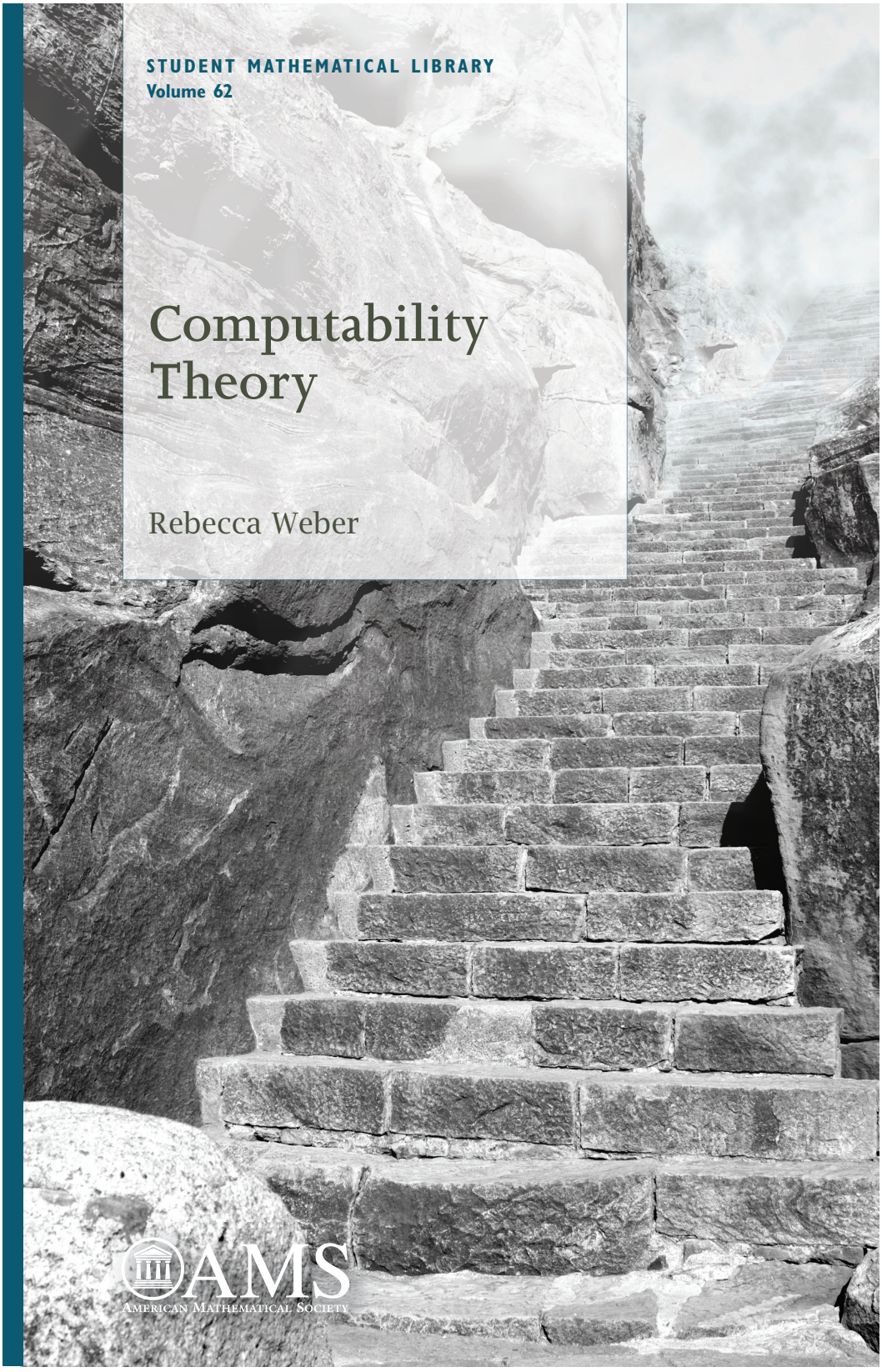

STUDENT MATHEMATICAL LIBRARY
Volume 62
Computability Theory
Rebecca Weber
AMS
American Mathematical Society

Chapter 1

Introduction

The bird's-eye view of computability: what does it mean, how does it work, where did it come from?

1.1. Approach

If I can program my computer to execute a function, to take any input and give me the correct output, then that function should certainly be called computable. That set of functions is not very large, though; I am not a particularly skilled programmer and my computer is nothing spectacular. I might expand my definition to say if a wizardly programmer can program the best computer there is to execute a function, that function is computable. However, programming languages and environments improve with time, making it easier to program complicated functions, and computers increase in processing speed and amount of working memory.

The idea of "computable" as "programmable" provides excellent intuition, but the precise mathematical definition needs to be an idealized version that does not change with hardware advances. It should capture all of the functions that may be automated even in theory.

To that end, computability theory removes any restriction on time and memory except that a successful computation must use only finitely much of each. When that change is made, the rest of the

details are fairly flexible: with sufficient memory, even the simple language of a programmable calculator is sufficient to implement all computable functions. An explicit definition is in §3.2.

Most computable functions, by this definition, are completely infeasible today, requiring centuries to run and more memory than currently exists on the planet. Why should we include them? One reason is that if we require feasibility, we have to draw a line between feasible and infeasible, and it is unclear where we ought to draw that line. A related reason is that if we do not require feasibility, and then we prove a function is noncomputable, the result is as strong as possible: no matter how much computing technology improves, the function can never be automated. A major historical example of such a proof is described in the next section, and more are in §4.5, from a variety of mathematical fields. These unsolvable problems are often simple statements about basic properties, so intuition may suggest they should be easy to compute.

A rigorous and general definition of computability also allows us to pin down concepts that are *necessarily* noncomputable. One example is randomness, which will be explored in more depth in §9.2. If a fair coin is flipped repeatedly, we do not expect to be able to win a lot of money betting on the outcome of each flip in order. We may get lucky sometimes, but in the long run whatever betting strategy we use should be essentially equivalent in success to betting heads every time: a fifty percent hit rate. That is an intuitive notion of what it means for the flip outcomes to be random. To turn it into mathematics we might represent betting strategies as functions, and say that a sequence of flips is random if no betting strategy is significantly better than choosing heads every time. However, for any given sequence of coin flips C, there is a function that bets correctly on every flip of C. If the class of functions considered to be betting strategies is unrestricted, no sequences will be random, but intuition also says *most* sequences will be random. One restriction we can choose is to require that the betting function be computable; this ties into the idea that the computable functions are those to which we have some kind of "access." If we could never hope to implement a function, it is difficult to argue that it allows us to predict coin flips.

I would like to highlight a bit of the mindset of computability theory here. Computability theorists pay close attention to *uniformity*. A nonuniform proof of computability proves some program exists to compute the function, and typically has some ad hoc component. A uniform proof explicitly builds such a program. A common example is showing explicitly that there is a program P that computes $f(n)$ provided $n \geq N$ for some fixed, finite N, and considering f computable despite the fact that P may completely fail at finding $f(n)$ for $0 \leq n < N$ (in computability theory, functions are on the natural numbers; see §3.4 for why that is not as stringent a restriction as it may seem). P is sufficient because for any given sequence of N numbers, there is a program Q that assigns those numbers in order as $f(0)$ through $f(N-1)$, and uses P to find the output for larger numbers. The function we want to compute must have *some* fixed sequence of outputs for the first N inputs, and hence via the appropriate Q it may be computed. It is computable, although we would need to magically (that is, nonuniformly) know the first N outputs to have the full program explicitly. Nonuniformity in isolation is not a problem, but it reduces the possible uses of the result. We will discuss uniformity from time to time as more examples come up.

Computability theorists more often work in the realm of the noncomputable than the computable, via approximations and partial computations. Programs that do not always give an output are seen regularly; on certain inputs such a program may just chug and chug and never finish. Of course, the program either gives an output or doesn't, and in many areas of mathematics we would be able to say "if the program halts, use the output in this way; if not, do this other computation." In fact we *could* do that in computability theory, but the question of whether any given program halts is a noncomputable one (see §4.1). Typically we want to complete our constructions using as little computational power as we can get away with, because that allows us to make stronger statements about the computability or noncomputability of the function we have built.

To succeed in such an environment, the construction must continue while computations are unfinished, working with incomplete and possibly incorrect assumptions. Mistakes will be made and need

to be repaired. The standard means of dealing with this situation is a *priority argument* (§6.2), which breaks the goals of the construction into small pieces and orders them. A piece (*requirement*) that is earlier in the ordering has higher priority and gets to do what appears at the moment to satisfy its goal even if that is harmful to later requirements. When done correctly, each requirement can be met at a finite stage and cease harming the lower-priority requirements, and each requirement can recover from the damage it sustains from the finitely many higher-priority requirements. Satisfaction of requirements cascades irregularly from the beginning of the order down.

1.2. Some History

The more early work in computability theory you read, the more it seems its founding was an inevitability. There was a push to make mathematics formal and rigorous, and find mechanical methods to solve problems and determine truth or falsehood of statements; hence there was a lot of thought on the nature of mechanical methods and formality. For more on this early work, I recommend John Hopcroft's article "Turing Machines" [41] and two survey papers by Martin Davis [20, 21]. If you wish to go deeper philosophically (and broader mathematically), try van Heijenoort [86] and Webb [88].

David Hilbert gave an address in 1900 in which he listed problems he thought should direct mathematical effort as the new century began [37]. His tenth problem, paraphrased, was to find a procedure to determine whether any given multivariable polynomial equation with integer coefficients (a *Diophantine* equation) has an integer solution. Hilbert asked for "a process according to which it can be determined by a finite number of operations" whether such a solution exists. For the specific case of single-variable Diophantine equations, mathematicians already had the rational root test; for an equation of the form $a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0 = 0$, any rational solution must be of the form $\pm\frac{r}{s}$ where r divides a_0 and s divides a_n. One may manually check each such fraction and determine whether any of them yields equality.

In 1910, the first volume of Alfred North Whitehead and Bertrand Russell's *Principia Mathematica* was published [89]. This ultimately

three-volume work was an effort to develop all mathematics from a small common set of axioms, with full rigor at every step. Russell and Whitehead wanted to remove vagueness and paradox from mathematics; every step in their system could be checked mechanically.

It seems this might take all creativity out of mathematics, as all possible theorems would eventually be produced by the mechanical application of logical deduction rules to axioms and previously generated theorems. However, in 1931 Gödel showed it was impossible for such a system to produce all true mathematical statements [30]. He used the mechanical nature of the system, intended for rigor, and showed it allowed a system of formula encoding that gave access to self-reference: he produced a formula P that says "P has no proof." If P is true, it is unprovable. If the negation of P is true, P has a proof, since that is the assertion made by P's negation. Therefore, unless Russell and Whitehead's system is internally inconsistent, P must be true, and hence unprovable. We will see a proof of Gödel's Incompleteness Theorem via undecidable problems in §5.3.

Principia Mathematica was a grand undertaking and one might expect it to fail to fulfill all its authors' hopes. However, Hilbert's quest was doomed as well. The proof that there can be no procedure to determine whether an arbitrary Diophantine equation has integer roots was not completed until 1973 [61], but in 1936 Church made the first response suggesting that might be the case [14]. He wrote:

> There is a class of problems of elementary number theory which can be stated in the form that it is required to find an effectively calculable function f of n positive integers, such that $f(x_1, \ldots, x_n) = 2$ is a necessary and sufficient condition for the truth of a certain proposition of elementary number theory involving $x_1, \ldots, x_n$ as free variables. [footnote: The selection of the particular positive integer 2 instead of some other is, of course, accidental and non-essential.]
>
> ... The purpose of the present paper is to propose a definition of effective calculability which is thought to correspond satisfactorily to the somewhat vague

> intuitive notion in terms of which problems of this class are often stated, and to show, by means of an example, that not every problem of this class is solvable.

Church's major contribution here is the point that we need some *formal* notion of "finite process" to answer Hilbert. He proposes two options in this paper: the lambda calculus, due to him and Kleene, and recursive functions, defined originally by Gödel [30] (after a suggestion by Herbrand) and modified by Kleene. Shortly thereafter Kleene proposed what we now call the partial recursive functions [44]. It was not widely accepted at the time that any of these definitions was a good characterization of "effectively computable," however. It was not until Turing developed his Turing machine [85], which *was* accepted as a good characterization, and it was proved that Turing-computable functions, lambda-computable functions, and partial recursive functions are the same class, that the functional definitions were accepted. All three of these formalizations of computability are studied in Chapter 3. The idea that not all problems are solvable comes up in Chapter 4, along with many of the tools used in such proofs.

Both Gödel's Incompleteness Theorem and Church's unsolvability result treat the limitations of mechanical, or algorithmic, procedures in mathematics. As is common in mathematics, these new ideas and tools took on a life of their own beyond answering Hilbert or finding the major flaw in Russell and Whitehead's approach to mathematics. The new field became known as recursion theory or computability theory. Chapters 5–8 explore some of the additional topics and fundamental results of the area, and Chapter 9 contains a survey of some areas of current interest to computability theorists.

1.3. Notes on Use of the Text

My intent is that Chapter 2 will be covered on an as-needed basis, and I have tried to include references to it wherever applicable throughout the text. However, the vocabulary in §§2.1 and 2.2 is needed throughout, so they should be read first if unfamiliar. Returning to §1.1 after reading through Chapter 4 may be helpful as well.

The core material is in Chapters 3 through 7. In those, without losing continuity, §§3.7, 4.5, 5.3, 6.2, and 6.3 may be omitted; if §6.2 is omitted, §5.4 may also be.

Chapters 8 and 9 should be covered as interest warrants. There is no interdependence between sections of these chapters except that §§9.4 and 9.5 both draw on §9.3, and §9.1 leans lightly on §8.3.

1.4. Acknowledgements and References

These notes owe a great debt to a small library of logic books. For graduate- and research-level work I regularly refer to *Classical Recursion Theory* by P. G. Odifreddi [68], *Theory of Recursive Functions and Effective Computability* by H. Rogers [75], and *Recursively Enumerable Sets and Degrees* by R. I. Soare [82]. The material in here owes a great deal to those three texts. More recently, I have enjoyed A. Nies' book *Computability and Randomness* [67]. As you will see, M. Davis' *The Undecidable* [18] was a constant presence on my desk as well.

In how to present such material to undergraduates, I was influenced by such books as *Computability and Logic* by Boolos, Burgess, and Jeffrey [10], *Computability* by Cutland [17], *A Mathematical Introduction to Logic* by Enderton [25], *An Introduction to Formal Languages and Automata* by Linz [57], and *A Transition to Advanced Mathematics* by Smith, Eggen, and St. Andre [81].

I have talked to many people about bits and pieces of the book, getting clarifications and opinions on my exposition, but the lion's share of gratitude must go to Denis Hirschfeldt. Many thanks are due also to the students in the three offerings of Computability Theory I gave as I was writing this text, first as course notes and then with my eye to publishing a book. With luck, their questions as they learned the material and their comments on the text have translated into improved exposition.

Chapter 2

Background

This chapter covers a collection of topics that are not computability theory per se, but are needed for it. They are set apart so the rest of the text reads more smoothly. If you are not familiar with logical and set notation, read §§2.1 and 2.2 now. The rest should be covered as needed when they become relevant.

2.1. First-Order Logic

In this section we learn a vocabulary for expressing formulas, logical sentences. This is useful for brevity ($x < y$ is much shorter than "x is less than y," and the savings grow as the statement becomes more complicated) but also for clarity. Expressing a mathematical statement symbolically can make it more obvious what needs to be done with it, and however carefully words are used they may admit some ambiguity.

We use lowercase Greek letters (mostly φ, ψ, and θ) to represent formulas. The simplest formula is a single symbol (or assertion) which can be either true or false. There are several ways to modify formulas, which we'll step through one at a time.

The *conjunction* of formulas φ and ψ is written "φ and ψ," "$\varphi \wedge \psi$," or "φ & ψ." It is true when both φ and ψ are true, and false otherwise. Logically "and" and "but" are equivalent, and so

are φ & ψ and ψ & φ, though in natural language there are some differences in connotation.

The *disjunction* of φ and ψ is written "φ or ψ" or "$\varphi \vee \psi$." It is false when both φ and ψ are false, and true otherwise. That is, $\varphi \vee \psi$ is true when *at least one* of φ and ψ is true; it is *inclusive or*. English tends to use *exclusive or*, which is true only when exactly one of the clauses is true, though there are exceptions. One such: "Would you like sugar or cream in your coffee?" Again, $\varphi \vee \psi$ and $\psi \vee \varphi$ are equivalent.

The *negation* of φ is written "not(φ)," "not-φ," "$\neg\varphi$," or "$\sim\varphi$." It is true when φ is false and false when φ is true. The potential difference from natural language negation is that $\neg\varphi$ must cover all cases where φ fails to hold, and in natural language the scope of a negation is sometimes more limited. Note that $\neg\neg\varphi = \varphi$.

How does negation interact with conjunction and disjunction? φ & ψ is false when φ, ψ, or both are false, and hence its negation is $(\neg\varphi) \vee (\neg\psi)$. $\varphi \vee \psi$ is false only when both φ and ψ are false, and so its negation is $(\neg\varphi)\&(\neg\psi)$. We might note in the latter case that this matches up with English's "neither...nor" construction. These two negation rules are called *De Morgan's Laws*.

Exercise 2.1.1. Simplify the following formulas.

(i) φ & $((\neg\varphi) \vee \psi)$

(ii) $(\varphi$ & $(\neg\psi)$ & $\theta) \vee (\varphi$ & $(\neg\psi)$ & $(\neg\theta))$

(iii) $\neg((\varphi$ & $\neg\psi)$ & $\varphi)$

There are two classes of special formulas to highlight now. A *tautology* is always true; the classic example is $\varphi \vee (\neg\varphi)$ for any formula φ. A *contradiction* is always false; here the example is φ & $(\neg\varphi)$. You will sometimes see the former expression denoted T (or $\top$) and the latter $\bot$.

To say φ *implies* ψ ($\varphi \to \psi$ or $\varphi \Rightarrow \psi$) means whenever φ is true, so is ψ. We call φ the *antecedent*, or assumption, and ψ the *consequent*, or conclusion, of the implication. We also say φ is *sufficient* for ψ (since whenever we have φ we have ψ, though we may also have ψ when φ is false), and ψ is *necessary* for φ (since it is impossible to

have φ without ψ). Clearly $\varphi \to \psi$ should be true when both formulas are true, and it should be false if φ is true but ψ is false. It is maybe not so clear what to do when φ is false; this is clarified by rephrasing implication as disjunction (which is often how it is defined in the first place). $\varphi \to \psi$ means either ψ holds or φ fails; i.e., $\psi \vee (\neg\varphi)$. The truth of that statement lines up with our assertions earlier, and gives truth values for when φ is false – namely, that the implication is true. Another way to look at this is to say $\varphi \to \psi$ is only false when *proven* false; i.e., when it has a true antecedent but a false consequent. From this it is clear that $\neg(\varphi \to \psi)$ is $\varphi \,\&\, (\neg\psi)$.

There is an enormous difference between implication in natural language and implication in logic. Implication in natural language tends to connote causation, whereas the truth of $\varphi \to \psi$ need not give any connection at all between the meanings of φ and ψ. It could be that φ is a contradiction, or that ψ is a tautology. Also, in natural language we tend to dismiss implications as irrelevant or meaningless when the antecedent is false, whereas to have a full and consistent logical theory we cannot throw those cases out.

Example 2.1.2. The following are true implications:

- If fish live in the water, then earthworms live in the soil.
- If rabbits are aquamarine blue, then earthworms live in the soil.
- If rabbits are aquamarine blue, then birds drive cars.

The negation of the final statement is "Rabbits are aquamarine blue but birds do not drive cars."

The statement "If fish live in the water, then birds drive cars" is an example of a false implication.

Equivalence is two-way implication and indicated by a double-headed arrow: $\varphi \leftrightarrow \psi$ or $\varphi \Leftrightarrow \psi$. It is an abbreviation for $(\varphi \to \psi) \,\&\, (\psi \to \varphi)$, and is true when φ and ψ are either both true or both false. Verbally we might say "φ if and only if ψ", which is often abbreviated to "φ iff ψ". In terms of just conjunction, disjunction, and negation, we may write equivalence as $(\varphi \,\&\, \psi) \vee ((\neg\varphi) \,\&\, (\neg\psi))$. Its negation is exclusive or, $(\varphi \vee \psi) \,\&\, \neg(\varphi \,\&\, \psi)$.

Exercise 2.1.3. Negate the following statements.

(i) 56894323 is a prime number.

(ii) If there is no coffee, I drink tea.

(iii) John watches but does not play.

(iv) I will buy the blue shirt or the green one.

Exercise 2.1.4. Write the following statements using standard logical symbols.

(i) φ if ψ.

(ii) φ only if ψ.

(iii) φ unless ψ.

As an aside, let us have a brief introduction to *truth tables*. These are nothing more than a way to organize information about logical statements. The leftmost columns are generally headed by the individual propositions, and under those headings occur all possible combinations of truth and falsehood. The remaining columns are headed by more complicated formulas that are built from the propositions, and the lower rows have T or F depending on the truth or falsehood of the header formula when the propositions have the true/false values in the beginning of that row. Truth tables aren't particularly relevant to our use for this material, so I'll leave you with an example and move on.

φ	ψ	$\neg\varphi$	$\neg\psi$	$\varphi \,\&\, \psi$	$\varphi \vee \psi$	$\varphi \to \psi$	$\varphi \leftrightarrow \psi$
T	T	F	F	T	T	T	T
T	F	F	T	F	T	F	F
F	T	T	F	F	T	T	F
F	F	T	T	F	F	T	T

If we stop here, we have *propositional* (or *sentential*) logic. These formulas usually look something like $[A \vee (B \& C)] \to C$ and their truth or falsehood depends on the truth or falsehood of the assertions A, B, and C. We will continue on to *predicate* logic, which replaces these assertions with statements such as $(x < 0)$ & $(x + 100 > 0)$, which will be true or false depending on the value substituted for the variable x. We will be able to turn those formulas into statements

which are true or false inherently via *quantifiers*. Note that writing $\varphi(x)$ indicates the variable x appears in the formula φ, and does not technically forbid φ containing other variables.

The *existential* quantification $\exists x$ is read "there exists x." The formula $\exists x \varphi(x)$ is true if for some value n the unquantified formula $\varphi(n)$ is true. *Universal* quantification, on the other hand, is $\forall x \varphi(x)$ ("for all x, $\varphi(x)$ holds"), true when no matter what n we fill in for x, $\varphi(n)$ is true.

Quantifiers must have a specified set of values to range over, because the truth value of a formula may be different depending on this *domain of quantification*. For example, take the formula

$$(\forall x)(x \neq 0 \to (\exists y)(xy = 1)).$$

This asserts every nonzero x has a multiplicative inverse. If we are letting our quantifiers range over the real numbers (denoted $\mathbb{R}$) or the rational numbers ($\mathbb{Q}$), this statement is true, because the reciprocal of x is available to play the role of y. However, in the integers ($\mathbb{Z}$) or natural numbers ($\mathbb{N}$) this is false, because $1/x$ is only in the domain when x is ± 1.

Introducing quantification opens us up to two kinds of logical formulas. If all variables are quantified over (*bound* variables), then the formula is called a *sentence*. If there are variables that are not in the scope of any quantifier (*free* variables), the formula is called a *predicate*. The truth value of a predicate depends on what values are plugged in for the free variables; a sentence has a truth value, period. For example, $(\forall x)(\exists y)(x < y)$ is a sentence, and it is true in all our usual domains of quantification. The formula $x < y$ is a predicate, and it will be true or false depending on whether the specific values plugged in for x and y satisfy the inequality.

Exercise 2.1.5. Write the following statements as formulas, specifying the domain of quantification.

(i) 5 is prime.

(ii) For any number x, the square of x is nonnegative.

(iii) There is a smallest positive integer.

Exercise 2.1.6. Consider $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, and $\mathbb{R}$. Over which domains of quantification are each of the following statements true?

(i) $(\forall x)(x \geq 0)$
(ii) $(\exists x)(5 < x < 6)$
(iii) $(\forall x)((x^2 = 2) \to (x = 5))$
(iv) $(\exists x)(x^2 - 1 = 0)$
(v) $(\exists x)(x^2 = 5)$
(vi) $(\exists x)(x^3 + 8 = 0)$
(vii) $(\exists x)(x^2 - 2 = 0)$

When working with multiple quantifiers the order of quantification can matter a great deal, as in the following formulas.

$$\varphi = (\forall x)(\exists y)(x \cdot x = y)$$

$$\psi = (\exists y)(\forall x)(x \cdot x = y)$$

φ says "every number has a square" and is true in our typical domains. ψ says "there is a number which is all other numbers' square" and is true only if your domain contains only 0 or only 1.

Exercise 2.1.7. Over the real numbers, which of the following statements are true? Over the natural numbers?

(i) $(\forall x)(\exists y)(x + y = 0)$
(ii) $(\exists y)(\forall x)(x + y = 0)$
(iii) $(\forall x)(\exists y)(x \leq y)$
(iv) $(\exists y)(\forall x)(x \leq y)$
(v) $(\exists x)(\forall y)(x < y^2)$
(vi) $(\forall y)(\exists x)(x < y^2)$
(vii) $(\forall x)(\exists y)(x \neq y \to x < y)$
(viii) $(\exists y)(\forall x)(x \neq y \to x < y)$

The order of operations when combining quantification with conjunction or disjunction can also make the difference between truth and falsehood.

Exercise 2.1.8. Over the real numbers, which of the following statements are true? Over the natural numbers?

(i) $(\forall x)(x \geq 0 \vee x \leq 0)$

(ii) $[(\forall x)(x \geq 0)] \vee [(\forall x)(x \leq 0)]$

(iii) $(\exists x)(x \leq 0 \ \& \ x \geq 5)$

(iv) $[(\exists x)(x \leq 0)] \ \& \ [(\exists x)(x \geq 5)]$

How does negation work for quantifiers? If $\exists x \varphi(x)$ fails, it means no matter what value we fill in for x the formula obtained is false; i.e., $\neg(\exists x \varphi(x)) \leftrightarrow \forall x(\neg\varphi(x))$. Likewise, $\neg(\forall x \varphi(x)) \leftrightarrow \exists x(\neg\varphi(x))$: if φ does not hold for all values of x, there must be an example for which it fails. If we have multiple quantifiers, the negation walks in one by one, flipping each quantifier and finally negating the predicate inside. For example:

$$\neg[(\exists x)(\forall y)(\forall z)(\exists w)\varphi(x, y, z, w)] \leftrightarrow (\forall x)(\exists y)(\exists z)(\forall w)(\neg\varphi(x, y, z, w)).$$

Exercise 2.1.9. Negate the following sentences.

(i) $(\forall x)(\exists y)(\forall z)((z < y) \rightarrow (z < x))$

(ii) $(\exists x)(\forall y)(\exists z)(xz = y)$

(iii) $(\forall x)(\forall y)(\forall z)(y = x \ \vee \ z = x \ \vee \ y = z)$
[Bonus: over what domains of quantification would this be true?]

A final notational comment: you will sometimes see the symbols $\exists^\infty$ and $\forall^\infty$. The former means "there exist infinitely many;" $\exists^\infty x \varphi(x)$ is shorthand for $\forall y \exists x(x > y \ \& \ \varphi(x))$ (no matter how far up we go, there are still examples of φ above us). The latter means "for all but finitely many;" $\forall^\infty x \varphi(x)$ is shorthand for $\exists y \forall x((x > y) \rightarrow \varphi(x))$ (we can get high enough up to bypass all the failed cases of φ). Somewhat common in predicate logic but less so in computability theory is $\exists! x$, which means "there exists a unique x." The sentence $(\exists! x)\varphi(x)$ expands into $(\exists x)(\forall y)(\varphi(x) \ \& \ (\varphi(y) \rightarrow (x = y)))$.

2.2. Sets

A *set* is a collection of objects. If x is a member, or *element*, of a set A, we write $x \in A$, and otherwise $x \notin A$. Two sets are *equal* if

they have the same elements; if they have no elements in common they are called *disjoint*. The set A is a *subset* of a set B if all of the elements of A are also elements of B; this is denoted $A \subseteq B$. If we know that A is not equal to B, we may write $A \subset B$ or (to emphasize the non-equality) $A \subsetneq B$. The collection of all subsets of A is denoted $\mathcal{P}(A)$ and called the *power set* of A.

We may write a set using an explicit list of its elements, such as {red, blue, green} or $\{5, 10, 15, \ldots\}$. When writing down sets, order does not matter and repetitions do not count, so $\{1, 2, 3\}$, $\{2, 3, 1\}$, and $\{1, 1, 2, 2, 3, 3\}$ are all representations of the same set. We may also use notation that may be familiar to you from calculus:

$$A = \{x : (\exists y)(y^2 = x)\}.$$

This is the set of all values we can fill in for x that make the logical predicate $(\exists y)(y^2 = x)$ true. The syntax of set definitions is often looser than in logical formulas generally, using commas to mean "and" and condensing clauses. For example, $\{(x, y) : x, y^2 \in \mathbb{Q}, x < y\}$ abbreviates $\{(x, y) : x \in \mathbb{Q} \,\&\, y^2 \in \mathbb{Q} \,\&\, x < y\}$.

We are always working within some fixed *universe*, a set which contains all of our sets. The domain of quantification is all elements of the universe, and hence the contents of the set A above will vary depending on what our universe is. If we are living in the integers it is the set of perfect squares; if we are living in the real numbers it is the set of all non-negative numbers.

Given two sets, we may obtain a third from them in several ways. First there is *union*: $A \cup B$ is the set containing all elements that appear in at least one of A and B. Next *intersection*: $A \cap B$ is the set containing all elements that appear in both A and B. We can subtract: $A - B$ contains all elements of A that are *not* also elements of B. You will often see $A \setminus B$ for set subtraction, but we will use ordinary minus because the slanted minus is sometimes given a different meaning in computability theory. Finally, we can take their *Cartesian product*: $A \times B$ consists of all ordered pairs that have their first entry an element of A and their second an element of B. We may take the product of more than two sets to get ordered triples, quadruples, quintuples, and in general *n-tuples*. If we take

the Cartesian product of n copies of A, we may abbreviate $A \times A \times \ldots \times A$ as A^n. A generic ordered n-tuple from A^n will be written $(x_1, x_2, \ldots, x_n)$, where x_i are all elements of A.

Example 2.2.1. Let $A = \{x, y\}$ and $B = \{y, z\}$. Then $A \cup B = \{x, y, z\}$, $A \cap B = \{y\}$, $A - B = \{x\}$, $B - A = \{z\}$, $A \times B = \{(x, y), (x, z), (y, y), (y, z)\}$, and $\mathcal{P}(A) = \{\emptyset, \{x\}, \{y\}, \{x, y\}\}$.

The sets we will use especially are $\emptyset$ and $\mathbb{N}$. The former is the *empty set*, the set with no elements. The latter is the *natural numbers*, the set $\{0, 1, 2, 3, \ldots\}$. In computability, we often use lowercase omega, ω, to denote the natural numbers, but in these notes we will be consistent with $\mathbb{N}$. On occasion we may also refer to $\mathbb{Z}$ (the integers), $\mathbb{Q}$ (the rational numbers), or $\mathbb{R}$ (the real numbers).

We will assume unless otherwise specified that our universe is $\mathbb{N}$; i.e., all of our sets are subsets of $\mathbb{N}$. When a universe is fixed we can define *complement*. The complement of A, denoted $\overline{A}$, is all the elements of $\mathbb{N}$ that are not in A; i.e., $\overline{A} = \mathbb{N} - A$.

Exercise 2.2.2. Convert the list or description of each of the following sets into notation using a logical predicate. Assume the domain of quantification is $\mathbb{N}$.

(i) $\{2, 4, 6, 8, 10, \ldots\}$

(ii) $\{4, 5, 6, 7, 8\}$

(iii) The set of numbers that are cubes.

(iv) The set of pairs of numbers such that one is twice the other (in either order).

(v) The intersection of the set of square numbers and the set of numbers that are divisible by 3.

(vi) [For this and the next two, you'll need to use $\in$ in your logical predicate.] $A \cup B$ for sets A and B.

(vii) $A \cap B$ for sets A and B.

(viii) $A - B$ for sets A and B.

Exercise 2.2.3. For each of the following sets, list (a) the elements of X, and (b) the elements of $\mathcal{P}(X)$.

(i) $X = \{1, 2\}$

(ii) $X = \{1, 2, \{1, 2\}\}$

(iii) $X = \{1, 2, \{1, 3\}\}$

Exercise 2.2.4. Work inside the finite universe $\{1, 2, \ldots, 10\}$. Define the following sets:

$A = \{1, 3, 5, 7, 9\}$

$B = \{1, 2, 3, 4, 5\}$

$C = \{2, 4, 6, 8, 10\}$

$D = \{7, 9\}$

$E = \{4, 5, 6, 7\}$

(i) Find all the subset relationships between pairs of the sets above.

(ii) Which pairs, if any, are disjoint?

(iii) Which pairs, if any, are complements?

(iv) Find the following unions and intersections: $A \cup B$, $A \cup D$, $B \cap D$, $B \cap E$.

We can also take unions and intersections of infinitely many sets. For sets A_i for $i \in \mathbb{N}$, these are defined as follows.

$$\bigcup_i A_i = \{x : (\exists i)(x \in A_i)\}$$

$$\bigcap_i A_i = \{x : (\forall i)(x \in A_i)\}$$

The i under the union or intersection symbol is also sometimes written "$i \in \mathbb{N}$."

Exercise 2.2.5. For $i \in \mathbb{N}$, let $A_i = \{0, 1, \ldots, i\}$ and let $B_i = \{0, i\}$. What are $\bigcup_i A_i$, $\bigcup_i B_i$, $\bigcap_i A_i$, and $\bigcap_i B_i$?

If two sets are given by descriptions instead of explicit lists, we must prove one set is a subset of another by taking an arbitrary element of the first set and showing it is also a member of the second set. For example, to show the set of people eligible for President of the United States is a subset of the set of people over 30, we might say: Consider a person in the first set. That person must meet the criteria listed in the U.S. Constitution, which includes being at least

35 years of age. Since 35 is more than 30, the person we chose is a member of the second set.

We can further show that this containment is proper, by demonstrating a member of the second set who is not a member of the first set. For example, a 40-year-old who is not a U.S. citizen.

Exercise 2.2.6. Prove that the set of squares of even numbers, $\{x : \exists y(x = (2y)^2)\}$, is a proper subset of the set of multiples of 4, $\{x : \exists y(x = 4y)\}$.

To prove two sets are equal, there are three options: show the criteria for membership on each side are the same, manipulate set operations until the expressions are the same, or show each side is a subset of the other side.

An extremely basic example of the first option is showing $\{x : \frac{x}{2}, \frac{x}{4} \in \mathbb{N}\} = \{x : (\exists y)(x = 4y)\}$. For the second, we have a bunch of *set identities*, including a set version of De Morgan's Laws.

$$\overline{A \cap B} = \overline{A} \cup \overline{B}$$

$$\overline{A \cup B} = \overline{A} \cap \overline{B}$$

We also have distribution laws.

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

To prove identities we have to turn to the first or third option.

Example 2.2.7. Prove that $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$.

We work by showing each set is a subset of the other. Suppose first that $x \in A \cup (B \cap C)$. By definition of union, x must be in A or in $B \cap C$. If $x \in A$, then x is in both $A \cup B$ and $A \cup C$, and hence in their intersection. On the other hand, if $x \in B \cap C$, then x is in both B and C, and hence again in both $A \cup B$ and $A \cup C$.

Now suppose $x \in (A \cup B) \cap (A \cup C)$. Then x is in both unions, $A \cup B$ and $A \cup C$. If $x \in A$, then $x \in A \cup (B \cap C)$. If, however, $x \notin A$, then x must be in both B and C, and therefore in $B \cap C$. Again, we obtain $x \in A \cup (B \cap C)$.

Notice that in the $\subseteq$ direction we used two cases that could overlap, and did not worry whether we were in the overlap or not. In the $\supseteq$ direction, we could only assert $x \in B$ *and* $x \in C$ if we knew $x \notin A$ (although it is certainly possible for x to be in all three sets), so forbidding the first case was part of the second case.

Exercise 2.2.8. Using any of the three options listed above, as long as it is applicable, do the following.

(i) Prove intersection distributes over union (i.e., for all A, B, C, $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$).

(ii) Prove de Morgan's Laws.

(iii) Prove that $A \cup B = (A - B) \cup (B - A) \cup (A \cap B)$ for any sets A and B.

Our final topic in the realm of sets is *cardinality*. The cardinality of a finite set is the number of elements in it. For example, the cardinality of the set of positive integer divisors of 6 is 4: $|\{1, 2, 3, 6\}| = 4$. When we get to infinite sets, cardinality separates them by "how infinite" they are. We'll get to its genuine definition in §2.3, but it is fine now and later to think of cardinality as a synonym for size. The way to tell whether set A is bigger than set B is to look for a one-to-one function from A into B. If no such function exists, then A is bigger than B, and we write $|B| < |A|$. The most important result is that $|A| < |\mathcal{P}(A)|$ for *any* set A (see §A.3).

If we know there is a one-to-one function from A into B but we don't know about the reverse direction, we write $|A| \leq |B|$. If we have injections both ways, $|A| = |B|$. It is a significant theorem of set theory that having injections from A to B and from B to A is equivalent to having a bijection between A and B; the fact that this requires work is a demonstration of the fact that things get weird when you work in the infinite world. Another key fact (for set theorists; not so much for us) is *trichotomy*: for any two sets A and B, exactly one of $|A| < |B|$, $|A| > |B|$, or $|A| = |B|$ is true.

For us, infinite cardinalities are divided into two categories. A set is *countably infinite* if it has the same cardinality as the natural numbers. The integers and the rational numbers are important examples of countably infinite sets. The term *countable* is used by some

authors to mean "countably infinite," and by others to mean "finite or countably infinite," so you often have to rely on context. We will mean the latter, but will try to be explicit. To prove that a set is countably infinite, you must demonstrate it is in bijection with the natural numbers – that is, that you can count the objects of your set 1, 2, 3, 4, ..., and not miss any. We'll come back to this in §3.4; for now you can look in the appendices to find Cantor's proofs that the rationals are countable and the reals are not (§A.3).

The rest of the infinite cardinalities are called *uncountable*, and for our purposes that's as fine-grained as it gets. The fundamental notions of computability theory live in the world of countable sets, and the only uncountable ones we get to are those which can be approximated in the countable world.

2.3. Relations

The following definition is not the most general case, but we'll start with it.

Definition 2.3.1. A *relation* $R(x, y)$ on a set A is a logical predicate that is true or false of each pair $(x, y) \in A^2$, never undefined.

We also think of relations as subsets of A^2 consisting of the pairs for which the relation is true. For example, in the set $A = \{1, 2, 3\}$, the relation $<$ consists of $\{(1, 2), (1, 3), (2, 3)\}$ and the relation $\leq$ is the union of $<$ with $\{(1, 1), (2, 2), (3, 3)\}$. Note that the order matters: although $1 < 2$, $2 \not< 1$, so $(2, 1)$ is not in $<$. The first definition shows you why these are called *relations*; we think of R as being true when the values filled in for x and y have some relationship to each other. The set-theoretic definition is generally more useful, however.

More generally, we may define *n-ary* relations on a set A as logical predicates that are true or false of any n-tuple (ordered set of n elements) of A, or alternatively as subsets of A^n. For $n = 1, 2, 3$ we refer to these relations as *unary*, *binary*, and *ternary*, respectively.

Exercise 2.3.2. Prove the two definitions of relation are equivalent. That is, prove that every logical predicate corresponds to a unique set, and vice-versa.

Exercise 2.3.3. Let $A = \{a, b, c, d, e\}$.

(i) What is the ternary relation R on A defined by $(x, y, z) \in R \Leftrightarrow$ (xyz is an English word)?

(ii) What is the unary relation on A which is true of elements of A that are vowels?

(iii) What is the complement of the relation in (ii)? We may describe it in two ways: as "the negation of the relation in (ii)," and how?

(iv) Define the 5-ary relation R by $(v, w, x, y, z) \in R \Leftrightarrow (v, w, x, y, z$ are all *distinct* elements of $A)$. How many elements does R contain?

(v) How many unary relations are possible on A? What other collection associated with A does the collection of all unary relations correspond to?

Exercise 2.3.4. How many n-ary relations are possible on an m-element set?

We tend to focus on binary relations, since most of our common, useful examples are binary: $<, \leq, =, \neq, \subset, \subseteq$. Binary relations may have certain properties:

- Reflexivity: $(\forall x)R(x, x)$
- Symmetry: $(\forall x, y)[R(x, y) \rightarrow R(y, x)]$
  i.e., $(\forall x, y)[(R(x, y)\ \&\ R(y, x))\ \vee\ (\neg R(x, y)\ \&\ \neg R(y, x))]$
- Antisymmetry: $(\forall x, y)[(R(x, y)\ \&\ R(y, x)) \rightarrow x = y]$
- Transitivity: $(\forall x, y, z)[(R(x, y)\ \&\ R(y, z)) \rightarrow R(x, z)]$

I want to point out that reflexivity is a property of possession: R must have the reflexive pairs (the pairs (x, x)). Antisymmetry is, loosely, a property of nonpossession. Symmetry and transitivity, on the other hand, are *closure* properties: **if** R has certain pairs, **then** it must also have other pairs. Those conditions may be met either by adding in the pairs that are consequences of the pairs already present, or omitting the pairs that are requiring such additions. In particular, the empty relation is symmetric and transitive, though it is not reflexive.

Exercise 2.3.5. Is $=$ reflexive? Symmetric? Antisymmetric? Transitive? How about $\neq$?

Exercise 2.3.6. For finite relations we may check these properties by hand. Let $A = \{1, 2, 3, 4\}$.

(a) What is the smallest binary relation on A that is reflexive?

(b) Define the following binary relations on A.

$$R_1 = \{(2, 3), (3, 4), (4, 2)\}$$

$$R_2 = \{(1, 1), (1, 2), (2, 1), (2, 2)\}$$

$$R_3 = \{(1, 1), (1, 2), (2, 2), (2, 3), (3, 3), (3, 4), (4, 4)\}$$

For each of those relations, answer the following questions.

(i) Is the relation reflexive? Symmetric? Antisymmetric? Transitive?

(ii) If the relation is not reflexive, what is the smallest collection of pairs that needs to be added to make it reflexive?

(iii) If the relation is not symmetric, what is the smallest collection of pairs that needs to be added to make it symmetric?

(iv) If the relation is not transitive, what is the smallest collection of pairs that needs to be added to make it transitive?

(v) If the relation is not antisymmetric, what is the smallest collection of pairs that could be removed to make it antisymmetric? Is this answer unique?

Exercise 2.3.7. Let $A = \{1, 2, 3\}$. Define binary relations on A with the following combinations of properties or say why such a relation cannot exist. Can such a relation be nonempty?

(i) Reflexive and antisymmetric but neither symmetric nor transitive.

(ii) Symmetric but neither reflexive nor transitive.

(iii) Transitive but neither reflexive nor symmetric.

(iv) Symmetric and transitive but not reflexive.

(v) Both symmetric and antisymmetric.

(vi) Neither symmetric nor antisymmetric.

(vii) Reflexive and transitive but not symmetric.

(viii) Reflexive and symmetric but not transitive.

(ix) Symmetric, antisymmetric, and transitive.

(x) Reflexive, symmetric, and transitive.

(xi) None of reflexive, symmetric, or transitive.

Exercise 2.3.8. Suppose R and S are binary relations on A. For each of the following properties, if R and S possess the property, must $R \cup S$ possess it? $R \cap S$?

(i) Reflexivity

(ii) Symmetry

(iii) Antisymmetry

(iv) Transitivity

Exercise 2.3.9. Each of the following relations has a simpler description than the one given. Find such a description.

(i) R_- on $\mathcal{P}(\mathbb{N})$ where $R_-(A, B) \leftrightarrow A - B = \emptyset$.

(ii) $R_{(\cap)}$ on $\mathbb{R}$ where $R_{(\cap)}(x, y) \leftrightarrow (-\infty, x) \cap (y, \infty) = \emptyset$.

(iii) $R_{[\cap]}$ on $\mathbb{R}$ where $R_{[\cap]}(x, y) \leftrightarrow (-\infty, x] \cap [y, \infty) = \emptyset$.

(iv) $R_{(\cup)}$ on $\mathbb{R}$ where $R_{(\cup)}(x, y) \leftrightarrow (-\infty, x) \cup (y, \infty) = \mathbb{R}$.

(v) $R_{[\cup]}$ on $\mathbb{R}$ where $R_{[\cup]}(x, y) \leftrightarrow (-\infty, x] \cup [y, \infty) = \mathbb{R}$.

We may visualize a binary relation R on A as a directed graph. The elements of A are the vertices, or nodes, of the graph, and there is an arrow (directed edge) from vertex x to vertex y if and only if $R(x, y)$ holds. The four properties we have just been exploring may be stated as:

- Reflexivity: every vertex has a loop.
- Symmetry: any pair of vertices is either directly connected in both directions or not directly connected at all.
- Antisymmetry: any two vertices have at most one edge directly connecting them.
- Transitivity: if there is a path of edges from one vertex to another (always proceeding in the direction of the edge), there is an edge directly connecting them, in the same direction as the path.

Exercise 2.3.10. Properly speaking, transitivity just gives the graphical interpretation "for any vertices x, y, z, if there is an edge from x to y and an edge from y to z, there is an edge from x to z." Prove that this statement is equivalent to the (a priori more general) one given for transitivity above.

We will consider two subsets of these properties that define classes of relations which are of particular importance.

Definition 2.3.11. An *equivalence relation* is a binary relation that is reflexive, symmetric, and transitive.

The quintessential equivalence relation is equality, which is the relation consisting of only the reflexive pairs. What is special about an equivalence relation? We can take a *quotient structure* whose elements are *equivalence classes*.

Definition 2.3.12. Let R be an equivalence relation on A. The *equivalence class* of some $x \in A$ is the set $[x] = \{y \in A : R(x, y)\}$.

Exercise 2.3.13. Let R be an equivalence relation on A and let x, y be elements of A. Prove that either $[x] = [y]$ or $[x] \cap [y] = \emptyset$.

In short, an equivalence relation puts all the elements of the set into boxes so that each element is unambiguously assigned to a single box. All possible pairings from within each box are in the relation, and no pairings that draw from different boxes are in the relation. We can consider the boxes themselves as elements, getting a quotient structure.

Definition 2.3.14. Given a set A and an equivalence relation R on A, the *quotient of A by R*, A/R, is the set whose elements are the equivalence classes of A under R.

Now we can define cardinality more correctly. The cardinality of a set is the equivalence class it belongs to under the equivalence relation of bijectivity, so cardinalities are elements of the quotient of the collection of all sets under that relation.

Exercise 2.3.15. Let A be the set $\{1, 2, 3, 4, 5\}$, and let R be the binary relation on A that consists of the reflexive pairs together with $(1, 2)$, $(2, 1)$, $(3, 4)$, $(3, 5)$, $(4, 3)$, $(4, 5)$, $(5, 3)$, $(5, 4)$.

(i) Represent R as a graph.

(ii) How many elements does A/R have?

(iii) Write out the sets [1], [2], and [3].

Exercise 2.3.16. A *partition* of a set A is a collection of disjoint subsets of A with union equal to A. Prove that any partition of A determines an equivalence relation on A, and every equivalence relation on A determines a partition of A.

Exercise 2.3.17. Let $R(m, n)$ be the relation on $\mathbb{Z}$ that holds when $m - n$ is a multiple of 3.

(i) Prove that R is an equivalence relation.

(ii) What are the equivalence classes of 1, 2, and 3?

(iii) What are the equivalence classes of -1, -2, and -3?

(iv) Prove that $\mathbb{Z}/R$ has three elements.

Exercise 2.3.18. Let $R(m, n)$ be the relation on $\mathbb{N}$ that holds when $m - n$ is even.

(i) Prove that R is an equivalence relation.

(ii) What are the equivalence classes of R? Give a concise verbal description of each.

The two exercises above are examples of *modular arithmetic*, which is also sometimes called *clock-face arithmetic* because its most widespread use in day-to-day life is telling what time it will be some hours from now. This is a notion that is used only in $\mathbb{N}$ and $\mathbb{Z}$. The idea of modular arithmetic is that it is only the number's remainder upon division by a fixed value that matters. For clock-face arithmetic, that value is 12; we say we are working *modulo 12*, or just *mod 12*, and the equivalence classes are represented by the numbers 0 through 11 (in mathematics; 1 through 12 in usual life). The fact that if it is currently 7:00 then in eight hours it will be 3:00 would be written as the equation $7 + 8 = 3 \pmod{12}$, where $\equiv$ is sometimes used in place of the equals sign.

Exercise 2.3.19. (i) Exercises 2.3.17 and 2.3.18 consider equivalence relations that give rise to arithmetic mod k for some k. For each, what is the correct value of k?

(ii) Describe the equivalence relation on $\mathbb{Z}$ that gives rise to arithmetic mod 12.

(iii) Let m, n, and p be integers. Prove that

$$n = m \pmod{12} \implies n + p = m + p \pmod{12}.$$

This shows that addition of equivalence classes via representatives is well-defined.

Our second important class of relations is partial orders.

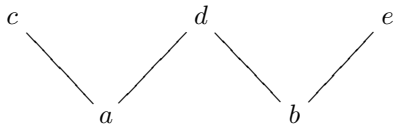
Definition 2.3.20. A *partial order* $\leq$ on a set A is a binary relation that is reflexive, antisymmetric, and transitive. A with $\leq$ is called a *partially ordered set*, or *poset*.

In a poset, given two nonequal elements of A, either one is strictly greater than the other or they are incomparable. If all pairs of elements are comparable, the relation is called a *total order* or *linear order* on A.

Example 2.3.21. Let $A = \{a, b, c, d, e\}$ and define $\leq$ on A as follows:

- $(\forall x \in A)(x \leq x)$
- $a \leq c$, $a \leq d$
- $b \leq d$, $b \leq e$

We could graph this as follows:



Technically, there are arrowheads pointing up and each element has a loop, but for partial orders we often assume that.

In Example 2.3.21, the elements c and d have no *upper bound*: no single element that is greater than or equal to both of them. We might add elements f and g, augmenting the relation to include $c \leq f$, $d \leq f$, $d \leq g$, and $e \leq g$ (making the diagram vertically symmetric), plus the necessary pairs to obtain transitivity (e.g., $a \leq f$; this is called taking the *transitive closure*). Then f is an upper bound for c

and d, and it is the least upper bound: any element that is greater than or equal to both c and d is also greater than or equal to f. The elements f and g are upper bounds for the pair a and b, but not the least upper bound; that is d. A *lower bound* for a pair of elements is defined symmetrically: an element that is less than or equal to both of the given elements. The greatest lower bound is a lower bound x such that any other lower bound is less than or equal to x. Note that even when lower and upper bounds exist, there need not be greatest lower or least upper bounds. This can happen in two ways: two incomparable bounds (think of a V, where the lower point is below both upper points, but the upper points are incomparable), or an infinite ascending or descending sequence of elements that limits to an element not in the poset (such as an open interval in the rational numbers with irrational endpoints).

Example 2.3.22. $\mathcal{P}(\mathbb{N})$ ordered by subset inclusion is a partially ordered set.

It is easy to check the relation $\subseteq$ is reflexive, transitive, and antisymmetric. Not every pair of elements is comparable: for example, neither $\{1, 2, 3\}$ nor $\{4, 5, 6\}$ is a subset of the other. This poset actually has some very nice properties that not every poset has: it has a top element ($\mathbb{N}$) and a bottom element ($\emptyset$), and every pair of elements has both a least upper bound (here, the union) and a greatest lower bound (the intersection).

If we were to graph this, it would look like an infinitely-faceted diamond with points at the top and bottom.

Example 2.3.23. Along the same lines as Example 2.3.22, we can consider the power set of a finite set, and then we can graph the poset that results.

Let $A = \{a, b, c\}$. Denote the set $\{a\}$ by a and the set $\{b, c\}$ by $\hat{a}$, and likewise for the other three elements. The graph is the transitive closure of the graph shown in Figure 2.1.

Exercise 2.3.24. Prove that every pair of elements in a poset has at most one greatest lower bound and at most one least upper bound.

Exercise 2.3.25. How many partial orders are possible on a set of two elements? Three elements?

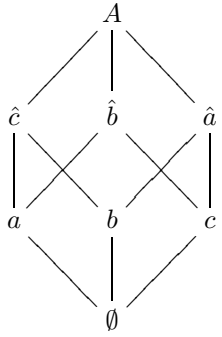


Figure 2.1. Subsets of $\{a, b, c\}$.

Our final note is to point out relations generalize functions. The function $f : A \to A$ may be written as a binary relation on A consisting of the pairs $(x, f(x))$. A binary relation R, conversely, represents a function whenever $[(x, y) \in R \;\&\; (x, z) \in R] \to y = z$ (the vertical line rule for functions).[1] We can ramp this up even further to multivariable functions, functions from A^n to A, by considering $(n+1)$-ary relations. The first n places represent the input and the last one the output. The advantage to this is consolidation; we can prove many things about functions by proving them for relations in general.

2.4. Bijection and Isomorphism

As mentioned, a function $f : A \to B$ may be thought of as a relation $G_f \subseteq A \times B$, the Cartesian product of its domain (also denoted $\operatorname{dom} f$) and codomain, though the property that every element of A appear in exactly one pair of G_f is not particularly natural to state in terms of relations. G_f is called the *graph* of f. Before we proceed recall also that the range of f, $\operatorname{rng} f$, is $\{f(x) : x \in A\} \subseteq B$.

When are two sets "the same?" How about two partial orders? The notion of *isomorphism* is, loosely, the idea that two mathematical structures may be essentially the same, even if cosmetically different. Recall that a function between two sets is *injective*, or *one-to-one*, if

[1] You might object that this does not require every element of A be in the domain of the function. We will not be concerned by that; see §3.1.2.

no two domain elements map to the same range element. It is *surjective*, or *onto*, if the codomain equals the range. Since no function can take a single domain element to multiple range elements, and by definition it must give an image (output) to every domain element (well... see §3.1.2), a function that is both one-to-one and onto uniquely pairs off the elements of the domain and codomain. Such functions are called *bijections*. Some authors will refer to a bijection as a "one-to-one correspondence;" they are also called *permutations*.

For sets, bijections are everything. It does not matter whether we use the letters A through Z or the numbers 1 through 26; a set of the same size is essentially the same set. Two sets (without structure) are *isomorphic* if there is a bijection between them. Two sets with additional structure may be bijective without being isomorphic.

When there are relations (including functions) on the elements, to say two structures are isomorphic we need a special kind of bijection: one that preserves the relations. For example, an isomorphism between two partial orders P and Q is a bijection $f : P \to Q$ such that for all $x, y \in P$, $x \leq_P y \leftrightarrow f(x) \leq_Q f(y)$. As a consequence, any logical statement about the relation will hold in one structure if and only if it holds in the other, such as $\forall x \exists y (x \leq y)$. Isomorphism must be defined separately for each kind of object, but it follows the same template each time: a bijection that preserves the structure.

Example 2.4.1. Let $A = \{\{1\}, \{1,2\}, \{1,2,3\}, \{1,2,3,4\}\}$ and $B = \{\emptyset, \{a\}, \{b\}, \{a,b\}\}$. A and B are in bijection, or *isomorphic as sets*. However, they are not isomorphic as partial orders, under the relation of subset inclusion. A is a linear order and B a diamond; A satisfies the statement $\forall x\ \forall y\ (x \subseteq y\ \vee\ y \subseteq x)$ and B satisfies its negation, $\exists x\ \exists y\ (\neg(x \subseteq y) \& \neg(y \subseteq x))$.

An isomorphism between a structure and itself is called an *automorphism*.

2.5. Recursion and Induction

Recursive definitions and proofs by induction are opposite sides of the same coin. Both have some specific starting point, and then a way to extend from there via a small set of operations. For induction, you

might be proving some property P is true of all the natural numbers. To do so, you prove that P is true of 0, and then prove that *if* P is true of some $n \geq 0$, *then* P is also true of $n+1$. To recursively define a class of objects C, you give certain simple examples of objects in C, and then operations that combine or extend elements of C to give results still in C. They relate more deeply than just appearance, though. We'll tackle induction, then recursion, then induction again.

2.5.1. Induction on $\mathbb{N}$. The basic premise of induction is that if you can start, and once you start you know how to keep going, then you will get all the way to the end. If I can get on the ladder, and I know how to get from one rung to the next, I can get to the top of the ladder.

Definition 2.5.1. The principle of mathematical induction, basic form, says the following.

If S is a subset of the positive integers containing 1 such that $n \in S$ implies $n+1 \in S$ for all n, then S contains all of the positive integers. [We may need the beginning to be 0 or another value depending on context.]

In general you want to use induction to show that some property holds no matter what integer you feed it, or no matter what size finite set you are dealing with. The proofs always have a *base case*, the case of 1 (or wherever you're starting). Then they have the *inductive step*, the point where you assume the property holds for some unspecified n and then show it holds for $n+1$.

Example 2.5.2. Prove that for every positive integer n,

$$1 + 3 + 5 + \ldots + (2n-1) = n^2.$$

Proof. Base case: For $n = 1$, the equation is $1 = 1^2$, which is true. Inductive step: Assume that $1 + 3 + 5 + \ldots + (2n-1) = n^2$ for some $n \geq 1$. To show that it holds for $n+1$, add $2(n+1) - 1$ to each side, in the simplified form $2n+1$.

$$1 + 3 + 5 + \ldots + (2n-1) + (2n+1) = n^2 + 2n + 1 = (n+1)^2$$

Since the equation above is that of the theorem, for $n+1$, by induction the equation holds for all n. □

The format of the proof above is typical of inductive proofs of summation formulas: use the inductive hypothesis to simplify a portion of the next value's sum.

For the next example we need to know a *convex* polygon is one where all the corners point out. If you connect two corners of a convex polygon with a straight line segment, the segment will lie entirely within the polygon, cutting it into two smaller convex polygons.

As you get more comfortable with induction, you can write it in a more natural way, without segmenting off the base case and inductive step portions of the argument. We'll do that here. Notice the base case is not 0 or 1 for this proof.

Example 2.5.3. For $n > 2$, the sum of angle measures of the interior angles of a convex polygon of n vertices is $(n-2) \cdot 180°$.

Proof. We work by induction. For $n = 3$, the polygon in question is a triangle, and it has interior angles which sum to $180° = (3-2)\cdot 180°$.

Assume the theorem holds for some $n \geq 3$ and consider a convex polygon with $n+1$ vertices. Let one of the vertices be named x, and pick a vertex y such that along the perimeter from x in one direction there is a single vertex between x and y, and in the opposite direction, $(n+1)-3 = n-2$ vertices. Join x and y by a new edge, dividing the original polygon into two polygons. The new polygons' interior angles together sum to the sum of the original polygon's interior angles. One of the new polygons has 3 vertices and the other has n vertices (x, y, and the $n-2$ vertices between them). The triangle has interior angle sum $180°$, and by the inductive hypothesis the n-gon has interior angle sum $(n-2) \cdot 180°$. The $n+1$-gon therefore has interior angle sum $180° + (n-2)180° = (n+1-2) \cdot 180°$, as desired. □

Notice also in this example that we used the base case as part of the inductive step, since one of the two polygons was a triangle. This is not uncommon.

Exercise 2.5.4. Prove the following statements by induction.

(i) For every positive integer n,

$$1 + 4 + 7 + \ldots + (3n-2) = \frac{1}{2}n(3n-1).$$

(ii) For every positive integer n,

$$2^1 + 2^2 + \ldots + 2^n = 2^{n+1} - 2.$$

(iii) For every positive integer n, $\dfrac{n^3}{3} + \dfrac{n^5}{5} + \dfrac{7n}{15}$ is an integer.

(iv) For every positive integer n, $4^n - 1$ is divisible by 3.

(v) The sequence $a_0, a_1, a_2, \ldots$ defined by $a_0 = 0$, $a_{n+1} = \frac{a_n+1}{2}$ is bounded above by 1.

Exercise 2.5.5. Recall that, for a binary operation $*$ on a set A, associativity is defined as "for any x, y, z, $(x*y)*z = x*(y*z)$." Use induction to prove that for any collection of n elements from A put together with $*$, $n \geq 3$, any grouping of the elements that preserves order will give the same result.

Exercise 2.5.6. A *graph* consists of *vertices* and *edges*. Each edge has a vertex at each end (they may be the same vertex). Each vertex has a *degree*, which is the number of edge endpoints at that vertex (so if an edge connects two distinct vertices, it contributes 1 to each of their degrees, and if it is a loop on one vertex, it contributes 2 to that vertex's degree). It is possible to prove without induction that for a graph the sum of the degrees of the vertices is twice the number of edges. Find a proof of that fact using

(i) induction on the number of vertices;

(ii) induction on the number of edges.

Exercise 2.5.7. The *Towers of Hanoi* is a puzzle consisting of a board with three pegs sticking up out of it and a collection of disks that fit on the pegs, each with a different diameter. The disks are placed on a single peg in order of size (smallest on top), and the goal is to move the entire stack to a different peg. A move consists of removing the top disk from any peg and placing it on another peg; a disk may never be placed on top of a smaller disk.

Determine how many moves it requires to solve the puzzle when there are n disks, and prove your answer by induction.

2.5.2. Recursion. To define a class *recursively* means to define it via a set of basic objects and a set of rules, called *closure operators*,

allowing you to extend the set of basic objects. We give some simple examples.

Example 2.5.8. The natural numbers may be defined recursively as follows:

- $0 \in \mathbb{N}$.
- If $n \in \mathbb{N}$, then $n + 1 \in \mathbb{N}$.

Example 2.5.9. The *well-formed formulas* (wffs) in propositional logic are a recursively defined class.

- Any propositional symbol P, Q, R, etc., is a wff.
- If φ and ψ are wffs, so are the following:
  (i) $(\varphi \,\&\, \psi)$
  (ii) $(\varphi \vee \psi)$
  (iii) $(\varphi \to \psi)$
  (iv) $(\varphi \leftrightarrow \psi)$
  (v) $(\neg\varphi)$

The important fact, which gives the strength of this method of definition, is that we may apply the closure operators repeatedly to get more and more complicated objects.

For example, $((A\&B) \vee ((P\&Q) \to (\neg A)))$ is a wff, as we can prove by giving a construction procedure for it. A, B, P, and Q are all basic wffs. We combine them into $(A\&B)$ and $(P\&Q)$ by operation (i), obtain $(\neg A)$ from (v), $((P\&Q) \to (\neg A))$ from (iii), and finally our original formula by (ii).

Exercise 2.5.10. (i) Prove that $((A \vee (B\&C)) \leftrightarrow C)$ is a wff.

(ii) Prove that $(P \to Q(\vee$ is *not* a wff.

Exercise 2.5.11. (i) Add a closure operator to the recursive definition of $\mathbb{N}$ to get a recursive definition of $\mathbb{Z}$.

(ii) Add a closure operator to the recursive definition of $\mathbb{Z}$ to get a recursive definition of $\mathbb{Q}$.

Exercise 2.5.12. Give two different recursive definitions of the set of all positive multiples of 5.

Exercise 2.5.13. Write a recursive definition of the rational functions in x, those functions which can be written as a fraction of two polynomials of x. Your basic objects should be x and all real numbers.

We may also define functions recursively. For that, we say what $f(0)$ is (or whatever our basic object is) and then define $f(n+1)$ in terms of $f(n)$. For example, $(n+1)! = (n+1)n!$, with $0! = 1$, is *factorial*, a recursively defined function you've probably seen before. We could write a recursive definition for addition of natural numbers as follows.

$$a(0,0) = 0$$
$$a(m+1,n) = a(m,n) + 1$$
$$a(m,n+1) = a(m,n) + 1$$

This looks lumpy but is actually used in logic in order to minimize the number of operations that we take as fundamental: this definition of addition is all in terms of *successor*, the plus-one function.

Exercise 2.5.14. Write a recursive definition of $p(m,n) = m \cdot n$, on the natural numbers, in terms of addition.

Exercise 2.5.15. Write a recursive definition of $f(n) = 2^n 3^{2n+1}$, on the natural numbers.

Definition 3.3.1 gives a recursively defined set of functions in which one of the closure operators gives a function defined recursively from functions already in the set.

2.5.3. Induction Again. Induction and recursion have a strong tie beyond just their resemblance. Proving a property holds of all members of a recursively defined class often requires induction. This use of induction is less codified than the induction on $\mathbb{N}$ we saw above. In fact, the induction in §2.5.1 is simply the induction that goes with the recursively defined set of natural numbers, as in Example 2.5.8.

To work generally, the base case of the inductive argument must match the basic objects of the recursive class. The inductive step comes from the closure operations that build up the rest of the class. You are showing the set of objects that have a certain property contains the basic objects of the class and is closed under the operations of the class, and hence must contain the entire class.

Example 2.5.16. Consider the class of wffs, defined in Example 2.5.9. We may prove by induction that for any wff φ, the number of positions where binary connective symbols occur in φ (that is, $\&, \vee, \rightarrow$, and $\leftrightarrow$) is one less than the number of positions where propositional symbols occur in φ.

Proof. For any propositional symbol, the number of propositional symbols is 1 and the number of binary connectives is 0.

Suppose by induction that $p_1 = c_1 + 1$ and $p_2 = c_2 + 1$ for p_1, p_2 the number of propositional symbols and c_1, c_2 the number of binary connectives in the wffs φ, ψ, respectively. The number of propositional symbols in $(\varphi Q \phi)$, for Q any of $\vee, \&, \rightarrow$, and $\leftrightarrow$, is $p_1 + p_2$, and the number of connective symbols is $c_1 + c_2 + 1$. By the inductive hypothesis we see that

$$p_1 + p_2 = c_1 + 1 + c_2 + 1 = (c_1 + c_2 + 1) + 1,$$

so the claim holds for $(\varphi Q \psi)$.

Finally, consider $(\neg\varphi)$. Here the number of binary connectives and propositional symbols has not changed, so the claim holds. □

Exercise 2.5.17. The length of a wff is the number of symbols it contains, including parentheses. Suppose φ is a wff not containing negation (that is, it comes from the class defined as in Example 2.5.9 but without closure operation (v)). Prove by induction that for each such φ, there is some $k \geq 0$ such that the length of φ is $4k+1$ and the number of positions at which propositional symbols occur is $k + 1$.

If we are careful, we can perform induction on $\mathbb{N}$ to get results about other recursively defined classes. For wffs, we might induct on the number of propositional symbols or the number of binary connectives, for instance.

Exercise 2.5.18. Recall from calculus that a function f is *continuous at* a if $f(a)$ is defined and equals $\lim_{x\to a} f(x)$. Recall also the *limit laws*, which may be summarized for our purposes as

$$\lim_{x\to a}(f(x)\square g(x)) = (\lim_{x\to a} f(x))\square(\lim_{x\to a} g(x)), \quad \square \in \{+, -, \cdot, /\},$$

as long as both limits on the right are defined and if $\square = /$ then $\lim_{x\to a} g(x) \neq 0$. Using those, the basic limits $\lim_{x\to a} x = a$ and

$\lim_{x\to a} c = c$ for all constants c, and your recursive definition from Exercise 2.5.13, prove that every rational function is continuous on its entire domain.

Exercise 2.5.19. Using the recursive definition of addition from the previous section ($a(0,0) = 0$; $a(m+1,n) = a(m,n+1) = a(m,n)+1$), prove that addition is commutative (i.e., for all m and n, $a(m,n) = a(n,m)$).

2.6. Some Notes on Proofs and Abstraction

2.6.1. Definitions. Definitions in mathematics are somewhat different from definitions in English. In natural language, the definition of a word is determined by the usage and may evolve. For example, "broadcasting" was originally just a way of sowing seed. Someone used it by analogy to mean spreading messages widely, and then it was adopted for radio and TV. For present-day speakers of English I doubt the original meaning is ever the first to come to mind.

In contrast, in mathematics we begin with the definition and assign a term to it as a shorthand. That term then denotes exactly the objects that fulfill the terms of the definition. To say something is "by definition impossible" has a rigorous meaning in mathematics: if it contradicts any of the properties of the definition, it cannot hold of an object to which we apply the term.

Mathematical definitions do not have the fluidity of natural language definitions. Sometimes mathematical terms are used to mean more than one thing, but that is a re-use of the term and not an evolution of the definition. Furthermore, mathematicians dislike that because it leads to ambiguity (exactly what is meant by this term in this context?), which defeats the purpose of mathematical terms in the first place: to serve as shorthand for specific lists of properties.

2.6.2. Proofs. There is no way to learn how to write proofs without actually writing them, but I hope you will refer back to this section from time to time. There are also a number of books available about learning to write proofs and solve mathematical problems that you can turn to for more thorough advice.

A proof is an object of convincing. It should be an explicit, specific, logically sound argument that walks step by step from the hypotheses to the conclusions. Avoid vagueness and leaps of deduction, and strip out irrelevant statements. Be careful to state what you are trying to prove in such a way that it does not appear you are asserting its truth prior to proving it. More broadly, make sure your steps are in the right order. Often, a good way to figure out how to prove something is to work backwards, in steps of "I get this as a conclusion if this other property holds; this property holds whenever the object is of this kind; and oh, everything that meets my hypothesis is an object of that kind!" However, the final proof should be written from hypothesis to kind of object to special property to conclusion.

True mathematical proofs are very verbal, bearing little to no resemblance to the two-column proofs of high school geometry. A proof which is just strings of symbols with only a few words is unlikely to be a good (or even understandable) proof. However, it can also be clumsy and expand proofs out of readability to avoid symbols altogether. For example, it is important for specificity to assign symbolic names to (arbitrary) numbers and other objects to which you will want to refer. Striking the symbol/word balance is a big step on the way to learning to write good proofs.

Make your proof self-contained except for explicit reference to definitions or previous results (i.e., don't assume your reader is so familiar with the theorems that you may use them without comment; instead say "by Theorem 2.5, ..."). Be clear; sentence fragments and tortured grammar have no place in mathematical proofs. If a sentence seems strained, try rearranging it, possibly involving the neighboring sentences. Do not fear to edit: the goal is a readable proof that does not require too much back-and-forth to understand. There is a place for words like *would*, *could*, *should*, *might*, and *ought* in proofs, but they should be kept to a minimum. Most of the time the appropriate words are *has*, *will*, *does*, and *is*. This is especially important in proofs by contradiction. Since in such a proof you are assuming something that is not true, it may feel more natural to use the subjunctive, but that can make things unclear. You assume some hypothesis; given that hypothesis other statements *are* or *are not* true. Be bold and let

the whole contraption go up in flames when it runs into the statement it contradicts.

Your audience is a person who is familiar with the underlying definitions used in the statement being proved, but not the statement itself. For instance, it could be yourself after you learned the definitions, but before you had begun work on the proof. You do not have to put every tiny painful step in the write-up, but be careful about what you assume of the reader's ability to fill in gaps. Your goal is to convince the reader of the truth of the statement, and that requires the reader to understand the proof. Along those lines, it is often helpful to insert small statements (I call it "foreshadowing" or "telegraphing") that let the reader know why you are doing what you are currently doing, and where you intend to go with it. In particular, when working by contradiction or induction, it is important to let the reader know at the beginning. More complicated proofs, the kinds that take several pages to complete, often benefit from an expository section at the beginning, that outlines the proof with a focus on the "why" of each step. A more technical portion that fills in all the details comes afterward.

You must keep to the definitions and other statements as they are written. In fact, a good strategy for finding a proof of a statement is first to unwrap the definitions involved. Be especially wary of mentally adding words like *only*, *for all*, *for every*, or *for some* which are not actually there. If you are asked to prove an implication it is likely the converse does not hold, so if you "prove" equivalence you will be in error. Statements that claim existence of an object satisfying certain hypotheses may be proved by producing an example, but if you are asked to prove something holds of *all* objects of some type, you cannot do so via a specific example. Instead, give a symbolic name to an arbitrary object and prove the property holds using only facts that are true for all objects of the given type. Similarly, the term *without loss of generality*, or WLOG, appears from time to time in proofs to indicate a simplifying but not restrictive assumption. If you use the term make sure the assumption truly does not restrict the cases. For example, one may assume without loss of generality that the coefficient of x in the equation of a plane is nonnegative, since if

it is negative another equation for the same plane may be obtained by multiplying through by -1. It is *not* without loss of generality to assume the coefficient of x is 1, since the plane defined by $y + z = 0$, for example, has no equation with an x-coefficient of 1.

Exercise 2.6.1. Here are some proofs you can try that don't involve induction:

(i) $\neg(\forall m)(\forall n)(3m + 5n = 12)$ (over $\mathbb{N}$)

(ii) For any integer n, the number $n^2 + n + 1$ is odd.

(iii) If every even natural number greater than 2 is the sum of two primes, then every odd natural number greater than 5 is the sum of three primes.

(iv) For nonempty sets A and B, $A \times B = B \times A$ if and only if $A = B$.

(v) $\sqrt{2}$ is irrational. (Hint: work by contradiction, assuming there is a fraction of integers in least terms that equals $\sqrt{2}$.)