds, Section 4.4.
copy	permutation (see end of Section 4.1.3)
derivative	See calculus, Section 4.5.

(*continued*)

TABLE G-1 (*Continued*)

English	Language of Mathematics
different order (sequence)	permutation (see end of Section 4.1.3)
each	and series or quantification (e.g., $[\wedge \ldots : \ldots : \ldots]$), also $\forall$
element of	$\in$
equal, equals	$=$
estimate	See probability theory, Section 4.6.
every	and series or quantification (e.g., $[\wedge \ldots : \ldots : \ldots]$), also $\forall$
exchange	permutation (see end of Section 4.1.3)
find	$=$
for all, for every, for each, for any	and series or quantification (e.g., $[\wedge \ldots : \ldots : \ldots]$), also $\forall$
for no, none	negated or series or quantification (e.g., $\neg[\vee \ldots : \ldots : \ldots]$) or equivalently and series or quantification with negated assertion $[\wedge \ldots : \ldots : \neg \ldots]$
for some	or series or quantification (e.g., $[\vee \ldots : \ldots : \ldots]$), also $\exists$
if … then …	$\Rightarrow$
if and only if	$=$, $\Leftrightarrow$
implies	$\Rightarrow$
in (element of a set)	$\in$
input	often suggests a finite state machine model (see Section 4.1.7)
integer, whole number	$\ldots \in \mathbb{Z}$
is present, present	$=$
is, are	See Sections 7.2 and 7.3.
limit	See convergence, limits, and bounds, Section 4.4.
mean, median	See probability theory, Section 4.6.2.
merge	permutation (see end of Section 4.1.3)
noun or noun phrase	non-Boolean variable or function, occasionally a value
number	whole, integer $\ldots \in \mathbb{Z}$; real $\ldots \in \mathbb{R}$
of	multiplication
only if	$\Leftarrow$
or	$\vee$ or other functions (see Section 6.2.4)
other	$\neq$
output	often suggests a finite state machine model (see Section 4.1.7)
participle (past or present)	See "adjective" above in English column.
per, percent, permille	division (see Section 7.4)
plural or a general singular reference suggesting plural	set $\{\ldots, \ldots, \ldots\}$ or sometimes sequence $[\ldots, \ldots, \ldots]$ or quantification $[\ldots : \ldots : \ldots]$ (see Sections 4.1.1, 4.1.3, 3.4.8, 7.2, and 7.3)
predicate adjective, predicate noun	See "adjective" or "noun" above in English column.
predict, prediction	See probability theory, Section 4.6.

TABLE G-1 (*Continued*)

English	Language of Mathematics
preposition, prepositional phrase in an adjective or noun phrase	See "adjective" or "noun" above in English column.
present, is present	$=$
probability	See probability theory, Section 4.6.
project (values in time), projection	See probability theory, Section 4.6.
rate, rate of change	See calculus, Section 4.5.
real number	$\ldots \in \mathbb{R}$
rearrange	permutation (see end of Section 4.1.3)
same, same as	$=$
search	$=$, quantification (see Section 3.4.8)
set, collection	$\{\ldots, \ldots, \ldots\}$ (see Section 4.1.1)
some	or series or quantification (e.g., $[\vee \ldots : \ldots : \ldots]$), also $\exists$
sort	permutation (see end of Section 4.1.3)
sorted	and series (e.g., $[\wedge\ i : i \in \mathbb{Z} \wedge 1 \leq i < n : A(i) \leq A(i+1)]$)
state	often suggests a finite state machine model (see Section 4.1.7)
statistics	See probability theory, Section 4.6.
there exists, there is (are)	or series or quantification (e.g., $[\vee \ldots : \ldots : \ldots]$), also $\exists$
transition	often suggests a transition function in a finite state machine model (see Section 4.1.7)
variance	See probability theory, Section 4.6.2.
verb phrase	See "clause …" above in English column.
when (or whenever) … then …	$\Rightarrow$
whole number, integer	$\ldots \in \mathbb{Z}$

APPENDIX H
Programming Languages and the Language of Mathematics

There are two main categories of computer programming languages, static and dynamic. Both are written in notational forms similar to—and probably largely borrowed from—some of the notation used in the Language of Mathematics. Both categories use specialized and very restricted notation.

Static programming languages, as the name implies, are based on a static view of the results of computation. Programs written in these languages are oriented primarily to the results of the computation, not to the mechanism or the temporal sequence of the steps in the computational process. Lacking the notion of time, static programming languages are more similar to the Language of Mathematics than are dynamic programming languages. Classical examples of static programming languages are the LISP languages and their modern derivatives.

More widely used in computer applications are the *dynamic programming languages*. As the name implies, time and the passage of time during the execution of a program are fundamental concepts in these languages. A program in one of these languages consists of a sequence of statements, each of which can be viewed as a command executed in a time interval. The statements are executed in a well-defined order, one after the other. Among the best known examples of this category of programming languages are the various machine languages, assembly languages, Fortran, Algol, Basic, Cobol, C, C++, Java, and so on.

Dynamic programming languages are typically based on individual statements (commands) of the following types:

- A declaration statement, which creates a programming variable with a name, a set, and a value
- A release statement, which deletes a programming variable
- An assignment statement, which calculates the value of an expression and changes the previous value of a variable to the newly calculated value
- A conditional (if) statement, which executes one of two sequences of statements depending on the value of a Boolean expression

- A loop statement, which repeatedly executes a sequence of statements as long as the value of a Boolean expression is true

Each specific programming language has a certain format for each of these types of statements as well as additional statements for handling input and output from and to external files (modified forms of the assignment statement outlined above). Various additional aspects are added to such languages as deemed desirable by the language designer. In some programming languages, the declaration and release statements are implicit and hence not apparent in the written program.

Each statement in a dynamic programming language is a command. Thus, many verbs arise in these languages which are not stative, in contrast to the Language of Mathematics. Many verbs are in the imperative mood, which is completely lacking in the Language of Mathematics itself. The verbs implicit in the Boolean conditions in the if and loop statements are in the present tense, referring to the point in time at which the condition is evaluated. Thus, the timeless stative verb of the Language of Mathematics is missing in dynamic programming languages and, vice versa, the verbs in the dynamic programming languages are missing in the Language of Mathematics.

Mathematics can be used to specify and to analyze programs written in such languages. The initial state and the states resulting from the execution of each command are modeled mathematically. The sequence of these states is modeled as a mathematical sequence. The characteristics of interest of the several states are usually the values of the program variables existing in each state. Relations between selected states in the sequence, especially between the initial and final states, are either specified or derived. If specified, such a relation is, in effect, both a specification for the program and a theorem about the effect of executing it. Alternatively, such a relation can be derived from the definitions of the statements in the program.

Each command in a computer program can be modeled mathematically as a function. The argument is the state of execution immediately before the command is executed. The value of the function is the state of execution immediately after the command is executed. The function corresponding to a sequence of commands is, then, the composition of the functions of the individual commands. In this way the result of the temporal—dynamic—execution of a program can be modeled as a timeless—static—function.

APPENDIX I
Other Literature

Many readers will ask what other books, articles, and so on, are relevant to the subject of this book and what literature they should read to extend their understanding of the Language of Mathematics and their ability to use it in practice. In this appendix they might expect to find a typical bibliography—references to such literature in the form of a list of authors, titles, publishers, and so on. This understandable desire is, I have concluded, impossible to satisfy to any useful extent.

On the one hand, I am not aware of any book, article, or monograph by any other author that is at all closely related to the specific view of the Language of Mathematics presented in this book. I am aware of books with "Language of Mathematics" and similar terms in their titles, but they do not treat the material in a way really comparable to this book's approach. An occasional comment can be found in a book or an article comparing or likening writing mathematical expressions to translating from one language to another, but going no further. *Mathematics* itself is their point of departure, whereas *language* is this book's point of departure. The book *How to Solve It* by George Polya (who, in turn, refers to *Arithmetica Universalis* by Isaac Newton) goes further into this subject than any other work I have seen, but even it devotes only four pages to this topic [pp. 174–177, "Setting Up Equations," in the second edition, "First Princeton Science Library Printing," 1988]. After likening setting up equations to translating from one language to another, it lists a few guidelines and presents three examples of an exclusively mathematical or geometrical nature, comparing in each the English description with the mathematical equations and their terms and variables. Nowhere in the literature have I seen the main concepts and ideas of the Language of Mathematics as presented in this book. Thus, a list of directly and closely related literature would be empty.

On the other hand, essentially every book on any aspect of mathematics is relevant in some way to some topic or some aspect of mathematics dealt with or referred to in this book. In addition, many books, articles, and other publications in each of the many application areas are relevant to some degree. A bibliography including such works would be too long to be of practical use, much too long for a book of this type, and impossible to compile.

As a compromise—which, I hope, will be both general enough and short enough to be useful—this appendix will briefly suggest mathematical topics and subdisciplines for the interested reader to look into more deeply. This will take the form of comments and key words for searching the mathematical literature, organized by the chapters and, in some cases, by the sections in this book. These passages will not direct the reader to any particular work, but to classes of works on topics of relevance.

Typically, a list of references also serves to credit the sources of the material presented in a book. Because I have learned and accumulated the mathematical material presented in this book over a period of decades, I simply cannot remember the many sources that I should cite. Two of my statements must suffice to acknowledge my great debt to present and past mathematicians and to distinguish between my contributions and those of others:

- Most of the material in this book about the *language* of mathematics specifically and *viewing* mathematics from a *linguistic* standpoint in the ways presented here is original.
- None of the material in this book about mathematics itself is original.

Linguistic concepts such as grammatical terms are, of course, not my work, but their application to the Language of Mathematics in the ways presented in this book is, to the best of my knowledge, original.

Suggestions for finding material relevant to the several chapters of this book follow.

Chapter 1 General books and articles on language and linguistics deal with the questions of what language is, what it consists of, how and what it is used for, and so on. Similarly, general works on mathematics and the philosophy underlying mathematics discuss what mathematics is, the advantages of applying it in practice, and to some extent at least, some of the notational forms used in mathematics. General works on the history of science and the history of mathematics are relevant. Some texts dealing with specific application areas also discuss the advantages, disadvantages, and limitations of mathematics in analyzing and solving practical problems. Look for books and articles intended for a general audience or for introductory works intended for specialists in the particular area of your interest.

Chapter 2 The examples in Chapter 2 were constructed specifically for this book. They are not completely fictitious, however, but are based loosely on actual or potential applications.

The examples of calculations for ancient construction projects were motivated by extensive records written in cuneiform on clay tablets in Sumerian or Akkadian and dating from around 2000 B.C.E. and later. To find actual examples of such calculations, search for such keywords as mathematics, ancient Mesopotamia, Babylon, Ur, Uruk, Sumer, Akkad, Egypt, and mathematical cuneiform texts. In the 1930s, Otto E. Neugebauer (1899–1990) published an extensive, multivolume work on mathematical texts in cuneiform which remains a major source of such material. More recently, Jöran Friberg has published new material on this subject. General encyclopedic works on ancient Mesopotamian, Assyrian, and Egyptian societies and culture contain some material on the mathematics of those times. Other books and papers on this general subject can be found by searching for appropriate combinations of the keywords history, ancient, mathematics, science, engineering, engineers, and technology. The last two decades have seen the entry of active new researchers into this field, so in the next few years more new material can be expected. There is no dearth of cuneiform clay tablets for them to work on—museums contain thousands of still unexamined tablets to study.

For more detailed information on the energy in the sunlight arriving at the Earth, search for keywords such as solar radiation, reflection, scattering, absorption, environment, ecology, Earth, albedo, biosphere, energy balance, and climate.

For information on mathematics in banking and finance, search for these keywords and others, such as discounted cash flow, compound interest, accounting, bookkeeping, financial mathematics, financial models, and mathematical models in finance.

Extensive material on the use of mathematics in mechanical engineering can be found in books and articles on such topics as mechanical engineering, statics, dynamics, stress, strain, bending, moment, shear, beams, structural elements, Mohr's circle, and physics.

Chapter 3 This chapter contains basic, introductory material on mathematics and mathematical notation. Additional information on these subjects can be found in general and introductory books and articles on mathematics and the specific topics mentioned in Chapter 3. The reader seeking further literature on these topics should search for the keywords of interest contained in the text in Chapter 3.

Books giving definitions and brief explanations of mathematical terms exist; they often have a word such as dictionary, encyclopedia, or handbook in their titles or descriptions. These types of books are useful reference works. They contain information relevant not only to the topics dealt with in Chapter 3, but to mathematics in general.

Much material on tabular notation can be found in literature on the application areas in which the various types of tables are used. Management- and business-oriented literature often contains examples of different types of tables used in those areas, as does literature on various engineering and technical fields, including computer science. The same comment applies to figures, drawings, and diagrams, many of which are specific to particular applications, such as Mohr's circle for analyzing tension, compression, and shear in a beam, and the Smith chart for determining the impedance of a transmission line of arbitrary length and terminated with an arbitrary impedance, just to mention two examples. Engineering drawings of different types of physical components and electrical circuits are topics in their own right, with extensive standards for their use and interpretation.

The subject of undefined values in expressions to be evaluated is not treated uniformly or extensively in the mathematical and application-oriented literature. No standard conventions have been defined. Various suggestions how to handle these problems in particular situations in practice can be found scattered throughout the relevant mathematical and application oriented literature. Different approaches are appropriate for different application needs. To find such literature, search for terms used in the relevant parts of Section 3.5 and for terms related to the particular application.

Chapter 4 Additional material on the topics covered in Chapter 4 can be found in the many books on topics such as calculus, probability theory, statistics, analysis, real analysis, algebra, linear algebra. Such works vary from basic, introductory treatments of these subjects to very advanced books and articles for experts. Information on the more general topics dealt with in Sections 4.1 through 4.4, 4.7, and 4.8 are usually contained in general books on mathematics and handbooks for engineers and scientists. In some cases, articles on specific topics in this area can be found in the literature.

Chapter 5 The topics dealt with in Chapter 5 belong to the general aspects of mathematics. More extensive information on these topics can be found scattered in various places in the general background literature on mathematics and, in some cases, in literature on particular application areas. Search for terms appearing in the text in this chapter.

The topic of Section 5.4, optimization, is, mathematically, a broad area. Some types of optimization problems are dealt with in general, introductory works on mathematics, especially calculus. In addition, many specialized books and articles on various mathematical types of optimization problems and on various applications of optimization exist. These works vary from introductory in nature to very specialized treatises for experts. Search for terms used in the text of this section and for terms related to the application area of interest.

Chapter 6 Those aspects of Chapter 6 dealing with the English language—and the grammar of natural languages in general—are topics dealt with in many books on the English language, especially its grammar, and the grammars of other languages as well. To find additional information on these topics, search for the relevant grammatical terms as well as the words English and grammar.

Those aspects of Chapter 6 dealing with mathematics fall into two subcategories. One category relates to mathematical definitions and characteristics of mathematical objects. Additional information on those topics can be found in books and articles on mathematics in general and on the specific mathematical areas. For such information, search for general mathematical works or for the terms connected with the particular subfield of mathematics of interest. The second category relates to the relationships between mathematical objects and structures and grammatical aspects of English and other natural languages. To the best of the author's knowledge, nothing in the currently existing literature deals with these topics.

Chapter 7 Generally, the comments in the paragraphs above on existing literature relevant to Chapter 6 apply also to existing literature relevant to Chapter 7, albeit with a different emphasis. The meanings of a number of mathematical terms are described in works on mathematics in English sentences. These descriptions often provide a few useful guidelines for translating English text into the Language of Mathematics. If the English text is not already phrased in terms of the mathematically oriented English phrases and terms, one should rephrase the original text accordingly before finally translating into the Language of Mathematics. Determining the terms and phrases to use in searching for relevant literature is not straightforward; a general familiarity with the English terms used to describe (and informally define) the various mathematical objects is helpful and sometimes even necessary. The glossary in Appendix G can be helpful in this regard.

Chapter 8 The examples in Chapter 8 cover a wide variety of application areas, a wide range of complexity, and many different mathematical objects and structures. To find existing literature on any of the aspects in these examples, search the literature for terms relating to the area of your interest, be they terms from the application area, terms used for the mathematical objects in the examples, or terms describing the application area itself.

Appendix A A moderate amount of literature exists on the history of words, symbols, and notational systems for representing numbers. Some of this literature is contained in historical works on particular human societies and political entities; some, in works on the history of numbers and mathematics; and some, in works on corresponding subdisciplines of linguistics. Properties and proofs of properties of particular number systems can also be found in the relevant mathematical literature. Search for books and articles containing keywords such as number systems, number symbols, numerical symbols, numbers, and number words. For detailed information on the number systems used by particular societies, search for the preceding key words in combination with the names of the societies, countries, political entities, geographical areas, and so on, of interest. For information on particular number systems, search for works on the specific number system of interest, such as binary, octal, decimal, hexadecimal, and sexagesimal.

For information on words for numbers in specific languages, look at dictionaries as well as books and articles on the grammar of the specific language of interest.

Appendix B Symbols used in mathematics can be found in reference books, dictionaries, and handbooks for mathematics in general, for particular subdisciplines of mathematics, for scientific disciplines, for engineering, and for other disciplines in which mathematics is applied. Most technical books include a list of mathematical symbols used within the book in question. Search for the name of the symbol or for the name of the function it represents.

Appendix C The several types of numbers are defined and described in some general works on mathematics. Specific types of numbers are defined and described in many books and articles on the particular subdisciplines of mathematics in which they are used. Search for the name appearing in Appendix C of the set of numbers corresponding to the area of interest.

Appendix D The examples in Appendix D illustrate only very few of the special structures defined in mathematics by building upon already defined structures. Perhaps some literature on the philosophy underlying mathematics might discuss this process, which is so commonly used in mathematics. In turn, it is an example of the process of generalization so important and useful in mathematics. The reader will encounter additional examples of extending mathematical structures in the course of delving into new areas of mathematics, as those extensions are useful in the various subdisciplines of mathematics.

Appendix E An extensive literature explicitly about mathematical logic has evolved over the last century or two. Still earlier literature contains an implicit treatment of logic, such as geometry in classical Greek (and presumably even earlier) times, but that logic was based ultimately on ideas (one might even say intuition) expressed in natural language, not on more precise and formal definitions.

For literature on the mathematical logic of the last century or two, search for terms such as mathematical logic, George Boole, Boolean algebra, Gottlob Frege, Bertrand Russell, Alfred North Whitehead, propositional logic, predicate logic, first-order logic, propositional calculus, predicate calculus, and formal logic. Some of this literature deals also with the philosophy of mathematics and with establishing a foundation from which all of mathematics could be formally derived. When searching using a person's name, look not only for works written by that person, but also for works referring to that person.

Appendix F Further literature on waves can be found both in the mathematical literature and in literature on the various application areas, including nautical engineering, aeronautical engineering, electrical engineering, mechanical engineering, civil engineering, physics, quantum mechanics, geology, seismology, and music. Among other key words are vibration, oscillation, waves, wave mechanics, and resonance. See also Appendix F for additional terms and names useful in search criteria.

Appendix G The terms in Appendix G are themselves key words for searching for additional literature on this subject. At the end of many mathematical books are tables defining the special symbols used. The entries in such tables can sometimes be used in reverse as a guide to translating from English to the Language of Mathematics.

Appendix H For additional information on computer programming languages, see the vast literature on computer programming in general and on particular programming languages. This literature covers the complete spectrum from scientific and professionally oriented treatises, through material intended for technical practitioners, to collections of details for amateur hackers. This literature is in the form of books, articles, and periodicals. Much of this literature is available both in printed form and via the Internet. Among the more commonly known programming languages, old and new, are the following, which can be used to search for relevant books, articles, and so on: machine language, machine code, assembly language, SOAP, programming languages, higher-level programming languages, Basic, Fortran, Algol, Cobol, Jovial, C, C++, Java, Ada, PL/1, APL, Eiffel, object-oriented programming, OOP, LISP, Modula, Pascal, Scheme, and Snobol. Many other specialized languages oriented to particular applications exist or existed, some of which illustrate particular language features of interest in some cases.

INDEX

The reader can also locate passages on specific topics by referring to the following:

- Table of Contents
- List of Tables
- Appendix B Symbols in the Language of Mathematics
- Appendix G Glossary: English to the Language of Mathematics