Arnold L. Rosenberg, Denis Trystram

Understand Mathematics, Understand Computing

Discrete Mathematics that All Computing Students Should Know

March 9, 2018

Springer

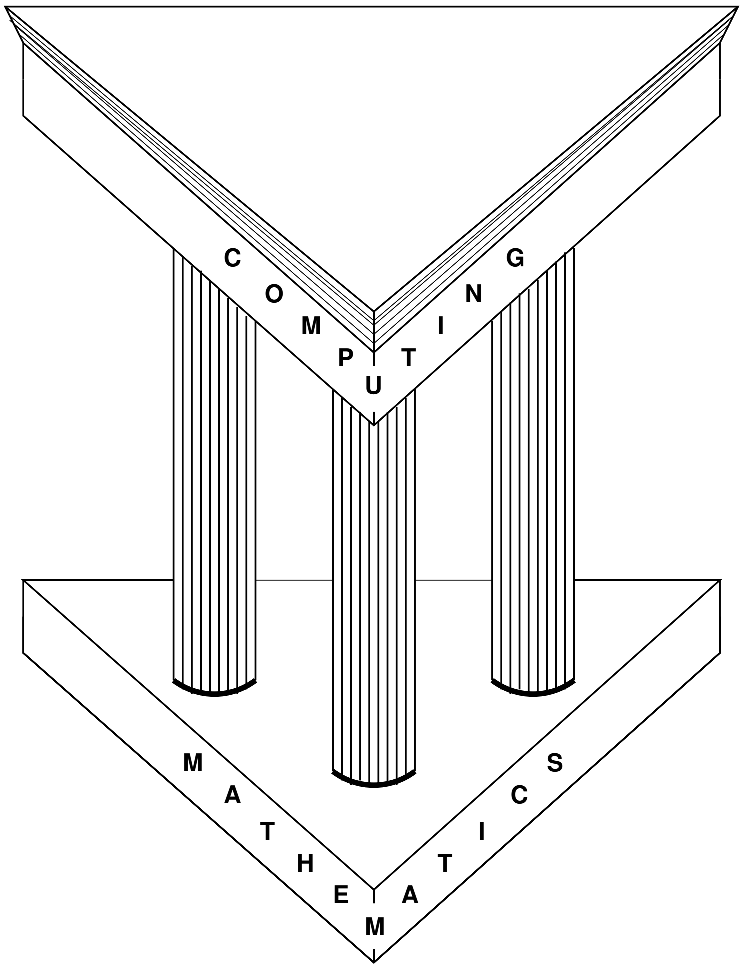
Understand Mathematics, Understand Computing

Discrete Mathematics that All Computing Students Should Know

Arnold L. Rosenberg
Distinguished University Professor Emeritus
University of Massachusetts
Amherst, MA 01003, USA
rsnbrg@cs.umass.edu

Denis Trystram
Distinguished Professor
University of Grenoble
Grenoble, FRANCE
denis.trystram@imag.fr

Mathematics is the foundation on which the edifice of Computing stands



Contents

Chapter 1
INTRODUCTION

1.1 Overview

In the early days of computing, all aspects of the field were considered the domain of the "techies"—the engineers and scientists and mathematicians who designed the early computers and figured out how to use them to solve a range of problems that is expanding even to this day. Back then, one expected every computer-oriented professional to have a mastery of many mathematical topics. Times—and the field of computing—have changed: classical computing curricula, which have traditional led to a BS degree within a school of science or engineering have been joined by curricula that lead to a BA degree or a degree in IT or in business or .... Within this new world, many aspiring computer professionals arguably need little knowledge of mathematics. However, many argue that the computing field suffers for this lack of mathematics, which has weakened the ability of many practitioners to design and perform experiments, to analyze their results, and to formulate well-reasoned conclusions based on these results. This situation has denied computer science the popular confidence enjoyed by other empirical disciplines; indeed, many would question the math-deprived practitioners' ability to reason rigorously about basic computational phenomena. Yet others, however, argue that such "scientific" acumen is not needed by practitioners in many of the newer segments of the computing field.

The preceding discussion is important to computing educators because we serve a large population of students, with quite diverse needs and aspirations. We strive to keep this diversity in mind within this essay. As we discuss a range of mathematical topics for possible inclusion as prerequisites for the undergraduate study of computing, we try to indicate why the selected topics are needed. Readers can then evaluate which topics are needed for students enrolled in their computing program.

1.2 The Elements of Rigorous Reasoning

1.2.1 Basic Reasoning

Distinguishing name from object. A fundamental stumbling block in the road to cogent reasoning arises from the inability to distinguish names from the objects they denote. A prime example within the world of computing resides in the inability to distinguish a function (which can be viewed as an infinite set of argument-value pairs) from a program, which can be viewed as a name for the function. **Note** that the often-used view of a function as a *rule* for assigning values to arguments should be avoided, because it suggests – *erroneously* – that an implementable such rule always exists!

Quantitative reasoning. Students should understand the foundational distinction between "growing without bound" and being innite. Within this theme, they should appreciate situations such as the following. Every integer, and every polynomial with integer coefficients, is finite, but there are infinitely many integers and infinitely many polynomials. Students should be able to verify (cogently but not necessarily via any particular formalism) assertions such as the following.

- Let us be given polynomials $p(x)$ of degree a and $q(x)$ of degree $b > a$, where a, b need not be integers. There must exist a constant $X_{p,q}$ (i.e., a constant that depends on the properties of polynomials p and q) such that for all $x > X_{p,q}$, $p(x) < q(x)$.
  Thus, polynomials having bigger degrees eventually *majorize*—i.e., have larger values than—polynomials having smaller degrees.
- Continuing with polynomial q of degree b: For any real number $c > 1$, there exists a constant $Y_{c;q}$ (i.e., a constant that depends on the properties of polynomial q and constant c) such that for all $x > Y_{c;q}$, $c^x > q(x)$.
  Thus, exponential functions eventually *majorize* polynomials.

1.2.1.1 The Elements of Formal Reasoning

induction
proof by contradiction

1.2.1.2 The Elements of Empirical Reasoning

Empirical reasoning does not convey the certitude that formal reasoning does.

1.3 Our Approach to Mathematical Preliminaries

"If your only tool is a hammer . . . "

We now review a broad range of mathematical concepts that are central to the study and practice of CS/CE. As we develop these concepts, we shall repeatedly observe instances of the following "self-evident truth" (which is what "axiom" means).

The conceptual axiom. *One's ability to think deeply about a complicated concept is always enhanced by having more than one way to think about the concept.*

Chapter 2
SETS, BOOLEAN ALGEBRA, AND LOGIC

2.1 Sets

2.1.1 Fundamental Set-Related Concepts

Sets are probably the most basic object of mathematical discourse. Sets exist to have *elements*, or *members*, the entities that *belong to* the set. The notion of set is surprisingly difficult to specify formally, so we just assume that *the reader knows what a set is and recognizes that some sets are finite, while others are infinite*. Speaking informally—a formal treatment will follow in later chapters—here are a few illustrative finite sets:

- the set of words in this book
  I do not know how big this set is, but I imagine that you as a reader have a better idea than I as an author.
- the set of characters in any JAVA program
  Note that while we are sure that this set is finite, we are not so confident about the number of seconds the program will run!
- the set consisting of *you*
  Paraphrasing the iconic television figure Mister Rogers, "You are unique." This set has just one element.
- the set of unicorns in New York City
  I will not argue with you about this, but I believe that this is the *empty set* $\emptyset$, which has zero members.

Some familiar infinite sets are:

- the set of *nonnegative integers*
- the set of *positive integers*
- the set of *all integers*
- the set of nonnegative *rational numbers*—which are quotients of integers
- the set of nonnegative *real numbers*—which can be viewed computationally as the set of numbers that admit infinite decimal expansions,

- the set of nonnegative *complex numbers*—which can be viewed as ordered pairs of real numbers,
- the set of *all* finite-length binary strings.
  A *binary string* is a sequence of 0s and 1s. When discussing computer-related matters, one often calls each 0 and 1 that occurs in a binary string a *bit* (for *binary digit*). The term "bit" leads to the term *bit string* as a synonym of *binary string*.

Despite this assumption, we begin the chapter by reviewing some basic concepts concerning sets and operations thereon.

As noted early, sets were created to contain members/elements. We denote the fact that element t *belongs to*, or, *is an element of* set T by the notation $t \in T$. A *subset* of a set T is a set S each of whose members belongs to T. The subset relation occurs in two forms, The *strong* form of the relation, denoted $S \subset T$, says that every element of S is an element of T, but *not* conversely; i.e., T contains (one or more) elements that S does not. The *weak* form of the relation, denoted $S \subseteq T$, is defined as follows:

$$[S \subseteq T] \quad \text{means:} \quad \Big[\textit{either} \quad [S = T] \quad \textit{or} \quad [S \subset T] \Big].$$

For any finite set S, we denote by $|S|$ the *cardinality* of S, which is the number of elements in S. Finite sets having three special cardinalities are singled out with special names. The limiting case of finite sets is the unique *empty set*, which we denote by $\emptyset$; thus, $\emptyset$ is characterized by the equation $|\emptyset| = 0$. (The empty set is often a limiting case of set-defined entities.) If $|S| = 1$, then we call S a *singleton*; and if $|S| = 2$, then we call S a *doubleton*.

It is often useful to have a convenient term and notation for *the set of all subsets of a set S*. This bigger set—it contains $2^{|S|}$ elements when S is finite—is denoted by $\mathscr{P}(S)$ and is called the *power set* of S.[1] Note carefully the two set-relations that we are talking about here:

A set T that is a subset *of set S is an* element *of the set* $\mathscr{P}(S)$.

You should satisfy yourself that the biggest and smallest elements of $\mathscr{P}(S)$ are, respectively, the set S itself and the empty set $\emptyset$.

2.1.2 Operations on Sets

Given two sets S and T, we denote by:

- $S \cap T$ the *intersection* of S and T: the set of elements that belong to *both* S and T.

$$[s \in S \cap T] \quad \text{means} \quad \Big[[s \in S] \ \textbf{and} \ [s \in T] \Big]$$

- $S \cup T$ the *union* of S and T: the set of elements that belong to S, or to T, *or to both*. (Because of the "or both" qualifier, this operation is sometimes called *inclusive*

[1] The name "power set" arises from the relative cardinalities of S and $\mathscr{P}(S)$ for finite S.

union.)

$$[s \in S \cup T] \quad \text{means} \quad \Big[[s \in S] \textbf{ or } [s \in T] \textbf{ or } [s \in S \cap T]\Big]$$

- $S \setminus T$ is the *(set) difference* of S and T: the set of elements that belong to S but not to T.

$$[s \in S \setminus T] \quad \text{means} \quad \Big[[s \in S] \textbf{ and } [s \notin T]\Big]$$

  (Particularly in the United States, one often encounters the notation "$S - T$" instead of "$S \setminus T$.")

We illustrate the preceding operations with the sets $S = \{a,b,c\}$ and $T = \{c,d\}$. For these sets:

$$\begin{aligned} S \cap T &= \{c\}, \\ S \cup T &= \{a,b,c,d\}, \\ S \setminus T &= \{a,b\}. \end{aligned}$$

In many set-related situations, the sets of interest will be subsets of some fixed "universal" set U.

We use the term "universal" as in "universe of discourse," not in the self-referencing sense of a set that contains all other sets as members, a construct (discussed by philosopher-logician Bertrand Russell) which leads to mind-bending paradoxes.

Given a universal set U and a *subset* $S \subseteq U$, we observe the set-inequalities

$$\emptyset \subseteq S \subseteq U.$$

When studying a context within which there exists a universal set U that contains all other sets of interest, we include within our repertoire of set-related operations also the operation of *complementation*

- $\overline{S} \overset{\text{def}}{=} U \setminus S$, the *complement* of S (relative to the universal set U).
  For instance, the set of odd positive integers is the complement of the set of even positive integers, relative to the set of all positive integers.

We note a number of basic identities involving sets and operations on them.

- $S \setminus T = S \cap \overline{T}$,
- If $S \subseteq T$, then
  1. $S \setminus T = \emptyset$,
  2. $S \cap T = S$,
  3. $S \cup T = T$.

Note, in particular, that[2]

$$[S = T] \ \text{ iff } \ \Big[[S \subseteq T] \text{ and } [T \subseteq S]\Big] \ \text{ iff } \ \Big[(S \setminus T) \cup (T \setminus S) = \emptyset\Big].$$

[2] "iff" abbreviates the common mathematical phrase, "if and only if."

The operations union, intersection, and complementation—and operations formed from them, such as set difference—are usually called the *Boolean (set) operations*, (named for the 19th-century English mathematician George Boole). There are several important identities involving the Boolean set operations. Among the most frequently invoked are the two "laws" attributed to the 19th-century French mathematician Auguste De Morgan:

$$\text{For all sets } S \text{ and } T\text{:} \quad \begin{cases} \overline{S \cup T} = \overline{S} \cap \overline{T}, \\ \\ \overline{S \cap T} = \overline{S} \cup \overline{T}. \end{cases} \tag{2.1}$$

(Algebraic) Closure. We end this section with a set-theoretic definition that occurs in many contexts. Let $\mathscr{C}$ be any (finite or infinite) collection of sets, and let S and T be two elements of $\mathscr{C}$. (Note that $\mathscr{C}$ is a set whose elements are sets.) Think, e.g., of the concrete example of set intersection.

We say that $\mathscr{C}$ is *closed* under intersection if whenever sets S and T (which could be the same set) both belong to $\mathscr{C}$, the set $S \cap T$ also belongs to $\mathscr{C}$. By De Morgan's laws, $\mathscr{C}$'s closure under union implies also its closure under intersection.

2.2 Binary Relations

2.2.1 The Formal Notion of Binary Relation

We begin our discussion of relations by adding a new (binary) set operation to our earlier repertoire. Given (finite or infinite) sets S and T we denote by $S \times T$ the *direct product* of S and T, which is the set of all *ordered pairs* whose first coordinate contains an element of S and whose second coordinate contains an element of T. For example, if $S = \{a,b,c\}$ and $T = \{c,d\}$, then

$$S \times T \ = \{\langle a,c\rangle, \langle b,c\rangle, \langle c,c\rangle, \langle a,d\rangle, \langle b,d\rangle, \langle c,d\rangle\}$$

The direct-product operation on sets affords us a simple, yet powerful, formal notion of binary relation.

Given (finite or infinite) sets S and T, a *relation* ρ *on* S *and* T (in that order) is any subset

$$\rho \ \subseteq \ S \times T.$$

When $S = T$, we often call ρ a *binary relation on (the set)* S ("*binary*" because there are *two* copies of set S being related by ρ).

Relations are so common that we use them in every aspect of our lives without even noticing them. The relations "equal to," "less than," and "greater than or equal to" are simple examples of binary relations on the integers. These same three relations apply also to other familiar number systems such as the rational and real numbers; only "equal," though, holds (in the natural way) for the complex numbers.

Some subset of the three relations "is a parent of," "is a child of," and "is a sibling of" probably are binary relations on (the set of people constituting) your family. To mention just one relation with distinct sets S and T, the relation "A is taking course X" is a relation on

$$(\text{the set of all students}) \times (\text{the set of all courses}).$$

By convention, when dealing with a binary relation $\rho \subseteq S \times T$, we often write "$s\rho t$" in place of the more stilted notation "$\langle s,t\rangle \in \rho$." For instance we (almost always) write "$5 < 7$" in place of the strange-looking (but formally correct) "$\langle 5,7\rangle \in <$."

The following operation on relations occurs in many guises, in almost all mathematical theories. Let ρ and ρ' be binary relations on a set S. The *composition* of ρ and ρ' (in that order) is the relation

$$\rho'' \stackrel{\text{def}}{=} \Big\{\langle s,t\rangle \in S \times S \mid (\exists t \in S)\Big[[s\rho t] \text{ and } [t\rho' u]\Big]\Big\}.$$

(Note that we have used both of our notational conventions for relations here. Note also our use of a common "shorthand" compound symbol "$\stackrel{\text{def}}{=}$": The sentence "$X \stackrel{\text{def}}{=} Y$" should be read X *is (or, equals), by definition*, Y.")
It is important to be able to assert that elements $s,t \in S$ are *not* ρ-related, i.e., $\langle s,t\rangle \notin S \times S$. Several notations have been developed for this purpose.

Relation	**Notation**	**Negation**	**Standard?**
set membership	$\in$	$\notin$	yes
equality	$=$	$\neq$	yes
less than (strong)	$<$	$\not<$ or $\geq$	yes
less than (weak)	$\leq$	$\not\leq$ or $>$	yes
greater than (strong)	$>$	$\not>$ or $\leq$	yes
greater than (weak)	$\geq$	$\not\geq$ or $<$	yes
generic	ρ	$\sim\rho$ or $\widetilde{\rho}$	no

(2.2)

There are several special classes of binary relations that are so important that we must single them out immediately, in the following subsections.

2.2.2 Order Relations

A binary relation ρ on a set S is a *partial order relation*, or, more briefly, is a *partial order* if ρ is transitive. This means that, for all elements $s,t,u \in S$,

$$\text{if } sRt \quad \text{and} \quad tRu \quad \text{then} \quad sRu. \tag{2.3}$$

The qualifier "partial" warns us that some pairs of elements of S do not occur in relation ρ. Number-related orders supply an easy illustrative example. Given any two distinct integers, m and n, one of them must be less than the other: either $m < n$, or $n < m$. In contrast, if we consider *ordered pairs* of integers, then there are pairs of pairs that are not related by the "less than" relation in any natural way. For instance, even though we may agree that, by a natural extension of the number-ordering relation "less than", $\langle 4,17\rangle$ is "less than" $\langle 22,19\rangle$, we might well not agree on which of $\langle 4,22\rangle$ and $\langle 19,17\rangle$ is less than the other—or, indeed, whether either is "less than" the other.

In many domains, order relations occur in two "flavors", *strong* and *weak*. For many such relations ρ—consider, e.g., "less than" on the integers—the weak version is denoted by underscoring the strong one's symbol. This will be our convention. Just as $\leq$ denotes the weak version of $<$, and $\geq$ denotes the weak version of $>$, we shall denote the weak version of a generic order ρ by $\underline{\rho}$.. Strong and weak versions of an order relation ρ (denoted, respectively, ρ and $\underline{\rho}$) are distinguished by their behavior under simultaneous membership. For illustration, instantiate the following template with ρ being "$<$" and with ρ being "$>$":

For a strong order ρ: **if** $[s\ \rho\ t]$, **then** $[t\ \widetilde{\rho}\ s]$
For the weak version $\underline{\rho}$ of ρ: **if** $[s\ \underline{\rho}\ t]$ **and** $[t\ \underline{\rho}\ s]$, **then** $[s = t]$.

2.2.3 Equivalence Relations

A binary relation R on a set S is an *equivalence relation* if it enjoys the following three properties:

1. R is *reflexive:* for all $s \in S$, we have sRs.
2. R is *symmetric:* for all $s, s' \in S$, we have sRs' whenever $s'Rs$.
3. R is *transitive:* for all $s, s', s'' \in S$, whenever we have sRs' and $s'Rs''$, we also have sRs''.

Sample familiar equivalence relations are:

- The equality relation, $=$, on a set S which relates each $s \in S$ with itself but with no other element of S.
- The relations $\equiv_{12}$ and $\equiv_{24}$ on integers, where[3]

  1. $n_1 \equiv_{12} n_2$ if and only if $|n_1 - n_2|$ is divisible by 12.
  2. $n_1 \equiv_{24} n_2$ if and only if $|n_1 - n_2|$ is divisible by 24.

  We use relation $\equiv_{12}$ (without formally knowing it) whenever we tell time using a 12-hour clock and relation $\equiv_{24}$ whenever we tell time using a 24-hour clock.

[3] As usual, $|x|$ is the *absolute value*, or, *magnitude* of the number x. That is, if $x \geq 0$, then $|x| = x$; if $x < 0$, then $|x| = -x$.

Closely related to the notion of an equivalence relation on a set S is the notion of a *partition* of S. A partition of S is a nonempty collection of subsets $S_1, S_2, \ldots$ of S that are

1. *mutually exclusive:* for distinct indices i and j, $S_i \cap S_j = \emptyset$;
2. *collectively exhaustive:* $S_1 \cup S_2 \cup \cdots = S$.

We call each set S_i a *block* of the partition.

One verifies the following Proposition easily.

Proposition 2.1 *A partition of a set S and an equivalence relation on S are just two ways of looking at the same concept.*

To verify this, we note the following.

Getting an equivalence relation from a partition. Given any partition $S_1, S_2, \ldots$ of a set S, define the following relation R on S:

sRs' if and only if s and s' belong to the same block of the partition.

Relation R is an equivalence relation on S. To wit, R is reflexive, symmetric, and transitive because collective exhaustiveness ensures that each $s \in S$ belongs to some block of the partition, while mutual exclusivity ensures that it belongs to only one block.

Getting a partition from an equivalence relation. To obtain the converse, focus on any equivalence relation R on a set S. For each $s \in S$, denote by $[s]_R$ the set

$$[s]_R \overset{\text{def}}{=} \{s' \in S \mid sRs'\};$$

we call $[s]_R$ *the equivalence class of s under relation R.*

The equivalence classes under R form a partition of S. To wit: R's reflexivity ensures that the equivalence classes collectively exhaust S; R's symmetry and transitivity ensure that equivalence classes are mutually disjoint.

The *index* of the equivalence relation R is its number of classes—which can be finite or infinite.

Let[4] $\equiv_1$ and $\equiv_2$ be two equivalence relations on a set S. We say that the relation $\equiv_1$ *is a refinement of* (or, *refines*) the relation $\equiv_2$ just when each block of $\equiv_1$ is a subset of some block of $\equiv_2$. We leave to the reader the simple verification of the following basic result.

Theorem 2.1. *The equality relation, $=$, on a set S refines every equivalence relation on S. In this sense, it is the finest equivalence relation on S.*

2.2.4 Functions

One learns early in school that a function from a set A to a set B is a rule that assigns a unique value from B to every value from A. Simple examples illustrate that this

[4] Conforming to common usage, we typically use the symbol $\equiv$, possibly embellished by a subscript or superscript, to denote an equivalence relation.

notion of function is more restrictive than necessary. Think, e.g., of the operation *division* on integers. We learn that division, like multiplication, is a function that assigns a number to a given pair of numbers. Yet we are warned almost immediately not to "divide by 0": The quotient upon division by 0 is "undefined." So, division is not quite a function as envisioned our initial definition of the notion. Indeed, in contrast to an expression such as "$4 \div 2$," which should lead to the result 2 in any programming environment,[5] expressions such as "$4 \div 0$" will lead to wildly different results in different programming environments. Since "wildly different" is anathema in any mathematical setting, we deal with situations such as just described by broadening the definition of "function" in a way that behaves like our initial simple definition under "well-behaved" circumstances and that extends the notion in an intellectually consistent way under "ill-behaved" circumstances. Let us begin to get formal.

A *(partial) function from set S to set T* is a relation $F \subseteq S \times T$ that is *single-valued;* i.e., for each $s \in S$, there is *at most* one $t \in T$ such that sFt. We traditionally write "$F : S \to T$" as shorthand for the assertion, "F is a function from the set S to the set T"; we also traditionally write "$F(s) = t$" for the more conservative "sFt." (The single-valuedness of F makes the nonconservative notation safe.) We often call the set S the *source (set)* and T the *target (set)* for function F. When there is always a (perforce, unique) $t \in T$ for each $s \in S$, then we call F a *total* function.

You may be surprisedf to encounter functions that are not total, because most of the functions you deal with daily are *total*. Our mathematical ancestors had to do some fancy footwork in order to make your world so neat. Their choreography took two complementary forms.

1. They expanded the target set T on numerous occasions. As just two instances:
   - They appended both 0 and the negative integers to the preexisting positive integers[6] in order to make subtraction a total function.
   - They appended the rationals to the preexisting integers in order to make division (by nonzero numbers!) a total function.

   The irrational algebraic numbers, the nonalgebraic real numbers, and the nonreal complex numbers were similarly appended, in turn, to our number system in order to make certain (more complicated) functions total.
2. They adapted the function. In programming languages, in particular, true undefinedness is anathema, so such languages typically have ways of making functions total, via devices such as "integer division" (so that odd integers can be "divided by 2") as well as various ploys for accommodating "division by 0."

The (20th-century) inventors of *Computation Theory* insisted on a theory of functions on nonnegative integers (or some transparent encoding thereof). The price for such "pureness" is that we must allow functions to be undefined on some arguments. Thus the theory renders such functions as "division by 2" and "taking square roots"

[5] We are, of course, ignoring demons such as round-off error.

[6] The great mathematician Leopold Kronecker said, "God made the integers, all else is the work of man"; Kronecker was referring, of course, to the *positive* integers.

as being *nontotal*: both are defined only on subsets of the positive integers (the even integers and the perfect squares, respectively).

Three special classes of functions merit explicit mention. For each, we give both a down-to-earth name and a more scholarly Latinate one.

A function $F : S \to T$ is:

1. *one-to-one* (or *injective*) if for each $t \in T$, there is at most one $s \in S$ such that $F(s) = t$;
   An injective function F is called an *injection*.
2. *onto* (or *surjective*) if for each $t \in T$, there is at least one $s \in S$ such that $F(s) = t$;
   A surjective function F is called a *surjection*.
3. *one-to-one, onto* (or *bijective*) if for each $t \in T$, there is precisely one $s \in S$ such that $F(s) = t$.
   A bijective function F is called a *bijection*.

2.3 Boolean Algebras

A *Boolean algebra* is a mathematical system **HERE

2.3.1 The Boolean Operations

2.3.2 The Axioms of Boolean Algebras

2.3.3 Two Special Boolean Algebras

2.3.3.1 The Algebra of Sets

2.3.3.2 Propositional Logic as an Algebra: The Propositional Calculus

A. The basic logical connectives. The Boolean set-related operations we discussed in Section 2.1.2 have important analogues within the context of Propositional logic. *logical* analogues of these operations for logical sentences and their logical *truth values*, TRUE and FALSE, often denoted 1 and 0, respectively:

- logical not ($\sim$) The operation not is the logical analogue of the set-theoretic operation of complementation. Because always writing "not" makes logical expressions long and cumbersome, we usually use the prefix-operator $\sim$ to denote not; i.e., we write

  $\sim P$ rather than the more cumbersome not P

  to denote the logical complementation of proposition P. Whichever notation we use, the defining properties of logical complementation are encapsulated in the following pair of equations

$$[\sim \text{TRUE} = \text{FALSE}] \quad \text{and} \quad [\sim \text{FALSE} = \text{TRUE}].$$

- logical or ($\vee$) The operation or—which is also called *disjunction* or *logical sum*—is the logical analogue of the set-theoretic operation of union. For convenience and brevity, we usually use the infix-operator $\vee$ to denote or in expressions. Whichever notation we use, the defining properties of logical disjunction are encapsulated as follows.

  $$[[P \vee Q] = \text{TRUE}] \quad \text{if, and only if,} \quad [P = \text{TRUE}] \text{ or } [Q = \text{TRUE}] \text{ or both.}$$

  Note that, as with union, logical or is *inclusive:* The assertion
  $[P \vee Q]$ is TRUE
  is true when *both* P and Q are true, as well as when only one of them is. Because such inclusivity does not always capture one's intended meaning, another, *exclusive* version of disjunction also exists, as we see next.
- logical xor ($\oplus$) The operation *exclusive or*—which is also called xor—is a version of disjunction that does not allow both disjuncts to be true simultaneously. For convenience and brevity, we usually use the infix-operator $\oplus$ to denote xor in expressions. Whichever notation we use, the defining properties of exclusive or are encapsulated as follows.

  $$[[P \oplus Q] = \text{TRUE}] \quad \text{if, and only if,} \quad [P = \text{TRUE}] \text{ or } [Q = \text{TRUE}] \textit{ but not } \text{both.}$$

  To emphasize the distinction between $\vee$ and $\oplus$: The assertion
  $[P \oplus Q]$ is TRUE
  is *false* when *both* P and Q are true.
- logical and ($\wedge$) The operation and—which is also called *conjunction* or *logical product*—is the logical analogue of the set-theoretic operation of intersection. For convenience and brevity, we usually use the infix-operator $\wedge$ to denote and in expressions. Whichever notation we use, the defining properties of logical conjunction are encapsulated as follows.

  $$[[P \wedge Q] = \text{TRUE}] \quad \text{if, and only if, } \textit{both} \quad [P = \text{TRUE}] \text{ and } [Q = \text{TRUE}]$$

- logical implication ($\Rightarrow$) The logical operation of implication, which is often called *conditional* and which we usually denote in expressions via the infix-operator $\Rightarrow$ differs from the other logical operations we have discussed in a way that the reader must always keep in mind. In contrast to not, or, and and, whose formal versions pretty much coincide with their informal versions, the formal version of implication, being formal, fixed, and precise, carries connotations that we do not always associate with the informal word "implies." The formal version of implication is defined as follows.

$[[P \Rightarrow Q] = \text{TRUE}]$ if, and only if, $[[\sim P] = \text{TRUE}]$ (inclusive) or $[Q = \text{TRUE}]$

This definition means, in particular, that

- If proposition P is false, then it implies *every* proposition.
- If proposition Q is true, then it is implied by *every* proposition.

- logical equivalence ($\equiv$) The final logical operation in our toolbox is variously called *equivalence,* as in:

  Proposition P is (logically) equivalent to Proposition Q

  and *biconditional*. We usually denote the operation in expressions via the infix-operator $\equiv$. The operation is defined as follows.

$[[P \equiv Q] = \text{TRUE}]$ if, and only if $[[P \Rightarrow Q] = \text{TRUE}]$ and $[[Q \Rightarrow P] = \text{TRUE}]$

We often use the term "connective" rather than "operation" to refer to what we have here called the logical operations of the Propositional Calclulus. This is because one often feels that logical propositions are static statements rather than active prescriptions for computations.

B. The logical connectives via truth tables If one is willing to view statements in the Propositional Calclulus as prescriptions for computations, then one can encapsulate the definitions of the basic logical connectives via functions that map logical expressions int the truth values true, or 1, and false, or 0. The following tables reproduce the definitions of Subsection A within this computational framework.

P	$\sim P$
0	1
1	0

P	Q	$P \vee Q$	$P \oplus Q$	$P \wedge Q$	$P \Rightarrow Q$	$P \equiv Q$
0	0	0	0	0	1	1
0	1	1	1	0	1	0
1	0	1	1	0	0	0
1	1	1	0	1	1	1

(2.4)

Note how the truth-value entries in the righthand truth table of (2.4) reinforce the lessons of Subsection A. This awareness is especially essential with the implication-related operations ($\Rightarrow$ and $\equiv$) that have informal counterparts in "real-life" reasoning. Think about whether your intuition agrees with the formal specifications of these operations.

C. The (Boolean) algebra of logical operations The operations of the Propositional Calclulus give rise to a Boolean algebra that manipulates (logical) propositions in much the way that the set-oriented Boolean algebra of Section 2.3.3.1 manipulates sets.

D. Logic via truth values As demonstrated in Subsection C, the Propositional Calculus is a Boolean Algebra when one interprets the logical connectives of Subsection A as operations on logical expressions that evaluate to either TRUE (or, 1) or FALSE (or, 0). But, the calculus forms a very special genre of Boolean Algebra, namely, a *free Boolean Algebra*.

The meaning of the qualifier *free* is easy to state but not so easy to understand. Here is a very informal try.

Any sort of algebra is defined as a set of objects accompanied by a set of basic operations, wherein the operations obey certain "self-evident" axioms.[7] In rough, but evocative, terms, the algebra is *free* if its operations do not relate to one another in any way that is not mandated by the axioms.

An easy illustration of this decidedly *not easy* concept is the following. There is a (quite important) genre of algebra called a *semi-group*. A semi-group $\mathscr{S}$ is specified via a set (of objects) S, together with a binary operation on S, which we denote $\oplus$. For the (algebraic) structure $\mathscr{S} = \langle S, \oplus \rangle$ to be a semi-group, it must obey the following two axioms.

1. $\mathscr{S}$ must be *closed* under operation $\oplus$; i.e., for all $s_1, s_2 \in S$, we must have
$$(s_1 \oplus s_2) \in S.$$
2. The operation $\oplus$ must obey the *associative law;* i.e., for all $s_1, s_2, s_3 \in S$, we must have
$$(s_1 \oplus s_2) \oplus s_3 \ = \ s_1 \oplus (s_2 \oplus s_3)$$

Now, it is easy to verify—we shall discuss this in Chapter 3—that the (algebraic) structure formed by the integers coupled with the operation of addition forms a semi-group (which the notation of Chapter 3 would denote $\langle \mathbb{Z}, + \rangle$). But this semi-group is *not* free, because integer addition obeys laws beyond just closure and associativity, namely:

1. Addition of integers is *commutative*.
2. Integer addition has an *identity*, namely 0.
3. Every integer $z \in \mathbb{Z}$ has an *additive inverse* (namely, $-z$).

So, it really is meaningful that the Boolean Algebra built upon the Porpositional calculus *is* free.

For us, the impact of the "freeness" of the Propositional algebra, *qua* Boolean Algebra, is manifest in the following *meta-theorem*[8] which is discussed and proved in [90].

Theorem 2.2. *A propositional expression $E(P,Q,\ldots,R)$ is a theorem of Propositional logic if, and only if, it to* TRUE *under all truth assignments to the propositions $P,Q,\ldots,R$.*

Proving Theorem 2.2 is beyond the scope of this book, but we now present a few illustrative instantiations, annotated to suggest their "real-world" messages. Each example is accompanied by a "real-life" interpretation and a verifying truth table.

The law of double negation

This is a formal analogue of the homely adage that "a double negative is a positive."

Proposition 2.2 *For any proposition P,*

[7] Indeed, that is what the word "axiom" means.

[8] A *theorem* exposes some truth within the mathematical structure being discussed. A *meta-theorem* exposes some truth about the way the discussion can proceed.

$$P \equiv \sim [\sim P]$$

Verification via truth table.

P	$\sim P$	$\sim [\sim P]$
0	1	0
1	0	1

(2.5)

Note that columns 1 and 3 of truth table (2.5) are identical. By Theorem 2.2, this fact verifies Proposition 2.2, the law of double negation. □

The law of contraposition

This is a very exciting example! Let us immediately convert this to a mathematical statement about the Boolean Algebra of Propositional logic and then prove the statement. We shall then contemplate the implications of this law for *logic* rather than *mathematics*.

Proposition 2.3 *For any propositions P and Q,*

$$[[P \Rightarrow Q] \equiv [\sim Q \Rightarrow \sim P]]$$

Verification via truth table.

P	$\sim P$	Q	$\sim Q$	$\underline{P \Rightarrow Q}$	$\underline{\sim Q \Rightarrow \sim P}$
0	1	0	1	1	1
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	0	1	1

(2.6)

Note that columns 5 and 6 of truth table (2.6) are identical. By Theorem 2.2, this fact verifies Proposition 2.3, the law of contraposition. □

Now let us reconsider the whole concept of contraposition, including this law, in the light of *logic and reasoning*.
asserts that the assertion
"Proposition Q implies Proposition Q"
is *logically equivalent* to the assertion
"the negation of Proposition Q implies the negation of Proposition P".
Think about this! In any system of reasoning in which a given proposition is either true or false

De Morgan's Laws:

Proposition 2.4 *For any propositions P and Q:*

- $[P \wedge Q] \equiv \sim [[\sim P] \vee [\sim Q]]$
- $[P \vee Q] \equiv \sim [[\sim P] \wedge [\sim Q]]$

Verification via truth table.

P	$\sim P$	Q	$\sim Q$	$[P \wedge Q]$	$[\sim P] \vee [\sim Q]$	$[P \vee Q]$	$[\sim P] \wedge [\sim Q]$
0	1	0	1	0	1	0	1
0	1	1	0	0	1	1	1
1	0	0	1	0	1	1	1
1	0	1	0	1	0	1	0

(2.7)

Note that columns 5 and 6 of truth table (2.7) are mutually complementary, as are columns 7 and 8. If we negate (or, complement) the entries of columns 6 and 8, then we can invoke Theorem 2.2 to verify proposition 2.4, which encapsulates De Morgan's laws for Propositional logic. □

The distributive laws for Propositional logic

In numerical arithmetic, multiplication distributes over addition, but not conversely, so we have a single distributive law for arithmetic (see Section 3.2.2). In contrast, each of logical multiplication and logical addition distributes over the other, so we have two distributive laws for Propositional logic.

- $P \vee [Q \wedge R] \quad \equiv \quad [P \vee Q] \wedge R$
- $P \wedge [Q \vee R] \quad \equiv \quad [P \wedge Q] \vee R$

P	Q	R	$[P \vee Q]$	$[P \wedge Q]$	$[Q \wedge R]$	$[Q \vee R]$	$P \vee [Q \wedge R]$	$[P \vee Q] \wedge [P \vee R]$	$P \wedge [Q \vee R]$	$[P \wedge Q] \vee [P \wedge R]$
0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0
0	1	0	1	0	0	1	0	0	0	0
0	1	1	1	0	1	1	1	1	0	0
1	0	0	1	0	0	0	1	1	0	0
1	0	1	1	0	1	0	1	1	1	1
1	1	0	1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1

(2.8)

Note that columns 8 and 9 of truth table (2.8) are identical, as are columns 10 and 11. By Theorem 2.2 this fact verifies the distributive laws for Propositional logic.

2.3.4 Connecting Mathematical Logic with Logical Reasoning

2.3.4.1 A formal notion of *implication*, and its implications

In everyday discourse, we all employ an intuitive notion of *implication*. When we make the assertion

Proposition A *implies* Proposition B

what we usually have in mind is

If Proposition A is true, then Proposition B istrue.

But this, or any, homespun meaning of the word/concept “implies” raises many questions.

- What if Proposition *A* is *not* true? Are there any inferences we can draw?
- Is there any relation between the assertion

  Proposition *A implies* Proposition *B*

  and its *converse* assertion

  Proposition *B implies* Proposition *A*?
- If we know that Proposition *B* is true, shouldn’t it be “implied by every other proposition—perhaps even a *false* one?

This section is devoted to discussing the *formal* notion of implication that was adopted by mathematical philosophers in the 19th century. The *advantage* of having such a formal notion is that it will answer all questions of the sort we have just posed. The *disadvantage* of having such a formal notion is that the way in which the notion answers some of our questions may be rather counter to one’s untutored intuition.

Converse
Contrapositive
Proof by contradiction

A. Integers and the number line.

Chapter 3
ARITHMETIC

Many, perhaps most, of us take for granted the brilliant notations that have been developed for the myriad arithmetic constructs that we use daily. Our mathematical ancestors have bequeathed us notations that are not only perspicuous but also convenient for computing and for discovering and verifying new mathematical truths. This chapter is dedicated to sharing this legacy with the reader.

3.1 Numbers and Numerals

Every reader will be familiar with the notion of *number* and with the familiar strings, called *numerals*, that name numbers within positional number systems.

> Numbers and numerals embody what is certainly the most familiar instance of a very important dichotomy that pervades our intellectual lives: the distinction between objects and their names:
>
> *Numbers are objects. Numerals are the names we use to refer to and manipulate numbers.*
>
> This is a crucially important distinction! You can "touch" a numeral: break it into pieces, combine two (or more) numerals via a large range of operations. Numbers are intangible abstractions: you cannot compute with them.

In our daily commerce, we typically deal with numerals formed within a *base-b positional number system,* i.e., by strings of *digits*, often embellished with other symbols, such as a *radix point*. We describe such systems in detail in Section 3.1.3.A. For now, we settle for a few examples: 123456789 (base 10, or, *decimal*), 10111.001 (base 2, or, *binary*), or 0.1267×8^{24} (base 8, or, *octal*).[1]

[1] The use of a period as the radix point is a US convention; in much of Europe, a comma denotes the radix point.

3.1.1 Numbers

We begin our study of arithmetic notions with a short taxonomy of numbers, the objects that arithmetic was invented to explicate and exploit. Although we assume that the reader is familiar with the most common classes of numbers, we do spend some time highlighting important features of each class, partly, at least, in the hope of heightening the reader's interest in this most basic object of mathematical discourse.

We present the four most common classes of numbers in what is almost certainly the chronological order of their discovery/invention.

Did humans invent these classes of numbers to fill specific needs, or did we just discover their pre-existing selves as needs prompted us to search for them? The great German mathematician Leopold Kronecker, as cited on page 477 of [3], shared his viewpoint on this question: "God made the integers; all else is the work of man."

A pleasing narrative can be fabricated to account for our multi-class system of numbers. In the beginning, the story goes, we needed to count things (sheep, bottles of oil, weapons, ...), and the positive *integers* were discovered. As accounting practices matured, we needed to augment this class with both zero (0) and the negative integers—and the full class of integers was born. As society developed, we had to start sharing subdivisible materials, so we needed to invent the *rational numbers*. Happily for the mathematically inclined, the rational numbers could be developed in a way that allowed one to view an integer as a special type of rational.

This quest for a single encompassing framework rather than a set of isolated concepts is a hallmark of mathematical thinking.

As the ancient Greeks (so the story goes) were inventing geometry and its place within architecture, they encountered the uncomfortable fact that the lengths of portions of eminently buildable structures were not "measurable," by which they meant "not rational." The poster child for this phenomenon was the hypotenuse of the isosceles right triangle with unit-length legs. As a response to this discomfort, the *real numbers* were invented. Once again, happily, one could develop the real numbers in a way that allowed one to view a rational number as a special type of real number. Time went on, and mathematics matured. Polynomials and their roots[2] were discovered, together with the next source of discomfort. To understand this discomfort, one must note that, since the invention of the real numbers, every polynomial $P_m(x) \overset{\text{def}}{=} x^2 - m$, where m is a nonnegative real number, had two roots, called, respectively, $+\sqrt{m}$ and $-\sqrt{m}$. But—here is the source of discomfort—*the polynomial $P_m(x)$ had no roots when m was negative, even a negative integer*. The response this time resided in the invention of a new *imaginary* number, called i, that was a root of the polynomial $P_{-1}(x) = x^2 + 1$; i was often defined via the equation, $i = \sqrt{-1}$; easily, $-i$ is also a root of $P_{-1}(x)$. By combining the imaginary number i with the real number system, the *complex numbers* were born.

[2] A number r is a *root* of a polynomial $P(x)$ if $P(r) = 0$.

The reader is possibly anticipating a new "discomfort" leading to an new augmentation of our number system, but, no, the story is now complete, in the sense expressed in the *Fundamental Theorem of Algebra*, which asserts:

Theorem 3.1 (The Fundamental Theorem of Algebra). *Every polynomial of degree n with complex coefficients has n roots over the complex numbers.*

We return to the later to the Theorem and its applications and implications. For now, though, it completes our historical tour, so we can finally begin to get acquainted with our four classes of numbers.

3.1.1.1 The integers

The most basic class of numbers are the *integers* (or, *whole numbers*, or, *counting numbers*). These are certainly the numbers that our prehistoric ancestors employed in the earliest days of our species.

A. Integers and the number line. We survey a number of the most important properties of the set $\mathbb{Z}$ that comprises *all integers* (the positive and negative integers and zero (0)) and the set $\mathbb{N}$ that comprises the *nonnegative integers* (the positive integers and 0). We estimate a property's importance from the vantage points of both mathematics and its manifold applications.

Several essential properties of $\mathbb{N}$ and $\mathbb{Z}$ are consequences of the sets' behavior under their natural order relations: strong ($<$) and weak ($\le$) and their converses.

- The set $\mathbb{Z}$ is *totally ordered*, also termed *linearly ordered*.

  This fact is embodied in the *Trichotomy Laws for integers*.

  The Trichotomy laws for integers.
  (a) *For each integer $a \in \mathbb{Z}$, precisely one of the following is true.*

  $$(1)\ \ a \text{ equals } 0\text{: } a = 0 \quad (2)\ \ a \text{ is } \textit{positive}\text{: } a > 0 \quad (3)\ \ a \text{ is } \textit{negative}\text{: } a < 0$$

  Consequently, $\mathbb{Z}$ can be visualized via the (2-way infinite) number line: $\ldots, -2, -1, 0, 1, 2, \ldots$.
  Analogously, $\mathbb{N}$ can be visualized via the (1-way infinite) number line: $0, 1, 2, \ldots$.

  (b) *For any integers $a, b \in \mathbb{Z}$, precisely one of the following is true.*

  $$(1)\ a = b \qquad (2)\ a < b \qquad (3)\ a > b$$

- The set $\mathbb{N}$ is *well-ordered*.

  The Well-ordering law for nonnegative integers. Every subset of $\mathbb{N}$ has a smallest element (under the ordering $<$).

- The set $\mathbb{Z}$ obeys the *"Between" Laws*

  The "Between" laws for integers.
  For any integers $a, b \in \mathbb{Z}$, there are finitely many $c \in \mathbb{Z}$ such that $a < c < b$.

  Any such $c \in \mathbb{Z}$ is *between* a and b, whence the name of the law.

B. Prime numbers We single out a subclass of the positive integers whose mathematical importance has been recognized for millennia but which have found important applications (e.g., within the domain of computer security) mainly within the past several decades.

Positive integer p *divides* positive integer n (or, *is a divisor of* n) if there is a positive integer q such that $n = p \cdot q$. Equivalently, we say that *n is divisible by p*. Thus, every positive integer n is divisible by $p = 1$ ($q = n$) and by $p = n$ ($q = 1$).

The class of integers we single out is defined by its divisibility characteristics.

An integer $p > 1$ is *prime* if its *only* positive integer divisors are 1 (which divides every integer) and itself (which is always a divisor).

We usually use the shorthand assertion, "p is a prime," instead of the longer, but equivalent, "p is a prime integer."

A very important way to classify a positive integer n is to list the primes that divide it, coupling each such prime p with its *multiplicity*, i.e., the number of times that p divides n. Let $p_1, p_2, \ldots, p_r$ be the distinct primes that divide n, and let each p_i divide n with multiplicity m_i. The *prime factorization* of n is the product $p_1^{m_1} \times p_2^{m_2} \times \cdots \times p_r^{m_r}$; note that this product satisfies the equation

$$n \;=\; p_1^{m_1} \times p_2^{m_2} \times \cdots \times p_r^{m_r} \tag{3.1}$$

When writing an integer n's prime factorization, it is traditional to list the primes $p_1, p_2, \ldots, p_r$ in increasing order, i.e., so that $p_1 < p_2 < \cdots < p_r$.
A positive integer n is totally characterized by its canonical prime factorization, as attested to by the following classical theorem, which we cite without proof.

Theorem 3.2. *The canonical prime factorization of every positive integer is unique.*

This theorem has been known for millennia and has been honored with the title *The Fundamental Theorem of Arithmetic*.

One very important application of Theorem 3.2 is as a mechanism for *encoding* sequences of positive integers as single integers! This works as follows. Consider the (infinite) ordered sequence of *all primes:*

$$(q_1 = 2), (q_2 = 3), (q_3 = 5), \ldots$$

Let

$$\mathbf{s} \;=\; \langle m_1, m_2, \ldots, m_k \rangle \tag{3.2}$$

be an arbitrary sequence of positive integers. Then Theorem 3.2 assures us that the (single) positive integer

$$\imath(\mathbf{s}) \;\stackrel{\text{def}}{=}\; q_1^{m_1} \times q_2^{m_2} \times \cdots \times q_k^{m_k}$$

is a (uniquely decodable) integer-representation of sequence $\mathbf{s}$.

We return to this idea of encoding-via-integers in a later chapter.

3.1.1.2 The rational numbers

Each augmentation of our number system throughout history has been a response to a deficiency with the then-current system. The deficiency that instigated the introduction of the rational numbers was the frequency with which a given integer q does not divide another given integer p. This fact meant that the operation of multiplication could often not be "undone", or "inverted", because division, the operation that would accomplish the undoing/inverting, would be a *partial operation:* There are pairs of integers neither of which divides the other. The defining characteristic of the expanded system is that, within it, every nonzero number divides every number.

The set $\mathbb{Q}$ of *rational* numbers consists of the number 0, plus the ratios p/q of all nonzero integers:

$$\mathbb{Q} \overset{\text{def}}{=} \{0\} \cup \{p/q \mid p, q \in \mathbb{Z} \setminus \{0\}\}$$

Each element of $\mathbb{Q}$ is called a *rational* number; each *nonzero* rational number p/q is often called a *fraction*, especially when $q > p$.

An alternative, mathematically more advanced, way of defining the set $\mathbb{Q}$ is to view it as the smallest set of numbers that contains the integers, i.e., the set $\mathbb{Z}$, and is *closed under the operation of division.* The word "closed" here means that, given any two numbers in $\mathbb{Q}$: r and $s \neq 0$, their quotient r/s belongs to $\mathbb{Q}$.

Numerous notations have been developed for "naming" rational numbers in terms of the integers they are "built from." Most of these notations continue our custom of employing the symbol "0" for the number 0. For the nonzero elements of $\mathbb{Q}$, we traditionally employ some notation for the operation of division and denote the

$$p/q \quad \text{or} \quad \frac{p}{q} \quad \text{or} \quad p \div q \tag{3.3}$$

The integer p in any of the expressions in (3.3) is the *numerator* of the fraction; the integer q is the *denominator*.

Obviously, every integer n can be viewed as a rational number, by letting n be the numerator of a fraction whose denominator is 1; i.e., $n = n/1$. Easily, this encoding preserves the special character of the numbers 0 and 1, because $0/1 = 0$ and $1/1 = 1$.

3.1.1.3 The real numbers

Each subsequent augmentation of our system of numbers inevitably gets more complicated than the last: one solves the easy problems first. The deficiency in the system of real numbers harkens back to historical time, roughly $2\frac{1}{2}$ millennia ago. The ancient Egyptians were prodigious builders who mastered truly sophisticated mathematics in order to engineer their temples and pyramids. The ancient Greeks perpetuated this engineering tradition, but they added to it the "soul" of mathematics.

Numbers were (literally) sacred objects to the Greeks, and they invented quite "modern" (to our perspective) ways of thinking about mathematical phenomena in order to understand *why* certain facts were true, in addition to knowing *that* they were true. One intellectual project in this spirit had to do with the way they designed contructions. They were attracted to geometric contructions that could be accomplished using only *straight-edges and compasses*. And—most relevant to our story—they preferred that the relative lengths of linear sections of their structures be *commensurable*, in the following sense. *Integers $x, y \in \mathbb{N}$ are* commensurable *if there exist $a, b \in \mathbb{N}$ such that*

$$ax = by \quad \text{or, equivalently,} \quad x = \frac{a}{b}y.$$

The desire to employ only commensurable pairs of integers, at least in moderately simple constructions, was shown to be impossible when one considered *the diagonal of the square with unit-length sides* or, equivalently, *the hypotenuse of the isosceles right triangle with unit-length legs*. In both situations, one encountered the unit lengths of the sides or legs of the structures, together with the *noncommensurable* length of the diagonal or hypotenuse, which, in current terminology, is $\sqrt{2}$. The Greek mathematicians, as reported by the renowned mathematician Euclid,[3] proved, using current terminology, that $\sqrt{2}$ is not rational. (We rephrase the proof imminently, in Proposition 3.1.) The conclusion from this proof is that the number system based on the rational numbers was inadequate. In response, they augmented this system by introducing *surds* or, as we more commonly term them, *radicals*, beginning a trajectory that culminated in the real number system. Since our intention has been to justify the journey along that trajectory, we leave our historical digression and turn to our real focus, the set $\mathbb{R}$ of *real numbers*.

For any integer $b > 1$, the real numbers are the numbers that can be named by em infinite strings built out of the digits $\{0, 1, \ldots, b-1\}$;[4] the resulting strings are called *b-ary numerals*. There are a couple of ways to form b-ary numerals; we shall discuss some of the most common ones in Section 3.1.3. For now, we define real numbers as those that can be represented by a base-b numeral, for some integer $b > 1$. Such a numeral has the form

$$\alpha_n \alpha_{n-1} \cdots \alpha_1 \alpha_0 . \beta_0 \beta_1 \beta_2 \cdots \tag{3.4}$$

and represents the (real) number

$$\underline{\alpha_n \alpha_{n-1} \cdots \alpha_1 \alpha_0 . \beta_0 \beta_1 \beta_2 \cdots} \ \stackrel{\text{def}}{=} \ \sum_{i=0}^{n} \alpha_i \cdot b^i \ + \ \sum_{j \geq 0} \beta_j \cdot b^{-j}.$$

By prepending a "negative sign" (or, "minus sign") $-$ to a numeral or a number, one renders the thus-embellished entity as negative.

The fact that every rational number (hence, also, every integer) is also a real number is manifest in the fact that integers and rationals can also be written as b-ary numerals as in (3.4). But with rational and integers, we are able to insist that their

[3] who wrote extensively on this and related subjects.

[4] This is not the traditional way that a mathematician would define the class of real numbers, but it is correct and adequate for thinking about the class.

numerals have special forms. We cite without proof the following classical results from the theory of arithmetic. The result for integers is easily phrased.

Theorem 3.3. *Integers as real numbers A real number is an integer if, and only if, it can be represented by a* finite-length *numeral.*

The result for rationals needs an introductory definition. An *infinite* sequence of numbers Σ is *ultimately periodic* if there exist two *finite* sequences of numbers, Γ and Δ, such that Σ can be written in the following form (spaces added to enhance legibility):

$$\Sigma = \Gamma\ \Delta\ \Delta\ \Delta\ \ldots\ \Delta\ \ldots$$

The intention here is that the finite sequence Δ is repeated *ad infinitum*.

Theorem 3.4. *Rationals as real numbers A real number is rational if, and only if, it can be represented by a numeral that is* ultimately periodic.

A clarification. Note that two types of sequences of 0s do not affect the value of the number represented by a numeral: (1) an *initial* sequence of 0s to the *left* of the radix point and of all non-0 digits; (2) a *terminal* sequence of 0s to the *right* of the radix point and of all non-0 digits.

One consequence of this fact is that we lose no generality by insisting that every numeral have the following form:

> a finite sequence of digits, followed by a radix point, followed by an infinite sequence of digits

Integers can then be singled out via numerals that have only 0s to the right of the radix point.

Theorems 3.3 and 3.4 show us that the three sets of numbers we have defined are a nested progression of successively more inclusive sets, in the sense that *every integer is a rational number* and *every rational number is a real number*. Those interested in the (philosophical) foundations of mathematics might quibble about the verb "is" in the highlighted sentences, but for all practical purposes, we can accept the sentences as written.

We close this section by verifying the earlier-mentioned assertion about the non-commensurability of the side-length of a square with the length of its diagonal or, equivalently, the leg-length of an isosceles right triangle with the length of its hypotenuse. The following proof is a simple application of Theorem 3.2, which can only suggest the range of the theorem's applications.

Proposition 3.1 *The real number* $\sqrt{2} = 2^{1/2}$ *is not rational.*

Verification. We prove the result by contradiction, a proof technique described in Chapter 2.3.4.

Let us assume, for contradiction, that $\sqrt{2}$ is rational. By definition, then $\sqrt{2}$ can be written as a fraction

$$\sqrt{2} = \frac{a}{b}$$

for positive integers a and b. In fact, we can also insist that a and b *share no common prime factor*. For, if a and b shared the prime factor p, then we would have $a = p \times c$ and $b = p \times d$. In this case, though, we would have

$$\sqrt{2} \;=\; \frac{a}{b} \;=\; \frac{p \times c}{p \times d} \;=\; \frac{c}{d}.$$

by cancellation of the common factor p. We can eliminate further common prime factors if necessary until, finally, we find a fraction for $\sqrt{2}$ whose numerator and denominator share no common prime factor. This must occur eventually because each elimination of a common factor leaves us with smaller integers, so the iterative elimination of common factors must terminate.

Let us say that, finally,

$$\sqrt{2} \;=\; \frac{k}{\ell} \tag{3.5}$$

where k and ℓ share no common prime factor. Let us square both expressions in (3.5) and multiply both sides of the resulting equation by ℓ^2. We thereby discover that

$$2\ell^2 \;=\; k^2. \tag{3.6}$$

This rewriting exposes the fact that k^2 is *even*, i.e., *divisible by* 2. But, Theorem 3.2 tells us that *if k^2 is divisible by* 2*, then so also is k*! This means that $k = 2m$ for some positive integer m, whoch allows us to rewrite (3.6) in the form

$$2\ell^2 \;=\; k^2 \;=\; (2m)^2 \;=\; 4m^2. \tag{3.7}$$

Hence, we can divide the first and last quantities in (3.7) by 2, to discover that

$$\ell^2 \;=\; 2m^2.$$

Repeating the invocation of Theorem 3.2 now tells us that the integer ℓ must be even.

We now see that *both k and ℓ are even, i.e., divisible by* 2. This contradicts our assumption that k and ℓ share no common prime divisor!

Since every step of our argument is ironclad—except for our assumption that $\sqrt{2}$ is rational, we conclude that that assumption is false! The Proposition is verified! $\square$

The proof of Proposition 3.1 is a classical (and early) example of *proof by contradiction*, as discussed in Section 2.3.4.

3.1.1.4 The complex numbers

$\mathbb{C}$ denotes the complex numbers

3.1.2 Order within the Number System

One of one's biggest friends when reasoning about numbers resides in the concept of *order*.

**HERE-ORDER

3.1.3 Numerals

We can identify distinct families of *operational* numerals, i.e., numerals that allow one to do things such as perform arithmetic (add, multiply, etc.).

THOUGHTS***** series expansions, strings created by a positional number system, and hybrids built on positional systems such as "scientific notation". *************

Of course, we are all familar with certain numbers that have *non-operational* names that we use all the time. Notable among these are (using terminology that anticipates future sections and chapters):

- π: the ratio of the circumference of a circle to its diameter; $\pi \approx 3.141592653\ldots$
- e: Euler's constant; the base of "natural" logarithms: $e \approx 2.718281828\ldots$
- i: the name of the number whose square is -1; i is appended to the real numbers in order to "complete" them to the complex numbers, within which system every polynomial of degree n has n roots.

A. Positional number systems. The most common way of forming numerals is via strings over a *number base*. We begin with an integer $b > 1$ that will serve as our base, and we define the set $B_b = \{0, 1, \ldots, b-1\}$ of *digits in base b*. To aid legibility, *within the context of base-b positional numerals*, we denote the digit $b-1$ as a single character, $\bar{b}$. We then form base-b numerals in the following way. This formation builds on *geometric sums*, a mathematical structure that we shall learn to manipulate, evaluate, and compute with in Section 3.3.3.

A base-b numeral is a string having three sections.

1. The numeral begins with its *integral part*, which is a finite string of digits from B_b: $\alpha_n \alpha_{n-1} \cdots \alpha_1 \alpha_0$.
   The base-b number represented by the numeral's integral part is[5]
   $$\underline{\alpha_n \alpha_{n-1} \cdots \alpha_1 \alpha_0} \ \stackrel{\text{def}}{=} \ \sum_{i=0}^{n} \alpha_i \cdot b^i$$
2. The numeral continues with a single occurrence of the *radix point* "."

[5] Our underlined notation for the numerical value of a numeral is not common, but we find it convenient.

3. The numeral ends with its *fractional part*, which is a string—*finite or infinite*—of digits from B_b: $\beta_0\beta_1\beta_2\cdots$.
   The base-b number represented by the numeral's fractional part is

$$\underline{.\beta_0\beta_1\beta_2\cdots} \overset{\text{def}}{=} \sum_{j\geq 0} \beta_j \cdot b^{-j}$$

In summary, then, the base-b number represented by the numeral $\alpha_n\alpha_{n-1}\cdots\alpha_1\alpha_0.\beta_0\beta_1\beta_2\cdots$ is

$$\underline{\alpha_n\alpha_{n-1}\cdots\alpha_1\alpha_0.\beta_0\beta_1\beta_2\cdots} \overset{\text{def}}{=} \sum_{i=0}^{n} \alpha_i \cdot b^i + \sum_{j\geq 0} \beta_j \cdot b^{-j}.$$

By prepending a "negative sign" (or, "minus sign") $-$ to a numeral or a number, one renders the thus-embellished entity as negative.

B. Scientific notation. The finite numerals in subsection A are all "exact" in the sense that changing any digit changes the value of the named number. We turn now to a class of numerals that abjure this "exactness" for the sake of expedience. There are a few reasons that one might be willing to do this.

- WHY SCIENTIFIC NOTATION

3.2 Arithmetic and Its Laws

Numbers are *adjectives*—you have five apples and three oranges—but in contrast to adjectives that are purely descriptive—the red ball, the big dog—numbers can be *manipulated*, using the tools of *arithmetic*.

3.2.1 The Tools of Arithmetic

The basic tools of arithmetic reside in a small set of operations, together with two special integers that play important roles with respect to the operations. Since these entities are so tightly intertwined, we discuss them simultaneously.

Two special integers The integers zero (0) and one (1), play special roles within all four of the classes of numbers we have described.

The operations of arithmetic Arithmetic on the four classes of numbers that we have described is built upon the following operations. When we say that an operation produces a number "of the same sort", we mean that it produces

- an integer result from integer arguments;
- a rational (number) result from rational (number) arguments;
- a real (number) result from real (number) arguments;

- a complex (number) result from complex (number) arguments;

The fundamental operations on numbers are, of course, familiar to the reader. Our goal in discussing them is to stress the laws that govern the operations.

3.2.1.1 Unary (single-argument) operations

A. Negating numbers. The operation of *negation:*

- is a *total function* on the sets $\mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$. It replaces a number a by its *negative*, a number of the same sort, denoted $-a$.
- is a *partial function* on the nonnegative subsets of $\mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$. It replaces a number a by its negative, $-a$, whenever both a and $-a$ belong to the nonnegative subset being operated on.

Zero (0) is the unique *fixed point* of the operation, meaning that 0 is the unique number a such that $a = -a$.

B. Reciprocating numbers. The operation of *reciprocating*

- is a *total function* on the sets $\mathbb{Q}, \mathbb{R}, \mathbb{C}$, which replaces each number a by its *reciprocal*, a number of the same sort, denoted $1/a$ or $\frac{1}{a}$. We shall employ whichever notation enhances legibility.
- is *undefined* on every integer a except for 1.

C. Floors and ceilings. The operations of *taking floors and ceilings* are total operations on the sets $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}$:

- The *floor* of a number a, also called *the integer part* of a, denoted $\lfloor a \rfloor$, is the largest integer that does not exceed a; i.e.,:

$$\lfloor a \rfloor \overset{\text{def}}{=} \max_{b \in \mathbb{N}} \Big[b \le a \Big]$$

- The *ceiling* of a number a of a, denoted $\lceil a \rceil$, is the smallest integer that is not smaller than a:

$$\lceil a \rceil \overset{\text{def}}{=} \min_{b \in \mathbb{N}} \Big[b \ge a \Big]$$

Thus, the operations of taking floors and ceilings are two ways to *round* rationals and reals to their "closest" integers.

D. Absolute values, magnitudes Let a be a real number. The *absolute value*, or, *magnitude*, of a, denoted $|a|$ equals either a or $-a$, whichever is positive. For a complex number a, the definition of $|a|$ is more complicated: it is a measure of a's "distance" from the "origin" complex number $0 + 0 \cdot i$. In detail:

$$|a| = \begin{cases} a & \text{if } [a \in \mathbb{R}] \text{ and } [a \ge 0] \\ -a & \text{if } [a \in \mathbb{R}] \text{ and } [a < 0] \\ \sqrt{b^2 + c^2} & \text{if } [a \in \mathbb{C}] \text{ and } [a = (b + ci)] \end{cases}$$

3.2.1.2 Binary (two-argument) operations

A. Addition and Subtraction. The operation of *addition* is a *total function* that replaces any two numbers a and b by a number of the same sort. The resulting number is the *sum of a and b* and is denoted $a+b$.
The operation of *subtraction* is a *total function* on the sets $\mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$, which replaces any two numbers a and b by a number of the same sort. The resulting number is the *difference of a and b* and is denoted $a-b$. On the nonnegative subsets of the sets $\mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$—such as $\mathbb{N}$, which is the largest nonnegative subset of $\mathbb{Z}$—subtraction is a *partial function*, which is defined only when $a \geq b$.

Subtraction can also be defined as follows. For any two numbers a and b, *the difference of a and b is the sum of a and the negation of b*; i.e.,

$$a-b \;=\; a+(-b)$$

The special role of 0 *under addition and subtraction.* The number 0 is the *identity* under addition and subtraction. This means that, for all numbers a,

$$a+0 \;=\; a-0 \;=\; a.$$

The special role of 1 *under addition and subtraction.* For any integer a, there is no integer between a and $a+1$ or between $a-1$ and a. For this reason, on the sets $\mathbb{Z}$ and $\mathbb{N}$, one often singles out the following special cases of addition and subtraction, especially in reasoning about situations that are indexed by integers. Strangely, these operations have no universally accepted notations.

- The *successor* operation is a *total function* on both $\mathbb{N}$ and $\mathbb{Z}$, which replaces an integer a by the integer $a+1$.
- The *predecessor* operation is a *total function* on $\mathbb{Z}$, which replaces an integer a by the integer $a-1$. It is a *partial function* on $\mathbb{N}$, which is defined only when the argument a is positive (so that $a-1 \in \mathbb{N}$).

The operations of addition and subtraction are said to be *inverse operations* of each other because each can be used to "undo" the other:

$$a \;=\; (a+b)-b \;=\; (a-b)+b$$

B. Multiplication and Division. The operation of *multiplication* is a *total function* that replaces any two numbers a and b by a number of the same sort. The resulting number is the *product of a and b* and is denoted either $a \cdot b$ or $a \times b$. We shall usually favor the former notation, except when the latter enhances legibility.

The operation of *division* is a *partial function* on all of our sets of numbers. Given two numbers a and b, the result of dividing a by b—*when that result is defined*—is the *quotient of a by b* and is denoted by one of the following three notations: a/b, $a \div b$, $\frac{a}{b}$. The *quotient of a by b* is defined precisely when *both*

(1) $b \neq 0$: one can never divide by 0
and
(2) there exists a number c such that $a = b \cdot c$.

Assuming that condition (1) holds, *condition (2) always holds when a and b belong to* $\mathbb{Q}$ *or* $\mathbb{R}$ *or* $\mathbb{C}$.

Division can also be defined as follows. For any two numbers a and b, *the quotient of a and b is the product of a and the reciprocal of b* (assuming that the latter exists); i.e.,

$$a/b \; = \; a \cdot (1/b).$$

Computing reciprocals of nonzero numbers in $\mathbb{Q}$ and $\mathbb{R}$ is standard high-school level fare; computing reciprocals of nonzero numbers in $\mathbb{C}$ requires a bit of calculational algebra which we do not cover. For completeness, we note that the reciprocal of the *nonzero* complex number $a + bi \in \mathbb{C}$ is the complex number $c + di$ where

$$c \; = \; \frac{a}{a^2 + b^2} \qquad \text{and} \qquad d \; = \; \frac{-b}{a^2 + b^2}.$$

The special role of 1 *under multiplication and division.* The number 1 is the *identity* under the operations of multiplication and division. This means that, for all numbers a,

$$a \cdot 1 \; = \; a \cdot (1/1) \; = \; a.$$

The special role of 0 *under multiplication and division.* The number 0 is the *annihilator* under multiplication. This means that, for all numbers a

$$a \cdot 0 \; = \; 0.$$

The operations of multiplication and division are said to be *inverse operations* because, when both operations can be applied, each can be used to "undo" the other:

$$a = (a \cdot b) \div b \; = \; (a \div b) \cdot b.$$

3.2.2 The Laws of Arithmetic

The student should understand the following laws of arithmetic on the reals, rationals, and reals—and be able to employ them cogently in rigorous argumentation.

- *The commutative law*. For all numbers x and y:

$$\begin{array}{ll} \textit{for addition:} & x + y \; = \; y + x \\ \textit{for multiplication:} & x \cdot y \; = \; y \cdot x \end{array}$$

- *The associative law*. For all numbers x, y, and z,

$$(x + y) + z \; = \; x + (y + z) \quad \textbf{and} \quad x \cdot (y \cdot z)(x \cdot y) \cdot z \; = \; x \cdot (y \cdot z).$$

This allows one, for instance, to write strings of additions or of multiplications without using parentheses for grouping.

- *The distributive law*. For all numbers x, y, and z,

$$x \cdot (y+z) \;=\; (x \cdot y) + (x \cdot z). \tag{3.8}$$

One commonly articulates this law as, "*Multiplication distributes over addition.*"

One of the most common uses of the distributive law reads equation (3.8) "backwards," thereby deriving a formula for *factoring* complex expressions that use both addition and multiplication.

Easily, addition does *not* distribute over multiplication; i.e., in general, $x+y\cdot z \neq (x+y)\cdot(x+z)$. Hence, when we see "$x+y\cdot z$", we know that the multiplication is performed before the addition. In other words, *Multiplication takes priority over addition.* This priority permits us to write the righthand side of (3.8) without parentheses, as in

$$x \cdot (y+z) \;=\; x \cdot y + x \cdot z.$$

Via multiple invocations of the preceding laws, we can derive a recipe for multiplying complicated expressions. We illustrate this via the "simplest" complicated expression, $(a+b)\cdot(c+d)$.

Proposition 3.2 *For all numbers a,b,c,d:*

$$(a+b)\cdot(c+d) \;=\; a\cdot c + a\cdot d + b\cdot c + b\cdot d \tag{3.9}$$

Verification. Note first that because multiplication takes priority over addition, the absence of parentheses in expressions such as (3.2) does not jeopardize unambiguity. Our proof of the proposition invokes the laws we have just enunciated multiple times.

$$\begin{array}{rll}
(a+b)\cdot(c+d) & = (a+b)\cdot c \;+\; (a+b)\cdot d & \text{distributive law} \\
& = c\cdot(a+b) \;+\; d\cdot(a+b) & \text{commutativity of multiplication } (2\times) \\
& = c\cdot a + c\cdot b + d\cdot a + d\cdot b & \text{distributive law } (2\times) \\
& = a\cdot c + b\cdot c + a\cdot d + b\cdot d & \text{commutativity of multiplication } (4\times) \\
& = a\cdot c + a\cdot d + b\cdot c + b\cdot d & \text{commutativity of addition}
\end{array}$$

□

We close our short survey of the laws of arithmetic with the following important two-part law.

- *The law of inverses*.
  - Every number x has an *additive inverse*, i.e., a number y such that $x+y=0$. This inverse is x's *negative* $-x$.
  - Every *nonzero* number $x \neq 0$ has a *multiplicative inverse*, i.e., a number y such that $x\cdot y = 1$. This inverse is x's *reciprocal*, $1/x$.

We close this section with another of our "fun" propositions.

Proposition 3.3 *Let n be any number that has a 2-digit decimal of the form $\delta 5$, where $\delta \in \{0,1,2,3,4,5,6,7,8,9\}$, so that*

$$n \;=\; 10\cdot\delta+5$$

Then

$$n^2 \;=\; 100\cdot\delta\cdot(\delta+1)+25.$$

In other words, one obtains a base-10 numeral for n^2 by multiplying δ by $\delta+1$ and appending 25 to the product.

Examples of Proposition 3.3 include $25^2=625$ (because $2\cdot 3=6$) and $75^2=5625$ (because $7\cdot 8=56$).
Verification (for general δ). We invoke Proposition 3.2 and the distributive law.

$$\begin{array}{rll} n^2 & = (10\cdot\delta+5)^2 & \text{Given} \\ & = 100\cdot\delta^2 \;+\; 100\cdot delta \;+\; 25 & \text{the proposition} \\ & = 100\cdot(\delta^2 \;+\; \delta) \;+\; 25 & \text{factoring: distributive law} \\ & = 100\cdot\delta\cdot(\delta+1) \;+\; 25 & \text{factoring: distributive law} \end{array}$$

□

3.2.3 Rational Arithmetic: A Worthwhile Exercise

In Section 3.1.1.2 we defined the rational numbers and reviewed why they were needed to compensate for the general lack of multiplicative inverses in the integers. But we did not review how to perform arithmetic on the elements of the set $\mathbb{Q}$. We correct this shortcoming now. Of course, the reader will have encountered rational arithmetic long ago—but we are now reviewing the topic in order to provide the reader with a set of worthwhile exercise to reinforce the mathematical thinking whose presentation is our main goal.

The rational numbers build their rules for arithmetic upon the corresponding rules for integers. For all p/q and r/s in $\mathbb{Q}$:

Addition:	$\frac{p}{q}+\frac{r}{s} = \frac{p\cdot s+r\cdot q}{q\cdot s}$
Subtraction:	$\frac{p}{q}+\frac{r}{s} = \frac{p}{q}+\frac{(-r)}{s}$
Multiplication:	$\frac{p}{q}\cdot\frac{r}{s} = \frac{p\cdot r}{r\cdot s}$
Division:	$\frac{p}{q}\div\frac{r}{s} = \frac{p}{q}\cdot\frac{s}{r}$

It is worth verifying that rational arithmetic as thus defined behaves in the required manner; in particular that rational arithmetic:

- works correctly when the argument rational numbers are, in fact, integers, i.e., when $q = s = 1$ in the preceding table.
- treats the number 0 appropriately, i.e., as an additive identity and a multiplicative annihilator; cf., Sections 3.2.1 and 3.2.2.
- obeys the required laws; cf., Section 3.2.2.
  Verifying the distributivity of rational multiplication over rational addition will be a particularly valuable exercise because of the required amount of manipulation.

3.3 Basic Algebraic Concepts and Their Manipulations

3.3.1 Powers and polynomials

A conceptually powerful notational construct is the operation of *raising a number to a power:* For real numbers a and b, the *bth power* of a, denoted a^b is defined by the system of equations

$$\begin{array}{ll} \text{for all numbers } a > 0 & a^0 = 1 \\ \text{for all numbers } a, b, c & a^b \cdot a^c = a^{b+c}. \end{array} \tag{3.10}$$

This deceptively simple definition has myriad consequences which we often take for granted.

- For all numbers $a > 0$, the number $a^0 = 1$.
  This follows (via cancellation) from (3.10) via the fact that
  $$a^b \cdot a^0 \;=\; a^{b+0} \;=\; a^b \;=\; a^b \cdot 1.$$
- For all numbers $a > 0$, the number $a^{1/2}$ is the *square root* of a, i.e., $a^{1/2}$ is the (unique, via cancellation) number b such that $b^2 = a$. Another common notation for The number $a^{1/2}$ is $\sqrt{a}$.
  This follows from (3.10) via the fact that
  $$a \;=\; a^1 \;=\; a^{(1/2)+(1/2)} \;=\; a^{1/2} \cdot a^{1/2} \;=\; \left(a^{1/2}\right)^2.$$
- For all numbers $a > 0$ and b, the number a^{-b} is the *multiplicative inverse* of a^b, meaning that $a^b \cdot a^{-b} = 1$
  This follows from (3.10) via the fact that
  $$a^b \cdot a^{-b} \;=\; a^{(b+(-b))} \;=\; a^0 \;=\; 1$$

When the power b is a positive integer, then definition (3.10) can be cast in the following attractive inductive form:

$$\begin{array}{ll} \text{for all numbers } a > 0 & a^0 = 1 \\ \text{for all numbers } a \text{ and integers } b & a^{b+1} = a \cdot a^b. \end{array} \tag{3.11}$$

Summing up, we now know about powers that are integral or fractional, positive, zero, or negative

We want the student to master the notions of polynomials and their associated notions, such as degrees and coefficients, and computations therewith, including polynomial summation and multiplication. While polynomial multiplication is often considered "non-elementary", it must be mastered in order to fully understand positional number systems; it is also essential, e.g., when discussing a range of topics relating to, say, fault tolerance and encryption).

3.3.2 Exponentials and Logarithms

This section introduces the fundamentals of two extremely important classes of functions which are functional inverses of each other, in the following sense. Functions f and g are *functional inverses* of each other if for all arguments x

$$f(g(x)) \;=\; x. \tag{3.12}$$

3.3.2.1 Basic definitions

A. Exponential functions. A function f is *exponential* if there is a positive number b such that, for all x,

$$f(x) \;=\; b^x. \tag{3.13}$$

The number b is the *base* of $f(x)$. The basic arithmetic properties of exponential functions are derivable from (3.10), so we leave these details to the reader and turn immediately to the functional inverses of exponential functions..

B. Logarithmic functions. Given an integer $b > 1$ (mnemonic for "base"), the *base-b logarithm* of a real number $a > 0$ is denoted $\log_b a$ and defined by the equation

$$a \;=\; b^{\log_b a}. \tag{3.14}$$

Logarithms are partial functions: $\log_b a$ is not defined for non-positive arguments.

The base $b = 2$ is so prominent in the contexts of computation theory and information theory that we commonly invoke one of two special notations for $\log_2 a$: (1) we often elide the base-2 subscript and write $\log a$; (2) we employ the specialized notation $\ln a$. Notationally:

$$\log_2 a \stackrel{\text{def}}{=} \log a \stackrel{\text{def}}{=} \ln a$$

We leave to the reader the easy verification, from (3.14), that the *base-b logarithmic function*, defined by

$$f(x) \;=\; \log_b x \tag{3.15}$$

is the functional inverse of the base-*b* exponential function.

3.3.2.2 Fun facts about exponentials and logarithms

Definition (3.14) exposes and—even more importantly—explains myriad facts about logarithms that we often take for granted.

Proposition 3.4 *For any base $b > 1$, for all numbers $x > 0$, $y > 0$,*

$$\log_b(x \cdot y) \;=\; \log_b x \,+\, \log_b y$$

Verification. Definition (3.14) tells us that $x = b^{\log_b x}$ and $y = b^{\log_b y}$. Therefore,

$$x \cdot y \;=\; b^{\log_b x} \cdot b^{\log_b y} \;=\; b^{\log_b x \,+\, \log_b y},$$

by the laws of powers. Taking base-*b* logarithms of the first and last terms in the chain yields the claimed equation. □

Many students believe that the following result is a *convention* rather than a consequence of the basic definitions. *The logarith of* 1 *to any base is* 0.

Proposition 3.5 *For any base $b > 1$,*

$$\log_b 1 \;=\; 0$$

Verification. We note the following chain of equalities.

$$b^{\log_b x} \;=\; b^{\log_b (x \cdot 1)} \;=\; b^{(\log_b x) + (\log_b 1)} \;=\; b^{\log_b x} \cdot b^{\log_b 1}$$

Hence, $b^{\log_b 1} \;=\; 1$. If $\log_b 1$ did not equal 0, then $b^{\log_b 1}$ would exceed 1. □

Proposition 3.6 *For all bases $b > 1$ and all numbers x, y,*

$$x^{\log_b y} \;=\; y^{\log_b x}$$

Verification. We invoke (3.14) twice to remark that

$$\left[x^{\log_b y} \;=\; b^{(\log_b x)\cdot(\log_b y)}\right] \quad \text{and} \quad \left[y^{\log_b x} \;=\; b^{(\log_b y)\cdot(\log_b x)}\right]$$

The commutativity of addition completes the verification. □

Proposition 3.7 *For any base $b > 1$,*

$$\log_b(1/x) \ = \ -\log_b x$$

Verification. This follows from the fact that $\log_b 1 = 0$, coupled with the product law for logarithms.

$$\log_b x + \log_b(1/x) \ = \ \log_b(x \cdot (1/x)) \ = \ \log_b 1 \ = \ 0$$

□

Proposition 3.8 *For any bases $a, b > 1$,*

$$\log_b x \ = \ (\log_b a) \cdot (\log_a x). \tag{3.16}$$

Verification. We begin by noting that, by definition, Note that

$$x \ = \ b^{\log_b x} \ = \ a^{\log_a x}. \tag{3.17}$$

Let us take the base-b logarithm of the second and third expressions in (3.17) and then invoke the product law for logarithms. From the second expression in (3.17), we find that

$$\log_b \left(b^{\log_b x} \right) \ = \ \log_b x. \tag{3.18}$$

From the third expression in (3.17), we find that

$$\log_b \left(a^{\log_a x} \right) \ = \ (\log_b a) \cdot (\log_a x). \tag{3.19}$$

We know from (3.17) that the righthand expressions in (3.18) and (3.19) are equal, whence (3.16) □

If we set $x = b$ in (3.16), then we find the following marvelous equation.

Proposition 3.9 *For any integers $a, b > 1$,*

$$(\log_b a) \cdot (\log_a b) \ = \ 1 \qquad \textit{or, equivalently,} \qquad \log_b a \ = \ \frac{1}{\log_a b}. \tag{3.20}$$

3.3.2.3 Exponentials and logarithms within information theory

The student should recognize and be able to reason about the following facts. If one has an alphabet of a letters/symbols and must provide distinct string-label "names" for n items, then at least one string-name must have length no shorter than $\lceil \log_a n \rceil$.

Proposition 3.10 *Say that one must assign distinct labels to n items, via strings over an alphabet of a letters. Then at least one string-label must have length no shorter than $\lceil \log_a n \rceil$.*

Verification. Let *Sigma* be an alphabet of a letters/symbols. For each integer $k \geq 0$ (i.e., for each $k \in \mathbb{N}$), let $\Sigma^{(k)}$ denote the set of all length-k strings over Σ. The bound of Proposition 3.10 follows by counting the number of strings of various lengths over Σ, because each such string can label at most one item. Let us, therefore, inductively evaluate the cardinality $|\Sigma^{(k)}|$ of each set $\Sigma^{(k)}$.

- $|\Sigma^{(0)}| = 1$
  This is because the null-string ε is the unique string in $\Sigma^{(0)}$, i.e., $\Sigma^{(0)} = \{\varepsilon\}$.
- $|\Sigma^{(k+1)}| = |Sigma| \cdot |\Sigma^{(k)}|$.
  This reckoning follows from the following recipe for creating all strings over Σ of length $k+1$ from all strings of length k.

  $$\Sigma^{(k+1)} \ = \ \{\sigma x \mid \sigma \in \Sigma, x \in \Sigma^{(k)}\}$$

  This recipe is correct because

  – Each string in $\Sigma^{(k+1)}$, as constructed, has length $k+1$.
    This is because the recipe adds a single symbol to a length-k string.
  – For each string $x \in \Sigma^{(k)}$, there are $|\Sigma|$ distinct strings in $\Sigma^{(k+1)}$, as constructed.
    This is because each string in $\Sigma^{(k+1)}$ begins with a distinct symbol from Σ.
  – $\Sigma^{(k+1)}$, as constructed, contains all strings of length $k+1$ over Σ.
    This is because for each $\sigma \in \Sigma$ and each $x \in \Sigma^{(k)}$, the string σx is in $\Sigma^{(k+1)}$, as constructed.

We thus have the following recurrence.

$$\begin{aligned} |\Sigma^{(0)}| &= 1 \\ |\Sigma^{(k+1)}| &= |\Sigma| \cdot |\Sigma^{(k)}| \quad \text{for } \ k \geq 0 \end{aligned}$$

Using the Master Theorem, we thus find explicitly that
For each $\ell \in \mathbb{N}$,

$$|\Sigma^{(\ell)}| \ = \ \frac{|\Sigma|^{\ell+1} \ - \ |\Sigma|}{|\Sigma| - 1} \ \leq \ c \cdot |\Sigma|^{\ell}$$

for some constant c. In order for this quantity to reach $n \in \mathbb{N}$, we must have

$$\ell \ > \ d \cdot \log_{|\Sigma|} n$$

for some small constant d. $\square$

**HERE

Focus on Say, inductively, that there are ℓ_k

Proposition 3.11 *The number of distinct strings of length k over an alphabet of a letters is a^k.*

3.3.3 *Arithmetic and geometric sequences and series*

The ability to sum – and perhaps approximate – simple series, including, *at least*, finite arithmetic series and both finite and infinite geometric series.

A. Arithmetic sequences and series. We define arithmetic sequences and learn how to calculate their sums.

$$\begin{array}{l} \text{An } n\text{-term arithmetic sequence:} \\ \quad a,\ a+b,\ a+2b,\ a+3b,\ \ldots, a+(n-1)b \\ \\ \text{The corresponding arithmetic series:} \\ \quad a+(a+b)+(a+2b)+(a+3b)+\cdots+(a+(n-1)b) \\ \qquad = \ an+b\cdot(1+2+\cdots+n-1) \end{array} \tag{3.21}$$

We can, thus, sum the arithmetic series in (3.21) by determining the sum of the first m positive integers; $m = n-1$ in (3.21). We accomplish this via a device known to Karl Friedrich Gauss as a pre-teen. Let S_m denote the desired sum.

$$\begin{array}{l} \text{Write the sum } S_m \text{ “forward”:} \\ \qquad S_m = 1 + \quad 2 \quad + \cdots + (m-1) + m \\ \text{Write } S_m \text{ in reverse:} \\ \qquad S_m = m + (m-1) + \cdots + \quad 2 \quad + 1 \end{array} \tag{3.22}$$

Now add the two versions of S_m in (3.22) *columnwise*. Because each column-sum equals $m+1$, we find that $2S_m = m(m+1)$, so that

$$S_m \ = \ 1+2+\cdots+(m-1)+m \ = \ \frac{1}{2}m(m+1) \ \stackrel{\text{def}}{=} \ \binom{m+1}{2}. \tag{3.23}$$

It follows that our original series in (3.21) sums as follows.

$$a+(a+b)+(a+2b)+(a+3b)+\cdots+(a+(n-1)b) \ = \ an+b\cdot\binom{n}{2}.$$

B. Geometric sequences and series. We define geometric sequences and learn how to calculate their sums.

$$\begin{array}{l} \text{An } n\text{-term geometric sequence:} \\ \quad a,\ ab,\ ab^2,\ \ldots, ab^{n-1} \\ \\ \text{The corresponding geometric series:} \\ \quad a+ab+ab^2+\cdots+ab^{n-1} \ = \ a(1+b+b^2+\cdots+b^{n-1}) \end{array} \tag{3.24}$$

Easily, we can sum the series in (3.24) by summing just the sub-series

$$S_b(n) \ \stackrel{\text{def}}{=} \ 1+b+b^2+\cdots+b^{n-1}. \tag{3.25}$$

We proceed as follows. Write $S_b(n)$ so that its terms are in *decreasing* order. We thereby isolate two cases.

1. Say first that $b > 1$. In this case, we write the series in the form

$$S_b^{b>1}(n) \;=\; b^{n-1} + b^{n-2} + \cdots + b^2 + b + 1,$$

and we note that

$$S_b^{b>1}(n) \;=\; b^{n-1} \;+\; \frac{1}{b} \cdot S_b^{b>1}(n) \;-\; \frac{1}{b}.$$

In other words, we have

$$\left(1 - \frac{1}{b}\right) S_b^{b>1}(n) \;=\; b^{n-1} - \frac{1}{b},$$

or equivalently,

$$S_b^{b>1}(n) \;=\; \frac{b^n - 1}{b - 1}. \tag{3.26}$$

2. Alternatively, if $b < 1$, then we write the series in the form

$$S_b^{b<1}(n) \;=\; 1 + b + b^2 + b^3 + \cdots + b^{n-1}.$$

and we note that

$$S_b^{b<1}(n) \;=\; 1 \;+\; b \cdot S_b^{b<1}(n) \;-\; b^n.$$

In other words,

$$(1 - b) S_b^{b<1}(n) \;=\; 1 \;-\; b^n$$

or equivalently,

$$S_b^{b<1}(n) \;=\; \frac{1 - b^n}{1 - b}. \tag{3.27}$$

Note that $S_b^{b>1}(n)$ and $S_b^{b<1}(n)$ actually have the same form. We have chosen to write them differently to stress their *approximate* values, which are useful in "back-of-the-envelope" calculations: For very large values of n, we have

$$S_b^{b>1}(n) \;\approx\; \frac{b^n}{b-1} \quad \text{while} \quad S_b^{b<1}(n) \;\approx\; \frac{1}{1-b}. \tag{3.28}$$

C. An amusing result about numerals. We illustrate a somewhat surprising nontrivial fact about integers that are "encoded" in their positional numerals. We hope that this "fun" result will inspire the reader to seek kindred numeral-encoded properties of numbers.

Proposition 3.12 *An integer n is divisible by an integer m if, and only if, m divides the sum of the digits in the base-$(m+1)$ numeral for n.*

The most familiar instance of this result is phrased in terms of our traditional use of base-10 (decimal) numerals.
An integer n is divisible by 9 if, and only if, the sum of the digits of n's base-10 numeral is divisible by 9.

Verification (for general base b). Of course, we lose no generality by focusing on numerals without leading 0's, for adding leading 0's does not alter a numeral's sum of digits.

To enhance legibility, let $b = m+1$, so that we are looking at the base-b numeral for n. Say that

$$n \ = \ \delta_k \cdot b^k + \delta_{k-1} \cdot b_{k-1} + \cdots + \delta_1 \cdot b + \delta_0,$$

so that the sum of the digits of n's base-b numeral is

$$s_b(n) \ \overset{\text{def}}{=} \ \delta_k + \delta_{k-1} + \cdots + \delta_1 + \delta_0.$$

We next calculate the difference $n - s_b(n)$. We proceed as follows, digit by digit.

$$\begin{array}{rcccccccc}
n & = & \delta_k \cdot b^k & + & \delta_{k-1} \cdot b^{k-1} & + \cdots + & \delta_1 \cdot b & + \ \delta_0 \\
s_b(n) & = & \delta_k & + & \delta_{k-1} & + \cdots + & \delta_1 & + \ \delta_0 \\
\hline
n - s_b(n) & = & \delta_k \cdot (b^k - 1) & + & \delta_{k-1} \cdot (b^{k-1} - 1) & + \cdots + & \delta_1 \cdot (b-1) &
\end{array} \tag{3.29}$$

We now revisit summation (3.26). Because b is a positive integer, so that $1 + b + \cdots + b^{a-2} + b^{a-1}$ is also a positive integer, we adduce from the summation that *the integer $b^a - 1$ is divisible by $b-1$.*

We are almost home. Look at the equation for $n - s_b(n)$ in the system (3.29). As we have just seen, every term on the righthand side of that equation is divisible by $b-1$. It follows therefore, that the lefthand expression, $n - s_b(n)$, is also divisible by $b-1$. An easy calculation, which we leave to the reader, now shows that this final fact means that n is divisible by $b-1$ if, and only if, $s_b(n)$ is. $\square$

3.3.4 *Linear recurrences*

By the time the reader has reached this paragraph, she has the mathematical tools necessary to prove and apply what is called *The Master Theorem for Linear Recurrences* [20]. This level of mathematical preparation should be adequate for most early-undergrad courses on data structures and algorithms, as well for for analyzing a large fraction of the algorithms that she is likely to encounter in daily activities.

Theorem 3.5 (The Master Theorem for Linear Recurrences). *Let the function F be specified by the following linear recurrence.*

$$F(n) \ = \ \begin{cases} aF(n/b) + c & \text{for } n \geq b \\ c & \text{for } n < b \end{cases} \tag{3.30}$$

Then the value of F on any argument n is given by

$$F(n) = \begin{cases} (1+\log_b n)c & \textit{if } a = 1 \\[2ex] \dfrac{1-a^{\log_b n}}{1-a} \approx \dfrac{1}{1-a} & \textit{if } a < 1 \\[2ex] \dfrac{a^{\log_b n}-1}{a-1} & \textit{if } a > 1 \end{cases} \tag{3.31}$$

Proof. In order to discern the recurring pattern in (3.30), let us begin to "expand" the specified computation by replacing occurrences of $F(\bullet)$ as mandated in (3.30).

$$\begin{array}{rcl} F(n) = & aF(n/b)+c & \\ = & a\big(aF(n/b^2)+c\big)+c & = \quad a^2F(n/b^2)+(1+a)c \\ = a^2\big(aF(n/b^3)+c\big)+(1+a)c & = & a^3F(n/b^3)+(1+a+a^2)c \\ & \vdots & \vdots \\ = & \Big(1+a+a^2+\cdots+a^{\log_b n}\Big)\,c & \end{array} \tag{3.32}$$

The segment of (3.32) "hidden" by the vertical dots betokens an induction that is left to the reader. Equations (3.26) and (3.27) now enable us to demonstrate that (3.31) is the case-structured solution to (3.30). □

3.4 Elementary counting

3.4.1 Binary Strings and Power Sets

Proposition 3.13 *For every integer $b > 1$, there are b^n b-ary strings of length n.*

Verification. The asserted numeration follows most simply by noting that there are always b times as many b-ary strings of length n as there are of length $n-1$. This is because we can form the set of b-ary strings of length n as follows. Take the set A_{n-1} of b-ary strings of length $n-1$, and make b copies of it, call them $A^{(0)}_{n-1}, A^{(1)}_{n-1}, \ldots, A^{(b-1)}_{n-1}$. Now, append 0 to every string in $A^{(0)}_{n-1}$, append 1 to every string in $A^{(1)}_{n-1}$, ..., append $\bar{b} = b-1$ to every string in $A^{(b-1)}_{n-1}$. The thus-amended sets $A^{(i)}_{n-1}$ are mutually disjoint (because of the terminal letters of their respective strings), and they collectively contain all b-ary strings of length n. □

Proposition 3.14 *The power set $\mathscr{P}(S)$ of a finite set S contain $2^{|S|}$ elements.*

Verification. Let us begin by taking an arbitrary finite set S—say of n elements—and laying its elements out in a line. We thereby establish a correspondence between S's elements and positive integers: there is the first element, which we associate with the integer 1, the second element, which we associate with the integer 2, and so on, until the last element along the line gets associated with the integer n.

Next, let's note that we can specify any subset S' of S by specifying a length-n *binary (i.e., base-*2*) string*, i.e., a string of 0's and 1's. The translation is as follows. If an element s of S appears in the subset S', then we look at the integer we have associated with s (via our linearization of S), and we set the corresponding bit-position of our binary string to 1; otherwise, we set this bit-position to 0. In this way, we get a distinct subset of S for each distinct binary string, and a distinct binary string for each distinct subset of S.

Let us pause to illustrate our correspondence between sets and strings by focussing on the set $S = \{a,b,c\}$. Just to make life more interesting, let us lay S's elements out in the order b,a,c, so that b has associated integer 1, a has associated integer 2, and c has associated integer 3. We depict the elements of $\mathscr{P}(S)$ and the corresponding binary strings in the following table.

Binary string	Set of integers	Subset of S
000	$\emptyset$	$\emptyset$
001	$\{3\}$	$\{c\}$
010	$\{2\}$	$\{a\}$
011	$\{2,3\}$	$\{a,c\}$
100	$\{1\}$	$\{b\}$
101	$\{1,3\}$	$\{b,c\}$
110	$\{1,2\}$	$\{a,b\}$
111	$\{1,2,3\}$	$\{a,b,c\} = S$

Back to the Proposition: We have verified the following: *The number of length-n binary strings is the same as the number of elements in the power set of S!* The desired numeration thus follows by the ($b = 2$) instance of Proposition 3.13. □

The binary string that we have constructed to represent each set of integers $N \subseteq \{0,1,\ldots,n-1\}$ is called the *(length-n) characteristic vector of the set N*. Of course, the finite set N has characteristic vectors of all finite lengths. Generalizing this idea, *every* set of integers $N \subseteq \mathbb{N}$, whether finite or infinite, has an *infinite* characteristic vector, which is formed in precisely the same way as are finite characteristic vectors, but now using the set $\mathbb{N}$ as the base set.

within discrete frameworks, including introducing discrete probability/likelihood as a ratio:

$$\frac{\text{number of targeted events}}{\text{number of possible events}}$$

3.5 Congruences and Modular Arithmetic

3.6 Numbers and Numerals

3.6.1 Number vs. Numeral: Object vs. Name

3.6.2 Geometric series and positional number systems

The relation between simple geometric series and numeration within positional number systems – including changing bases in such systems.

3.7 Useful Nonalgebraic Notions

3.7.1 Nonalgebraic Notions Involving Numbers

If the intended curriculum will approach more sophisticated application areas such as robotics or data science or information retrieval or data mining (of course, at levels consistent with the students' preparation), then one would do well to insist on familiarity with notions such as:

3.8 Advanced Topics

3.8.1 Measures of distance in tuple-spaces

including the following norms/metrics: L_1 (Manhattan, or, rook's-move distance), L_2 (Euclidean distance); L_∞ (King's-move distance).

3.8.2 Edit-distance: a measure of closeness in string spaces

Chapter 4
GRAPH-THEORETIC ENTITIES

4.1 Basic Graph-Theoretic Topics

Graphs provide one of the richest technical and conceptual frameworks in the world of computing. They provide concrete representations of manifold data structures, hence must be well understood in preparation for a "Data Structures and Algorithms" course. They embody tangible abstractions of relationships of all sorts, hence must be well understood in order to discuss entities as varied as web-search engines and social networks with precision and rigor. As with most of the topics we are discussing, graph-oriented concepts must be taught "in layers". All students must be conversant with the use of graphs to represent and reason about a variety of complicated relationships, but the degree of sophistication that an individual student requires depends both on the abilities of the student and the range of applications that will appear in the student's program. The fundamental concepts in this essay should be understood by all students—although each can be developed with more texture and nuance within the context of specific application domains.

Many developments in computing technology over recent decades have made it imperative that graphs no longer be viewed by students as the static objects introduced, e.g., as abstractions of data struntures. Applications ranging from databases to web-search engines to social networks demand an appreciation of graphs as dynamic objects. This change in perspective affects many aspects of the mathematical prerequisites for any CS/CE program.

4.1.1 Fundamental Notions

The basic components of graphs are *nodes/vertices* (one encounters both terms in the literature) and the *edges* that interconnect them. When a graph is *undirected*, an edge connotes some sort of sibling-like relationship among nodes of "equal" status. When a graph is *directed* (sometimes referred to as a *digraph*), an edge connotes an "unequal" relationship such as parenthood or priority or dependence; edges in directed graphs are often termed *arcs*. In many situations involving directed graphs, it is important to recognize The *dual* of a digraph G is a digraph obtained by reversing all of G's arcs. One sometimes encounters situations when arguments about, or operations on, a digraph G can be "translated" to arguments about, or operations on, G's dual with only clerical effort. A *subgraph* G' of a graph G is a graph whose nodes are a subset of G's and whose edges are a subset of G's that interconnect only nodes of G'. A *path* in an undirected graph is a sequence of nodes within which every adjacent pair is connected by an edge. A path is a *cycle* if all nodes in the sequence are distinct, except for the first and last, which are identical. Paths and cycles in directed graphs are defined similarly, except that every adjacent pair of nodes must be connected by an arc, and all arcs must "point in the same direction."

Getting formal. A *directed graph* (*digraph*, for short) $\mathcal{G}$ is given by a set of *nodes* $\mathcal{N}_{\mathcal{G}}$ and a set of *arcs* (or *directed edges*) $\mathcal{A}_{\mathcal{G}}$. Each arc has the form $(u \rightarrow v)$, where $u, v \in \mathcal{N}_{\mathcal{G}}$; we say that this arc goes *from u to v*. A *path* in $\mathcal{G}$ is a sequence of arcs that share adjacent endpoints, as in the following path from node u_1 to node u_n:

$$(u_1 \rightarrow u_2),\ (u_2 \rightarrow u_3),\ \ldots,\ (u_{n-2} \rightarrow u_{n-1}),\ (u_{n-1} \rightarrow u_n). \tag{4.1}$$

It is sometimes useful to endow the arcs of a digraph with labels from an alphabet Σ. When so endowed, the path (4.1) would be written

$$(u_1 \overset{\lambda_1}{\rightarrow} u_2),\ (u_2 \overset{\lambda_2}{\rightarrow} u_3),\ \ldots,\ (u_{n-2} \overset{\lambda_{n-2}}{\rightarrow} u_{n-1}),\ (u_{n-1} \overset{\lambda_{n-1}}{\rightarrow} u_n),$$

where the λ_i denote symbols from Σ. If $u_1 = u_n$, then we call the preceding path a *cycle*.

An *undirected graph* is obtained from a directed graph by removing the directionality of the arcs; the thus-beheaded arcs are called *edges*. Whereas we say:

the *arc* (u, v) goes *from* node u *to* node v

we say:

the undirected edge $\{u, v\}$ goes *between* nodes u and v

or, more simply:

the undirected edge $\{u, v\}$*connects* nodes u and v.

Undirected graphs are usually the default concept, in the following sense: *When $\mathcal{G}$ is described as a "graph," with no qualifier "directed" or "undirected," it is understood that $\mathcal{G}$ is an undirected graph.*

4.1.2 Graph evolution and decomposition

Any course on algorithms will discuss graphs, especially trees, that evolve over time. Such structures arise in the study of "classical" algorithmic topics such minimum-spanning-tree and branch-and-bound algorithms, as well as in the study of more modern topics such as social networks and internetworks. For the most part, the mathematics already appearing in this essay suffices for these studies: the students must assimilate new algorithmic notions, not new mathematics. That said, students will be challenged to utilize the mathematical devices that they have (hopefully) mastered in new, more sophisticated, ways. Instructors who wish to lead their students to even a casual understanding of emerging modalities and platforms for computing are thereby challenged to teach required mathematical preliminaries using exemplars that include these new modalities and exemplars.

A fundamental variety of relevant notions within the study of graphs reside in the notions known in various contexts via terms such as *graph separators* or *graph bisection*. The key idea that underlies these notions is that certain graph-theoretic structures interconnect their graphs' constituent nodes more or less densely — and the type and level of interconnectivity has important algorithmic consequences. In such situations, the student must understand how the phenomenon/a modeled by the graphs of interest are elucidated by the way a graph can be broken into subgraphs by excising nodes or edges. When discussing communication-related structures, for instance, graph are often used to model the individual pairwise communications that must occur in order to accomplish the desired overall communication (such as a broadcast). There is often a provable tradeoff between the number of such pairwise communications and the overall time for the completion of the overall task. As another example, when discussing social networks, one can pose questions such as: which node in a network is best to connect to (when joining the network) in order to best facilitate one's interactions or influence within the community. The latter topic leads, e.g., to the study of *power-law* networks, a topic that would not be studied in depth in any early course; indeed, the structure of these networks is not yet well understood even in advanced settings.

4.1.3 Trees

Within the world of graphs, trees occupy a place of honor. In their undirected form, the are defined by the proerty of containing no cycles. They then provide the minimal interconnection structure that provides a path connecting every pair of nodes. Of course, trees appear also in directed form. Among directed trees, a very important subclass are those that have a *root*, i.e., a node that has a directed path to every other node of the tree. In applications, rooted trees represent hierarchical organizations (as do families when viewed generationally).

More formally: *rooted trees* are a class of *acyclic* digraphs. Paths in trees that start at the root are often called *branches*. The *acyclicity* of a tree $\mathscr{T}$ means that for any branch of $\mathscr{T}$ of the form (4.1), we cannot have $u_1 = u_n$, for this would create a cycle. Each rooted tree $\mathscr{T}$ has a designated *root node* $u_n \in \mathscr{N}_{\mathscr{T}}$ that resides at the end of a branch (4.1) that starts at $r_{\mathscr{T}}$ (so $u_1 = r_{\mathscr{T}}$) is said to reside at *depth* $n-1$ in $\mathscr{T}$; by convention, $r_{\mathscr{T}}$ is said to reside at depth 0. $\mathscr{T}$'s root $r_{\mathscr{T}}$ has some number (possibly 0) of arcs that go from $r_{\mathscr{T}}$ to its *children,* each of which thus resides at depth 1 in $\mathscr{T}$; in turn, each child has some number of arcs (possibly 0) to its children, and so on. (Think of a family tree.) For each arc $(u \to v) \in A_{\mathscr{T}}$, we call u a *parent* of v, and v a *child* of u, in $\mathscr{T}$; clearly, the depth of each child is one greater than the depth of its parent. Every node of $\mathscr{T}$ except for $r_{\mathscr{T}}$ has precisely one parent; $r_{\mathscr{T}}$ has no parents. A childless node of a tree is a *leaf*. The transitive extensions of the parent and child relations are, respectively, the *ancestor* and *descendant* relations. The *degree* of a node v in a tree is the number of children that the node has, call it c_v. If every nonleaf node in a tree has the same degree c, then we call c the *degree of the tree*.

4.2 Hypergraphs: an advanced topic

When studying certain computing-related topics, one will need a generalization of graphs called *hypergraphs*. A hypergraph has nodes that are analogous to the nodes of a graph, but in place of edges a hypergraph has *hyperedges*. Each hyperedge is a set of nodes whose size is not restricted to 2; thus, hypergraphs represent internode relationships that are not "binary". A simple example can be framed by describing *bus-connected* systems, such as occur in certain genres of digital circuits and certain message-passing systems. The underlying idea is that nodes that coreside in a hyperedge represent agents that "hear" all messages simultaneously, or equipotential points in a network. Because of their inherent complexity, hypergraphs as graph-theoretic objects are usually relegated to advanced courses, but specific concrete instantiations, exemplified by bus-oriented communication and digital circuits should be accessible even to early students.

Chapter 5
ASYMPTOTICS

Asymptotics can be viewed as a language and a system of reasoning that allow one to talk in a *qualitative* voice about *quantitative* topics. We thereby generalize to arbitrary growth functions terms such as “linear”, “quadratic”, “exponential”, and “logarithmic”.

Such a language and system are indispensable if one needs to reason about computational topics over a range of situations, such as a range (“all existing”?) computer architectures and software systems. As two simple examples: (1) Carry-ripple adders perform additions in time linear in the lengths n of the summands (measured in number of bits) no matter what these lengths are. (2) Comparison-based sorting algorithms can sort lists of n keys in time proportional to $n\log n$, but no faster—where the base of the logarithm depends on the characteristics of the computing platform. More precise versions of the preceding statements require specication of the number n and other details, possibly down to the clock speeds of the host computer’s circuitry.

5.1 The language of asymptotics

The language of asymptotics, which has its origins in the field of Number Theory in the late 19th century, builds on the following terminology, which is likely what one would cover in an early undergraduate course. More advanced aspects of the language would likely by beyond the needs of most students of computing, aside from specialists in advanced courses. The basics of the language build on three primitive notations and notions. Standard sources, such as any text on algorithm design and analysis, flesh out the following ideas.

- *The big-O notation*. The assertion $f(x) = O(g(x))$ says, intuitively, that the function f grows no faster than function g. It is, thus, the asymptotic analogue of “less than”.
  Formally: $f(x) = O(g(x))$
  means

$(\exists c > 0)(\exists x^{\#})(\forall x > x^{\#})[f(x) \leq c \cdot g(x)]$

- *The big-Ω notation.* The assertion $f(x) = \Omega(g(x))$ says, intuitively, that the function f grows at least as fast as function g. It is, thus, the asymptotic analogue of "greater than".
  Formally: $f(x) = \Omega(g(x))$
  means
  $(\exists c > 0)(\exists x^{\#})(\forall x > x^{\#})[f(x) \geq c \cdot g(x)]$

- *The big-Θ notation.* The assertion $f(n) = \Theta(g(n))$ says, intuitively, that the function f grows at the same rate as does function g. It is, thus, the asymptotic analogue of "equal to".
  Formally: $f(x) = \Theta(g(x))$
  means
  $(\exists c_1 > 0)(\exists c_2 > 0)(\exists x^{\#})(\forall x > x^{\#})[c_1 \cdot g(x) \leq f(x) \leq c_2 \cdot g(x)]$

One renders the preceding intuitive explanations precise by pointing out that the three specifies relations (*a*) take hold *eventually*, i.e., only for large arguments to the functions f and g, and (*b*) hold up to an unspecified constant of proportionality.

5.2 The "uncertainties" in asymptotic relationships

The formal definitions of all three of our asymptotic relationships are bracketed by two important quantifiers:

$$\text{“}(\exists c > 0)'' \quad \text{and “}(\forall x > x^{\#})''.$$

The former, *uncertain-size* quantifier, asserts that asymptotic notions describe functional behavior "in the large". Thus, in common with more common qualitative descriptors of quantitative growth such as linear, quadratic, cubic, quartic, exponential, logarithmic, etc., asymptotic relationships give no infomation about constants of proportionality. *We are not saying that constant factors do not matter! We are, rather, saying that we want to discuss growth patterns $\underline{\text{in the large}}$.*

The latter, *uncertain-time* quantifier asserts that asymptotic relationships between functions are promised to hold only "eventually", i.e., "for sufficiently large values of the argument x". Therefore, in particular, asymptotic notions cannot be employed to discuss or analyze quantities that can never grow beyond a fixed finite value. The fact that all instances of a quantity throughout history have been below N is immaterial, as long as it is conceivable that an instance larger than N could appear at some time in the future.

These quantifiers in particular distinguishes claims of asymptotic relationship from the more familiar definite inequalities such as "$f(x) \leq g(x)$" or $f(x) \geq 7 \cdot g(x)$. In fact, it is often easier to think about our three asymptotic bounding assertions as establishing *envelopes* for $f(x)$:

- Say that $f(x) = O(g(x))$. If one draws the graphs of the functions $f(x)$ and $c \cdot g(x)$, then as one traces the graphs with increasing values of x, one eventually reaches a point $x^{\#}$ beyond which the graph of $f(x)$ never enters the territory *above* the graph of $c \cdot g(x)$.
- Say that $f(x) = \Omega(g(x))$. This situation is the up-down mirror image of the preceding one: just replace the highlighted "*above*" with "*below*."
- Say that $f(x) = \Theta(g(x))$. We now have a two-sided envelope: beyond $x^{\#}$, the graph of $f(x)$ never enters the territory *above* the graph of $c_1 \cdot g(x)$ and never enters the territory *below* the graph of $c_2 \cdot g(x)$.

In addition to allowing one to make familiar growth-rate comparisons such as "$n^{14} = O(n^{15})$" and "$1.001^n = \Omega(n^{1000})$," we can now also make assertions such as "$\sin x = \Theta(1)$," which are much clumsier to explain in words.
Beyond the big letters. There are "small"-letter analogues of the preceding "big"-letter asymptotic notations, but they are only rarely encountered in discourse about real computations (although they do arise in the analysis of algorithms).

5.3 Inescapable complications

The story we have told thus far is covered in many sources and courses. Two complications to the story are covered less faithfully, although lacking them, one cannot perform cogent asymptotic reasoning. Both complications involve the notion of *uniformity*.
1. *Multiple functions*. Say that we have four functions, f, g, h, k, and we know that both

$$f(n) = O(g(n)) \quad \text{and} \quad h(n) = O(k(n))$$

It is intuitive that

$$f(n) + h(n) = O(g(n) + k(n))$$

— but is it true?

In short, the answer is YES, but verifying that requires a bit of subtlety, because, absent hitherto undisclosed information, the proportionality constants $c_{f,g}$ and $N_{f,g}$ that witness the big-O relationship between functions f and g have no connection with the constants, $c_{h,k}, N_{h,k}$ that witness the analogous relationship between functions h and k. Therefore, in order to verify the posited relationship between functions $f + h$ and $g + k$, one much find witnessing constants $c_{f+h,g+k}$ and $N_{f+h,g+k}$. Of course, this task requires only elementary reasoning and manipulation — but it must be done!

2. *Multivariate functions*. Finally, we discuss the scenario that almost automatically accompanies the transition from a focus on sequential, single-agent computing to a focus on PDC. Within this broadened context, most functions that describe a system have one or more variables that describe the computing system — its number of processors or of agents or the sizes of its memory modules or the communication-radii of its transponders or ..., in addition to the one or more variables that describe the data that the system is processing. Within such scenarios, every assertion of an asymptotic relationship, of the form

$$f(\mathbf{m};\mathbf{n}) = O(g(\mathbf{m};\mathbf{n}))$$

must explicitly specify the following information:

- which variables can grow without bound;
- among such unbounded variables, which participate in the posited asymptotic relation;
- for each participating unbounded variable x, what are the constants c_x and N_x that witness the posited asymptotic relationship(s).

Clearly the complexity of cogent asymptotic reasoning — hence also the complexity of teaching about such reasoning — gets much more complicated in the multivariate settings engendered by PDC. But, the benefits of being able to reason qualitatively about the quantitative aspects of computing increase at least commensurately!

Chapter 6
PROBABILITY AND STATISTICS

Elements of probability theory and statistics infuse every area of computing. The practicality of many algorithms that are experientially the most efficient for their target tasks depend on the *distribution* of inputs in "real" situations. Design methodologies for crucial complex circuits must acknowledge the *mean times to failure* of the critical components of the circuits. Sophisticated searching algorithms must take into account the relative *likelihoods* of finding one's goal in the various optional search directions. Analyzing and understanding large corpora of data requires a variety of methodologies that build on the concepts of *clustering* and/or *decomposition*.

A student needs at least an introduction to the foundations of probability and statistics to even understand, all the moreso to master, the terms highlighted in the preceding paragraph. We outline many of the key concepts that a student must be exposed to in the following subsections.

6.1 The Elements of Combinatorial Probability

Perhaps the easiest and most engaging way to introduce "probability via counting" is by calculating the comparative likelihoods of various deals in 5-card poker and of various rolls of a pair of dice. The arithmetic required for this discussion is elementary and the "application" to gambling of interest even to non-gamblers: "Why is such a deal in poker (say, a straight) worth more than another (say, three of a kind)?" One can also introduce in this setting concepts such as randomness, bias, etc., that are so important in the design of experiments and the analysis of their outcomes.

6.2 Toward a Basic Understanding of Statistics

Most students whose interest tend to the empirical will likely "do" statistics with the aid of apps, rather than by explicitly writing programs that perform the required calculations. That said, all students should understand the crucial notion of *random variable* and should be conversant with the most common statistical distributions. "Conversant" in this context should include complete understandings of the (low-numbered) moments of *at least* the *uniform* and *exponential* distributions. They should know how to compute, say, the means and variances of various distributions and, most importantly, they should *understand* the sense in which the variance of a distribution give *important* information that is not available from the mean. All of this is prerequisite to rigor in experimentation.

6.2.0.1 The Elements of Empirical Reasoning

Empirical reasoning does not convey the certitude that formal reasoning does. Students should understand how to craft experiments in a way that collects the "right" data. The should then be able—perhaps just with statistical packages—to interpret the results they collect and to understand what conclusions are justifiable. *It is essential that all students understand the ditinction between* positive correlation *and* causation*!* (Most of the public would seem to flunk that test.)

In order to satisfy the preceding demands, students should understand enough about statistics—including definitions and meanings related to distributions and their moments—to understand what conclusions can be made based on experimental results, and to understand how to describe conclusions in a way that is supported by the statistics.

6.3 Beyond the Basics

As students are introduced to modern topics within computing, whether at the level of a Computing Literacy course or a post-core technical course, they will have to master a variety of more specialized topics that combine pieces of the elements we have discussed in this essay. While these topics are beyond the level of generality aimed at in this essay, some may be appropriate prerequisites to programs that have some specialized foci.

- Issues relating to *clustering* find application in applicatios as diverse as: *linear-algebraic computations*, *data mining*, *design and layout of digital circuitry*.
- Issues building on *graph separation/decomposition* are encountered when studying: *linear-algebraic computing*, *fault-tolerant design*, *load balancing*.
- Many issues relating to *fault/failure tolerance* and *data analytics* benefit from study using *random walks* (at least in one dimension).

- Many useful ideas regarding the *encoding and manipulation of data* can be gleaned from the elements of *information theory* and *computer arithmetic*.

The preceding list is really endless. Hopefully readers will be inspired by our few examples to compile a longer version that is appropriate for their particular environments.

References

1. F.E. Allen and J. Cocke (1976): A program data flow analysis procedure. *Comm. ACM 19* 137–147.
2. D.N. Arden (1960): *Theory of Computing Machine Design*. Univ. Michigan Press, Ann Arbor, pp. 1–35.
3. E.T. Bell (1986): *Men of Mathematics*. Simon and Schuster, New York.
4. F. Bernstein (1905): Untersuchungen aus der Mengenlehre. *Math. Ann. 61*, 117–155.
5. G. Birkhoff and S. Mac Lane (1953): *A Survey of Modern Algebra*, Macmillan, New York.
6. E. Bishop (1967): *Foundations of Constructive Analysis*, McGraw Hill, New York.
7. W.W. Boone, F.B. Cannonito, R.C. Lyndon (1973): *Word Problems: Decision Problem in Group Theory*, North-Holland, Amsterdam.
8. R. Buyya, D. Abramson, J. Giddy (2001): A case for economy Grid architecture for service oriented Grid computing. *10th Heterogeneous Computing Wkshp.*
9. G. Cantor (1874): Über eine Eigenschaft des Inbegriffes aller reellen algebraischen Zahlen. *J. Reine und Angew. Math. 77*, 258–262.
10. G. Cantor (1878): Ein Beitrag zur Begründung der transfiniter Mengenlehre. *J. Reine Angew. Math. 84*, 242–258.
11. A.L. Cauchy (1821): *Cours d'analyse de l'École Royale Polytechnique, 1ère partie: Analyse algébrique*. l'Imprimerie Royale, Paris. Reprinted: Wissenschaftliche Buchgesellschaft, Darmstadt, 1968.
12. N. Chomsky (1956): Three models for the description of language. *IRE Trans. Information Theory 2*, 113–124.
13. N. Chomsky (1959): On certain formal properties of grammars. *Inform. Contr. 2*, 137–167.
14. A. Church (1941): *The Calculi of Lambda-Conversion. Annals of Math. Studies 6*, Princeton Univ. Press, Princeton, NJ.
15. A. Church (1944): *Introduction to Mathematical Logic, Part I. Annals of Math. Studies 13*, Princeton Univ. Press, Princeton, NJ.
16. W. Cirne and K. Marzullo (1999): The Computational Co-Op: gathering clusters into a metacomputer. *13th Intl. Parallel Processing Symp.*, 160–166.
17. E.F. Codd (1970): *A relational model of data for large shared data banks*. *Comm. ACM 13*, 377–387.
18. S.A. Cook (1971): The complexity of theorem-proving procedures. *ACM Symp. on Theory of Computing*, 151–158.
19. I.M. Copi, C.C. Elgot, J.B. Wright (1958): Realization of events by logical nets. *J. ACM 5*, 181–196.
20. T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein (2001): *Introduction to Algorithms (2nd ed.)*. MIT Press, Cambridge, MA.
21. H.B. Curry (1934): Some properties of equality and implication in combinatory logic. *Annals of Mathematics, 35*, 849–850.

22. H.B. Curry, R. Feys, W. Craig (1958): *Combinatory Logic. Studies in logic and the foundations of mathematics*. North-Holland, Amsterdam.
23. M. Davis (1958): *Computability and Unsolvability.* McGraw-Hill, New York.
24. J.R. Douceur (2002): The Sybil attack. *1st Intl. Wkshp. on Peer-to-Peer Systems*.
25. R.W. Floyd (1962): Algorithm 97: Shortest Path. *Comm. ACM 5*, 345.
26. R.W. Floyd (1967): Assigning meanings to programs. *Proc. Symposia in Applied Mathematics 19*, 19–32.
27. R. Fueter and G. Pólya (1923): Rationale Abzählung der Gitterpunkte. *Vierteljschr. Naturforsch. Ges. Zürich 58*, 380–386.
28. M.R. Garey and D.S. Johnson (1979): *Computers and Intractability*. W.H. Freeman and Co., San Francisco.
29. A. Gilat (2004): *MATLAB: An Introduction with Applications (2nd ed.)*. J. Wiley & Sons, New York.
30. K. Gödel (1931): Über Formal Unentscheidbare Sätze der Principia Mathematica und Verwandter Systeme, I. *Monatshefte für Mathematik u. Physik 38*, 173–198.
31. B. Goldberg (1996): Functional programming languages, *ACM Computing Surveys 28*, 249–251.
32. O. Goldreich (2006): On teaching the basics of complexity theory. In *Theoretical Computer Science: Essays in Memory of Shimon Even. Springer Festschrift series, Lecture Notes in Computer Science 3895*, Springer, Heidelberg.
33. G.H. Golub, C.F. Van Loan (1996): *Matrix Computations* (3rd ed.) Johns Hopkins Press, Baltimore.
34. P.R. Halmos (1960): *Naive Set Theory*. D. Van Nostrand, New York.
35. D. Harel (1987): *Algorithmics: The Spirit of Computing*. Addison-Wesley, Reading, MA.
36. J. Hartmanis and R.E. Stearns (1966): *Algebraic Structure Theory of Sequential Machines*. Prentice Hall, Englewood Cliffs, NJ.
37. L.S. Heath, F.T. Leighton A.L. Rosenberg (1992): "Comparing queues and stacks as mechanisms for laying out graphs." *SIAM J. Discr. Math. 5*, 398–412.
38. F.C. Hennie (1966): On-line Turing machine computations. *IEEE Trans. Electronic Computers, EC-15*, 35–44.
39. F.C. Hennie and R.E. Stearns (1966): Two-tape simulation of multitape Turing machines. *J. ACM 13*, 533–546.
40. S. Homer and A.L. Selman (2001): *Computability and Complexity Theory*. Springer, New York
41. J.E. Hopcroft, R. Motwani, J.D. Ullman (2001): *Introduction to Automata Theory, Languages, and Computation* (2nd ed.). Addison-Wesley, Reading, MA.
42. J.E. Hopcroft and J.D. Ullman (1979): *Introduction to Automata Theory, Languages, and Computation* (1st ed.) Addison-Wesley, Reading, MA.
43. N. Immerman (1988): Nondeterministic space is closed under complementation. *SIAM J. Comput. 17*, 935–938.
44. *The Intel Philanthropic Peer-to-Peer program.* ⟨`www.intel.com/cure`⟩.
45. K.E. Iverson (1962): *A Programming Language*. J. Wiley & Sons, New York.
46. J. Jaffe (1978): A necessary and sufficient pumping lemma for regular languages. *SIGACT News*, 48–49.
47. R.M. Karp (1967): Some bounds on the storage requirements of sequential machines and Turing machines. *J. ACM 14*, 478–489.
48. R.M. Karp (1972): Reducibility among combinatorial problems. In *Complexity of Computer Computations* (R.E. Miller and J.W. Thatcher, eds.) Plenum Press, NY, pp. 85–103.
49. S.C. Kleene (1936): General recursive functions of natural numbers. *Math. Annalen 112*, 727–742.
50. S.C. Kleene (1952): *Introduction to Metamathematics.* D. Van Nostrand, Princeton, NJ.
51. S.C. Kleene (1956): Realization of events in nerve nets and finite automata. In *Automata Studies* (C.E. Shannon and J. McCarthy, Eds.) *[Ann. Math. Studies 34]*, Princeton Univ. Press, Princeton, NJ, pp. 3–42.

52. D. König (1936): *Theorie der endlichen und unendlichen Graphen.* Lipzig: Akad. Verlag.
53. D.E. Knuth (1973): *The Art of Computer Programming: Fundamental Algorithms* (2nd ed.) Addison-Wesley, Reading, MA.
54. E. Korpela, D. Werthimer, D. Anderson, J. Cobb, M. Lebofsky (2000): SETI@home: massively distributed computing for SETI. In *Computing in Science and Engineering* (P.F. Dubois, Ed.) IEEE Computer Soc. Press, Los Alamitos, CA.
55. P.J Landin (1964): The mechanical evaluation of expressions. *Computer J. 6*, 308–320.
56. L. Levin (1973): Universal search problems. *Problemy Peredachi Informatsii 9*, 265–266. Translated in, B.A. Trakhtenbrot (1984): A survey of Russian approaches to perebor (brute-force search) algorithms. *Annals of the History of Computing 6*, 384–400.
57. J.S. Lew and A.L. Rosenberg (1978): Polynomial indexing of integer lattices, I. *J. Number Th. 10*, 192–214.
58. J.S. Lew and A.L. Rosenberg (1978): Polynomial indexing of integer lattices, II. *J. Number Th. 10*, 215–243.
59. H.R. Lewis and C.H. Papadimitriou (1981): *Elements of the Theory of Computation*. Prentice-Hall, Englewood Cliffs, NJ.
60. P. Linz (2001): *An Introduction to Formal Languages and Automata* (3rd ed.) Jones and Bartlett Publ., Sudbury, MA.
61. A.A. Markov (1949): On the representation of recursive functions (in Russian). *Izvestiya Akademii Nauk S.S.S.R. 13*. English translation: Translation 54, Amer. Math. Soc., 1951.
62. W.S. McCulloch and W.H Pitts (1943): A logical calculus of the ideas immanent in nervous activity. *Bull. Mathematical Biophysics 5*, 115–133.
63. R. McNaughton and H. Yamada (1964): Regular expressions and state graphs for automata. In *Sequential Machines: Selected Papers* (E.F. Moore, ed.) Addison-Wesley, Reading, MA, PP. 157–176.
64. G.H. Mealy (1955): A method for synthesizing sequential circuits. *Bell Syst. Tech. J. 34*, 1045–1079.
65. K. Mehlhorn (1984): *Data Structures and Algorithms 2: Graph Algorithms and NP-Completeness.* Springer-Verlag, Berlin.
66. C.F. Miller (1991): Decision problems for groups – survey and reflections. In *Algorithms and Classification in Combinatorial Group Theory*, Springer, New York, pp. 1–60.
67. M. Minsky (1967): *Computation: Finite and Infinite Machines.* Prentice-Hall, Inc., Englewood Cliffs, NJ.
68. E.F. Moore (1956): Gendanken experiments on sequential machines. In *Automata Studies* (C.E. Shannon and J. McCarthy, eds.) *[Ann. Math. Studies 34]*, Princeton Univ. Press, Princeton, NJ, pp. 129–153.
69. B.M. Moret (1997): *The Theory of Computation*. Addison-Wesley, Reading, MA.
70. J. Myhill (1957): Finite automata and the representation of events. WADD TR-57-624, Wright Patterson AFB, Ohio, pp. 112–137.
71. T. Naur (2006): Letter to the Editor. *Comm. ACM 49*, 13.
72. A. Nerode (1958): Linear automaton transformations. *Proc. AMS 9*, 541–544.
73. I. Niven and H.S. Zuckerman (1980): *An Introduction to the Theory of Numbers.* (4th ed.) J. Wiley & Sons, New York.
74. *The Olson Laboratory Fight AIDS@Home project.* ⟨`www.fightaidsathome.org`⟩.
75. G.H. Ott, N.H. Feinstein (1961): Design of sequential machines from their regular expressions. *J. ACM 8*, 585–600.
76. C.H. Papadimitriou (1994): *Computational Complexity*. Addison-Wesley, Reading, MA.
77. M.O. Rabin (1963): Probabilistic automata. *Inform. Control 6*, 230–245.
78. M.O. Rabin (1964): The word problem for groups. *J. Symbolic Logic 29*, 205–206.
79. M.O. Rabin and D. Scott (1959): Finite automata and their decision problems. *IBM J. Res. Develop. 3*, 114–125.
80. H. Rogers, Jr. (1967): *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, New York. Reprinted in 1987 by MIT Press, Cambridge, MA.
81. A.L. Rosenberg (1971): Data graphs and addressing schemes. *J. CSS 5*, 193–238.

82. A.L. Rosenberg (1974): Allocating storage for extendible arrays. *J. ACM 21*, 652–670.
83. A.L. Rosenberg (1975): Managing storage for extendible arrays. *SIAM J. Comput. 4*, 287–306.
84. A.L. Rosenberg (2003): Accountable Web-computing. *IEEE Trans. Parallel and Distr. Systs. 14*, 97–106.
85. A.L. Rosenberg (2003): Efficient pairing functions—and why you should care. *Intl. J. Foundations of Computer Science 14*, 3–17.
86. A.L. Rosenberg (2006): State. In *Theoretical Computer Science: Essays in Memory of Shimon Even* (O. Goldreich, A.L. Rosenberg, A. Selman, eds.) *Springer Festschrift series, Lecture Notes in Computer Science 3895*, Springer, Heidelberg, pp. 375–398.
87. A.L. Rosenberg (2009): *The Pillars of Computation Theory: State, Encoding, Nondeterminism*. Universitext Series, Springer, New York
88. A.L. Rosenberg and L.S. Heath (2001): *Graph Separators, with Applications*. Kluwer Academic/Plenum Publishers, New York.
89. A.L. Rosenberg and L.J. Stockmeyer (1977): Hashing schemes for extendible arrays. *J. ACM 24*, 199–221.
90. J.B. Rosser (1953): *Logic for Mathematicians.* McGraw-Hill, New York.
91. *The RSA Factoring by Web Project.* ⟨`http://www.npac.syr.edu/factoring`⟩ (with Foreword by A. Lenstra). Northeast Parallel Architecture Center.
92. B. Russell, Bertrand (1903). Principles of Mathematics. Cambridge: Cambridge University Press.
93. W. Savitch (1969): Deterministic simulation of non-deterministic Turing machines. *1st ACM Symp. on Theory of Computing*, 247–248.
94. M. Schönfinkel (1924): Über die Bausteine der mathematischen Logik. *Math. Annalen 92*, 305–316.
95. A. Schönhage (1980): Storage modification machines. *SIAM J. Computing 9*, 490–508.
96. E. Schröder (1898): Über zwei Definitionen der Endlichkeit und G. Cantor'sche Sätze. *Nova Acta Academiae Caesareae Leopoldino-Carolinae (Halle a.d. Saale) 71*, 303–362.
97. E. Schröder (1898): Die selbständige Definition der Mächtigkeiten 0, 1, 2, 3 und die explicite Gleichzahligkeitsbedingung. *Nova Acta Academiae Caesareae Leopoldino-Carolinae (Halle a.d. Saale) 71*, 365–376.
98. M. Sipser (1997): *Introduction to the Theory of Computation*. PWS Publishing, Boston, MA.
99. R.E. Stearns, J. Hartmanis, P.M. Lewis, II (1972): Hierarchies of memory limited computations. *J. Symbolic Logic 37*, 624–625.
100. L.J. Stockmeyer (1973): Extendible array realizations with additive traversal. IBM Research Report RC-4578.
101. R. Szelepcsényi (1987): The method of forcing for nondeterministic automata. *Bull. EATCS 33*, 96–100.
102. R.E. Tarjan (1972): Sorting using networks of queues and stacks. *J. ACM 19*, 341–346.
103. M. Taufer, D. Anderson, P. Cicotti, C.L. Brooks (2005): Homogeneous redundancy: A technique to ensure integrity of molecular simulation results using public computing. *19th Intl. Parallel and Distributed Processing Symp.*
104. R.P. Tewarson (1973): *Sparse Matrices*. In *Mathematics in Science & Engineering*. Academic Press, New York.
105. A.M. Turing (1936): On computable numbers, with an application to the Entscheidungsproblem. *Proc. London Math. Soc.* (ser. 2, vol. 42) 230–265; Correction *ibid.* (vol. 43) 544–546.
106. S. Warshall (1962): A theorem on Boolean matrices. *J. ACM 9*, 11-12.
107. Wikipedia: The Free Encyclopedia (2005):
`http://en.wikipedia.org/wiki/Pumping_lemma`
108. R. Zach (1999): Completeness before Post: Bernays, Hilbert, and the development of propositional logic. *Bull. Symbolic Logic 5*, 331–366.

Index