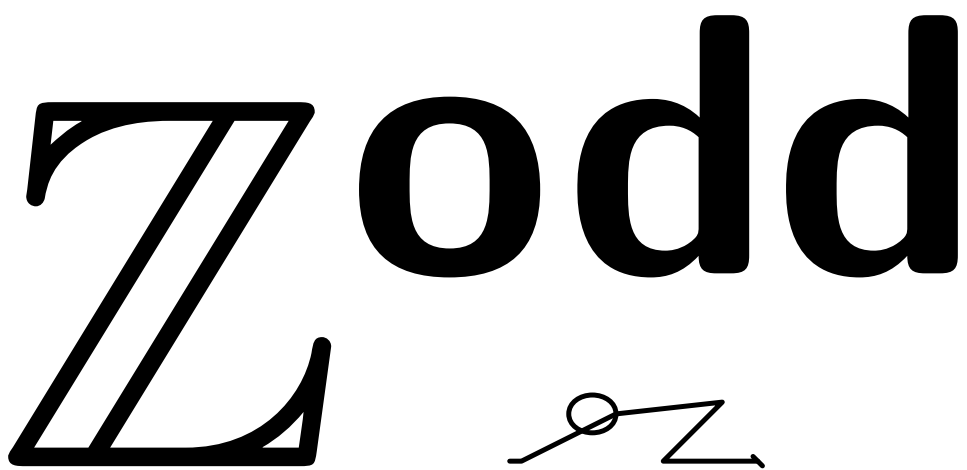


— Please — Please — **DO NOT Distribute!** — Thank You! — Thank You! —

Kneel Before Zodd



Discrete Structures
for
Computer Science

Lester I. McCann

— Draft — Draft — Draft — August 16, 2023 — Draft — Draft — Draft —

Preface

This is a collection of draft chapters of what may someday become a full discrete structures textbook. I roughly write roughly two rough chapters per summer, during those summers that I have time to write chapters, at which rate this book will be finished sometime after the year 2022. By the time it is finished, I expect that people will be even less inclined to read books than they are now, which suggests that I may be writing this only for myself. It's a good thing I'm happy with it so far.

While this book is still incomplete, I ask that you, dear student of CSc 245 in the Department of Computer Science at the University of Arizona, **do not** share it with anyone outside of the class. This means that you **may not** post it on your web page, upload it to the cloud, ship it to the New York Times to be reviewed, send it to a major motion picture studio, use it as the basis for a new political party, or accept it as the holy book of a fledgling religion.¹

You **may** attempt to learn from it. I'll grant you that doing so would be easier if I had spent some time writing exercises. The reality is that writing exercises is absolutely no fun, which explains why most discrete structures books have so few. Until creating them becomes fun, or at least becomes interesting enough to compete with cleaning bathrooms or staring vacantly out a window, this book won't come with any exercises. And no, I'm not accepting contributions.

These chapters were created with the help of the following software tools:

- Donald Knuth's T_EX document preparation system (<http://tug.org/>).
- Leslie Lamport's L^AT_EX macro collection (<http://latex-project.org>).
- John Hobby's MetaPost picture-drawing language (<http://ect.bell-labs.com/who/hobby/MetaPost.html>).
- L. Peter Deutsch's Ghostscript software suite (<http://www.ghostscript.com/>).
- GIMP, the GNU Image Manipulation Program (<http://www.gimp.org/>).
- A whole bunch of add-on L^AT_EX packages, including but not limited to:

¹These days, there are people who consider the last two to be the same thing.

- `graphicx` (<http://ctan.org/pkg/graphicx>).
- `mdframed` (<http://www.ctan.org/pkg/mdframed>).
- `marginnote` (<http://www.ctan.org/pkg/marginnote>).
- `hyperref` (<http://ctan.org/pkg/hyperref>).

Two notes about the effects of these packages. First, whenever there’s a reference a figure or example, the number of that item is actually a link to it. If you’re using a capable PDF viewer, you can click on the number and jump directly to the source – if the source is in the same file. Jumping back . . . well, you’ll have to handle that yourself. Second, `mdframed` allows the examples to span page breaks. As you read the chapters, you’ll probably see examples that just barely start on one page or just barely extend to the following page. This doesn’t look wonderful, and it can be avoided, but while this book is in draft form (that is, while I’m still making frequent tweaks to the content), the fix is more trouble than is the problem. No doubt such quirks of formatting will make you love discrete math all the more.

Whenever I remembered to do it, I included the \LaTeX commands used to create special symbols used in the chapters. These commands are generally provided with the first appearance of the symbol, especially if you consider the Math Review appendix to be Chapter 0. That’s not unreasonable; it was the first part to be written, in the summer of 2012.

The decision not to reset the example-numbering at the start of each chapter was intentional. On many occasions, students have emailed me to ask about ‘example n ,’ but they forgot to say which chapter’s example n they mean. That won’t happen in this book! Besides, having examples number into the hundreds looks really impressive.

Also intentional was my decision to close quotes before punctuation symbols when the quoting was for a term, a constant, or the like. If the quotes are for an actual quote, then I tried to follow the normal rules. For example, I’d write that the first term of a sequence is ‘3’, but would instead write, when quoting a trenchant observation, “Oh, what sad times are these when passing ruffians can say ‘Ni’ at will to old ladies.”²

Finally, I acknowledge that I have acquired permission to use the various images and quoted material found in this text from exactly none of the copyright holders. Why not? First, I’m using those images neither to make money nor to harm the copyright holders. Second, I’m distributing this book via a

²Roger the Shrubber, “Monty Python and the Holy Grail.” See? Normal quoting rules there, too. I’m not a total heathen!

password-protected web page so that the only people who see it are instructors and students of CSc 245 at the University of Arizona. Third, and most important, using images such as these in a work such as this is covered by the ‘fair use’ exception to the U.S. copyright laws, an exception that has long allowed instructors to use pieces of copyrighted material in their teaching for educational purposes.³

³So, kindly be educated (or at least amused) by them!

Contents

Contents	v
1 Logic	1
1.1 What is Logic?	1
1.2 Propositional Logic	2
1.3 Logical Connectives	5
1.4 More English \Leftrightarrow Logic Conversions	25
1.5 Logical Equivalence	29
2 Quantified Expressions	41
2.1 Propositions to Predicates	41
2.2 Evaluating Predicates	44
2.3 More Detail on Universal Quantification	47
2.4 More Detail on Existential Quantification	52
2.5 Mixing Universal and Existential Quantification	58
2.6 Generalized De Morgan's Laws	61
2.7 "Exactly n " Expressions	62
3 Logical Arguments	71
3.1 Reasoning	72
3.2 Categorizing Deductive Arguments	77
3.3 Rules of Inference for Propositions	80
3.4 Rules of Inference for Predicates	91
3.5 Fallacies	98
4 Direct Proofs	107
4.1 The Heartbreak of Probarephobia	107
4.2 Proof Preliminaries	109
4.3 Direct Proofs	116
4.4 Disproving Conjectures	132

5	Indirect (“Contra”) Proofs	137
5.1	Proof by Contraposition	137
5.2	Proof by Contradiction	141
5.3	Proving Biconditional Conjectures	148
5.4	Summary of Direct and Indirect Proof Techniques	151
6	Additional Set Concepts	153
6.1	What is So Important about Sets?	153
6.2	Logical Representation of Basic Set Operators	157
6.3	Some Useful Set Concepts	161
6.4	Set Identities	169
6.5	Choosing a Set Representation	177
7	Matrices	183
7.1	The Utility of Matrices	183
7.2	Matrix Fundamentals	185
7.3	Numeric Matrix Operations	193
7.4	Matrix Powers and the Identity Matrix	204
7.5	Matrix Operations in Orthographic Projections	214
7.6	Logical Matrices	218
8	Relations	229
8.1	(Binary) Relations	229
8.2	Creating Relations from Relations	232
8.3	Graph Representations of Relations	242
8.4	Four Properties of Binary Relations	245
8.5	Matrix Representations of Relations	262
8.6	Equivalence Relations	272
8.7	Partial Orders	278
8.8	Total Orders	286
A	Math Review	291
A.1	Fractions	291
A.2	Rational Numbers	292
A.3	Set Basics	294
A.4	Associativity, Friends and a Distant Cousin	300
A.5	Properties of Inequalities	303
A.6	Summation and Product Notations	306
A.7	Integer Division	310
A.8	Evens and Odds	312

A.9 Logarithms and Exponents	313
A.10 Working with Quadratic Equations	316
A.11 Positional Number Systems	321

Chapter 1

Logic

A major topic in most discrete structures books, this one included, is proof techniques. Proofs are arguments, arguments are built from inferences, inferences are implications, and an implication is an operator that applies to logical propositions. To understand proofs, we need to start with logic.

1.1 What is Logic?

In brief, *logic* is the art of intellectual persuasion, and comes in many forms. For our purposes, we consider the branches known as *philosophical* and *mathematical* logic.

logic

Definition 1: Philosophical Logic
<i>Philosophical Logic</i> is the study of thought and reasoning as expressed in natural languages.

philosophical logic

Definition 2: Mathematical Logic
<i>Mathematical Logic</i> is the use of formal languages to represent reasoning and computation.

mathematical logic

Neither of these definitions is universally accepted by philosophers or mathematicians, but they are adequate to show that logic as used in computer sci-

ence is not easily pigeonholed.¹ In this book, we will use concepts from both categories.

propositional logic

Our path to proofs will begin with *propositional logic*, also known as *sentential calculus*.² As the name suggests, the key idea is that of propositions, which are logical formulae, often compounded with logical operators, that evaluate to the values of true or false. Of particular use to us will be *first-order logic*. In FOL (also known as *(first-order) predicate calculus*, or FOPC), the formulae are propositions that can be augmented with quantified variables, as we will see in Chapter 2. The variables represent values drawn from specified domains.

first-order logic

For most of the rest of this chapter, we will consider variables to be ‘illegal’ within propositions. As mentioned above, we will be using them within quantified expressions, but that’s the next chapter.

second-order logic

First-order logic is all we need to present the material in this book, but logic doesn’t stop there. Immediately beyond FOL is *second-order logic* (SOL³). SOL extends FOL by permitting variables to represent entire sets, and even functions, rather than just values drawn from sets. There are even higher-order logics if SOL isn’t enough for your needs.

An example from programming might help make the FOL – SOL distinction more clear. Most programming languages allow you to (and often insist that you) declare variables before they are used. In C-like languages, you do this with a statement of the form `int a;`. In subsequent statements, `a` can hold values from the domain of integers. That’s also how variables work in FOL. Now imagine that you could declare a variable that can hold entire categories of numbers, rather than individual values. One moment, variable `b` represents all integers; the next, it represents all rational numbers. That sort of thing is possible with SOL variables.

1.2 Propositional Logic

A formal definition of ‘proposition’ is a good place to start:

¹Oooo, foreshadowing!

²‘Sentential’ from ‘sentence’ (another substitute for ‘proposition’), and ‘calculus’ meaning any calculation or computation technique (not just the one with derivatives and integration!)

³I know, I know; grow up!

Definition 3: Proposition

A *proposition* (alternatively, *statement*) is a claim that is either true or false within a given context.

proposition

We use ‘claim’ in this definition to make it applicable outside of the realm of mathematical expressions.

1.2.1 Simple Propositions

Propositions come in subtypes, of which the simplest is the aptly-named simple proposition.

Definition 4: Simple Proposition

A *simple proposition* is a proposition that includes no logical operators.

simple proposition

Distinguishing propositions from non-propositions can be a bit tricky. A few examples should help.

Example 1:

The following claims are propositions; they are either true or false:

- Radon is a noble gas. (True; one of the six that are natural)
- Bill Gates graduated from Harvard. (False; he dropped out)
- $3 * (4 * 5) = (3 * 4) * 5$ (True; by associativity)
- 9.95 is irrational. (False; it is rational)

These ‘claims’ are not propositions, for the reasons given:

- $2^{\log_8 x} = 4$ (We can’t evaluate its truth without a value for x)
- $a * b = b * a$ (What types are a and b ? Does ‘*’ apply to them?)
- Is your refrigerator running? (That’s a question, not a claim.)

- Don't open your present early! (A command, not a claim.)

The first two non-propositions suffer from a lack of context; the variables aren't explained. The last two aren't statements of fact; they do not evaluate to true or false. Please remember that a claim is a proposition when it evaluates to true or false. A proposition is still a proposition whether or not *you* know to which of those it evaluates! Your lack of knowledge does not change the categorization of the statement.

proposition labels

Giving propositions (simple or otherwise) a 'name' aids identification. These names are called *proposition labels* or *statement labels* and are traditionally single lower-case letters written ahead of the proposition and separated with a colon. For example, "r: Radon is a noble gas".

There are two acceptable schemes for choosing proposition labels. The first is to use, in order, the letters 'p', 'q', 'r', 's', etc.⁴ The second is to choose a letter that reminds you of the proposition. In the above example, 'r' is the first letter of 'radon'. Yes, we could use longer (more meaningful) names, but soon we will be creating long compound propositions for which short names are a significant convenience.

1.2.2 Compound Propositions

To represent more complex claims, we use compound propositions.

compound proposition

Definition 5: Compound Proposition

A proposition that is a logical combination of simple propositions is a *compound proposition*.

Definition 5 leads to an obvious question: "With what do we combine them?" The answer is, in part, familiar to generations of children who grew up watching the "Schoolhouse Rock" educational videos. ABC's 1973 three-minute grammar animation called "Conjunction Junction", although created to explain a specific part of English sentence structure, is also a nice introduction to two logical operators.⁵ Figure 1.1 might bring back some memories.

⁴Why start with 'p'? 'p' is for 'proposition'! It's in every one of those alphabet books for children ... or should be.

⁵You can see this and several others at www.youtube.com/user/SchoolhouseRockVids



Figure 1.1: ABC’s “Conjunction Junction” educational cartoon explained ‘and’ and ‘or’ to children using a train car metaphor. Credit: The Walt Disney Company.

1.3 Logical Connectives

This section introduces the commonly-used logical connectives⁶ (a.k.a. logical operators) shown in Table 1.

Table 1: Common Logical Connectives			
Connective	In Brief	Symbol(s)	\LaTeX
conjunction	and	$p \wedge q$	<code>\land</code>
inclusive disjunction	or	$p \vee q$	<code>\lor</code>
exclusive disjunction	xor	$p \oplus q, p \underline{\vee} q$	<code>\oplus, \veebar</code>
negation	not	$\neg p, \overline{p}$	<code>\neg, \overline{\{}}</code>
implication	if-then	$p \rightarrow q$	<code>\to</code>
biimplication	iff	$p \leftrightarrow q$	<code>\leftrightarrow</code>

Before we get to the connectives, we start with a useful tool for examining the situations in which propositions are true or false: Truth tables.

⁶Negation doesn’t do any connecting, as it does not bring together two propositions, but it is a logical operator, and as such is frequently grouped with the others.

1.3.1 Truth Tables

You can think of truth tables as interrogation tools used to get propositions to tell us everything they know. That is, their purpose is to show the complete set of possible evaluations for propositions. Knowing exactly when a proposition evaluates to ‘true’ is often quite useful.

<i>variables</i> \Rightarrow	<i>p</i>	<i>q</i>	$p \vee q$	$q \wedge (p \vee q)$	\Leftarrow <i>proposition sequence</i>
	T	T	T	T	
<i>all possible</i> \Rightarrow	T	F	T	F	\Leftarrow <i>proposition</i>
<i>truth value</i>	F	T	T	T	<i>evaluations</i>
<i>combinations</i>	F	F	F	F	

A truth table is divided into four quadrants, as illustrated in Table 2:

- **Upper-Left:** The variables used by the proposition being evaluated, often ordered to aid the evaluation of the proposition. Note that each variable heads its own column.
- **Lower-Left:** This quadrant lists all of the possible combinations of values that can be assigned to the variables, one combination per row. It is traditional in both logic and mathematics to use the symbols ‘T’ for ‘true’ and ‘F’ for ‘false’, and to order the rows such that the first row is all ‘true’, the last is all ‘false’, and in-between the values change in the same manner as binary digits are incremented. We can look at the pattern another way. The right-most variable’s column alternates between ‘T’ and ‘F’. The values of its neighbor to the left alternate also, but in groups of two (‘TT’, ‘FF’, ‘TT’, ‘FF’, etc.). If there is a third column, we alternate by groups of four (yes, this is based on powers of two).
- **Upper-Right:** Here we use a sequence of propositions to ‘build up’ to the proposition in question. Typically, each proposition adds an operator to the previous one. As we will demonstrate later, it is sometimes convenient to repeat a column to avoid making silly mistakes.
- **Lower-Right:** This is where we evaluate each proposition in the sequence, left to right, ending with the evaluation of the proposition in

question. Thus, the final (right-most) column is the one in which we are likely the most interested.

Categories of Propositions

The following definitions provide three categories of propositions, based on how often they evaluate to true and false.

Definition 6: Tautology

A *tautology* is a proposition that always evaluates to true.

tautology

Definition 7: Contradiction

A proposition that always evaluates to false is a *contradiction*.

contradiction

Definition 8: Contingency

A proposition that is neither a tautology nor a contradiction is known as a *contingency*.

contingency

Example 2:

Problem: Is the compound proposition $q \wedge (p \vee q)$ a tautology, a contradiction, or a contingency?

Solution: It's a contingency, as Table 2 shows. The column of boolean values beneath the proposition includes both true and false, which makes $q \wedge (p \vee q)$ a contingency.

Truth tables are excellent tools for categorizing propositions, because they show all possible evaluations. Tautologies and contradictions can also be determined by applications of logical equivalences; keep reading!

1.3.2 Conjunction

Consider this English sentence:

Amit bought a tablet and installed a chess app.

This sentence contains two separate propositions (*Amit bought a tablet* and *[he] installed a chess app*). We know that these are propositions because they are either true or false — Amit either bought a tablet or he did not, and he did or did not install a chess app. The ‘and’ between them makes the complete sentence a *conjunction*, ($p \wedge q$, `\land`) which is the type of compound proposition that is true only when both participating propositions are true. The truth table for conjunction can be found in Table 3, along with those of the other common connectives.

conjunction

Table 3: Truth Tables of the Common Logical Connectives

p	q	<i>negation</i> $\neg q$	<i>conjunction</i> $p \wedge q$	<i>inclusive disjunction</i> $p \vee q$	<i>exclusive disjunction</i> $p \oplus q$	<i>implication</i> $p \rightarrow q$	<i>biimplication</i> $p \leftrightarrow q$
T	T	F	T	T	F	T	T
T	F	T	F	T	T	F	F
F	T	-	F	T	T	T	F
F	F	-	F	F	F	T	T

Turning an English sentence into the logical notation shown in Table 3 is a four-step process: (1) Identify the simple propositions, (2) assign proposition labels to them, (3) identify the logical operators, and (4) make all of the substitutions. Example 3 demonstrates this process.

Example 3:

Problem: Convert to logical notation this sentence: *Bobby found, bought, and wrapped the present.*

Solution: As with the first example, this sentence is missing some parts. As English speakers, we are used to this, and can insert the missing parts if necessary. We will do that here to make the conversion steps

plain: *Bobby found the present and Bobby bought the present and Bobby wrapped the present.* This is awkward but correctly-structured English.

Step 1: Identify the simple propositions. Having expanded the original sentence, the three simple propositions are easy to see: *Bobby found the present, Bobby bought the present, and Bobby wrapped the present.*

Step 2: Assign proposition labels. To select meaningful labels, look at the subjects, verb, and objects of the propositions. Here, the subjects and objects ('Bobby' and 'present') are the same in all three, but the verbs all start with different letters, making our label choices easy:

f : Bobby found the present
 b : Bobby bought the present
 w : Bobby wrapped the present

Step 3: Identify the logical operators. This step is not much of a challenge, as we have covered only one: conjunction. With three simple propositions, we need two conjunctions to tie them together, and we have them.

Step 4: Make all of the substitutions. Replacing the English propositions with our labels and using ' \wedge ' for 'and' produces the answer.

The logical notation version of *Bobby found, bought, and wrapped the present* is $f \wedge b \wedge w$, using the labels defined above.

1.3.3 Inclusive and Exclusive Disjunction

Consider this compound proposition:

He won the game by defeating the boss or by collecting all of the relics.

What are the circumstances under which this claim could be true? If he defeated the boss, but missed a relic, we would consider this to be true, because the connective ('or') allows for either one to be true. Similarly, if he found all of the relics but let the boss win, the claim is still true because, again, one part of it is true. If neither action took place, we would consider the statement to be false – there were two ways to win, and he didn't achieve either of them.

inclusive disjunction
exclusive disjunction

That leaves one option: What if both parts are true? If the ‘or’ is an *inclusive disjunction*, ($p \vee q$, `\lor`) then yes, we consider the claim to be true. But if the ‘or’ is an *exclusive disjunction*, ($p \oplus q$, `\oplus`) the claim is considered to be false. The names provide an easy way to remember this distinction: inclusive disjunction includes the case where both parts are true, but exclusive disjunction excludes that case. Table 3 shows that this is the only difference between these two types of disjunction.

Unfortunately, in English the ‘or’ by itself does not tell the reader which interpretation is correct. Sometimes, we can tell by context. In other situations, adding a few extra words makes the difference.

Example 4:

Adding “or both” to the end of a disjunction tells the reader to assume inclusive disjunction: *You can type your ID number or your full name, or both.* Similarly, “but not both” means that exclusive disjunction is intended. Prefixing the statement with “either” gives a hint of exclusivity, but still leaves the reader in some doubt: *Either rake the yard this afternoon or this evening.* Can the raker do half of the yard in the afternoon and the other half in the evening? Even with the addition of ‘either’, it’s hard to tell. Ah, the joys of natural language! Sometimes, we have to ask for additional clarification to get the meaning correct.

Example 5:

Problem: Determine the types of disjunction used in each of these sentences:

- (a) Ramone or Elias scored the winning hockey goal.
- (b) Fill in the circles with pen or pencil.
- (c) The meal includes mashed or french-fried potatoes.

Solution: (a) is exclusive; the scorekeeper must credit the goal to just one player. (b) is inclusive; as the type of mark does not matter, a combination is acceptable. (c) is likely intended to be exclusive, though it is possible the waiter could arrange for you to receive some of each variety.

Example 6:

Problem: Why is Wei late? Express the answer *Wei overslept or missed the 8 a.m. bus* in logic.⁷

Solution: Using the four-step process given above (see page 8 and demonstrated in Example 3):

1. The simple propositions are *Wei overslept* and *Wei missed the 8 a.m. bus*.
2. o : Wei overslept m : Wei missed the 8 a.m. bus
3. It's possible that Wei was late for both reasons, making inclusive OR the best disjunction option here.
4. In logic: $o \vee m$.

This is a good time for a word to the wise: *Do not 'over-think' the meanings of propositions*, especially not in this book! When people learn about converting from English to logic, they sometimes let their imaginations get the best of them, and start inventing interpretations that lead to indecision and ultimately an indefensible answer.

Example 7:

Problem: True or False: The sky is blue with one bright light source or black with lots of them.

'Solution': "Hmmm. Sounds like an exclusive 'or'; it is either daylight (blue sky) or night (black). But there are two transition periods, with the sky moving between blue and black. Then again, at different places on Earth, the sky can be blue and black at the same time, which is inclusive. And is the moon a light source? Sure, it only reflects sunlight, but it does reflect it well enough to be seen when the sky is blue, so that

⁷Might as well; Wei's not here yet, so you've got time to kill.



Figure 1.2: “Need to type up your discrete structures homeworks? Feel free to use any typeface, point size, and symbols you wish . . . so long as this keyboard⁸ has ‘em.” Credit: Wikimedia Commons

could be a second light source. Wait – if we were on the moon, the sky is always black; it cannot be blue at all! And what about alien planets with sulphuric atmospheres, perpetual clouds and two suns? Boy, what a stupid question . . . I’ll just say ‘Yes.’”

Solution: An answer of ‘Yes’ is unacceptable when the question asks for ‘true’ or ‘false’! Go with the most straight-forward interpretation: True, by exclusive ‘or’. We are not trying to trick anyone with our questions.

1.3.4 Negation

unary connective

binary connective

As shown in Table 3, negation takes just one operand, making its truth table half the size of the others. It is a *unary connective*, unlike conjunction and disjunction, which take two operands and are therefore termed *binary connectives*.

Over time, several different notations for negation have been used. In the days of typewriters, the set of available symbols was quite limited (see Figure 1.2), and as a result most of them had multiple meanings. Combine that with the limits on the exchange of ideas, and it is not hard to see how variety would flourish. In the case of negation, the modern notations are the

⁸This is a picture of an old European QWERTZ typewriter keyboard. Yes; QWERTZ, not QWERTY. Seriously!

negation symbol ($\neg p$, `\neg`) and the *overline* or *overbar* (\bar{p} , `\overline{}`). Older negation notations that are still used occasionally include $\sim p$ and p' .⁹

Example 8:

Problem: Use a truth table to show that $(p \vee q) \vee \bar{p}$ is a tautology.

Solution: Recall that a tautology always evaluates to true. We can use a truth table to show that, regardless of the combinations of true and false that are assigned to p and q , the evaluation of that expression will be true.

p	q	$p \vee q$	\bar{p}	$(p \vee q) \vee \bar{p}$
T	T	T	F	T
T	F	T	F	T
F	T	T	T	T
F	F	F	T	T

Occasionally, students go a little crazy when they negate propositions. Example 9 demonstrates both right and wrong ways to do it, and Example 10 shows that not all propositions have easily-constructed negations.

Example 9:

Problem: Let m label the proposition *The mountain is tall*. In English, express \bar{m} .

Correct Solution: *The mountain is not tall*. The safest way to express the opposite of ‘tall’ is by saying ‘not tall’.

A Probably Acceptable Solution: *The mountain is short*. ‘Short’ is usually considered to be an antonym of ‘tall’, making this an acceptable negation in most situations. Trouble occurs when you start asking annoying questions such as, “What about ordinary, average-height mountains?”

⁹ \sim is called a *tilde* (`\sim`)

Incorrect Solution: The valley is tall. The characteristic of mountains known as ‘height’ is what we need to negate. Changing the subject does not address the problem.

Another Incorrect Solution: The mountain was tall. Changing the verb tense does not address the problem, either.

Example 10:

Problem: Negate the proposition *Education funding has decreased and test scores are down.*

Solution: Let p represent the first simple proposition and q the second. We know that the truth table of $p \wedge q$ reads T, F, F, F from top to bottom (see Table 1 if you need to refresh). The negation $[\neg(p \wedge q)]$ needs to evaluate to F, T, T, T to be a complete negation.

If we try *Education funding has increased and test scores are up* as the negation, we have two problems. First, is ‘increased’ the negation of ‘decreased’? How does ‘staying the same’ fit in? (The same can be said for ‘up’ and ‘down’.) We will take our earlier advice and not overthink these. Second, the truth table column for this version $[\neg p \wedge \neg q]$ is F, F, F, T instead of the needed F, T, T, T.

As we will learn in Section 1.5.2, a pair of logical equivalences known as De Morgan’s Laws tell us that the correct negation (still ignoring the annoying ‘stay the same’ possibility) is: *Education funding has increased or test scores are up* $[\neg p \vee \neg q]$.

A classic way to express negation in English is to prefix the statement with “It is not the case that”. If we try that on the proposition from Example 10, we produce: *It is not the case that education funding has decreased and test scores are down*, but what is being negated is not clear. Are we negating just the first simple proposition or the entire compound proposition? Natural languages are notoriously imprecise.

Something else to remember: It is very easy to overlook an overbar, particularly one that covers an entire compound proposition. Be sure to watch for them!

1.3.5 A Digression: Well-Formed Formulae

It is likely that you have typed a mathematical expression into an assignment statement of a program, attempted to compile it, and had the compiler complain that your expression is malformed. Chances are, the compiler complained because your expression was not a correctly structured. A *well-formed formula* (frequently abbreviated ‘*wff*’) is a grammatically-correct expression of the intended language. Most often, the term is used in mathematics, logic, and programming, but it applies to any language usage, even the construction of English sentences.

well-formed formula

Example 11:

Consider this assignment statement, which follows the syntax of a variety of programming languages:

```
result = 8 + * 9;
```

`8 + * 9` is not a well-formed formula.¹⁰ Multiplication of integers is a binary operation, and in a language that uses infix notation (that is, nearly all of them), the proper form is *operand * operand*. The 9 is the second operand of the multiplication operator, but the first is missing.

Compound logical propositions are statements of a language, just as arithmetic expressions are. Fortunately, the grammatical rules of the language of logic are much the same as those of arithmetic and algebra. Briefly, assuming that p and q are propositions, then based on the logical operators we’ve seen so far:

1. If p and q are propositions, then they are wffs.
2. If r and s are wffs, then so are (r) , $\neg r$, $r \wedge s$, $r \vee s$, and $r \oplus s$.

¹⁰However, `foo = 8 * + 9;` is usually considered to be well-formed. If you don’t see why, put it into a program and look at the value assigned to `foo`.

Example 12:

Problem: Show that $\overline{t \oplus u}$ is a wff in logic, given that t and u are propositions.

Solution: By (1), above, t and u are wffs. By (2), $t \oplus u$ is a wff. And, also by (2), so is $\overline{t \oplus u}$.

Example 12's demonstration shows the foundation of the process used by programming language compilers to check that a program is a correct expression of that language.

1.3.6 Implication (a.k.a. Conditional Proposition)

Programming languages provide selection statements that allow computers to execute one or more commands based on the evaluation of a condition. In most languages, the form is something like `if <condition> <action(s)>`, which mimics one of the forms conditionals take in English. The interpretation is much like the English version as well: The action(s) are performed only when the condition is true, and ignored when it is false.

Take this English sentence:¹¹

If you start the car, then the engine will run.

We can convert this compound proposition to logic as we did with our previous examples. Here we again have two (simple) propositions (s : *you start the car* and e : *the engine will run*). The symbol for *implication* is a right-pointing arrow ($s \rightarrow e$, `\LATEX: \to`). Rewriting this is easy; understanding how to interpret the logic is harder.

implication

Assume that both s and e are true; that is, you do start the car, and as a result the engine does run. You would accept that $s \rightarrow e$ is true. Now consider that the engine does not run (e is false) even though you started the car. You would have no trouble accepting that the sentence is false. But how do we interpret the truth of the statement when s is false? Certainly, we do not expect the engine to spring to life all by itself.¹² More to the point, is the entire *if you start the car, then the engine will run* statement a true statement when we do not have a chance to see it in action? The answer

¹¹... please! (en.wikipedia.org/wiki/Henny_Youngman)

¹²Outside of cartoons and horror movies, that is.

is, perhaps surprisingly, ‘yes’! We have no justification for saying that the statement is false, so we accept that it is true. This assumption of truth is known as *vacuous truth*.

Definition 9: Vacuous Truth

Given a statement that can be expressed as an implication $p \rightarrow q$, the statement is *vacuously true* when p is false.

vacuous truth

In English, ‘vacuous’ is synonymous with ‘empty,’ explaining why vacuous truths are sometimes known as empty truths.

Example 13:

Question: Is the statement “If Godzilla is an astronaut, he is wearing blue ear plugs” vacuously true?

Answer: Godzilla is a fictional monster. As such, the set of astronaut Godzillas is empty. Because “Godzilla is an astronaut” is false, and because $\mathbf{F} \rightarrow \text{anything}$ is true, yes, the statement is vacuously true.

You could just memorize the implication truth table (which can be found in Table 3), but looking at `if` statements in programming languages provides another perspective on this interpretation. When you write a program and the computer translates it (to machine language or bytecode or whatever), the translator verifies that the structure of the code meets the specifications of the language. That is, your `if` statement is deemed to be ‘correct’ or ‘true’, even though you have yet to execute the program. Similarly, in English, you can check that the sentence syntax is correct without knowing the meaning of all of the words in it, so long as you know their roles (noun, verb, etc.).

The ‘if’ part and the ‘then’ part of implications go by a variety of pairs of names, as shown in Table 4:

Table 4: Names of p and q in “if p , then q ”

	p	—	q
(a)	hypothesis	—	conclusion
(b)	antecedent	—	consequent
(c)	sufficient	—	necessary

Avoid the temptation to “mix and match” the terms in Table 4. That is, do not pair ‘sufficient’ with ‘consequent’, or people might wonder about your education.

English, being a language with much flexibility, supports many ways of writing “if p , then q ”. Table 5 covers nineteen (including the ‘when’ and ‘whenever’ substitutions), but even so it is by no means an exhaustive list.

Table 5: Some Other Ways of Saying “if p , then q ”

- | | |
|-------------------------------|------------------------------|
| (a) if p , q | (f) q if p |
| (b) p implies q | (g) q is implied by p |
| (c) p infers q | (h) q follows from p |
| (d) p only if q | (i) q unless \bar{p} |
| (e) p is sufficient for q | (j) q is necessary for p |

NOTE: You can replace the word **if** with **when** or **whenever** without changing the logical interpretation.

Example 14:

Problem: Express $(\bar{o} \wedge c) \rightarrow a$ in conversational English using the q is sufficient for p form, given that:

- o : Wei oversleeps
- c : Wei catches the 8 a.m. bus
- a : Wei attends the meeting

Solution: The negation of *Wei oversleeps* is *Wei does not oversleep*. Dropping the rest of the propositions in place, adding the operators, using the requested form, and making tweaks to the English to make it read (reasonably) conversationally, we get: *Wei not oversleeping and catching the 8 a.m. bus is sufficient for him to attend the meeting.*

Let's face it: It's really hard to make sentences that use *is sufficient for* and *is necessary for* beautifully conversational!

Example 15:

Problem: Rewrite each of these English expressions of implication in “if antecedent then consequent” form:

- (a) You'll hurt your back if you lift that box.
- (b) The light comes on only if the circuit is not open.
- (c) An installed battery is necessary for the remote control to work.

Solution: Part (a) is straight-forward; it matches the “consequent if antecedent” form. In if-then form: *If you lift that box, then you'll hurt your back.*

Part (b) is a bit more complex. There are four forms in Table 5 that include the word “if”. In just one of those does the hypothesis not immediately follow the “if”, and that is the “only if” case. Be mindful of that word “only”! With that in mind, the hypothesis and conclusion are already in the proper orientation for an if-then: *If the light comes on, then the circuit is not open.* Notice that the “not” just stays where it is; compound propositions can serve as hypotheses and conclusions, too.

Part (c) demonstrates that just because you can use a particular form does not mean that you should. Outside of nerdy sitcoms and logic classes, very few people talk like that (that is, the form is not very conversational). Happily, we can convert it to something more reasonable: *If the remote control works, then a battery is installed.* Do not worry about matching verb tense and word arrangements within the component propositions, so long as the result has the same basic meaning and is correctly-structured English.

“Only if” sentences, such as part (b) of Example 15, often raise a question: What is wrong with putting the pieces in the other order? In this case, that would produce the sentence *if the circuit is closed, then the light comes on*, which sounds pretty reasonable. But is anything else needed for the light to

come on? Sure; for starters, the filament (or LED or . . .) must not be broken and the power source must supply adequate power. That is, the circuit being closed is not sufficient by itself for the light to come on. The given answer (*if the light comes on, then the circuit is closed*) is logical; given that the light came on, we can reasonably conclude that the circuit was closed.

People sometimes play on the flawed understanding some people have of conditionals to influence their opinions. The practice of “push” polling in politics is a classic example.

Example 16:

Ahead of the 2000 U.S. presidential primary in South Carolina, a telephone poll (apparently authorized by the George W. Bush campaign team) asked this question: *Would you be more likely or less likely to vote for John McCain for president if you knew he had fathered an illegitimate black child?* The poll was designed to suggest to ill-informed voters that McCain’s adopted Bangladeshi daughter, Bridget, was biologically his. Notice the ‘if’ in the question. There is no claim that such an act occurred, but listener susceptible to suggestion could be left with that impression. You know better: First, you know that the truth of a conditional depends on the truth of the antecedent, and second, you know that questions are not propositions.

This survey technique is known as “push” polling because it attempts to sway (‘push’) the listener to a particular point of view. Signatories of the American Association of Political Consultant’s Code of Ethics are prohibited from using “false or misleading attacks”.

Converse, Inverse, and Contrapositive

Three adjustments to the basic $p \rightarrow q$ implication have enough utility to be worth special mention.

converse

- The *converse* of $p \rightarrow q$ is the implication $q \rightarrow p$. To form the converse, just swap the hypothesis and the conclusion.

inverse

- The *inverse* of $p \rightarrow q$ is the same implication with both the antecedent and the consequent negated. That is, the inverse of $p \rightarrow q$ is $\bar{p} \rightarrow \bar{q}$.

- Finally, the *contrapositive* is the original implication with both converse and inverse applied to it – the hypothesis and conclusion are each negated and swap positions. Symbolically, the contrapositive of $p \rightarrow q$ is $\bar{q} \rightarrow \bar{p}$.

contrapositive

Many people find inverse and converse to be hard to keep straight. Here is a mental trick that might help: *Inverse includes negation*. Because contrapositive includes both adjustments, it is easy to distinguish from the other two.

Why are these worth knowing? Apart from jokes about athletic footwear, archaic ways of describing mating, and bad puns,¹³ they are useful examples when discussing logical equivalence, as we will see in Examples 24 and 32.

1.3.7 Biimplication (a.k.a. Biconditional Proposition)

Does this sentence sound pretentious to you?

I will eat truffles if and only if they are in a French wine reduction.

The topic certainly raises the pretentiousness level, but what about that “if and only if” part? Why not just say “if” once instead of reiterating it?

The answer is that “if and only if” actually is not repeating the “if”. Rather, it is meant to be taken literally: It is a combination of the “if” and the “only if” expressions of implication. We can see the logical interpretation by breaking down the example. Assume that t labels *I will eat truffles* and w labels *[the truffles] are in a French wine reduction*.

if	and	only if	
t if w		t only if w	← as separate propositions
if w , then t		if t , then w	← in if-then form
$(w \rightarrow t)$	\wedge	$(t \rightarrow w)$	← converted to logic notation

Now the meaning is (hopefully!) clear: “if and only if” combines two implications into one statement, providing a compact way of writing this form of compound proposition. It may sound pretentious, but it is really just a shortcut.¹⁴ This logical operator is known as a *biconditional* or a *biimplication*,

biconditional

¹³Converse is an athletic apparel company, and a rarely-used word to describe sex. (“No worries, Mom; we . . . conversed last night, that’s all. It was logic homework, I swear!”) As for puns, how about “A negative poet writes inverse”?

¹⁴Now you can be pretentious, educated, and lazy all at the same time!

biimplication

and is represented with the symbol \leftrightarrow (`LATEX`: `\leftrightharpoon`). Another way of writing *if and only if* in English is with the abbreviation *iff*,¹⁵ which you may have seen in other mathematics classes because that is where it is most often used.¹⁶

Table 3 (all the way back on page 8) shows that biimplication is true only when the operands match (sometimes stated as “have the same parity”). That is, a biconditional is true when its arguments are both true or when they are both false. Remember that you do not have to rely on your memory of this fact; if need be, you can reconstruct the above derivation and then create the truth table for $(p \rightarrow q) \wedge (q \rightarrow p)$.

People frequently use a conditional expression when a biconditional would correctly give more useful information, as Example 17 demonstrates.

Example 17:

Problem: Which is more helpful: $x = 2y$ if $y = x/2$, or $x = 2y$ iff $y = x/2$, when x and y are integers?

Solution: First, both are correct; you can safely state either one. The difference is that the first version is just half of the second; that is, the second includes the first, and adds *if $x = 2y$, then $y = x/2$* . As both parts are correct over the integers, the second version provides more information and it therefore more helpful.

1.3.8 Precedence and Associativity of Logical Connectives

You are likely familiar with the ideas of operator precedence and associativity from mathematics and the programming languages you have learned. *Precedence* tells us which of a group of operators to perform first in an expression when no parentheses are provided, and *associativity* provides an evaluation order when the operators have the same precedence.

Most discrete structures books provide a precedence table for the logical operators. They agree on the rough structure (negation has highest precedence, followed by conjunction and disjunction, with the forms of implication

¹⁵Speaking of pretentious: See how the two f’s in ‘iff’ are joined? In typography, that’s called a *ligature*. `LATEX` uses them automatically, but that behavior can be overridden.

¹⁶In `LATEX`, `\iff` produces the symbol \iff instead of \leftrightarrow . Some people like the double-line arrows for implication and biimplication. We like the simpler single-line arrows.

precedence
associativity

next), but they often disagree on the smaller details (e.g., does conjunction have higher precedence than disjunction, or are they the same?). Instead of adding to the confusion, for this book we adopt a simple rule: *Use parentheses to avoid confusion.*

Associativity is much less troublesome: *All of the binary logical connectives are left-associative.* That is, assuming that \diamond is a left-associative binary logical connective, when evaluating the expression $a \diamond b \diamond c$, $a \diamond b$ (the left-most operator) is evaluated first. To force right-associativity, add parentheses, as in $a \diamond (b \diamond c)$. Negation is naturally right-associative. Given the expression $\neg\neg p$, the only way it can be evaluated is to perform the right-most negation first; if we try to make it left-associative, we would be using a negation operator as the operand to the other negation operator.

1.3.9 A Digression: Logical Bit-Operators in Programming Languages

So-called “systems” programming languages (that is, languages that are designed to create programs that interface with hardware more so than with people) include operators that consider individual bits to be their operands. Of course, most computers work with memory in larger ‘chunks’ (words) than individual bits, but these operators still apply to the component bits of those words.

The C family of programming languages (which includes C++ and Java) provides several bit-level operators. Due to the fact that bits have just two possible values (0 and 1, or ‘off’ and ‘on’, or ‘false’ and ‘true’), these operators include negation, conjunction, and disjunction.¹⁷

Table 6 lists and demonstrates four of the logical operators as they appear in C as bit-level operators, along with two bit-shifting operators as examples of additional bit-level operators. The right-most column shows most clearly how the corresponding pairs of bits from the operands are processed by the binary operators.

Table 6: Logical Bit Operators in C-family Languages

¹⁷As we will soon see, implication is a composite operator – we can build it from disjunction and negation – and so programming languages do not need to provide it as a separate operator.

Operator	Name	Example (Dec.)	Example (Binary)
\sim	Complement	$\sim 12 = -13$	$\sim 00001100 = 11110011$
$\&$	AND	$12 \& 10 = 8$	$\begin{array}{r} 1100 \\ \& 1010 \\ \hline 1000 \end{array}$
$ $	OR	$12 10 = 14$	$\begin{array}{r} 1100 \\ 1010 \\ \hline 1110 \end{array}$
\wedge	XOR	$12 \wedge 10 = 6$	$\begin{array}{r} 1100 \\ \wedge 1010 \\ \hline 0110 \end{array}$
\gg	Shift Right	$33 \gg 1 = 16$	$00100001 \gg 1 = 00010000$
\ll	Shift Left	$33 \ll 2 = 132$	$00100001 \ll 2 = 10000100$

Those of you who have programmed with a C-family language may have wondered why logical AND, for example, is represented by a pair of ampersands rather than a just one. Table 6 answers the question: The single-ampersand version is used for bit-level conjunction, and the double-ampersand version is used for variable-level conjunction. If you are wondering why $\sim 12 = -13$ instead of -12 and how shifting bits can be useful, those topics are often covered in computer architecture classes.¹⁸

Knowledge of bit-wise logical operators is useful in some interesting places. Example 18 introduces one such situation.

Example 18:

UNIX and UNIX-like operating systems maintain a 9-bit default permission value (frequently $110\ 110\ 110$, or 666_8) and each process maintains another bit string called the *umask*. Together, these bit patterns are used to give new files created by the process initial access permissions. Here's how.

Say that the default permission string is as stated above and the umask value is 037_8 ($= 000\ 011\ 111$ in binary). The result of the bit-wise AND-ing of the complement (a.k.a. negation) of the umask to the default

¹⁸Too much work? Fine; be that way: Computers usually use a representation called *two's complement* for signed integers, and shifting is an efficient way to multiply and divide by powers of two. You and <http://www.wikipedia.com> can take it from there.

permission string is the permission string assigned to newly-created files. In this case:

	000	011	111	[umask]
	111	100	000	[complement of umask]
&	110	110	110	[default permissions]
	110	100	000	[the file's permissions]
	rw-	r--	---	[as reported by <code>ls -l</code>]

What does this mean? The new file can be read or written by the file's owner, can be read (but not written to) by users with group access, and is inaccessible to anyone else. Users can adjust their command shell's umask. Now you know how to choose a suitable value.

1.4 More English \Leftrightarrow Logic Conversions

Remember that we are studying the basics of logic to prepare for proofs. Statements that we wish to prove correct are sometimes expressed in logic or mathematical notation for us, but they can also be expressions in natural languages (like English). Before we can apply logical principles to the statements, we have to convert them to logic to make them easier to work with and to solidify their meanings. Converting from logic to a natural language is an equally-important skill. It is good to be able to produce a logical result, but it is even better to be able to explain the meaning of the result to those not versed in logical notation.

In section 1.3.2 (see Example 3 in particular), we showed how to convert from English to logic using a four-step process: (1) Identify the simple propositions, (2) assign proposition labels to them, (3) identify the logical operators, and (4) make all of the substitutions. Examples 19 and 20 further demonstrate the challenges of English to logic conversions.

Example 19:

Problem: Express this sentence in logic notation: *He loses the election if he loses Ohio or Virginia.*

Solution: Steps 1 and 2 identify the component (simple) propositions and label them. When negations are involved, think positively. That is, state the propositions without negations. (Why? Because negations are operators, and that's Step 3.) Following this philosophy, we have three simple propositions – e : *He wins the election*, o : *He wins Ohio*, and v : *He wins Virginia*.

Now for the operators: We clearly have an implication (the 'if' in the middle) and a disjunction (the 'or' between the states) – but which variety of disjunction is it? Presumably, if the candidate loses both states he will lose the election by an even wider margin, making inclusive disjunction the reasonable choice. Another question: How many negations do we have, two or three? One applies to e , but is there just one covering the disjunction of o and v , or is there one negation for each proposition? We can answer this by expanding the antecedent from *he loses Ohio or Virginia* to *he loses Ohio or he loses Virginia*. Three negations it is.

At last, step 4 produces the final answer: $(\neg o \vee \neg v) \rightarrow \neg e$.

You may be wondering if the 'one negation vs. two' issue even matters. It does, as a truth table would make clear. $\bar{o} \vee \bar{v}$ and $\overline{o \vee v}$ differ when o and v have different values.

Example 20:

Problem: Express this sentence in logic notation: *The sum of two odd integers is also an odd integer.*

Solution: This problem is challenging for two reasons: First, it deals with math, which we have yet to try to express in logic. Second, it is missing all of our logic keywords! This is the sort of sentence we will see when we discuss proofs.

To express this sentence in logic, we need to rewrite it so that the propositions and logical connectives are clear. When working with math, it can help to add variables. In this example, we can introduce two variables (i and j will do) to represent the odd integers being added. This means

that our propositions are $p: i \in \mathbb{Z}^{odd}$, $q: j \in \mathbb{Z}^{odd}$, and $r: (i + j) \in \mathbb{Z}^{odd}$.

Now to connect them. The implicit assumption is that both i and j are provided to us; given them, we add them and claim that the result is odd. That is, we are assuming that they are odd integers – *if* they are both odd, *then* their sum is odd. ‘Both’ suggests conjunction, and the if-then is implication. Thus, we can rewrite the original sentence: *If i and j are odd integers, then $i + j$ is an odd integer.* In this form, and having already identified the propositions and connectives, the conversion to logic is straight-forward: $(p \wedge q) \rightarrow r$.

You may be wondering if $p \wedge q \wedge r$ is an acceptable alternative to $(p \wedge q) \rightarrow r$ in Example 20. It is not. The sum is a result of a process (here, addition), not something that is also provided to us. As there is a conclusion to be reached, implication best fits this situation.

When converting from logic to English, the trick is to make the result conversational; that is, to write the English version so that it sounds as though it were written for a person by a person, rather than by a mechanical conversion process, while retaining the correct meaning. Example 21 should help make this distinction clear.

Example 21:

Problem: Express this logical expression in conversational English: $\overline{\overline{i} \wedge \overline{u}}$, where i : *education funding has increased* and u : *test scores are up*

Solution: You may have noticed that this is a slightly-restructured version of Example 10. In particular, notice that there is an overline over the entire expression. Expressing in conversational English the negations of i and u is easy if we aren’t too picky, as we have already seen in that earlier example: \overline{i} : *education funding has decreased* and \overline{u} : *test scores are down*.

Expressing the negation of the conjunction is more challenging. The prefix phrase *it is not the case that* is the lazy (and not very conversational) solution for expressing a negated proposition. Using that here produces a difficult-to-interpret English version: *It is not the case that education funding has decreased and test scores are down.* Does the prefix apply

to both parts or just the first part? The lack of a comma ahead of the ‘and’ suggests both, but still leaves doubt. We can create a more clear version by being a little more creative: *That both education funding has decreased and test scores are down is not true.* Adding ‘both’ and moving the negation to the end makes the meaning more plain – and acceptably conversational.¹⁹

To repeat what Example 10 said: We will see another way to express this situation when we cover logical equivalences. In the meantime, we hope you appreciate why we want to express ideas in logic rather than in English. Natural languages are not famous for their precision.

Example 22:

Problem: Express this logical expression in conversational English: $(p \wedge q) \rightarrow r$, where $p: i \in \mathbb{Z}^{\text{odd}}$, $q: j \in \mathbb{Z}^{\text{odd}}$ and $r: (i + j) \in \mathbb{Z}^{\text{odd}}$.

Solution: Yes, this is just Example 20 in reverse. We are not merely lazy; we are also doing this to make a point.

Translating the logic directly to English is possible but produces a result that is unsatisfying: *If i is an odd integer and j is an odd integer, then $i + j$ is an odd integer.* It is difficult to complain about the correctness of the English version, but we can make it less awkward – more conversational – by reducing the repetition: *If i and j are both odd integers, then $i + j$ is also odd.* And, for those who feel that a sentence is not English if it contains variables and math expressions, we can use the version that started Example 20: *The sum of two odd integers is also an odd integer.*

What was our point? We have two, actually: First, converting directly to English is likely to be unsatisfactory if your goal is conversationality, and second, there are different levels of informal expression – what one person feels is

¹⁹Want it to be more conversational? Imagine a political candidate’s debate: “My knuckle-dragging, loose-jawed and lightly-educated opponent’s claim that both education funding has decreased and test scores are down has as much truth as Hollywood movie accounting . . . but fewer mysterious explosions.”

conversational may not be acceptable to another. Unfortunately, distinguishing the two perspectives is likely to be difficult.

1.5 Logical Equivalence

1.5.1 Two Definitions of Logical Equivalence

As there are multiple ways of saying the same thing in English, there are multiple ways to express logical propositions so that they all produce the same column in a truth table. This idea is the core of our first definition of logical equivalence.

Definition 10: Logical Equivalence

Two propositions p and q are *logically equivalent* (written $p \equiv q$, `\equiv`) when both evaluate to the same result when presented with the same input.

logical equivalence

Example 23:

Back in section 1.3.1, we used a truth table in Table 2, but only for labeling its sections. Here is the unadorned table:

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

You may not have noticed at the time, but the columns for q and $q \wedge (p \vee q)$ are identical. As the truth table covers all possible inputs to these two propositions, and as their evaluations match on all of those inputs, $q \equiv q \wedge (p \vee q)$.

Example 24:

Problem: Show that $r \rightarrow s \equiv \neg s \rightarrow \neg r$ and that the converse of $r \rightarrow s$ is equivalent to its inverse.

Solution: A single truth table can show both of these claims are, perhaps surprising, correct:

r	s	(Original) $r \rightarrow s$	(Converse) $s \rightarrow r$	(Inverse) $\neg r \rightarrow \neg s$	(Contrapositive) $\neg s \rightarrow \neg r$
T	T	T	T	T	T
T	F	F	T	T	F
F	T	T	F	F	T
F	F	T	T	T	T

It's important to remember that, while the inverse and the converse are equivalent to each other, neither is equivalent to the original implication. Only the contrapositive is equivalent to the original.

This is a good time to clear up a common point of confusion: *In logic, equality is not the same as equivalence.* q and $q \wedge (p \vee q)$ are equivalent, as Example 23 shows, but they are not equal because they are different expressions (one is a simple proposition and the other is compound). Later, we will further distinguish equality and equivalence by introducing the idea of *equivalence relations*.

You may be tempted to use this to argue that, in algebra, “ $a+b$ ” and “ $b+a$ ” should be equivalent but not equal. Restate this in logic (that is, change “+” to “ \wedge ” and the domain from real to boolean) and we would agree with you. Logic and algebra share common characteristics (such as commutativity), but they are not the same field of study.

Example 25:

As a computer programmer, no doubt you have found yourself, probably very late at night, seemingly unable to construct a condition for an if statement that will make your program run correctly. The more desperate you became, the wilder your modifications became. Eventually, you stumbled on a condition that worked, and you never looked at that code

again.

If you had, you might have discovered a really good logical equivalence example! Imagine that this pseudocode `if` statement holds your hard-found condition:

```
if (temp > tmp or tmp2 = temp) and tmp < temp then ...
```

(You really need to start using more meaningful variable names, by the way.) This compound proposition – yes, that’s what it is – includes the same condition twice; `tmp < temp` is the same as `temp > tmp`. With the application of commutativity on both the `and` and `or`, this turns out to be ... the proposition $q \wedge (p \vee q)$ from Example 23! Because that is known to be equivalent to just q , we could replace the original `if` statement with:

```
if tmp < temp then ...
```

and make our code both more efficient (fewer operators to evaluate) and easier to understand (especially with better variable names).

True, back when we defined propositions (section 1.2), we said that expressions with variables were not acceptable because we did not know what the variables represented. In a correctly-structured computer program, when a condition is evaluated, the variables all have values. If we imagine replacing the variables with their values, it is easy to see why a condition in a program is actually a proposition.

Recall that bimplication is true only when the operands have the same value. (Or, refresh your memory with a look back at Table 3.) This fact leads to an alternative definition of logical equivalence.

Definition 11: Logical Equivalence

Propositions p and q are *logically equivalent* (written $p \equiv q$, $\text{\LaTeX} : \backslash\text{equiv}$) if $p \leftrightarrow q$ is a tautology.

logical equivalence

Example 26:

Here is the truth table from Example 23, augmented with an additional column:

p	q	$p \vee q$	$q \wedge (p \vee q)$	$q \leftrightarrow [q \wedge (p \vee q)]$
T	T	T	T	T
T	F	T	F	T
F	T	T	T	T
F	F	F	F	T

Because $q \equiv q \wedge (p \vee q)$, their results will always match, their bimplication will always be true, and thus it is a tautology.

1.5.2 Common Logical Equivalences

An infinite number of logical equivalences can be created. A few dozen of those turn out to be particularly useful in problem-solving. Listed below is our collection, grouped in tables based on the logical connectives they use. Names are provided for many but not all. Be aware that other resources may use alternate names (for example, the Negation Laws are sometimes known as the Law of Contradiction and the Law of the Excluded Middle, respectively).

Table 7: Some Logical Equivalences using AND (\wedge) and OR (\vee)

(a)	$p \wedge p \equiv p$ $p \vee p \equiv p$	Idempotent Laws
(b)	$p \wedge \mathbf{F} \equiv \mathbf{F}$ $p \vee \mathbf{T} \equiv \mathbf{T}$	Domination Laws
(c)	$p \wedge \mathbf{T} \equiv p$ $p \vee \mathbf{F} \equiv p$	Identity Laws
(d)	$p \wedge q \equiv q \wedge p$ $p \vee q \equiv q \vee p$	Commutative Laws
(e)	$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$ $(p \vee q) \vee r \equiv p \vee (q \vee r)$	Associative Laws
(f)	$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$ $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$	Distributive Laws
(g)	$p \wedge (p \vee q) \equiv p$ $p \vee (p \wedge q) \equiv p$	Absorption Laws

Table 8: Some More Logical Equivalences (adding \neg)

(a)	$\neg(\neg p) \equiv p$	Double Negation
(b)	$p \wedge \neg p \equiv \mathbf{F}$ $p \vee \neg p \equiv \mathbf{T}$	Negation Laws
(c)	$\neg(p \wedge q) \equiv \neg p \vee \neg q$ $\neg(p \vee q) \equiv \neg p \wedge \neg q$	De Morgan's Laws

Table 9: Some More Logical Equivalences (adding \rightarrow)

--	--	--

(a)	$p \rightarrow q \equiv \neg p \vee q$	Law of Implication
(b)	$p \rightarrow q \equiv \neg q \rightarrow \neg p$	Law of the Contrapositive
(c)	$p \equiv \mathbf{T} \rightarrow p$	“Law of the True Antecedent”
(d)	$\neg p \equiv p \rightarrow \mathbf{F}$	“Law of the False Consequent”
(e)	$p \rightarrow p \equiv \mathbf{T}$	Self-implication (a.k.a. Reflexivity)
(f)	$p \rightarrow q \equiv (p \wedge \neg q) \rightarrow \mathbf{F}$	Reductio Ad Absurdum
(g)	$\neg p \rightarrow q \equiv p \vee q$	
(h)	$\neg(p \rightarrow q) \equiv p \wedge \neg q$	
(i)	$\neg(p \rightarrow \neg q) \equiv p \wedge q$	
(j)	$(p \rightarrow q) \vee (q \rightarrow p) \equiv \mathbf{T}$	Totality
(k)	$(p \wedge q) \rightarrow r \equiv p \rightarrow (q \rightarrow r)$	Exportation Law (a.k.a. Currying)
(l)	$(p \wedge q) \rightarrow r \equiv (p \rightarrow r) \vee (q \rightarrow r)$	
(m)	$(p \vee q) \rightarrow r \equiv (p \rightarrow r) \wedge (q \rightarrow r)$	
(n)	$p \rightarrow (q \wedge r) \equiv (p \rightarrow q) \wedge (p \rightarrow r)$	
(o)	$p \rightarrow (q \vee r) \equiv (p \rightarrow q) \vee (p \rightarrow r)$	
(p)	$p \rightarrow (q \rightarrow r) \equiv q \rightarrow (p \rightarrow r)$	Commutativity of Antecedents

Table 10: Some More Logical Equivalences (adding \oplus and \leftrightarrow)

(a)	$p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$	Definition of Bimplication
(b)	$p \leftrightarrow q \equiv (p \wedge q) \vee (\neg p \wedge \neg q)$	
(c)	$p \leftrightarrow q \equiv \neg p \leftrightarrow \neg q$	
(d)	$p \oplus q \equiv (p \wedge \neg q) \vee (\neg p \wedge q)$	Definition of Exclusive Or
(e)	$p \oplus q \equiv \neg(p \leftrightarrow q)$	
(f)	$p \oplus q \equiv p \leftrightarrow \neg q \equiv \neg p \leftrightarrow q$	

1.5.3 Applications of Logical Equivalences

Having a few dozen logical equivalences is nice, but what can we do with them? In a later chapter we will see how they help with arguments, but for now we can use them to verify additional equivalences, tautologies, and contradictions.

Example 27:

Problem: Using a truth table, demonstrate that $p \rightarrow (q \vee p)$ is a tautology.

Solution: We are doing this as a reminder: You don't need to use logical equivalences to demonstrate these relationships.

p	q	$[p]$	$q \vee p$	$p \rightarrow (q \vee p)$
T	T	[T]	T	T
T	F	[T]	T	T
F	T	[F]	T	T
F	F	[F]	F	T

As $p \rightarrow (q \vee p)$ evaluates to true for all possible combinations of values of p and q , it is a tautology.

We copied over the p column so that the q and p columns are in the same order as those operands are in $q \vee p$. This isn't a necessary step, especially not for operators like \vee that are commutative, but we think that it is a good habit to acquire. Many people have a hard time accurately re-ordering operands of implication (which is not commutative) in their heads.

Note that you can say that the truth table shows that $p \rightarrow (q \vee p)$ is equivalent to true (that is, $p \rightarrow (q \vee p) \equiv \mathbf{T}$). That is a symbolic way of saying that the expression is a tautology.

Let us repeat that example, but this time we will apply logical equivalences.

Example 28:

Problem: Using a sequence of logical equivalences, demonstrate that $p \rightarrow (q \vee p)$ is a tautology.

Solution: The statement of the problem tells us where we need to go: We need to show that the proposition is equivalent to true. Can we find a known equivalence that both matches the form of $p \rightarrow (q \vee p)$ and seems a likely route to our desired answer? Yes! Table 9, line n, has the same form. True, it uses p , q , and r , but as these are just place-holders for propositions, we can substitute p for r . After applying that equivalence,

the next two steps are straight-forward.

$$\begin{aligned}
 p \rightarrow (q \vee p) &\equiv (p \rightarrow q) \vee (p \rightarrow p) && \text{[Table 9(n)]} \\
 &\equiv (p \rightarrow q) \vee \mathbf{T} && \text{[Self-Implication]} \\
 &\equiv \mathbf{T} && \text{[Domination]}
 \end{aligned}$$

Because $p \rightarrow (q \vee p)$ is equivalent to true, it is a tautology.

Justifying each equivalence in the sequence is essential if the reader is to trust that the demonstration is correct. Writing one equivalence per line gives space to the right to either direct the reader to one of the tables on pages 33–34, or by giving its name (if it has one).

Frequently, there are other ways to string together equivalences to reach the same conclusion. Example 29 repeats Example 28 but chooses a different starting equivalence.

Example 29:

Problem: Using a sequence of logical equivalences different from that used in Example 28, demonstrate that $p \rightarrow (q \vee p)$ is a tautology.

Solution: There are no other given logical equivalences that have the same form. We have to be more creative. It might be tempting to just use self-implication directly, but its form ($p \rightarrow p$) shows that the same proposition must appear on both sides of the implication. Instead, we will start with the Law of Implication (Table 9(a)) and see where that leads us.

$$\begin{aligned}
 p \rightarrow (q \vee p) &\equiv \neg p \vee (q \vee p) && \text{[Law of Implication]} \\
 &\equiv q \vee (\neg p \vee p) && \text{[Commutativity of } \vee \text{]} \\
 &\equiv q \vee \mathbf{T} && \text{[Negation Law]} \\
 &\equiv \mathbf{T} && \text{[Domination]}
 \end{aligned}$$

Once again, as $p \rightarrow (q \vee p)$ is equivalent to true, it is a tautology.

The second step is actually more than just one step. We remove the parentheses, which is acceptable because \vee is associative. Next, we swap (commute) $\neg p$ and q , and we insert parentheses (associativity of \vee again) just to highlight the location of our next step. If you do not care for this lack of complete clarity, you can include a line for each of these steps. However, this

kind of short-cutting is very commonly done, and so you should be prepared to mentally fill-in these small steps when studying examples.

At the risk of over-doing this problem, we wish to show one more technique for solving it. This approach is still based on logical equivalences, but considers the possible values of the operands in turn to reason about the solution. We call the technique *case-based reasoning*.

case-based reasoning

Example 30:

Problem: Demonstrate that $p \rightarrow (q \vee p)$ is a tautology by using case-based reasoning.

Solution: Being a logical proposition, p can be either true or false. We consider each in turn.

Case 1: Let p be true. Our expression becomes $\mathbf{T} \rightarrow (q \vee \mathbf{T})$. By domination, this simplifies to $\mathbf{T} \rightarrow \mathbf{T}$, which is true by the definition of implication.

Case 2: Let p be false. $\mathbf{F} \rightarrow (q \vee \mathbf{F})$ reduces to $\mathbf{F} \rightarrow q$ by the identity laws (Table 7(c)). By the definition of implication, any implication with false as the antecedent evaluates to true.

Thus, for both possible values of p , the proposition $p \rightarrow (q \vee p)$ is true, demonstrating that it is a tautology.

We will see this type of case-based reasoning again when we study proofs.

Knowledge of logical equivalences can help you make your programs more readable by other programmers (yes, this is a good thing!).

Example 31:

Problem: Write a condition that evaluates to true when the variable **sensor** is outside of the acceptable tolerance range defined by the constants **ZERO** and **OMG**. Assume that $\text{OMG} > \text{ZERO}$.

Solution: Assuming that a reading of **OMG** is unacceptable, ‘outside’ means

that the value is either lower than ZERO or at/above OMG. That is, the sensor reading is not within the acceptable range. We can code that as:

```
if sensor < ZERO or sensor => OMG then ...
```

This is perfectly correct, but the code would be better if the condition could be more easily ‘read’ to mean “not acceptable” by the programmers who will be tasked with maintaining your code (and remember, one of them could well be you!). In logic, we can express that condition as $p \vee q$, with p : `sensor < ZERO` and q : `sensor ≥ OMG`. $p \vee q \equiv \overline{\overline{p \vee q}}$ by double negation (Table 8(a)). Applying the second De Morgan’s Law (Table 8(c)) to the inner negation produces $\overline{\overline{p} \wedge \overline{q}}$. The negation of p is `sensor ≥ ZERO` and the negation of q is `sensor < OMG`. As a program condition, it looks like ...

```
if not (sensor ≥ ZERO and sensor < OMG) then ...
```

...and is logically equivalent to the first version. Yes, this has one more operator (the ‘not’) than does the first version, but it is a direct translation of “not acceptable” into code. Code that ‘reads’ well (“if the measurement is not acceptable, do something!”) is more likely to be understood, and more likely to be maintained correctly. Give this some thought the next time you choose a cryptic identifier name (yes, like ‘OMG’) or receive a CT scan.²⁰

One more example. We have already seen that conversions between English and logic can be challenging. Example 32 demonstrates how your knowledge of logical equivalences can help make sure your conversion to logic is correct.

Example 32:

Problem: Express this sentence in logical notation: *If he is honest, he didn’t open his eyes and sneeze.*

Solution: Thinking positively, we start by identifying and labeling our

²⁰One of the most famous examples of a software failure of a medical device is the Therac 25 story: <http://en.wikipedia.org/wiki/Therac-25>

predicates: h : *he is honest*, o : *he did open his eyes*, and s : *he did sneeze*. On the surface, the sentence is ambiguous. We don't know if *he didn't open his eyes and sneeze* should be expressed as $\bar{o} \wedge \bar{s}$ (*he did not open his eyes and he did not sneeze*, meaning that neither action occurred) or as $\overline{o \wedge s}$ (*he did not both open his eyes and sneeze*, meaning that he could have done either one separately or neither). Somehow, we have to choose between $h \rightarrow (\bar{o} \wedge \bar{s})$ and $h \rightarrow \overline{o \wedge s}$ as our answer. Perhaps logical equivalences can help us decide between them.

We know that the contrapositive of an implication is equivalent to the original (see Table 9). Considering our first option, we find that $h \rightarrow (\bar{o} \wedge \bar{s}) \equiv \overline{\bar{o} \wedge \bar{s}} \rightarrow \bar{h} \equiv \overline{o \vee s} \rightarrow \bar{h} \equiv (o \vee s) \rightarrow \bar{h}$ (by contraposition, De Morgan's, and double negation). In conversational English, this could be written as: *If he opened his eyes or sneezed, he's lying*. This version does not make much sense; disjunction provides two or three options for truth, making lying unlikely.

Moving to the second option, applying contraposition and double negation to it shows that $h \rightarrow \overline{o \wedge s} \equiv (o \wedge s) \rightarrow \bar{h}$. In English: *If he opened his eyes and sneezed, then he is lying*. That is a reasonable statement; you could sneeze with your eyes open, but you'd almost certainly have to physically hold them open to do so.²¹

Thanks to logical equivalences, the correct logical interpretation appears to be $h \rightarrow \overline{o \wedge s}$.

Remember: *Do not overthink conversions between natural languages and logic!* It is easy to ask, "What if he opens his eyes, pauses, and closes them to sneeze?" Easy, but unjustified. Take the statement as given; don't waste time dreaming up embellishments. This material is challenging enough as-is.

²¹Thank you, Mythbusters! <http://dsc.discovery.com/fansites/mythbusters/db/human-body/sneezing-eyeballs-pop-out.html>

Chapter 2

Quantified Expressions

We learned in Chapter 1 that variables are unwelcome in propositions. Variables are, however, incredibly useful in math and programming, making their utility in logic no surprise. In this chapter we add variables (and their associated baggage) to propositions, as we continue on the trail to proofs.

2.1 Propositions to Predicates

Flashback: We said in Chapter 1 that we “consider variables to be ‘illegal’ within propositions.” That’s because adding variables to a proposition requires the introduction of several new concepts that explain and bind the variables. We had enough to talk about as it was.

Without the additional complications, predicates and propositions are very similar.

Definition 12: Predicate

A statement that includes at least one variable and will evaluate to either true or false when all variables are assigned values is a *predicate* (a.k.a. a *propositional function*).

predicate

As we did with propositions, we will be giving predicates labels for ease of reference. To help distinguish predicates from propositions, we will use upper-case letters for predicates. Also, we will follow the predicate label with a list

of the variables used by the definition of the predicate, much as a program's invocation of a subprogram is followed by the parameters of the call.

Example 33:

Here are two examples of basic predicates:

$$H(t) : (0 \leq t) \wedge (t < 24)$$

$$A(x, y) : x \text{ is on the } y$$

The first predicate evaluates to true when the argument is in the range $[0 \dots 23]$, which are the legal hour values in military (a.k.a. 24-hour) time. The second example demonstrates a two-variable predicate dealing with spacial orientation.

Something's missing from those examples: Declarations of the variables. Many programming languages require that variables be declared (added to the code's scope, assigned a type, etc.) before they are assigned values. Other languages permit variables to take on such characteristics dynamically during execution. In logic, we take the first approach: You must state the *domain* of a variable when you use it in a predicate.

domain

Definition 13: Domain

The *domain* (or *universe*) of a variable is the collection of values from which its value may be drawn.¹

You can think of variables in a program as having domains, too. For example, in Java a variable declared to be of type `int` has a domain consisting of the integers from -2,147,483,648 through 2,147,483,647.

¹More formally, the term is *domain of discourse* or *universe of discourse*, but who wants to say or write those?

Example 34:

Let's fix the predicates in Example 33:

$$H(t) : (0 \leq t) \wedge (t < 24), t \in \mathbb{Z}$$

We could have used \mathbb{R} or even \mathbb{Z}^* for t 's domain, but \mathbb{Z} is all we really need – we want to exclude non-integer hours, and the predicate's condition already deals with negative values.

$$A(x, y) : x \text{ is on the } y, x, y \in \text{Furniture}$$

x and y are taken from the same domain in the second example. That's not a requirement; they can be from separate domains if that suits your needs, as in:

$$A(x, y) : x \text{ is on the } y, x \in \text{Toys}, y \in \text{Furniture}$$

which allows us to consider the logical values of $A(\text{ball}, \text{table})$ and $A(\text{go-kart}, \text{piano})$.

To avoid misunderstandings, here are the rules for domains that we will follow in this book:

1. Each variable's domain is stated after the predicate.
2. Domains may be no more than one step more specific than “Things.”
3. Operators may not be concealed within domains.

Like most rules, these have lots of room for loopholes. Example 35 tries to clear up some of the more common ones.

Example 35:

Consider this predicate:

$$G(x) : x \text{ contains gossip}, x \in \text{supermarket tabloids}$$

The domain “supermarket tabloid” runs afoul of rules (2) and (3). This

domain is for a specific kind of publication, which suggests that the domain ought to be “magazines” or “periodicals.” Further, it merges the characteristics of “sold in supermarkets” and “is a tabloid.” As the last sentence suggests, the merger is best accomplished with AND. We should be clear about that by creating separate predicates for each of those characteristics:

$G(x)$: x contains gossip, $x \in$ magazines

$S(x)$: x is sold in supermarkets, $x \in$ magazines

$T(x)$: x is a tabloid, $x \in$ magazines

We can combine them with ANDs to recapture the original meaning: $S(x) \wedge T(x) \wedge G(x)$, $x \in$ magazines — a supermarket tabloid that contains gossip.

The quantification examples in sections 2.3 and 2.4 will show the value of these rules. You may be viewing them as causing extra clutter (“Now we need multiple predicates instead of just one!”), but that ‘clutter’ helps extract the meaning that is being expressed by the statement.

2.2 Evaluating Predicates

This may seem straight-forward; if you want to evaluate a predicate, plug in a value and produce a result. But that’s not the only way to do it.

2.2.1 Evaluating Predicates on Specific Domain Elements

Much as we don’t know the return value of a square root function until we pass it a value, until we assign the variables of a predicate appropriate values, we cannot determine its the truth value.

Example 36:

Consider this predicate:

$O(x)$: x is evenly divisible by 2, $x \in \mathbb{R}$

Supplying $O()$ with any multiple of two will make it true: $O(8)$, $O(-4)$,

and $O(0)$ are all true. All other real values cause $O()$ to be false, including 1, $\frac{5}{7}$, and π .

Notice that when we supply values for the variables, we are converting predicates to propositions. *x is evenly divisible by 2* is a fine predicate (especially with a domain!), but an unacceptable proposition. However, *7 is evenly divisible by 2* is a proposition, one that happens to be false.

2.2.2 Quantification: Evaluating Predicates on Sets of Domain Elements

We aren't restricted to assigning predicate variables individual values to treat the predicates as propositions. A more general approach that permits us to express some common situations compactly works by considering certain subsets of the variables' domains instead of specific individual values. The idea is called *quantification*, and we will consider two types:

quantification

1. *Universal Quantification* (\forall , `\forall`) requires that the entire domain of values be considered .
2. *Existential Quantification* (\exists , `\exists`) considers one or more (which could include all) members of the domain.

When applying quantification to a predicate, we will use one type of quantifier or the other (but never both) with each variable of the predicate. We consider quantified variables to be *bound* by the quantifier. In some contexts, it's possible for some variables not to be quantified; such variables are considered to be *free* (a.k.a. *unbound*). Sorry; none of those contexts appear in this book. We'll be quantifying all of our predicates' variables.

Definition 14: Bound Variable

A quantified variable within a predicate is a *bound variable*.

bound variable

free variable

Definition 15: Free Variable

An unquantified variable within a predicate is a *free variable*.

Table 11 shows the situations in which quantified expressions are true or false.

Table 11: Evaluating \forall and \exists

This expression:	is True when:	is False when:
$\forall x P(x), x \in D$	every $d \in D$ makes $P()$ true	at least one $d \in D$ makes $P()$ false
$\exists x P(x), x \in D$	at least one $d \in D$ makes $P()$ true	every $d \in D$ makes $P()$ false

Quantified expressions are used frequently in human languages. For example, in English:²

- “Most politicians are slimy weasels”
- “None of those peaches are ripe”
- “Few students like taking discrete math exams”
- “That bag contains at least two pieces of candy”

There are quantifications that cannot be adequately expressed in first-order logic. For example, how many politicians can be considered to be “most,” and how many students are there in a “few?” Many quantifications can be expressed in FOL, although it is sometimes a challenge. In particular, the second and fourth of those examples can be expressed; we’ll see how before the end of this chapter.

The next two sections cover \forall and \exists in detail. For now, here’s a simple example that should help distinguish them.

²A joke: “What do you call a person who speaks three languages? Trilingual. What do you call a person who speaks two languages? Bilingual. What do you call a person who speaks one language? American.” If you want examples in other languages, you need a different author!

Example 37:

Let $S(x) : x^2 < 9, x \in \mathbb{Z}^+$.

Are there any members of the set \mathbb{Z}^+ that make $S(x)$ true? Yes; try $x = 2$. $S(2)$ is the proposition “ $4 < 9$ ”, which is true. Because at least one member of the domain makes $S()$ true, we can say that $\exists x S(x)$ is true. To read that aloud, we would say “There exists a value for x such that $S(x)$ is true.”

In order for $\forall x S(x)$ to be true, *all* members of the domain must make $S()$ evaluate to true — no exceptions. Showing this can be a significant task, one often requiring that formal reasoning techniques be applied. However, to show that a universally quantified expression is false, all we need is one domain member that makes $S()$ false. For this predicate, that’s easy: Any integer above 3 (or under -3) makes $S()$ false. Thus, $S()$ is not true for all domain elements; the expression $\forall x S(x)$ is false.

Let’s take the example a step further by changing the domain from \mathbb{Z} to $\{0, 1, 2\}$. Because every member of this new domain makes $x^2 < 9$ true, $\forall x S(x)$ is now true. Read aloud: “For all x in the domain, $S(x)$ is true.” $\exists x S(x)$ is also true, because at least one element of the domain makes $S()$ true. If $\forall x S(x)$ is true, then $\exists x S(x)$ is also true.

2.3 More Detail on Universal Quantification

We already know (from Section 2.2) that $\forall x P(x)$ is true only when all members of the domain make $P()$ true. Otherwise, the quantification is false. Happily (?), universal quantification is more interesting than that.

2.3.1 Identifying Universal Quantification in English

There are some words and phrases in English that can be indications of universal quantification, including “All,” “For all,” “Every,” and “Any.” Unfortunately, it’s not as simple as doing a text search for these phrases to location universal quantification.

Example 38:

Here’s a sentence that uses “all” in a non-universal way:

Darling, I will come home with all possible speed!

Unless this person is planning to make the trip an uncountably infinite³ number of times, ‘with all possible speed’ is just a fancy way of saying ‘as soon as possible.’

Example 39:

Sometimes, none of those phrases are used, yet the sentence is still clearly universal in nature. For instance:

Car doors open outward.

The point: You have to read statements carefully to understand their intended meaning. You can’t simply learn and follow a tidy set of rules.

2.3.2 Nesting Universal Quantifiers

Recall that it’s possible for a predicate to have more than one variable, as in $L(a, b) : a < b$. When working with such predicates, we have to quantify all of their variables. But what does it mean for an expression to have two universal quantifiers?

Let’s start with the notation. Authors have a small variety of ways to write nested quantifiers. We’ll stick with the classic: $\forall a \forall b L(a, b)$. Other options include $(\forall a)(\forall b)L(a, b)$, $\forall a \forall b : L(a, b)$, and $\forall a, b L(a, b)$.

In English, assuming a domain of real numbers for both a and b , we’d express that nested quantification as “For all pairs of real numbers, the first is less than the second.” This isn’t true, of course, but that’s OK – we need to be able to represent both true and false expressions.

³Will be covered in a future chapter. Really!

Example 40:

When we subtract one integer from another (e.g., $4 - 6$), the result is also an integer. Mathematicians say that \mathbb{Z} is *closed* under subtraction. How can we express it as a quantified expression?

We need to say that subtracting any integer from any other integer produces an integer. We don't want to create an expression that is overly restrictive. For instance, it's possible for all three integers to be the same ($0 - 0 = 0$). We don't want to create an expression that excludes such possibilities.

Consider this attempt: $\forall x [(x - x) \in \mathbb{Z}], x \in \mathbb{Z}$. x represents integers, and so $x - x$ represents the subtraction of two integers, right? Close: $x - x$ represents the action of subtracting an integer from itself (as in $19 - 19$ or $(-4) - (-4)$). We need to say that the two integers have no other connection (besides both being integers). Here's the expression we need:

$$\forall x \forall y [(x - y) \in \mathbb{Z}], x, y \in \mathbb{Z}.$$

Adding y allows the expression to represent the subtraction of two integers that may or may not be the same integer.

Example 41:

Integers are closed under subtraction, but not under division. $4/3$, for instance, doesn't produce an integer result.⁴ Real numbers are closed under division, but we have to be careful to exclude division by 0. How can we add that exception to the expression? Here's how:

$$\forall a \forall b [(b \neq 0) \rightarrow ((a/b) \in \mathbb{R})], a, b \in \mathbb{R}.$$

This can be read as, "For any real numbers a and b , if b is not zero, the division of a by b produces a real number," or, more conversationally, "Dividing a real by a non-zero real produces a real." The implication 'protects' the division from the case that $b = 0$, much as we'd use an `if` statement in a program for the same purpose.

Example 41 demonstrates the difference between *logical English* and *conversational English*. If someone looks over your shoulder, sees you writing a quantified expression, and asks, “What’s that mean?”, you’d want to provide a conversational answer. In many examples in this chapter, we will provide both logical and conversational versions, because it’s reasonably easy to create the former from a quantified expression, but not the latter, at least not without practice. Moving from logic to conversational English via logical English usually makes the task easier (while cutting down on errors).

If you are still having trouble wrapping your mind around this nested quantification idea, viewing it from the perspective of nested loops in a programming language may help. Consider this nested loop pseudocode example that returns `True` only if all nine products are negative:

```

for i in {1,2,3}:
    for j in {-2,-4,-6}:
        if (i * j >= 0)
            return False
        endif
    endfor
endfor
return True

```

The loops are a short-cut that help us avoid having to write nine separate *if* statements. They systematically pair every member of the first set (*i*’s domain, if you will) with all members of the second set. This is also what nested quantification represents. Here’s the logical notation version:

$$\forall i \forall j (i * j < 0), i \in \{1, 2, 3\}, j \in \{-2, -4, -6\}.$$

Each quantifier can be thought of as representing a loop that covers the corresponding variable’s domain. The condition that we want to establish is the negation of the code’s condition, because we want the code to stop checking as soon as we know what the result will be — remember, to show that a universal quantification is false, all we need is one exception.

⁴Integers are closed under *integer* division ($4 \setminus 3$), but this example is using real division.

2.3.3 An Extended Universal Quantification Example

The next example brings together many of the quantification ideas we have covered so far.

Example 42:

Problem: Express the sentence “*Everyone who reads this book smells fresh*” in logical notation.

Solution: We will (intentionally!) make several errors as we work toward the correct answer to this problem, so that you can avoid making the same errors yourself.

The first word (“everyone”) suggests that this is a universal quantification situation, and thus that we should be using predicates instead of propositions. But, what are the predicates? It’s tempting to create a single predicate:

$$P(x) : x \text{ is a fresh-smelling reader of this book, } x \in \text{People}$$

The problem: There are two ideas covered by that predicate — readers of this book, and people who smell fresh. They need to be separated.⁵ Attempt #2:

$$R(x) : x \text{ reads this book, } x \in \text{People who smell fresh}$$

Nice try, but as we talked about earlier, we want the domain to be no more than one step removed from “Things.” “People” is a fine domain, but in isolation. This means that we need a second predicate for fresh-smellers. Attempt #3:

$$R(x) : x \text{ reads this book, } x \in \text{People}$$

$$F(x) : x \text{ smells fresh, } x \in \text{People}$$

Not bad, but $R()$ needs some help. “This book” is pretty vague. We can tidy this up by making $R()$ into a binary predicate. Attempt #4:

$$R(x, y) : x \text{ reads } y, x \in \text{People, } y \in \text{Books}$$

$F(x) : x$ smells fresh, $x \in \text{People}$

That looks good. We have the quantification, we have the predicates, and when we use $R()$ in our final expression, we'll use the name of this book in place of y . One last step: We just need to combine the predicates with a suitable logical operator. The given sentence combines these two characteristics, which makes AND seem like a reasonable operator choice:

$$\forall x [R(x, \text{"Discrete Structures for Computer Science"}) \wedge F(x)],$$

$x \in \text{People}$

But what does that mean? In English, we would read this as, "Everyone reads this book and smells fresh." Everyone? All ~ 7 billion of us? That's overstepping the meaning of the sentence. We just want to say that those people who read this book smell fresh. Put another way: If a person reads this book, they smell fresh. Looks like a job for . . . implication!

$$\forall x [R(x, \text{"Discrete Structures for Computer Science"}) \rightarrow F(x)],$$

$x \in \text{People}$

In not-very-conversational English: "Considering all people, if a person reads this book, that person smells fresh." Stated conversationally: "Everyone who reads this book smells fresh," which is exactly what we were given.

Example 42 showed that implication was the appropriate operator. That is often the case: In universally-quantified expressions, when you appear to have a choice between AND and implication, the correct choice is usually implication.

2.4 More Detail on Existential Quantification

Recall from Section 2.2.2 that existentially quantified expressions are true so long as a member of the domain makes it true. More than one is fine, but we need at least one. To make such existential quantifications false, none of the domain members can make them true.

⁵No, not because fresh-smelling people want nothing to do with readers of this book . . .

2.4.1 Identifying Existential Quantification in English

Like universal quantification, there are some key words and phrases in English that can suggest existential quantification as the appropriate choice. These include “Some,” “A few,” “More than one,” and “Several.” These words and phrases could also have nothing to do with quantification, and there are many other ways to suggest ‘there exists’ — always read the English statement carefully to understand its intended meaning.

Example 43:

You know, quite a few people need food to survive . . .

“Quite a few” is clearly existential, stopping short of universal. But looking at the complete sentence, universal certainly seems more appropriate, under the reasonable assumption that all people need food to live. The tone of the sentence is one of sarcasm, as though the speaker is chiding another person for an obvious error in reasoning. Such undercurrents of meaning are a key complication when converting English to logic.

Example 44:

That was some game!

This is the sort of remark one friend might say to another at the end of a 8-7 (that is, unusually high-scoring) soccer game. “Some” suggests existential quantification, but after more thought it’s clear that this sentence is devoid of definite content. It’s just not worth trying to express logically, because we don’t know the information supporting the remark. We can imagine converting “The total number of goals scored in today’s game exceeds the 98th percentile for soccer matches” to logic, but it’s much harder to imagine it being used to break an awkward silence in casual post-game conversation.

2.4.2 Nesting Existential Quantifiers

Having covered nested universal quantifiers in section 2.3.2, this section should be easier to follow, because the ideas are much the same. We'll even use the same starting example: $L(a, b) : a < b, a, b \in \mathbb{R}$. Using nested existential quantification, we create $\exists a \exists b L(a, b), a, b \in \mathbb{R}$, which can be expressed in English as “There exist two real numbers a and b such that $a < b$,” or more conversationally as “There are two reals such that the first is less than the second.” We need just one true example to verify the truth of that expression, such as $2.15 < 6.736$.

Example 45:

Problem: Convert $\exists c \exists d \neg D(c, d), c \in \text{Plants}, d \in \text{Defenses}$ to conversational English, where $D(c, d)$ represents “ c is protected by d .”

Solution: A good place to start is to remember how to handle negations. The negation of “ c is protected by d ” is “ c is not protected by d .”

From there, we can construct a formal English version: “There exists a plant and there exists a defense such that the plant is not protected by that defense.” Conversationally, “Some plants don't have some defenses” means the same thing.

In section 2.6, we will learn how to correctly relocate negations within quantified expressions. Doing so is often helpful when interpreting the meaning of a complex expression.

Example 46:

Problem: Express *There are three integers such that the sum of the squares of the first two equals the square of the third* in logic.

Solution: Let's let $r, s,$ and t represent the three integers. The sentence says that we need to add together the squares of r and s ($r^2 + s^2$) to get a result that equals the square of other integer (t^2). That is, $\exists r \exists s \exists t [r^2 + s^2 = t^2], r, s, t \in \mathbb{Z}$. If you remember your basic geometry, you'll know that this expression is true — it's the Pythagorean Theorem.

There's no limit on the quantity of variables (and therefore on the quantity of quantifiers) we can have in an expression. Use as many as you need to say what needs to be said.

2.4.3 An Extended Existential Quantification Example

Ready to see a more involved example of nested existential quantification? Sure, you say that now ...

Example 47:

Problem: Express this sentence in logic: *There are some tow trucks in some of the downtown parking garages.*

Solution: Compared to our examples so far, this sentence has a lot of detail. So that the structure of the answer doesn't get obscured by that detail, we will temporarily suspend our "one concept per predicate" rule. Once we have the structure in place, we'll reinstate that rule and produce the final answer.

We have three basic ideas for which predicates are needed: Tow trucks, downtown parking garages, and storing the trucks in the garages.

$T(x)$: x is a tow truck, $x \in \text{Vehicles}$

$G(x)$: x is a downtown parking garage, $x \in \text{Buildings}$

$I(x, y)$: x is inside of y , $x \in \text{Vehicles}$, $y \in \text{Buildings}$

We need to say that some trucks are in some garages. Maybe this will work:

$$\exists v \exists b I(T(v), G(b)), v \in \text{Vehicles}, b \in \text{Buildings}$$

No, it won't. $I()$ accepts a vehicle and a building, but $T()$ and $G()$ return boolean values. It's like trying to pass a character string to a square root function. OK, let's separate the predicates and try combining them with implications (because that's what we ended up using in

the extended universal quantification example of section 42):

$$\exists v [T(v) \rightarrow \exists b (G(b) \rightarrow I(v, b))], v \in \text{Vehicles}, b \in \text{Buildings}$$

This is a WFF, but still isn't correct. In logical English, it reads as "There exists a vehicle such that, if the vehicle is a tow truck, then there exists a building such that, if the building is a downtown parking garage, then the vehicle is in the garage."⁶ The problem is that the conditionals don't help here. They were important for the universal quantification example, because we needed to cover every member of a subset of the domain. Here, with existential quantification, we don't need to say *if* a tow truck exists; rather, we merely need to say that one does. For that purpose, AND is all we need to express our desired meaning:

$$\exists v [T(v) \wedge \exists b (G(b) \wedge I(v, b))], v \in \text{Vehicles}, b \in \text{Buildings}$$

In logical English, and playing fast and loose with plurals to keep it from being an even bigger mess, we have "There exists at least one vehicle such that the vehicles are tow trucks and there exists at least one building such that the buildings are downtown parking garages and the vehicles are in the garages," or, conversationally, "Some tow trucks are parked in a few downtown parking garages." That's not exactly the same wording as we were given, but it's logically equivalent, which is fine.

Now that we have the structure the way we want it, we need to fix the logical expression so that each predicate covers just one detail. We need several more predicates; here's the complete set:

$T(x)$: x is a truck, $x \in \text{Vehicles}$

$W(x)$: x is for towing, $x \in \text{Vehicles}$

$D(x)$: x is located downtown, $x \in \text{Buildings}$

$G(x)$: x is a garage, $x \in \text{Buildings}$

$P(x)$: x is a vehicle parking location, $x \in \text{Buildings}$

$I(x, y)$: x is inside of y , $x \in \text{Vehicles}$, $y \in \text{Buildings}$

Adding them to the expression is easy; they are placed in groups based on the part of the sentence they are describing:

$$\exists v [T(v) \wedge W(v) \wedge \exists b (D(b) \wedge G(b) \wedge P(b) \wedge I(v, b))],$$

$v \in \text{Vehicles}, b \in \text{Buildings}$

And, finally, we're done!

Tying up some loose ends from Example 47:

- *That last expression is a little overwhelming. Can you give me a breakdown to help me make sense of it?* Sure; take a look at Figure 2.1.
- *Seems that \rightarrow goes with \forall and \wedge goes with \exists , right?* Basically, yes. This isn't to say that we can't mix and match when the situation warrants, but in many circumstances this rule of thumb works.
- *Why did you move $\exists b$ to the middle of the expression?* You've probably written a counter-controlled WHILE loop as part of a program. Where did you declare the variable, at the start of the subprogram or just ahead of the loop? Probably the latter, because that's where you needed the variable – why declare it until you need it? The same idea applies here. We didn't need b right away, so the quantifier wasn't needed right away, either.
- *So ... we can put all of the quantifiers at the front of the expression?* Yes, you can, but it's best that you don't. Readers will expect you to hold off on inserting quantifiers until they are needed. Why make them remember more variables before those variables are necessary?⁷ Also, delaying the quantifiers helps the expression 'read' a little more naturally.
- *I really like compact expressions. Is that 'one concept per predicate' rule really necessary?* In this book it is! Here's one way to look at it: If you could dump everything into a single predicate, how would you ever learn how to use logical operators? It's like a child 'cleaning' her room by stuffing all of her toys into her closet.
- *Do we have to write the domains after every ... single ... expression?* Nope! You can instead begin by saying something like "In the following expressions, assume $x \in \text{Vehicles}$ and $y \in \text{Buildings}$." But if you get in the habit of putting domains after every expression, you (and the reader) will be less likely to overlook them.

⁶Yuck!

⁷'Cognitive load' is the phrase that describes this. You could also use it as an insult: *My friend, no one can say that you're a cognitive load.*

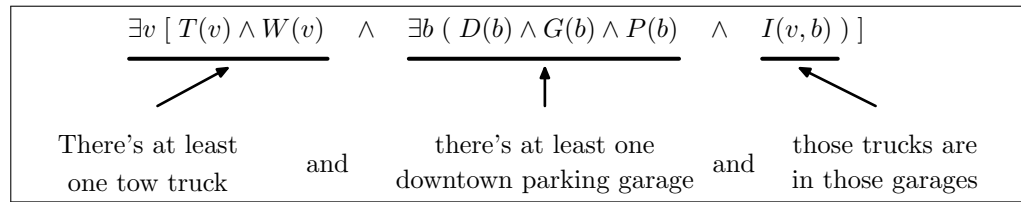


Figure 2.1: A breakdown of the final expression from Example 47.

2.5 Mixing Universal and Existential Quantification

Nesting matching quantifiers (e.g., $\exists a \exists b$) is relatively straight-forward and creates expressions whose meanings aren't too hard to understand. Nesting mismatched quantifiers (e.g., $\forall a \exists b$) is also often necessary. Unfortunately, making sense of such mixed nestings usually isn't as easy, even though the process is the same.

Example 48:

Problem: Express both $\forall p \exists q L(p, q)$ and $\exists p \forall q L(p, q)$ in conversational English, where $L(p, q)$ represents “ p laughs at q ,” and p and q share the domain of people.

Solution: $\forall p \exists q L(p, q)$ means, in logical English, “Considering all people, there exists at least one person that all people laugh at.” We can condense that to the more conversational, “There’s someone that we all laugh at.”⁸ Notice that this means a universally-laughed-at person also laughs at himself or herself.

$\exists p \forall q L(p, q)$ means “there exists a person such that, no matter which person you name, the first person will laugh at them.” Conversationally, “There’s someone who laughs at everyone” (again, including himself or herself).

⁸Maybe one of those hapless burglars who gets stuck upside-down in a restaurant’s kitchen exhaust ductwork. They must laugh at themselves . . . eventually.

A common question: *Does the order of the quantifiers' variables have to match the order of variables' appearances in the expression? That is, could I write " $\forall b \exists a Q(a, b)$ " instead of " $\exists a \forall b Q(a, b)$ "?* Technically, 'yes,' but practically 'no.' It's very rare for people to change the order. Why? Because it makes interpretation of mixed-quantifier expressions unnecessarily difficult. That's reason enough for us not to do it. By the way, the order of the variables in nested *matching*-quantifier expressions (such as $\forall x \forall y$ vs. $\forall y \forall x$), by the way, doesn't matter logically, but again, it's still best to quantify the variables in the order in which they appear in the expression being quantified.

Example 49:

Problem: Express this sentence in logic: *The tow trucks are in the downtown parking garages.*

Solution: This is the sentence of Example 47 with some small changes. Perhaps the most intimidating change is that the linguistic clues of quantification are missing — words such as 'some' and 'all' aren't provided. We need to deduce the intended quantifiers using our knowledge of English.

The subject "tow trucks" includes no limitations, indicating that the meaning is *all* tow trucks. There are also no limitations on the parking garages. It's tempting to assume that the intended meaning is again "all"; that is, *all* of the tow trucks are in *all* of the parking garages. But . . . must all of the parking garages contain tow trucks? The statement doesn't quite say that; all we know is that, if the statement is true, the tow trucks can be found in the garages — *some* of the garages, possibly all of them. Thus, we need to universally quantify the vehicles and existentially quantify the buildings. The only real difference between this sentence and the one of Example 47 is the change in the vehicle quantification.

Using the same predicates as used in Example 47 and our deciphering of the quantifiers, we can create a corresponding logical version of the statement. Not surprisingly, it's just the solution to Example 47 with the swap of quantifiers and of one AND for an implication (as we've seen, \rightarrow usually goes with \forall):

$$\forall v [(T(v) \wedge W(v)) \rightarrow \exists b (D(b) \wedge G(b) \wedge P(b) \wedge I(v, b))],$$

$v \in \text{Vehicles}, b \in \text{Buildings}$

In somewhat condensed logical English: “For all vehicles, if the vehicle is a tow truck, then there exists a downtown parking garage and the tow truck is in it.”

You may be wondering if the intended meaning of this statement could be exactly that of Example 47. Perhaps, but that seems unlikely, given how people use English. Based on the wording of the sentence, the all-trucks/some-garages interpretation is the most likely. Try to keep this issue of interpretation in mind the next time you have to write down a recipe, create a software specification, or read a legal document.

For reference, Table 12 summarizes when mixed quantifications are true and false.

Table 12: When Are Mixed Quantifications True/False?

Quantification	Is True When ...	Is False When ...
$\forall m \exists n P(m, n)$	no matter the m , you can find an n that makes $P()$ true	there’s an m for which no n exists to make $P()$ true
$\exists m \forall n P(m, n)$	some m can be paired with any n yet $P()$ is true	we try every m , but no n exists that makes $P()$ true

The ‘true’ case of $\forall m \exists n P(m, n)$ is worth an additional mention. You don’t have to find a single n that works with all possible values of m – that is, each of the m values can have its own n value. For example, no matter which non-zero integer you give me ($\forall i$), I can find a rational number ($\exists j$) such that their product is 3. That rational number is based on i ($j = 3/i$), and is therefore different every time, but that’s all we need to make $\forall i \exists j (i \cdot j = 3)$ true.

2.6 Generalized De Morgan's Laws

In Chapter 1 we learned a pair of equivalences known as DeMorgan's Laws: $\neg(p \wedge q) \equiv (\neg p \vee \neg q)$ and $\neg(p \vee q) \equiv (\neg p \wedge \neg q)$. They (as well as the rest of the equivalences) apply to predicates as well as they do to propositions.

Example 50:

Consider $\exists x[P(x) \wedge Q(x)]$ (it doesn't matter what $P()$ and $Q()$ represent). $P(x) \wedge Q(x)$, using an x from its domain, is just a compound proposition. We can apply Double Negation and De Morgan to it: $P(x) \wedge Q(x) \equiv \overline{\overline{P(x) \wedge Q(x)}} \equiv \overline{\overline{P(x)} \vee \overline{Q(x)}}$. Thus, $\exists x[P(x) \wedge Q(x)] \equiv \exists x[\overline{\overline{P(x)} \vee \overline{Q(x)}}]$.

But what if we want to move negations into (or out of) quantifiers rather than just into or out of parentheses? De Morgan's Laws don't work across quantifiers, but we can generalize those laws to handle quantification. The result is sometimes called *Generalized De Morgan's Laws*.

*generalized
de morgan*

The generalization comes from a pair of observations, of which we will discuss just one. Consider the expression $\forall x \neg P(x), x \in \{1, 2, \dots, n\}$. $\forall x \neg P(x)$ is a notation for $\neg P(1) \wedge \neg P(2) \wedge \dots \wedge \neg P(n)$. If we were to apply De Morgan's to that, we would produce the equivalent expression $\neg(P(1) \vee P(2) \vee \dots \vee P(n))$. Ignoring the leading negation for a moment, the disjunction can be represented by $\exists x P(x)$. Thus, $\forall x \neg P(x)$ is logically equivalent to $\neg \exists x P(x)$. Notice that the moving the negation from inside to outside 'flipped' the quantifier from universal to existential.

Both Generalized De Morgan's Laws are given in Table 13.

Table 13: Generalized De Morgan's Laws

1. $\neg \forall x P(x) \equiv \exists x \neg P(x)$
2. $\neg \exists x P(x) \equiv \forall x \neg P(x)$

A common use for Generalized De Morgan's Laws is to move negations inside of quantifiers to create a result that is easier to express in English.

Example 51:

Problem: Construct an expression that is equivalent to $\neg\forall c\forall d D(c, d)$ by moving the negation inside of the quantifiers.

Solution: First, let's think about what $\neg\forall c\forall d$ means. $\neg\forall c$ means 'not all' – that is, some or none. The negation does not also apply to $\forall d$. (If you need both to be negated, you need a second negation, as in $\neg\forall c\neg\forall d$.) In English, we could try to say, "For not all c and for all d , ..." Or, we could try, "For some c or no c , but for all d , ..." Either way, it's pretty awkward.

Applying the first Generalized De Morgan's Law (GDM #1) twice (once per quantifier), we can move the negation inside:

$$\begin{aligned}\neg\forall c\forall d D(c, d) &\equiv \exists c\neg\forall d D(c, d) && \text{[GDM \#1]} \\ &\equiv \exists c\exists d\neg D(c, d) && \text{[GDM \#1]}\end{aligned}$$

This is easy to express in English; in fact, we already did, back in Example 45: "Some plants don't have some defenses."

2.7 “Exactly n ” Expressions

*uniqueness
quantifier*

If you've studied quantifiers previously, you may have encountered the *uniqueness quantifier*, $\exists!x$. It was created to represent the concept of “exactly one.” This is a useful concept, of course, but we don't need a new quantifier to express that idea – we can express it using the quantifiers and logical operators we already have. Learning how to do it is a worthwhile activity, not only as additional practice with quantification, but also as a starting point for extensions of the idea.

2.7.1 Exactly One

To represent this idea, we need a way to exclude “or more” from the “there exists one or more” meaning of \exists . One way to do that is to think of what “one or more” means. For the sake of illustration, consider integers. If x represents a quantity of items and is an integer, then $x \geq 1$ represents “one or more” of those items. Similarly, $x = 1$ represents “exactly one.” To reduce $x \geq 1$ to

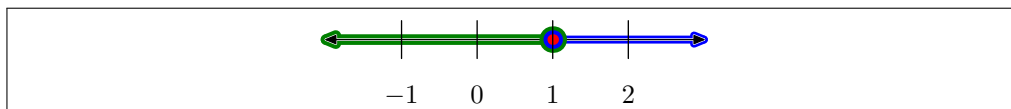


Figure 2.2: $(x \leq 1) \wedge (x \geq 1) \equiv (x = 1)$

$x = 1$, we can AND $x \geq 1$ with $x \leq 1$ – their intersection is $x = 1$. Figure 2.2 illustrates this on a number line.

Now all we have to do is figure out how to express $x \leq 1$ (in English, “no more than one”) logically. Imagine that you thought you had two cans of beans in the cupboard, but you can find only one. The following conversation with your roommate ‘Slash’ results:

You: “Seriously, your name is ‘Slash?’ ”
Slash: “Deal with it, OK?”
You: “Whatever. Look, I can’t find the second can of beans. Do you see it?”
Slash: (takes a look) “Idiot, it’s sitting right in front!”
You: “That’s the first can, genius!”⁹

It may be hard to believe, but there *is* a point to that touching story: If you’ve think you’ve found two or more items, but they all turn out to be the same item, then you have exactly one item.

This “no more than one” idea can be turned into logic fairly easily. The following description takes several liberties, but should help clarify the structure of the final logical expression. Let y be you and s be Slash. You search through all of the cans ($\forall y$) and find one of beans ($B(y)$ is true). Slash does the same ($\forall s$) and also finds one ($B(s)$ is true). But if there is only one can of beans, then the can you found is the same can he found ($y = s$). That is, any time more than one can seems to have been found, just one was found. As a quantified expression: $\forall y \forall s [(B(y) \wedge B(s)) \rightarrow (y = s)]$, where the domain of y and of s is the content of the cupboard (that is, the things we’re trying to count). In English: “Whenever two cans of beans are found, they’re really the same can.” Or, “There is at most one can of beans in the cupboard.” (‘No cans’ is a viable option; this is a conditional expression.)

Almost done. All we have left to do is to AND that expression with one for “at least one” and we’ll have the structure of an expression for “exactly

⁹Tune in next week, when you and Slash try to find your marbles, and hilarity ensues!

one.” And that’s easy:

$$\exists a P(a) \wedge \forall b \forall c [(P(b) \wedge P(c)) \rightarrow (b = c)], \quad a, b, c \in \text{The Domain}$$

$\exists a P(a)$ gives the “at least one” meaning, and the rest gives “at most one.” Together, we have “exactly one.” The two halves are independent expressions glued together.

There are several ways to express “exactly one” in first-order logic. One manages to shorten the version above by reusing the \exists variable within the \forall half:

$$\exists a (P(a) \wedge \forall b [P(b) \rightarrow (b = a)]), \quad a, b \in \text{The Domain}$$

In logical English: “There exists a domain element that has the property and, for all domain elements, if an element has the property, it’s the same element as the first one.” Conversationally: “Exactly one domain element has the property.” You can think of a as representing the “exactly one” candidate, and b as representing all of the challengers.

Now that we have two general structures that we can use, converting “exactly one” expressions to logic is fairly easy: Identify the domain and necessary predicates for the given situation, and use them instead of the placeholders in either of the two general structures, with adjustments as needed for the characteristics of the problem.

Example 52:

Problem: Express “Earth has exactly one moon” as a quantified logical expression.

Solution: We are trying to count moons, so we will start by using a domain of “moons,” but will show how to change it to the more general “heavenly bodies” later.¹⁰ We also need a way to say that a moon belongs to Earth. “ $O(x) : x$ orbits Earth” will do the job. Notice that our choice of domain for x prevents consideration of artificial satellites. All that remains is to insert these pieces into one of the two general expressions developed above. Let’s use the second. The domain becomes ‘Moons’ and $O()$ goes in place of $P()$:

$$\exists m (O(m) \wedge \forall n [O(n) \rightarrow (n = m)]), m, n \in \text{Moons}$$

To use the more general “heavenly bodies” as the domain, we need to add a predicate that describes moons. Easy: “ $M(x)$: x is a moon.” But where does this predicate go within the expression? Also easy: Alongside each occurrence of $O()$. The reason is that we are talking about Earth moons. $M()$ handles moons, and $O()$ gives them to Earth; together, they describe moons of Earth. The expression:

$$\exists h (O(h) \wedge M(h) \wedge \forall i [(O(i) \wedge M(i)) \rightarrow (i = h)]), \\ h, i \in \text{Heavenly Bodies}$$

One more adjustment: Some of you are probably not happy with $O()$ including Earth as a constant, and would prefer that $O()$ be more general so that our Earth restriction is more obvious. No problem: We can change $O()$ to have two arguments – “ $O(x, y)$: x orbits y ” – and can use ‘Earth’ for y in the expression:

$$\exists h (O(h, \text{Earth}) \wedge M(h) \wedge \forall i [(O(i, \text{Earth}) \wedge M(i)) \rightarrow (i = h)]), \\ h, i \in \text{Heavenly Bodies}$$

Example 53:

Problem: Express this “exactly one” situation in logic: For all positive integers p , there’s exactly one non-negative integer i such that $p \cdot i = 0$.

Solution: You’re probably thinking, “Well, that’s a long-winded way of saying that anything times zero is zero!” And it is. But, the point of the example is not long-windedness.¹¹

At first glance, the structure of the statement makes this exactly one situation appear to be different than the previous examples. This statement begins with a \forall and the “exactly one” is buried inside of it. Thanks to the flexibility of English, we can rewrite it to match the structure we expect to see: *Exactly one non-negative integer i makes $p \cdot i = 0$, where*

¹¹If you’re expecting a joke about swimsuit models, you’ll have to make your own. I’m not looking for trouble!

p is any positive integer.

Assuming that we'll again use the shorter “exactly one” structure, we can see that i should be the existentially quantified variable and can guess that p will be universally quantified (one quantifier seeks one variable for a logical marriage, right?). But what should the predicate(s) be? We're only dealing with integers, but both of the variables have certain restrictions – i is non-negative and p is positive. Perhaps they can be our predicates:

$$\exists i ((i \geq 0) \wedge \forall p [(p > 0) \rightarrow (p \cdot i = 0)]), i, p \in \mathbb{Z}$$

Something's missing – nothing here says that there's exactly one i . There's at least one, but there can be more. (If this doesn't make sense, try converting the expression to English.)

The problem is that we allowed p to fill the role of the challenger to i , but p is really just a bystander who got caught up in the excitement. We erred by trying to blindly fit our problem to the exactly-one structure we wanted to use, forgetting to think about what the parts of that structure represent. To do “exactly one,” we need candidate and challenger variables that come from the same domain and represent the same kind of items. Compare the above attempt to this:

$$\exists i ((i \geq 0 \wedge p \cdot i = 0) \wedge \forall j [(j \geq 0 \wedge p \cdot j = 0) \rightarrow (j = i)]), i, j, p \in \mathbb{Z}$$

This looks like progress; if nothing else, the structure looks better. In English: “There's an integer that's non-negative that we can multiply with p to produce zero and, for all integers, if we find such an integer, it's the same as the first one.” But we're not finished – p has two issues. First, we gave p a domain of \mathbb{Z} , but we need it to be a positive integer. Second, we didn't quantify p , which means it's free and we don't allow free variables (see Section 2.2.2). Because p can be any positive integer, p should be universally quantified.

As long as we're fixing those problems, let's also consider the $i \geq 0$ conditions. We can eliminate those by using a domain of \mathbb{Z}^* for i and j . You can argue that by using such a domain we're ignoring our ‘one step above things’ rule for domains, but as \mathbb{Z}^* , \mathbb{Z}^+ , and the like are well-

accepted and commonly-used in mathematics, we can make an exception without feeling too guilty. And, allowing the use of \mathbb{Z}^* means we’re free to use \mathbb{Z}^+ to restrict p to be a positive integer. After making all of these adjustments, our final expression is:

$$\forall p \exists i ((p \cdot i = 0) \wedge \forall j [(p \cdot j = 0) \rightarrow (j = i)]), i, j, \in \mathbb{Z}^*, p \in \mathbb{Z}^+$$

It’s no wonder mathematicians call $p \cdot 0 = 0$ the *zero product property*; giving it a name is so much easier.

2.7.2 Exactly Two

Understanding how to express “exactly one” is a challenge. Happily, with “exactly one” covered, “exactly two” won’t be as difficult. In fact, we’ll approach it the same way, starting with almost the same equivalence: $(x \leq 2) \wedge (x \geq 2) \equiv (x = 2)$.

Let’s start with $x \geq 2$. We don’t have a quantifier for that, like we do for $x \geq 1$. We need to construct our own expression possessing the ‘more than one’ meaning. \exists means ‘at least one,’ as we know. A pair of \exists s means ‘at least two’ so long as they are representing two distinct items. In symbols: $\exists a \exists b (P(a) \wedge P(b) \wedge (a \neq b))$. Because a and b are different items both possessing the property of interest ($P()$), we have two. Because there’s no limit on more such items existing, we also still have the ‘or more’ meaning.

Now consider $x \leq 2$. To get $x \leq 1$, we created an expression that means, essentially, “if we think we found two, we were wrong.” We’ll just change ‘two’ to ‘three.’ $\forall c \forall d \forall e [(P(c) \wedge P(d) \wedge P(e)) \rightarrow ((c = d) \vee (c = e) \vee (d = e))]$. This says that if c , d , and e all have the property, we’re at least double-counting (two of the three are the same), and maybe even triple-counting (all three are the same). Double-counting means there are two items with the property, triple-counting means there is just one, and the implication allows for there to be none. In short, we have captured the meaning ‘at most two.’

The simple way to get “exactly two” is to AND these two expressions together, just as we did to create the first version of our “exactly one” structure:

$$\begin{aligned} & \exists a \exists b (P(a) \wedge P(b) \wedge (a \neq b)) \wedge \\ & \forall c \forall d \forall e [(P(c) \wedge P(d) \wedge P(e)) \rightarrow ((c = d) \vee (c = e) \vee (d = e))], \end{aligned}$$

¹¹Surprised? ☹

$a, b, c, d, e \in \text{The Domain}$

Also just like “exactly one,” we can create a shorter version that reuses the \exists variables:

$$\exists a \exists b (P(a) \wedge P(b) \wedge (a \neq b) \wedge \forall c [P(c) \rightarrow ((c = a) \vee (c = b))]),$$

$a, b, c \in \text{The Domain}$

Example 54:

Problem: Express in logical notation: “My bicycle has exactly two wheels.”

Solution: Because we’re counting bicycle wheels, that will be our domain.¹² We need a predicate to say that the wheels belong to my bike: “ $B(x) : x$ is part of my bike” will suffice. We’re ready to put them in place:

$$\exists w \exists x (B(w) \wedge B(x) \wedge (w \neq x) \wedge \forall y [B(y) \rightarrow ((y = w) \vee (y = x))]),$$

$w, x, y \in \text{bicycle wheels}$

That’s it! No intentional errors, no new twists. Many English \rightarrow Logic problems are straight-forward. But, there is one lingering ‘thing’ thing . . .

“Bicycle wheels” is more than a single step above ‘Things.’ “Wheels” would be better, with the addition of a predicate that’s true when the given wheel is used on bicycles. If you’ve followed the previous examples where we’ve simplified domains (examples 42 and 52, say), you know how to adjust this expression. Do it! And, don’t be afraid to adopt this approach yourself on future problems – use a complex domain until you get the logical structure in place, then simplify the domain and add the corresponding predicate(s).

2.7.3 Exactly n

We can use this $(x \leq n) \wedge (x \geq n) \equiv (x = n)$ approach to handle any $n \geq 1$ we need. Unfortunately, the logic gets polynomially messier as n increases. The

¹²I know what you’re thinking; patience . . .

number of comparisons in the \forall half of the uncombined expressions follows a sequence known as the *Triangular numbers*: 0, 1, 3, 6, 10, 15, . . . The n -th Triangular number equals $\frac{1}{2}(n^2 + n)$. For instance, if you had to express ‘exactly four,’ you’d have five variables in the \forall section, and would need to express that at least two of them are the same. There are 10 ways to pair up those five variables: $\{(a, b), (a, c), (a, d), (a, e), (b, c), (b, d), (b, e), (c, d), (c, e), (d, e)\}$.¹³

On the bright side, it’s unusual to need to express situations above “exactly two,” even in a discrete structures class. But now you know how to do it if you need to.

¹³This set is an example of a *binary relation*. We cover a lot of material on such sets in Chapter 8.

Chapter 3

Logical Arguments

Monty Python’s Flying Circus, a late-1960s/early-1970s British sketch-comedy television show, was created by a group of five Oxford- and Cambridge-educated comedy writers and one American animator. Several of the sketches from that show have acquired comedic immortality, including the “Argument Clinic” from the episode “The Money Programme” that aired in November of 1972. Figure 3.1 is a screen-capture. In the bit, a man (“Man,” played by Sir Michael Palin¹) pays for what turns out to be an unsatisfying argument with a second man (“Mr. Vibrating,” John Cleese²). If you have never seen it, you should stop reading this and go watch it.³

Welcome back! In the middle of the non-argument argument, Palin gives a very good definition, which we adopt here.

Definition 16: Argument

“An *argument* is a connected series of statements to establish a definite proposition.”

argument

Our demonstrations of logical equivalence from Chapter 1 are covered by this definition, as we used a sequence of logical expressions to establish that all expressions in the sequence are logically equivalent. Arguments can be much

¹Palin was appointed a Knight Commander of the Most Distinguished Order of Saint Michael and Saint George in 2019, for services to travel, culture and geography.

²Because I know you’re wondering: Cleese has routinely declined royal honors.

³Many of Monty Python’s sketches, including the Argument Clinic, can be found on YouTube: <https://www.youtube.com/user/MontyPython>



Figure 3.1: Michael Palin, left, argues arguments with John Cleese in Monty Python’s “Argument Clinic” sketch. Credit: BBC.

more involved. This chapter introduces common argument structures and shows how logical equivalences can be combined with rules of logical inference to create complex arguments, our final stop on the road to proofs.

Please be aware that what are often called ‘arguments’ on television and radio talk shows rarely possess a logical foundation. This chapter’s arguments will be based on logic, will not cover politics, and will require no raised voices or finger-pointing.

3.1 Reasoning

Have you ever written down an answer to an exam question for no better reason than it felt like the correct answer? If so, you relied on your *intuition*, for better or worse. You probably feel more confident in answers that you can convincingly justify. The process of drawing justifiable conclusions from given information using principles of logic is *reasoning*.

reasoning

There are many forms of reasoning, some more formal than others. Here, we have need of two forms of reasoning: *deductive* and *inductive*.

3.1.1 Deductive Reasoning

One night, before you go to bed, you set your phone on the nightstand, fully expecting that it will still be there when you wake up in the morning. You have that expectation because you have two facts: You placed the phone on the nightstand, and Newton's First Law of Motion (a motionless object will remain motionless until a force is applied).⁴ Assuming that nature, pets, and practical-joking roommates fail to apply adequate forces, those two facts will convince you that your phone will be on the nightstand in the morning. The logical connection between those facts and your conviction is *deduction*.

deduction

Definition 17: Deductive Argument

A *deductive argument* uses general principles of logic, applied to a given set of facts, to draw a conclusion from those facts.

deductive argument

Example 55:

Problem: You have a dog as a pet. You have dropped a piece of bread on the floor. You conclude that you won't have to pick up the piece of bread. Is that a deductive argument?

Solution: Yes, that is a deductive argument. Anyone who has ever had a dog knows that dogs will eat any food – or anything that remotely mimics food – that reaches the ground. Combine that knowledge with the given facts, and you know that the bread is as good as gone.

Example 55's solution is incomplete – we don't (yet!) know the general principle of logic that is being applied to those facts to reach that conclusion. Logical justifications are an essential component of a complete deductive argument.

All of the proof forms presented in this book will be forms of deductive arguments . . . even the ones named after induction.

⁴Second definition: An unguarded Fig Newton will move into the nearest mouth.

3.1.2 Inductive Reasoning

You've probably noticed that the sun rose in the east two days ago. It did the same thing yesterday morning, and again this morning. Based on that, you probably expect it to rise in the east again tomorrow morning, and every day thereafter.⁵ This is an example of reasoning by *induction*: You started with some observations, and formed a general conclusion based on those observations.

induction

inductive argument

Definition 18: Inductive Argument

An *inductive argument* reaches a general conclusion from a set of specific observations.

Example 56:

Episodes of the Fox television show “Bones” usually start with unsuspecting people finding human remains, and conclude with the perpetrator being captured by an FBI agent and a forensic anthropologist. The following dialog appears in the first-season episode “Woman in the Tunnel” as the team examines a virtual representation of tunnels under a city:

⁵Until the Earth's magnetic poles flip and the sun starts rising in the west. That won't happen for thousands of years . . . probably.

Booth: Well, we found that Civil War victim near a cave-in. Maybe the treasure's on the other side?

Goodman: Inductive, reductive or deductive?

Brennan: Deductive.

Goodman: As you wish. Ms. Montenegro, please remove all tunnels containing power, cable or utility lines.
[...]

Booth: Oh, what about diamond dust? You said that there was diamond dust in the old tunnels. There was also diamond dust on the Civil War guy, so ... what? I'm not allowed to help now?

Goodman: That's inductive logic.

Brennan: We agreed on deductive.

Booth: I'm sorry, I'm just, you know, trying to think outside your box.

The Goodman character deduces from the facts that the body was from the U.S. Civil War period and that people in the 1860s did not have amenities such as cable television to decide that a body from that time could not have been resting in a tunnel with modern wiring. Later, Booth tries to generalize from specific examples of the appearance of diamond dust to draw a conclusion. That's inductive reasoning.

The dialog used in Example 56 includes a reference to *reductive* reasoning, which is also a useful if somewhat confusing way to form an argument. We won't discuss it in this chapter, but we will in Chapter 4.

Defining inductive arguments isn't as straight-forward as defining deductive arguments. In philosophy, inductive reasoning allows for uncertainty – an inductive argument strongly suggests that the conclusion is correct, but doesn't claim that it is correct. In mathematics and the sciences, we crave certainty, but the structure of inductive reasoning is appealing because it is directly applicable to a variety of provable situations. To address this dilemma, *mathematical induction* wraps deductive reasoning in a shell of inductive reasoning, giving us the form of induction but the certainty of deduction. We will cover mathematical induction in a later chapter.

3.1.3 Abductive Reasoning

The American city of Tucson, Arizona appears to be roughly surrounded by mountain ranges, including the Rincons to the east, the Santa Catalinas to



Figure 3.2: Mountain ranges near Tucson, Arizona. Credit: Arizona Daily Star.

the northeast, the Tortolitas to the northwest, and the Tucson Mountains to the west (see Figure 3.2). It's tempting to assume that this current geologic arrangement was formed from the collapse of an ancient and massive volcano's caldera, with the current mountain ranges being the uncollapsed edges of the volcano.⁶

abduction

Such an assumption can arise from abductive reasoning. In *abduction*, we start with the end result and propose possible situations that could have produced that result. This is roughly the converse of the deductive reasoning process combined with the uncertainty of inductive reasoning. As such, it isn't useful logically, although it can be helpful for brainstorming ideas that might turn out to be provable.

Example 57:

Problem: In Gotham City, Detective Montoya is investigating a homicide.

⁶Tempting, but likely incorrect. Based on the types of rocks, the current thinking is that the volcano was where the Catalinas are, and detachment faults caused the Tucson Mountains to 'slide' to the southwest. The city is built on the mile-deep sediment that rests in-between.

While searching the victim, she finds a joker playing card in his hand. Without hesitating, she pulls out her phone and calls Commissioner Gordon. Why did she feel it necessary to ignore her chain of command and call him directly?

Solution: Through abductive reasoning, of course! In Gotham, a dead man holding a joker could mean that the Joker has escaped from Arkam Asylum and has killed again. Or, perhaps the Penguin is the real culprit and is trying to shift the blame. Either way, it's time to call Batman. The brainstorming that occurs when using abductive reasoning is helpful when investigating crimes.⁷

Given its speculative nature, we will consider abduction no further.

3.2 Categorizing Deductive Arguments

For the rest of this chapter, we will consider only deductive arguments. Some deductive arguments are more speculative than others. In this section we will discuss that distinction.

3.2.1 Writing Deductive Arguments

In section 1.3.6, we learned that in an implication, the left operand can be called the hypothesis and the right the conclusion. Deductive arguments can be viewed as implications with multiple hypotheses (some given to us and some deduced by us) and a single ultimate conclusion.

Example 58:

We can rewrite the deductive argument from Example 55 as an if-then sentence: *If you have a dog and you dropped a piece of bread on the floor, then you did not need to pick up that piece of bread.* This makes the individual hypotheses of the compound hypothesis, and the intended conclusion, clear. Written in logic, we have $(d \wedge b) \rightarrow \neg n$, where d is 'you have a dog', b is 'you dropped a piece of bread on the floor', and n is 'you need to pick up that piece of bread'.

⁷Even fictional ones. Yes, Batman is a fictional character. Deal with it.

Although the hypotheses and the conclusion are clear, the justification that allows us to reach that conclusion from those hypotheses is not. As mentioned earlier in this chapter, a complete deductive argument also includes justifications for its steps. To make it easy for us to include these explanations, we will adopt a vertical format for our argument components. For example, here's the vertical form of the logical expression $(d \wedge b) \rightarrow \neg n$ from Example 58:

$$\frac{\begin{array}{c} d \\ b \end{array}}{\therefore \neg n}$$

The vinculum⁸ separates the hypotheses from the conclusion, and the hypotheses are assumed to be ANDed together. The symbol \therefore (`\LATEX : \therefore`) means ‘therefore’ and is used to highlight the conclusion. For such a short expression, we could adopt the horizontal equivalent: $d, b / \therefore \neg n$. The advantage of the vertical form is that we have room to the side to provide each line with a justification. While we're at it, we can also number the lines, to make referencing them easy:

$$\begin{array}{l} (1) \quad d \quad [\text{Given}] \\ (2) \quad b \quad [\text{Given}] \\ \hline (3) \quad \therefore \neg n \quad [\text{Ummmm ... magic?}] \end{array}$$

Sadly, we still don't know how to get to the conclusion. But we will!

3.2.2 Valid and Sound Arguments

For an argument to be convincing, we must be able to see that the truth of the conclusion is based on the truth of the available hypotheses. Such an argument is said to be *valid*.

valid argument

Definition 19: Valid Argument

Any deductive argument of the form $(p_1 \wedge p_2 \wedge \dots \wedge p_n) \rightarrow q$ is *valid* if the truth of conclusion q can be derived from the truth of hypotheses p_1 through p_n .

⁸The horizontal line, but you already knew that because you've read the math review appendix ... right?

What the definition of a valid argument doesn't say is whether or not the hypotheses are actually true! This might appear to be a huge problem, but it isn't. If you think back to abductive reasoning for a moment, you'll recall that it was speculative – we thought of one or more situations that could have caused the conclusion to result. We can assume the truth of the hypotheses that describe one of those situations, correctly apply logic to those hypotheses, and determine if the conclusion follows. If it does, the validity of the argument has been logically established, even though the hypotheses are not yet known to be true.

Example 59:

Consider the following argument:

- | | | |
|-----|--|--|
| (1) | 63 is evenly divisible by 2 | |
| (2) | 63 is evenly divisible by 3 | |
| | | |
| (3) | \therefore 63 is evenly divisible by 6 | |

What do you think of that argument? If you are dismissing it because 63 is not evenly divisible by two, you're missing the point. The argument itself is valid: When you have a number that is divisible by both two and three, it will also be divisible by six. Remember, in a valid argument, the hypotheses need only be assumed true.

Valid arguments, then, have room for improvement: We could insist that the hypotheses actually be true! For the argument to be anything more than an exercise in logical construction, they need to be. If they are, we have more than a valid argument – we have a *sound* one.

Definition 20: Sound Argument

Any valid deductive argument is *sound* when its hypotheses are true.

sound argument

Because 63 is not actually evenly divisible by two, the argument of Example 59 is not sound. Changing 63 to any multiple of six will make it both valid and sound. Please note that a sound argument is necessarily valid, just as a multiple of six is necessarily a multiple of two.

3.3 Rules of Inference for Propositions

Let's reflect on what a valid argument tells us: When the hypotheses are true, the conclusion is true. Notice that a valid argument tells us nothing about the truth of the conclusion when one or more of the hypotheses are false.

Because the hypotheses of a valid argument supply enough information to reach the conclusion, we can demonstrate that validity with a truth table.

Example 60:

Consider the following trivial but valid argument:

$$\begin{array}{l} (1) \quad p \\ (2) \quad q \\ \hline (3) \quad \therefore p \wedge q \end{array}$$

We can rewrite this argument in the form of the logical expression $(p \wedge q) \rightarrow (p \wedge q)$. We don't need to create a truth table to see that this expression is true when both p and q are true, but we'll do it anyway to make a point:

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

First, look at the third column (labeled $p \wedge q$), which represents our conclusion. When both of the hypotheses are true (the first row), the conclusion is true. This shows that the argument $p, q / \therefore p \wedge q$ is valid.

Now consider the fourth column, which represents the entire argument. It's a tautology! In this example, that's also not much of a surprise; $p \rightarrow p \equiv \mathbf{T}$ is one of the logical equivalences ("Self-implication") presented in Chapter 1. Still, keep this observation in mind; we will consider it again later in this section and also in section 3.5.1.

Using truth tables to verify the validity of arguments is possible, as we just showed, but is tedious for all but the smallest of arguments. Fortunately, we have an alternative.

Remember, back in Chapter 1, when we used a given⁹ set of logical equivalences to show that more complex expressions were also equivalences? We can do the same sort of thing with arguments: We can construct large arguments using smaller arguments as building blocks. We will also make use of those logical equivalences to restate expressions into forms that suit our needs.

3.3.1 Eight Rules of Inference

The tiny valid argument used in Example 60 is known as a *rule of inference*. Many such tiny valid arguments exist, but only a few of them are used often enough in this book to be worth learning. Some, such as the one used in Example 60, are clearly valid. Others can be shown to be, again using a truth table.

rule of inference

Table 14: Some Rules of Inference Worth Knowing
--

⁹But verifiable – you don't have to trust us!

	Name (Alternate Name)	Argument
(1)	Conjunction (Conjunction Introduction)	$\frac{p}{q}$ $\therefore p \wedge q$
(2)	Simplification (Conjunction Elimination)	$\frac{p \wedge q}{p}$ $\therefore p$
(3)	Addition (Disjunction Introduction)	$\frac{p}{p \vee q}$ $\therefore p \vee q$
(4)	Disjunctive Syllogism (Disjunction Elimination)	$\frac{p \vee q}{\bar{p}}$ $\therefore q$
(5)	Modus Ponens (Affirming the Antecedent)	$\frac{p}{p \rightarrow q}$ $\therefore q$
(6)	Modus Tollens (Denying the Consequent)	$\frac{\bar{q}}{p \rightarrow q}$ $\therefore \bar{p}$
(7)	Hypothetical Syllogism (Transitivity of Implication)	$\frac{p \rightarrow q}{q \rightarrow r}$ $\therefore p \rightarrow r$
(8)	Resolution	$\frac{p \vee q}{\bar{p} \vee r}$ $\therefore q \vee r$

syllogism

The word *syllogism* is a synonym for deductive reasoning, making its use in the name of an argument a bit redundant. But, those are the names most people use, and so shall we.

If you take the time to verify the validity of these rules, you will probably notice that they are all tautologies, as we'd hinted at in Example 60. This isn't coincidence. Recall that arguments (including rules of inference) have the form $(p_1 \wedge p_2 \wedge \dots \wedge p_n) \rightarrow q$. We know that, for an argument to be valid, when the hypotheses are all true, the conclusion *must* be true. In this case, the complete argument will also be true $((\mathbf{T} \wedge \mathbf{T} \wedge \dots \wedge \mathbf{T}) \rightarrow \mathbf{T} \equiv \mathbf{T} \rightarrow \mathbf{T} \equiv \mathbf{T})$. But even if one or more of the hypotheses are false, the argument will still be true $((\mathbf{T} \wedge \mathbf{F} \wedge \dots \wedge \mathbf{T}) \rightarrow \textit{whatever} \equiv \mathbf{F} \rightarrow \textit{whatever} \equiv \mathbf{T})$. Thus, valid

arguments (including rules of inference) must always evaluate to true; in other words, they are always tautologies.

Occasionally, people lump rules of inference together with logical equivalences and attempt to use the former as if they were the latter. Rules of inferences do **not** claim that the hypotheses are logically equivalent to the conclusions. That may sometimes be true (as it is with Conjunction, for example), but most often it is not.

3.3.2 Using Rules of Inference in Arguments

We know about logical equivalences, valid arguments, and rules of inference. It's time to put them to work producing conclusions from given hypotheses. We will start with a simple argument.

Example 61:

Problem: Given $\neg b$ and $a \rightarrow b$, what can we conclude about a ?

Solution: By modus tollens, we can conclude that a must be false. Here's the complete argument, in our preferred vertical form:

$$\begin{array}{lll} (1) & \neg b & [\text{Given}] \\ (2) & a \rightarrow b & [\text{Given}] \\ \hline (3) & \therefore \neg a & [1, 2, \text{Modus Tollens}] \end{array}$$

By listing $\neg b$ as a hypothesis, we are assuming it to be true. For the negation of b to be true, b itself must be false. This is how we can say something is false when every expression is assumed true. In the same way, the conclusion of $\neg a$ says that we have determined a to be false.

Notice that we justified the conclusion by including hypothesis line numbers with the rule of inference we used. We recommend that you adopt this practice, too, because doing so makes it easier for the reader to follow the logic of the argument. In a small example such as this, it doesn't really help, but in longer arguments it will.

The next example is almost as simple, but we need to perform some con-

versions from English to Logic.¹⁰

Example 62:

Problem: You are ready to go to bed, but see some light coming from the other end of the hallway. That means that you forgot to turn off either the dining room light or the kitchen light. You stick your head out the bedroom door and see that the kitchen is dark – the kitchen light is off. How are you able to conclude that the dining room light is on?

Solution: It's tempting to answer, "Because the kitchen light's not on, duh!" While it's true that the kitchen light isn't on, that fact alone does not explain the source of the light. We need to construct a valid and completely justified argument from the given hypotheses, showing that the conclusion is true if the hypotheses are.

First step: Identify the hypotheses and conclusion. In other words, what do we know and what do we hope to show? We know that the dining room light is on or the kitchen light is on. We also know that the kitchen light is off. We hope to show that the dining room light is on. These statements will be easier to work with if we give their component propositions labels, remembering that the fact that the kitchen light is off is not a third proposition, but rather the negation of the second:

d : The dining room light is on
 k : The kitchen light is on

We can re-write our hypotheses and conclusion as logical expressions using these labels:

Hypotheses:

"the dining room light is on or the kitchen light is on" $\Rightarrow d \vee k$
 "the kitchen light is off" $\Rightarrow \neg k$

Conclusion:

"the dining room light is on" $\Rightarrow d$

How can we show that the conclusion follows from those hypotheses?

¹⁰Bet you wished you'd never have to do that again! Like it or not, it's a necessary step in many arguments, not to mention proofs.

Table 14 includes a rule of inference that fits our situation: Disjunctive Syllogism. In fact, it fits so well that it completes the argument all by itself. All we have to do is write out the argument with our adornments:

(1)	$d \vee k$	[Given]
(2)	$\neg k$	[Given]
(3)	$\therefore d$	[1, 2, Disjunctive Syllogism]

If you are particularly attentive, you may be concerned that the order of the labels in our first proposition is the opposite of the order used in the form of disjunctive syllogism in Table 14. That is, in the table, we have $p \vee q$ and $\neg p$, meaning that the negation of the left operand is true. In the example, we have $d \vee k$ and $\neg k$, where the negation of the right operand is true. This really isn't a problem. Disjunctive syllogism says that if we have an inclusive disjunction and one of the operands of this disjunctions is false, the other must be true. It doesn't matter which one is false, because inclusive-OR is commutative. If this implicit step bothers you, you can add a line to the argument that commutes the disjunction (and uses 'Commutativity of Disjunction' to justify it) before applying disjunctive syllogism.

Similarly, we could have listed $\neg k$ as line (1) and $d \vee k$ as line (2) without causing any trouble. Disjunctive syllogism would still be applicable because the hypotheses are being ANDed together (see Definition 19), and conjunction is also commutative.

Example 63:

Problem: What is wrong with this valid argument?

If seven is larger than eight, then seven minus eight is a positive number. Seven is larger than eight. Therefore, seven minus eight is a positive number.

Solution: There's nothing wrong ... with the argument! It's a perfectly lovely valid argument, as the following demonstrates:

s : Seven is larger than eight
 m : Seven minus eight is a positive number

(1)	$s \rightarrow m$	[Given]
(2)	s	[Given]
(3)	$\therefore m$	[1, 2, Modus Ponens]

If we assume the hypotheses to be true, the conclusion logically follows, thanks to modus ponens. If you were thinking that s 's offensive lack of truth invalidated the argument, you were thinking about a sound argument, where the hypotheses need to be true. In valid arguments, we can indulge flights of fancy.

Ready for a more complex argument?

Example 64:

Problem: Your car is parked facing the sun. You do not have a windshield sun shade. When your car is facing the sun without a sun shade, the steering wheel is in the sun. The steering wheel is hot, or it is not in the sun. Is the steering wheel hot?

Solution: It certainly seems reasonable that the steering wheel is hot, but we need to show how to reach that conclusion from the given information and our knowledge of logic.

As always, the first step is to identify and label the propositions, remembering to stay positive so as not to hide any negations:

- f : The car is facing the sun
- s : The car has a sun shade
- w : The steering wheel is in the sun
- h : The steering wheel is hot

Next, express the hypotheses and conclusion in logic notation:

The car is facing the sun	\Rightarrow	f
The car has no sun shade	\Rightarrow	$\neg s$
When the car faces the sun without a sun shade, the steering wheel is in the sun	\Rightarrow	$(f \wedge \neg s) \rightarrow w$
The steering wheel is hot or is not in the sun.	\Rightarrow	$h \vee \neg w$
The steering wheel is hot	\Rightarrow	h

Now the fun starts: We need to see what new truths we can uncover from the hypotheses; hopefully, h will be one of them. One way to approach this task is to start with what we're given and see where it leads us. Hopefully, it will lead us close enough to the conclusion that we'll be able to see how to complete the argument.

Looking at the third hypothesis, we can see how modus ponens can be used to establish the truth of w , if we knew that $f \wedge \neg s$ were true. Ah, but we can show that, from the first two hypotheses!

(1)	f	[Given]
(2)	$\neg s$	[Given]
(3)	$f \wedge \neg s$	[1, 2, Conjunction]
(4)	$(f \wedge \neg s) \rightarrow w$	[Given]
(5)	w	[3, 4, Modus Ponens]

Great ... but does knowing the truth of w help? The fourth hypothesis includes a $\neg w$; can that help? It almost fits the form of disjunctive syllogism, but not quite. Maybe we can find a logical equivalence that can help ... Ah! The 'something ORed with the negation of something else' form appears in the Law of Implication:

(6)	$h \vee \neg w$	[Given]
(7)	$\neg w \vee h$	[6, Commutativity of \vee]
(8)	$w \rightarrow h$	[7, Law of Implication]

(Again, many people would not include line 7 in the argument, but you can't go wrong by including it.)

At this point, you can probably see how to finish it off: Applying modus ponens to lines 5 and 8 shows that h is true. Here's the complete argument:

(1)	f	[Given]
(2)	$\neg s$	[Given]
(3)	$f \wedge \neg s$	[1, 2, Conjunction]
(4)	$(f \wedge \neg s) \rightarrow w$	[Given]
(5)	w	[3, 4, Modus Ponens]
(6)	$h \vee \neg w$	[Given]
(7)	$\neg w \vee h$	[6, Commutative Laws]
(8)	$w \rightarrow h$	[7, Law of Implication]
(9)	$\therefore h$	[5, 8, Modus Ponens]

Yes, the steering wheel hot, and we have explained why.

There's lots to say about Example 64:

1. A complete argument is frequently constructed of many small arguments. Here, we built our argument from two rules of inference and two logical equivalences.
2. Each use of a rule of inference or a logical equivalence created a new (derived) hypothesis that helped move us to our goal. Remember, both logical equivalences and rules of inference produce an expression that we can assume to be true.
3. Numbering the lines of the argument helps!
4. We used modus ponens twice in that one argument. Re-using rules of inference and logical equivalences in the same argument is fine.
5. Some people like to list all of the givens at the start of the argument (i.e., the g givens appear on lines 1 through g), while others prefer to write them into the argument just before they are needed (as we demonstrated here). Either approach is fine.

6. It's not uncommon to reason your way into a dead-end – an expression that doesn't show any sign of helping you reach your conclusion. Don't panic! Back up a step or two, and look through your rules of inference and logical equivalences again. You might find another option that will turn out to be more helpful.
7. Not every desired conclusion can be reached from every initial set of hypotheses. Example arguments in textbooks are usually (though not always) constructed to work.

The argument in Example 64 used a “top-to-bottom” technique. We can also build arguments “bottom-to-top,” as the next example demonstrates on the same set of hypotheses and the same conclusion.

Example 65:

Problem: Can you construct a different argument based on the hypotheses of Example 64 that reaches the same conclusion?

Solution: Definitely! The more complex the argument, the more ways there will be to reach the conclusion (assuming that reaching the conclusion is possible, of course).

We have the same hypotheses and desired conclusion, repeated here for convenience:

$$\begin{array}{l}
 \text{Hypotheses: } f \\
 \qquad \qquad \neg s \\
 \qquad \qquad (f \wedge \neg s) \rightarrow w \\
 \qquad \qquad h \vee \neg w \\
 \hline
 \text{Conclusion: } h
 \end{array}$$

This time, let's start at the conclusion. Can we see a way to reach h from the hypotheses? h appears only in the fourth hypothesis. We'd noted in Example 64 that disjunctive syllogism didn't quite fit that hypothesis, but it's not hard to make it fit. The structure of disjunctive syllogism needs one of the operands in the disjunction to be 'combined' with its negation. As we need to keep h , we need to combine $\neg w$ with its negation. The negation of $\neg w$ is $\neg\neg w$, which (by double negation)

is equivalent to w . Thus, if we can show that w is true, we can apply double negation and disjunctive syllogism to show the truth of h . This is how the argument will end:

$(n - 4)$	w	[we hope!]
$(n - 3)$	$\neg\neg w$	[$n - 4$, Double Negation]
$(n - 2)$	$h \vee \neg w$	[Given]
$(n - 1)$	$\neg w \vee h$	[$n - 2$, Commutative Laws]
(n)	$\therefore h$	[$n - 3, n - 1$, Disjunctive Syllogism]

Now, how can we establish the truth of w ? Well, why not the same way we did it in Example 64? We could, but we've been there and done that – let's be different. We can apply the law of implication to $(f \wedge \neg s) \rightarrow w$, producing $\neg(f \wedge \neg s) \vee w$. We already know $f \wedge \neg s$ is true. This is set up perfectly for disjunctive syllogism, giving us a final argument that is almost completely different than that of Example 64:

(1)	$(f \wedge \neg s) \rightarrow w$	[Given]
(2)	$\neg(f \wedge \neg s) \vee w$	[1, Law of Implication]
(3)	f	[Given]
(4)	$\neg s$	[Given]
(5)	$f \wedge \neg s$	[3, 4, Conjunction]
(6)	$\neg\neg(f \wedge \neg s)$	[5, Double Negation]
(7)	w	[2, 6, Disjunctive Syllogism]
(8)	$\neg\neg w$	[7, Double Negation]
(9)	$h \vee \neg w$	[Given]
(10)	$\neg w \vee h$	[9, Commutative Laws]
(11)	$\therefore h$	[8, 10, Disjunctive Syllogism]

The final argument is two steps longer than the one developed in Example 64, but is just as correct, and correctness is more important than brevity.

We've already mentioned that people often leave out commutativity as a step in arguments, assuming that the concept is so basic that it doesn't need to be stated. People often assume the same with double negation. Even if

you're not required to include them, remember that your arguments will be more complete – and easier for people to follow – with those steps in them.

To finish this section, let's fulfill a promise by (finally!) completing the argument introduced in Examples 55 and 58.

Example 66:

Problem: Which additional hypotheses must be assumed true to conclude that you don't need to pick up a piece of dropped bread, assuming that if you have a dog and you dropped the piece of bread on the floor, you don't need to pick it up?

Solution: If you followed the recent argument examples, this should be easy. Here are the labels introduced in Example 58:

- d : You have a dog
- b : You dropped a piece of bread on the floor
- n : You need to pick up that piece of bread

To reach the conclusion ($\neg n$) from the one known hypothesis ($(d \wedge b) \rightarrow \neg n$), we can use modus ponens ...if we could assume that $d \wedge b$ is true. Thus, the answer to the question is: The hypotheses d and b must both be assumed to be true. Here's the complete argument:

(1)	d	[Given]
(2)	b	[Given]
(3)	$d \wedge b$	[1, 2, Conjunction]
(4)	$(d \wedge b) \rightarrow \neg n$	[Given]
(5)	$\therefore \neg n$	[3, 4, Modus Ponens]

And now you know why that puddle of slobber is in the middle of your kitchen floor and a very attentive dog is sitting next to you with a hopeful look on its face.

3.4 Rules of Inference for Predicates

The eight rules of inference from the last section can be applied to predicates just as easily as they can be applied to propositions. The following example's

argument is that of Example 62, but using predicates instead of propositions.

Example 67:

Problem: Restate the argument from Example 62 using predicates instead of propositions.

Solution: Converting propositions to predicates isn't too difficult. The general concept expressed by the proposition becomes the predicate, and the 'subject' is generalized to be the variable. In Example 62, there's just one concept, that of a light being on. Thus, our single predicate is:

$$L(x) : x \text{ is on, } x \in \text{Lights}$$

The argument involves two specific lights: Those of the dining room and the kitchen. Those lights will be constants passed to the predicate to be used to express the hypotheses and conclusion. Here's the re-written argument:

(1)	$L(\text{dining room}) \vee L(\text{kitchen})$	[Given]
(2)	$\neg L(\text{kitchen})$	[Given]
(3) $\therefore L(\text{dining room})$		
		[1, 2, Disjunctive Syllogism]

This application of disjunctive syllogism is correct because “ $L(\text{dining room})$ ” and “the dining room light is on” are just two ways of supplying the same information; that is, two ways of saying the same thing. We think that using predicates when they aren't needed (as in this example) is a bad idea. The predicates add a layer of abstraction and additional syntactic clutter without supplying any benefit. Don't use predicates unless you have to.

3.4.1 Four Rules of Inference for Quantified Expressions

When might you need predicates in an argument? We introduced predicates in Chapter 2 because quantifiers apply to variables and predicates accept variables. Should we need to employ a rule of inference that depends on a

quantifier, we won't have any choice but to express the argument in terms of predicates.

Such rules of inference are not hypothetical. There are four such rules for quantified expressions, two per quantifier, that you may find useful in arguments. Table 15 lists them.

Table 15: Rules of Inference for Quantified Expressions		
	Name	Argument
(1)	Universal Instantiation	$\frac{\forall x P(x), x \in D}{\therefore P(d), d \in D}$
(2)	Universal Generalization	$\frac{P(d) \text{ for every } d \in D}{\therefore \forall x P(x), x \in D}$
(3)	Existential Instantiation	$\frac{\exists x P(x), x \in D}{\therefore P(d) \text{ for some } d \in D}$
(4)	Existential Generalization	$\frac{P(d) \text{ for some } d \in D}{\therefore \exists x P(x), x \in D}$

3.4.2 Using Quantified Expressions in Arguments

Example 68:

Problem: Bicycle Repair Man is Mr. F. G. Superman's secret identity.¹¹ Mr. Superman is a Briton. Does there exist at least one Bicycle Repair Man who is also a Briton?

Solution: It seems very likely that the answer is 'yes,' but how do we know? Is this the right place for an argument?¹²

Our desired conclusion is an existentially-quantified expression, so we need to use predicates to construct the argument. Let's use these ...

$R(x)$: x is Bicycle Repair Man, $x \in \text{People}$

$B(x)$: x is a Briton, $x \in \text{People}$

... to express our hypotheses ($R(\text{F. G. Superman})$ and $B(\text{F. G. Superman})$) and desired conclusion ($\exists x(R(x) \wedge B(x))$). With all of that in place, the argument itself is straight-forward, thanks to existential generalization. Note that we've shortened Mr. Superman's name to get the table rows to be of reasonable lengths:

(1)	$R(\text{F. G. S.})$	[Given]
(2)	$B(\text{F. G. S.})$	[Given]
(3)	$R(\text{F. G. S.}) \wedge B(\text{F. G. S.})$	[1, 2, Conjunction]
(4)	$\therefore \exists x(R(x) \wedge B(x)), x \in \text{People}$	[3, Existential Gen.]

Yes, there is a Briton who is Bicycle Repair Man, to the relief of bicyclists all across England.

You might be wondering why we didn't have a binary predicate (e.g., " $S(x, y)$: The secret identity of x is y "), instead of $R(x)$, so that we could represent the secret identities of many people. We certainly could have, but as we only needed to worry about one superhero, we didn't a more flexible predicate.

Example 69:

Problem: If Alice knows how to swim, so does Bob, but Bob doesn't know how to swim. Create an argument demonstrating that the statement "Someone knows how to swim" is false.

Solution: The first word ('someone') of the desired conclusion makes it clear that we need a predicate covering the ability of a person to swim. Less clear is how far-reaching 'someone' should be – all people or just Alice and Bob? As it's quite a jump from Alice and Bob to every human,

¹¹See the Bicycle Repair Man sketch from Episode 3 ("How to Recognise Different Types of Trees From Quite a Long Way Away") of Monty Python's Flying Circus.

¹²"I've told you once." ☺

we'll assume the set of size two. As a convenience, we use D to label the domain set $\{\text{Alice}, \text{Bob}\}$.

We need just one predicate with which to express our two hypotheses and single conclusion:

Predicate: $S(x) : x$ knows how to swim, $x \in D$

Hypotheses: $S(\text{Alice}) \rightarrow S(\text{Bob})$ (*If Alice can swim, Bob can swim*)

$\neg S(\text{Bob})$ (*Bob can't swim*)

Conclusion: $\neg \exists x(S(x)), x \in D$ (*No one can swim*)

There's a clear first step here: Apply modus tollens to the two hypotheses to show that Alice can't swim. This means that they both can't swim, or, stated another way, everyone in our small domain cannot swim. That's not quite what we were asked to show, but an application of generalized De Morgan's laws will get us the rest of the way:

(1)	$S(\text{Alice}) \rightarrow S(\text{Bob})$	[Given]
(2)	$\neg S(\text{Bob})$	[Given]
(3)	$\neg S(\text{Alice})$	[1, 2, Modus Tollens]
(4)	$\neg S(\text{Alice}) \wedge \neg S(\text{Bob})$	[2, 3, Conjunction]
(5)	$\forall x(\neg S(x)), x \in D$	[4, Universal Generalization]
(6)	$\therefore \neg \exists x(S(x)), x \in D$	[5, Gen. De Morgan's Laws]

Line (4) isn't strictly necessary; we could justify the universally quantified statement directly from lines (2) and (3). But, as universal quantification means that the expression is true for every member of the set, putting both $\neg S(\text{Alice})$ and $\neg S(\text{Bob})$ on the same line – showing that every member of the domain makes $\neg S()$ true – allows the jump to the quantified expression be a little easier to see.

Guess what Alice and Bob's parents will sign them up to learn this summer (besides cryptography)?¹³

Getting tired of writing the same domain over and over? Example 69 demonstrates one way of speeding things up: Give the domain a single-letter

¹³'Alice' and 'Bob' are the traditional names used by cryptographers for the people exchanging encrypted messages. 'Eve' might soon be in the class, too ...

label. You can also write the domain once and state that it is used for the entire answer. We'll demonstrate this in the next example.

The first two examples in this section started with facts that, until this chapter, we would have written as propositions, and ended with conclusions that are quantified expressions. It's entirely possible to start an argument with a quantified expression and end with a thinly-disguised proposition.

Example 70:

Problem: The entire Sharma family is escaping the afternoon heat at the mall. Everyone at the mall has sore feet. Demonstrate that Diya Sharma has sore feet.

Solution: You know what to do first: Identify the predicates needed to express the hypotheses and the conclusion, and then express them.

(Note: The domain of x is "People" in this solution.)

Predicates: $S(x)$: x is a Sharma family member
 $M(x)$: x is at the mall
 $F(x)$: x has sore feet

Hypotheses: $\forall x (S(x) \rightarrow M(x))$ (*You're a Sharma, you're malling*)
 $\forall x (M(x) \rightarrow F(x))$ (*You're malling, your feet hurt*)
 $S(\text{Diya})$ (*Diya's a Sharma family member*)

Conclusion: $F(\text{Diya})$ (*Poor Diya has sore feet*)

The shortest way to construct this argument is to start by combining the two quantified expressions. We can do this with hypothetical syllogism because they are both universally quantified and both have the same domain. Add in a little universal instantiation and a pinch of modus ponens, and we've got an argument. Because we've already stated that $x \in \text{People}$ for the entire answer, we won't include the domain in the argument.

(1)	$\forall x (S(x) \rightarrow M(x))$	[Given]
(2)	$\forall x (M(x) \rightarrow F(x))$	[Given]
(3)	$\forall x (S(x) \rightarrow F(x))$	[1, 2, Hypothetical Syllogism]
(4)	$S(\text{Diya})$	[Given]
(5)	$S(\text{Diya}) \rightarrow F(\text{Diya})$	[3, Universal Instantiation]
(6)	$\therefore F(\text{Diya})$	[4, 5, Modus Ponens]

For practice: Construct a second argument that also uses hypothetical syllogism, but later in the argument. Want more practice? Make a third argument that doesn't use hypothetical syllogism at all. Both of these versions will be longer than the argument given here, but validity also matters more than brevity.

This is a good time for a reminder: Don't overthink the context of your arguments, just as we warned you not to overthink expressions of predicates. Don't worry about details such as when the Sharmas are at the mall, or if Diya is even old enough to walk. Our purpose here is to make you comfortable with logical reasoning. Accept the context of the example, work within that framework, and you'll avoid introducing irrelevant details and concerns that will slow you down by complicating the problem unnecessarily.

Example 70 demonstrated that we can instantiate predicates to constants (e.g., $\forall x(S(x) \rightarrow F(x))$ to $S(\text{Diya}) \rightarrow F(\text{Diya})$). Often, we do not know the names of the elements of our domains. In those situations, we can use a placeholder symbol to represent a sample member of the domain. The next example uses this approach.

Example 71:

Problem: All cars are not plaid or are heavy (or both). There is a car that is plaid. Using this information, show that a heavy car must exist.

Solution: We start as usual: Define suitable predicates, and express the hypotheses and conclusion in terms of those predicates.

(Note: All domains are “Cars.”)

Predicates: $P(x) : x$ is plaid
 $H(x) : x$ is heavy

Hypotheses: $\exists x P(x)$ (*A plaid car exists*)
 $\forall x (\neg P(x) \vee H(x))$ (*All cars aren't plaid or are heavy*)

Conclusion: $\exists x H(x)$ (*Some car is heavy*)

Time for the argument. Unlike the previous examples, we don't have a specific set element (that is, an identifiable car) in mind. We'll assign an identifier when we need to instantiate a car; we'll say more about that below.

(1)	$\exists x P(x)$	[Given]
(2)	$P(c)$	[1, Existential Instantiation]
(3)	$\forall x (\neg P(x) \vee H(x))$	[Given]
(4)	$\neg P(c) \vee H(c)$	[3, Universal Instantiation]
(5)	$H(c)$	[2, 4, Disjunctive Syllogism]
(6)	$\therefore \exists x H(x)$	[5, Existential Generalization]

In Line 2, we selected c to represent a plaid car (thanks to Line 1, we know that at least one such car exists). We can use c again in Line 4 because Line 3 is universally quantified; every car, including c , makes this expression true. We cannot stop with Line 5 because we need to match the form of the desired conclusion; yes, the truth of $H(c)$ means that such a car exists, but we need the final step to conclude the argument appropriately.

Not a fan of Disjunctive Syllogism and don't mind an extra step? Using the Law of Implication, $\neg P(c) \vee H(c)$ can be shown to be equivalent to $P(c) \rightarrow H(c)$, and then Modus Ponens can be used to show that $H(c)$ is true.

3.5 Fallacies

Until now, our arguments have been firmly supported by logic. Detailing that support was a bit tedious, perhaps, but wasn't too difficult with the help of rules of inference and logical equivalences. Not many people have the knowledge or the patience to verify that their conclusions follow from a logical argument. An unfortunate consequence is that people can create – and worse,

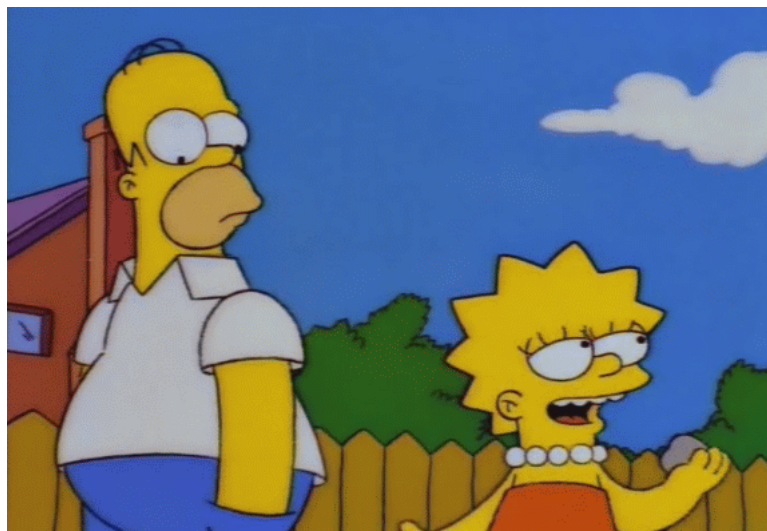


Figure 3.3: Lisa tries to explain specious reasoning to Homer in “Much Apu About Nothing,” a seventh season episode of “The Simpsons.” Credit: Fox.

believe and act upon – invalid arguments that appear to be valid due to lack of critical examination. Such arguments are examples of *specious reasoning*.

specious reasoning

A noted creator of, and believer in, specious arguments is the animated character Homer Simpson in the long-running American television series “The Simpsons.” In the seventh season episode “Much Apu About Nothing,” in the scene pictured in Figure 3.3, Homer and his intelligent (and surprisingly well-educated) daughter Lisa have a conversation about the impact of a recently-implemented bear patrol in Springfield:

Homer: Ah, not a bear in sight. The Bear Patrol must be working like a charm.

Lisa: That's specious reasoning, Dad.

Homer: Thank you, honey.

Lisa: By your logic I could claim that this rock keeps tigers away.

Homer: Oh, how does it work?

Lisa: It doesn't work.

Homer: Uh-huh.

Lisa: It's just a stupid rock.

Homer: Uh-huh.

Lisa: But I don't see any tigers around here, do you?

Homer: Lisa, I want to buy your rock.

Lisa is trying to demonstrate that correlation doesn't mean causation. While it's true (at least for that episode) that the rock is in their neighborhood and tigers are not, there is no support for the implication that the former implies the latter. When an argument is based upon an incorrect or unsupported inference, the argument is a *fallacy*.

logical fallacy

Definition 21: Logical Fallacy

A *logical fallacy* is an argument built with an incorrect inference.

informal fallacy

(Another name for logical fallacy is *propositional fallacy*.)

Be aware that there are also *informal fallacies*, those that demonstrate less specific errors of reasoning. We mention a few famous examples of such 'illogical' fallacies in section 3.5.2. First, let's cover some logical fallacies.

3.5.1 Logical Fallacies

Recall that all rules of inference are based on tautologies – when the hypotheses are true, the conclusion must also be true. Fallacies are often based on implications that mimic actual rules of inference, suggesting to the reader that the fallacy is also logically supported.

The following examples present three common logical fallacies.

Affirming the Conclusion

Example 72:

Problem: Does the following argument use a valid rule of inference?

Whenever it is sunny, Rodrigo wears sunglasses. Today he's wearing a pair, therefore it must be sunny.

Solution: No, this argument doesn't use a valid rule of inference – but the example does seem to make sense, because why else would anyone wear sunglasses? Actually, there are multiple reasons: The clouds are thin, it's windy, Rodrigo has an unsightly stye, the future's so bright, etc.¹⁴ Any of those can explain the wearing of sunglasses on a less-than-sunny day.

The fallacious argument of Example 72 can be expressed in logic notation like this:

$$\begin{array}{l} (1) \quad s \rightarrow g \\ (2) \quad g \\ \hline (3) \quad \therefore s \end{array}$$

In this form, it looks a lot like modus ponens, but of course it isn't. Because this fallacy claims the truth of the hypothesis (s) by assuming the truth of the conclusion (g), it is known by the name *Affirming the Conclusion* (a.k.a. *Affirming the Consequent*).

But why isn't this a valid argument? We are assuming that g is true, and that $s \rightarrow g \equiv s \rightarrow \mathbf{T}$ is also true. The problem is that the truth of $s \rightarrow \mathbf{T}$ doesn't mean that s must be true. In fact, $s \rightarrow \mathbf{T} \equiv \mathbf{T}$ regardless of the value of s ($\mathbf{T} \rightarrow \mathbf{T} \equiv \mathbf{F} \rightarrow \mathbf{T} \equiv \mathbf{T}$, by the definition of implication). We cannot conclude anything about the truth of s from the truth of those two hypotheses.

Another way to see that Affirming the Conclusion is a fallacy is by remembering that rules of inference are based on tautologies (see subsection 3.3.1). To show that $((s \rightarrow g) \wedge g) \rightarrow s$ is not a tautology, we need to find a circumstance in which conjunction of the hypotheses $((s \rightarrow g) \wedge g)$ is true but the conclusion (s) is false.

¹⁴Greetings from Timbuk3!

*affirming the
conclusion*

You could build a truth table to do this, but instead we'll just reason it through. For the logical AND to be true, both operands have to be true. That means g must be true and $s \rightarrow g$ ($\equiv s \rightarrow \mathbf{T}$) must also be true, which it is when s is false. Thus, we've found a row in the truth table for which $((s \rightarrow g) \wedge g) \rightarrow s \equiv \mathbf{T} \rightarrow \mathbf{F} \equiv \mathbf{F}$, demonstrating that $((s \rightarrow g) \wedge g) \rightarrow s$ is not a tautology, and verifying that Affirming the Conclusion is a fallacy rather than a rule of inference.

Denying the Hypothesis

Remember those word analogy questions from standardized tests of vocabulary? They are often phrased like this: "This discrete math book is to fine literature as expired vinegar is to (a) overly-sweetened soft drinks, (b) discount floor wax, (c) the purest spring water, or (d) a bottle of 1787 Chateau Margaux?"¹⁵ That question type is applicable here. We just saw that Affirming the Conclusion is much like modus ponens. Similarly, the fallacy known as *Denying the Hypothesis* is much like modus tollens.

denying the hypothesis

Example 73:

Problem: Does the following argument use a valid rule of inference?

You don't have a deck of cards. If you had one, you'd play poker.
So, no poker game for you.

Solution: Give yourself two demerits if you answered 'yes'. Sounds pretty good, though, doesn't it? Here's the argument in logical notation:

$$\begin{array}{ll} (1) & \neg d \\ (2) & d \rightarrow p \\ \hline (3) & \therefore \neg p \end{array}$$

It's close to modus tollens, but not close enough. If you apply the Law of Contraposition to $d \rightarrow p$, you'll get $\neg p \rightarrow \neg d$, giving the argument exactly the same form as Affirming the Conclusion, and we've just shown that it isn't a valid argument.

¹⁵A bottle of which, insured for \$225,000, was accidently broken by its owner while he was showing it off to guests at a 1989 dinner in New York. Oopsie!

Affirming a Disjunct

Our third logical fallacy, *Affirming a Disjunct*, is, as the name suggests, based on a disjunction rather than upon an implication. It confuses people because it is similar to disjunctive syllogism.

*affirming a
disjunct*

Example 74:

Problem: Does the following argument use a valid rule of inference?

To graduate with a computer science degree from Wossamotta U.,¹⁶ you need to successfully complete either Compilers or Operating Systems. You complete Compilers. Therefore, you did not complete Operating Systems.

Solution: Shockingly, the answer is still ‘no.’ Here’s the form:

$$\begin{array}{l} (1) \quad c \vee o \\ (2) \quad c \\ \hline (3) \quad \therefore \neg o \end{array}$$

The problem with this one is in the implied assumption that you must choose exactly one of c or o to make $c \vee o$ true, when we know that there’s a third way to make inclusive-OR true: $\mathbf{T} \vee \mathbf{T}$. That is, o could be true or false; we cannot claim that it must be false.

For thought: Can you turn this fallacy into a rule of inference by replacing the inclusive-OR with an exclusive-OR?

3.5.2 Informal Fallacies

When an argument has an error of reason other than a poor logical inference, it is termed an *informal fallacy*. There are many of these; we offer two famous examples.

informal fallacy

Begging the Question

You may have heard someone use a form of this phrase in conversation. For example: “The existence of seedless watermelons begs the question: How

¹⁶Famous for having offered a football scholarship to B. J. Moose.

do they reproduce?” That’s a literal interpretation of the phrase “begs the question,” in which the speaker is imploring you to ask the suggested question.

As a fallacy, the phrase has a much different meaning: The argument’s conclusion is assumed true rather than shown to be true.

Example 75:

Problem: What’s wrong with the reasoning used in this argument?

Ladies and gentlemen of the jury: Let’s assume that Mr. Naidoo ate his mid-day meal of bunny chow¹⁷ on the day in question. Let us further assume that he paid in cash, as he claims, and left a generous tip. By his own admission, he was so sated that he didn’t eat again until the next morning. Ladies and gentlemen, you must agree: Mr. Naidoo ate that lunch!

Solution: You can almost believe that this was a portion of a lawyer’s concluding remarks to a jury, but even though it sounds good, you shouldn’t be convinced of the truth of the conclusion. The problem is that the jury is asked to assume that the man ate lunch, and then to conclude that the man ate lunch. That is, the argument has this form: $\dots, p, \dots / \therefore p$.

Logically, this works – having assumed p to be true, we can correctly conclude that p is true under that assumption. But, practically, such an argument is a waste of our time.

circular reasoning

Begging the Question is also known as *circular reasoning*, which makes sense given that we end up exactly where we started – our conclusion is one of our hypotheses.

An interesting property of this argument form is that, like rules of inference, it *is* based on a tautology (specifically, $p \rightarrow p$). Thus, in terms of its structure, it’s a valid argument, just not one that justifies any new truths. That’s why Begging the Question is considered to be an informal rather than formal fallacy.

You may be wondering: Why is Simplification considered to be a rule of inference when it sure as heck looks like Begging the Question in a stylish Halloween costume? Yes, we could rewrite Simplification ($p \wedge q / \therefore p$) as

¹⁷A common South African dish. For humans. Ever have a stew served in a “bread bowl?” Same thing, really.

$p, q / \therefore p$, because the notation p, q represents $p \wedge q$. However, the purpose of Simplification is to justify the truth of a single operand of the compound proposition $p \wedge q$ after we have discovered (or have been told) that $p \wedge q$ is true. If we are told (or have discovered ourselves) that p is true, we don't need any other justification for accepting p 's truth.

No True Scotsman

This informal fallacy can be viewed as a combination of Begging the Question and another, known as *equivocation*, in which the argument is based on an expression that has multiple meanings or interpretations. Because of these origins, the No True Scotsman fallacy isn't a distinct fallacy, but that doesn't lessen its popularity, or make it less worthy of consideration.

equivocation

Philosopher Anthony Flew is credited with the naming of this fallacy. His example involves sex,¹⁸ so to remain G-rated we will present a less-offensive version.

Example 76:

Problem: What's wrong with the reasoning used in this argument?

Forbes was taken aback. Sitting in the local coffee house with his wife, Isla, he had just watched a neighbor pour sugar in his morning coffee. "No Scotsman puts *sugar* in his coffee," Forbes muttered. Isla gave him a familiar look. "Forbes, love, your cousin Angus also takes his coffee with sugar." Now Forbes was appalled. "I'm telling you, no **TRUE** Scotsman puts sugar in his coffee!"

Solution: To see this as a form of Begging the Question, notice that Forbes is assuming (in the face of mounting evidence to the contrary) that no Scotsman puts sugar in coffee, yet concludes the same thing, albeit with an insignificant additional word and an increase in volume. It's also equivocal, in that Forbes intends "Scotsman" and "true Scotsman" to have different meanings in an attempt to salvage his point.

¹⁸Simmer down! It has the word 'sex' in it; that's as salacious as it gets.

Chapter 4

Direct Proofs

At last, we are ready to begin learning about writing formal proofs. Just as the previous chapters' material built upon that of the chapters that preceded them, so will this chapter's material build upon that of its predecessors. If you are jumping directly to this chapter and find that some ideas are difficult to follow, please consider returning after you read about logic, quantified expressions, and arguments. A solid foundation in those topics will serve you well as you learn to write proofs.

This chapter focuses on direct proofs, the most common variety. It also explains proof by cases, a technique that is often used in conjunction with direct proofs, but which is also useful with other proof techniques. At the end, we introduce some approaches for showing that a conjecture cannot be proven. If you are looking for indirect (a.k.a. 'contra') proof techniques, see the next chapter . . . but read this one first!

4.1 The Heartbreak of Probarephobia¹

We have a confession to make. Remember truth tables? Sequences of logical equivalences? Valid arguments using rules of inference? Those are all proofs. You've been soaking your brain in proofs for a few chapters already.²

¹A 1960s ad campaign for Tegrin skin care products introduced the phrase "Heartbreak of Psoriasis" to attract the attention of people with dry, itchy skin.

²A 1981 commercial for Palmolive dish-washing liquid featured a woman soaking her fingers in a small bowl of it, in preparation for a manicure, as an illustration of how gentle Palmolive is on your hands. The memorable phrase from Madge the Manicurist was, "You're soaking in it!" We thought about titling this chapter "Direct Proofs: You've Been Soaking in Them!"

There are two reasons we didn't call them proofs. The first is that the kinds of arguments that most people think of as 'proofs' are less constrained, usually more challenging to construct, and often harder to follow than are the examples we've seen so far. The first two of those three justifications come with the territory. You know more now than you did at the start of Chapter 1, and that knowledge allows us to start working with more challenging hypotheses and conclusions.

As for being harder to follow, that's a problem that proof-writers can address by writing their proofs to be readable by a knowledgeable audience. We will try to follow that advice, which means that we will no longer justify every step of every argument. This is necessary to keep the lengths of our proofs manageable, and is standard practice. The content of the proofs will contain justifications for only the less-obvious steps. We will rely on your knowledge of logic, arguments, and the rest of your education to allow you to fill in most of the remaining details.

The second reason for avoiding the word 'proof' until now: Fear. College-bound students are advised to complete a reasonably well-defined sequence of math classes during high school (or the equivalent), but too frequently don't receive a significant introduction to proofs along the way. There are many reasons for this omission, but the consequence is what matters. Students often end up believing that a proof is a mysterious and impossibly difficult construct that isn't important: "If learning to read and write proofs were useful skills, someone would have taught them to us by now!" You're in luck: 'Someone' has already started, and in this chapter 'someone' will continue the job.

In an effort to show that proofs aren't scary (and to entertain the world), writer Ken Keeler of the American animated television show *Futurama* penned an episode ("The Prisoner of Benda") that included the full text of a proof (Figure 4.1) showing that the effects of Professor Farnsworth's mind-switching machine could be reversed.³ Keeler, who earned a Ph.D. degree in applied mathematics, wrote the proof specifically for this episode. If a proof can star in a cartoon, you know they don't need to be frightening.

As far as we can tell, no one has bothered to create a word for the fear of proofs. We modestly suggest 'probarephobia,' from the Latin *probare* (to show to be true, to demonstrate) and the Greek *phobos* (fear). There are words for related fears – for example, 'arithmophobia' is the fear of numbers,

probarephobia

³The full text of the proof is available online; one source is http://theinfosphere.org/Futurama_theorem

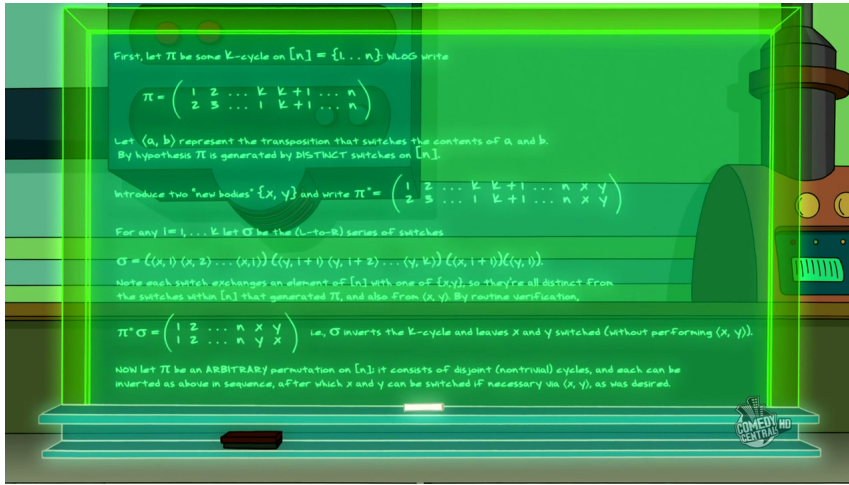


Figure 4.1: Futurama writer Ken Keeler’s mind restoration proof from the episode “The Prisoner of Benda.” Credit: The Curiosity Company.

and ‘phronemophobia’ is the fear of thinking.⁴ – but until now none for a fear of the content of this chapter⁵

4.2 Proof Preliminaries

Before we write any of these challengingly less-constrained proofs, we have some terminology to introduce and some advice to impart.

4.2.1 Terminology

Proofs come with a handful of associated terms. If you’re like most people, you’ve heard of them, but you’re not quite sure how they differ. We start by distinguishing *conjecture* from *theorem*.

Definition 22: Conjecture

conjecture

⁴Some phobia names contain words related to proofs, but describe very different fears. ‘Theologicophobia’ is the fear of theology, while ‘amathophobia’ is the fear of . . . dust.

⁵Suggested t-shirt and bumper-sticker slogan: Got Probarephobia? Relax; It’s a Given.

A statement with an unknown truth value is a *conjecture*.

theorem

Definition 23: Theorem

A *theorem* is a conjecture whose truth has been demonstrated.

Example 77:

The existence of intelligent life elsewhere in the universe is a conjecture.⁶ When an extraterrestrial visitor arrives to ask us to stop littering this corner of the galaxy with our space probes and radio waves, it will become a theorem.

disproven conjecture

In this book, we will label as conjectures all statements we are about to try to prove. As soon as we can successfully prove one, from that point forward we'll call it a theorem. If we can disprove it, it's a *disproven conjecture*. If we can do neither, it remains merely a conjecture.

Now we can define 'proof.'

proof

Definition 24: Proof

A sound argument that demonstrates the truth of a theorem is a *proof*.

Notice that we are insisting on a sound argument, not just a valid one. We can't claim that we've demonstrated that a conjecture is true by beginning with information we're just assuming to be true – we need to start with truth to establish truth.

Our last two definitions are other names for theorems. They are used to describe theorems that are created as parts of, or extensions of, other theorems. We will provide examples of these in section 4.3.

⁶And has only tenuously been accepted to exist here.

Definition 25: Lemma

A *lemma* is a theorem created to aid in the construction of another theorem.

lemma

Sometimes, while writing a proof, you reach a point where you say to yourself, “Self, I could easily finish this proof if I knew that this statement were true.” You start trying to prove that it is, and if you succeed, you will have created a new theorem – a lemma – that you can use to justify a step in your original proof. You can think of a lemma as a ‘sub-theorem,’ much as you might create a subprogram (e.g., a method or a function) in a programming language for the main program to invoke.

Mathematicians sometimes reserve ‘theorem’ to describe important results and use ‘lemma’ for less significant results. In this book, we’ll be calling every proven conjecture a theorem, reserving ‘lemma’ for situations matching our definition.

Definition 26: Corollary

A *corollary* is a theorem whose truth follows almost immediately from the truth of another theorem.

corollary

The difference between a lemma and a corollary, then, is that lemmas appear in the middle of proofs of other theorems, while corollaries appear immediately after such proofs. They are both usually smaller results (and usually easier to prove) than their associated theorems.

4.2.2 Proof-Writing Advice

Before we cover any proof techniques, we want to try to put you in an appropriate frame of mind. As we’ve mentioned, many people have a deep-seated fear of proofs. A common manifestation of this fear is a tendency to ‘lock up’ – to have no idea what to do, or even what to try, to move the argument forward. This is similar to “writer’s block,” which occurs when an author can’t decide what to write next. When you get stuck while writing a proof and aren’t sure of the next step, try to remember these pieces of advice.

1. **Not all conjectures can be proven.** Always ask yourself: “Does this conjecture seem to be true?” If you have doubts, start by trying to disprove the conjecture – that is, try to show that the conjecture is false. Techniques for disproof are covered in section 4.4.
2. **Dead-ends are expected.** When writing a proof, it’s very common to reason your way to a conclusion that doesn’t seem useful for completing the proof. Maybe it isn’t useful, or maybe you’re just not seeing how it can be useful. If you feel yourself getting frustrated, take a break. If you can, take a walk. During an exam, work on another question and come back to the proof later.
3. **Choose the right tool for the job.** We cover the direct proof technique in this chapter, two more techniques in the next chapter, and a fourth several chapters later. If the technique you chose initially doesn’t seem to be a good fit for the conjecture, try another.
4. **“Mind what you have learned. Save you it can.”**⁷ To Yoda you listen! Any of your education in mathematics, including what you’ve covered in this book, can be useful in proof-writing. Keep your mind open.
5. **There are multiple ways to say the same thing.** Some of those ways can be more useful than others. For example, we know that $10 \mid x$ is true when x is a multiple of 10. But also, $x \% 10 = 0$ and the decimal representation of x ends in a zero when x is a multiple of 10.
6. **Practice!** An unfortunate consequence of standardized mathematics curricula and standardized assessment examinations in grade school is that students learn to expect all math problems on a given topic to look the same and to be solved the same way. When they get to college, they are often unprepared to be creative and to embrace flexibility, two characteristics that proof-writing requires. The more conjectures you examine, and the more proofs you write, the more prepared you will be.

4.2.3 Proofs of $p \rightarrow q$

Conjectures are written, or can be re-written, as implications. We have some given information (the hypothesis or hypotheses) and the conclusion we hope

⁷Star Wars: Episode V – The Empire Strikes Back, Lucasfilm, 1980. (1980!?!)

to justify. We are also told the circumstances under which the conclusion needs to be true, though sometimes we need to infer them ourselves.

Because we are proving conjectures expressible as implications, it is common to refer to them using the notation $p \rightarrow q$. Often, the conjecture needs to be shown to hold over all members of a domain, such as the positive integers. In such “for all” cases, we can use $\forall x(P(x) \rightarrow Q(x)), x \in D$ instead of $p \rightarrow q$, but as the idea is the same with either notation, most of the time we will stick with the simpler $p \rightarrow q$ when we need to use notation.

Methods of Proof

To correctly reason from the hypotheses to the conclusion, we need acceptable proof techniques that can show the truth of conditional statements. Having constructed proofs in the earlier chapters, there must have been at least one underlying technique, and there was:

- (a) *Direct proof*. Outside of textbooks, it’s safe to say that most proofs are direct proofs. All of the logical equivalence and rule of inference arguments we examined in the previous chapters are direct proofs. We will revisit those, and cover additional structurings of direct proofs, in section 4.3. *direct proof*

The direct proof method is also the foundation for three related proof techniques, the first two of which are covered in the next chapter:

- (b) *Proof by Contraposition*. Also known as *proof of the contrapositive*, this technique is a very slight variation of direct proof. It will be covered in Section 5.1. *proof by contraposition*
- (c) *Proof by Contradiction* sacrifices the definite conclusion of direct and contraposition proofs to gain a compound hypothesis (a sacrifice which is more useful than it may sound). This method is covered in Section 5.2. *proof by contradiction*
- (d) *Proof by Induction*, which, despite the name, has a satisfying, comfort-food core of deduction. Proof by induction is different enough to warrant a chapter of its own. *proof by induction*

Many other proof techniques exist. These can all be considered to be forms of direct proof, but they have separate names to help us keep the variations straight. We won’t have a special section for these; instead, we will demonstrate them as we have need of them:

- vacuous proof* (e) *Vacuous Proofs* are based on the idea of vacuous truth we defined in Chapter 1. They often appear when we have a definition of a property in the form of an implication and try to apply the definition to a situation in which the antecedent cannot be applied.
- trivial proof* (f) *Trivial Proofs* are those in which the implication is true because the consequent is true; that is, the truth of the antecedent is irrelevant. For example: *If I am both Batman and Superman, then 2 is even.*
- proof by cases* (g) *Proof by Cases* (also known as *proof by exhaustion*⁸) applies when we can show that the conjecture is true for every member of the domain by testing each member individually or by testing partitions of the domain. Proofs by cases often appear within other proof techniques, as we will see.
- constructive proof* (h) *Constructive Proofs* identify a member of the domain that makes an existential conjecture (that is, one that merely claims a satisfying member exists) true. A related technique is the *non-constructive proof*, which manages to demonstrate that an existential conjecture is true without ever identifying a specific example.
- non-constructive proof*
- combinatorial proof* (i) *Combinatorial proofs* show the number of ways to count an activity, such as finding the number of subsets of size n of a set of size m . We won't see one of these proofs until the counting chapter.⁹

Additional proof methods exist, but this collection is enough for a discrete mathematics book.

Content of a Complete Proof (As Far As We're Concerned)

Everyone has their own favorite way to write a proof, and there is no single set of proof-writing rules that pleases everyone. Here are the principles to which we will (usually!) adhere for the proofs in this chapter, and in the rest of the book. We reserve the flexibility to ignore them when we need to make a point.

1. Start by writing “Proof (*proof technique*):” at the top of the proof. For example: “Proof (Direct):”. This lets the reader know how you have

⁸Magically, at 3 a.m. on a homework due date, every proof technique temporarily shares this name.

⁹You're welcome!

structured your argument, allowing him or her to more easily understand it.

2. State your hypotheses and any other given information. We don't consider this to be a necessity, but is especially helpful (both to you and to the reader) when using a technique other than direct proof. It's also a good way to stave off, if not eliminate, writer's block.
3. Make your proofs comprehensible. You've almost certainly had the experience of trying to read a proof in a textbook, only to mutter, "Wait . . . where'd *that* come from?!?" It's possible that you didn't have the knowledge or experience to immediately understand the step, but it's also possible that the author abbreviated the proof for publication and lost too much detail in the process. Proofs should be clear, not cloudy.
4. Justify the less-obvious steps of your argument. Telling the reader that $x + y = y + x$ by commutativity of addition will help few readers, but mentioning that $e^{i\pi} + 1 = 0$ is Euler's Identity will help most anyone who is not a math major.
5. "Declare" your variables. Ever written a program in a language that requires the programmer to declare variables before they are used in statements? One reason for that requirement is to help the language translator correctly convert your statements to a lower-level representation. Similarly, telling the reader of your proof that a variable's value is drawn from a particular domain helps the reader understand your argument.
6. When we wrote our logical equivalence and rules of inference arguments, we used columnar representations. If your proof's clarity would be enhanced by using such a representation, please use it. Proof-writers often prefer to have their proofs look as much like traditional paragraphs of text as possible, but we've already seen that the extra hassle of formatting a few columns can be well worth the trouble.
7. To show that your proof is complete, finish it by writing the word "therefore", a comma, and the original conjecture. We think is this a better way for beginning proof-writers to end a proof than are the traditional endings of "Q.E.D."¹⁰ or a 'tombstone' marker (such as ' \square ' (`\(\square\)` in `LATEX` :

¹⁰Q, E, and D are the first letters of the words in the Latin phrase *quod erat demonstrandum*, which means "this was to be demonstrated." "Quite easily done" is a popular sarcastic alternative.

\Box)), because restating the conjecture helps you remember what you were trying to prove. When you are deep in the middle of an argument, it's easy to forget what you were arguing about in the first place.

4.3 Direct Proofs

As the name suggests, a direct proof is quite straight-forward. To prove a conjecture of the form $p \rightarrow q$, we assume that the (often compound) hypothesis (p) is true, and demonstrate that the conclusion (q) must also be true. In short:

To prove $p \rightarrow q$: Assume p , show q .

4.3.1 Examples of Direct Proofs

Our first direct proof example is very straight-forward, so that we can focus on the construction of the proof rather than on the difficulty of the argument. We will ease into every new proof technique in the same way.

Example 78:

Problem: Using a direct proof, prove this conjecture: If x and y are odd integers, the product xy is also an odd integer.

Solution: We are told to use a direct proof. This means that we are to assume that the hypothesis is true and show that the truth of the conclusion must follow. Our first task, then, is to identify the hypothesis and the conclusion. In this example, finding them is trivial, because the conjecture is expressed in an if-then form: Our hypothesis is “ x and y are odd integers,” and our conclusion is “the product xy is an odd integer.” The word ‘also’ doesn’t change the conclusion; it can be dropped.

We know from the math review appendix (Section A.8). that each odd integer is one more than twice some integer. That is, we can say $x = 2a + 1$ and $y = 2b + 1$. Using this form of expression for odds requires the introduction of the new integer variables a and b . In the proof, we’ll have to remember to explicitly declare them to be integers.

Now that we have representations for x and y , we can multiply those polynomials to create a representation for xy : $xy = (2a + 1)(2b + 1) =$

$$4ab + 2a + 2b + 1.$$

Having created such an expression, you might be tempted to say, “OK, but ...so what?” Remember what we are trying to show: That xy is odd. We know that odds have the form $2z + 1$. By factoring a 2 from the first three terms of our xy representation, we can achieve that form: $4ab + 2a + 2b + 1 = 2(2ab + a + b) + 1$. Products and sums of integers are also integers, and so xy is an odd integer because it has the $2z+1$ form.

We could prepend “Proof (Direct):” and append “Therefore, ...” lines to the preceding paragraphs and call it a proof. Such a proof would be unnecessarily wordy, even for us. Instead, we’ll condense it into a more compact, though not minimal, version. Here’s the result:

Proof (Direct): We may assume that x and y are odd integers. Because they are odd, x and y are each one more than twice some integer. Let those integers be a and b , and let $x = 2a + 1$ and $y = 2b + 1$. Using those representations, $xy = (2a + 1)(2b + 1) = 4ab + 2a + 2b + 1 = 2(2ab + a + b) + 1$. This expression shows that xy can be represented as one more than twice an integer, demonstrating that xy is an odd integer.

Therefore, if x and y are odd integers, the product xy is also an odd integer.

This proof includes a bit more detail we will use in most proofs in this book – in general, we won’t explain everything, but we will explain the key steps. The issue of detail is also addressed in Example 79, below.

The proof in Example 78 needed two new variables to create the representations of x and y , not just one. If we tried to use a single variable, we’d say that $x = 2c + 1$ and $y = 2c + 1$. By transitivity, this means that $x = y$. The conjecture does not require that x and y be the *same* odd integer, meaning that the proof’s representations of x and y can’t require it, either.

Example 78 demonstrated the level of detail we will use in most proofs in this book. The next example addresses a related concern of students.

Example 79:

Problem: My instructor's exams are really long;¹¹ I don't have time to write proofs with that much detail! Can I get away with writing just the math?

Solution: Probably not, but of course that depends on your instructor. Here's a shortened version of the proof given in Example 78 that we feel has an acceptable level of detail for a proof on an exam:

Proof (Direct): Let $x = 2a + 1$ and $y = 2b + 1$, $x, y \in \mathbb{Z}^{\text{odd}}$.
 $xy = (2a + 1)(2b + 1) = 4ab + 2a + 2b + 1 = 2(2ab + a + b) + 1$.
 This shows that xy is an odd integer.

Therefore, if x and y are odd integers, the product xy is also an odd integer.

This form is still acceptable because it shows a logical progression from the hypothesis to the conclusion, and includes enough detail for the reader to accept that the conclusion follows from the hypothesis, although the reader is expected to fill in some minor details mentally.

Note that extending the algebra through to the $2(2ab + a + b) + 1$ expression is important. If you stop with $4ab + 2a + 2b + 1$, you're not showing that xy has the form of an odd-integer, leaving the reader to wonder if you really know how to complete the proof. Most instructors won't give you the benefit of the doubt on something like this. Be safe; be sure to include enough detail to make it clear that you know what you're doing. The points you save will be your own.

Our next direct proof example requires a little more imagination. It's also rather lengthy, to make a point about how proofs are often constructed.

¹¹The author has never heard such a comment from his students. Never happened. Not even once. In unrelated news, rumors of the author's selective hearing and recollection are undoubtedly wild exaggerations.

Example 80:

Problem: Using a direct proof, prove this conjecture: $ac > bd$ when $a > b > 0$, $c \geq d > 0$, where $a, b, c, d \in \mathbb{R}$.

Solution: The first step is to identify the hypothesis and the conclusion. This conjecture uses a “ q if p ” form, meaning that $ac > bd$ is the conclusion and the rest forms the compound hypothesis.

Next, we need to make sure we understand what the notation is trying to tell us. $a > b > 0$ contains three individual pieces of information: $a > b$, $b > 0$, and, by transitivity of $>$, $a > 0$. Similarly, we are given that $c \geq d$, $d > 0$, and $c > 0$. Pay close attention to the inequalities; all of them are $>$ except for the \geq in $c \geq d$.

We’re ready to think about the proof. We need some way to introduce the products ac and bd . It’s tempting to say, “Well, just multiply $a > b$ by $c > d$, and we’re done!” Unfortunately, that isn’t legal; inequalities are boolean expressions, and multiplication doesn’t apply to boolean operands. We’ll have to create ac and bd another way.

Let’s start with $a > b$. We can try to create ac by multiplying the left side by c , producing $ac > bc$. But is that true? It may not be, as c could be less than one, which could make ac less than or even equal to b . But we’re on the right track: We need to multiply both a and b by c to produce $ac > bc$. This is true because we’re multiplying both sides by the same amount, and the direction of the inequality stays the same because $c > 0$.

We can create bd in the same way: Multiplying both sides of $a > b$ by d produces $ad > bd$. Now we have to figure out how to combine $ac > bc$ and $ad > bd$. We can do this with transitivity of $>$ if we knew that $bc > ad$. . . but we don’t. We could try to prove that it is true and introduce that result as a lemma in this proof, but before we go to that trouble, let’s stop and think: Can we create bd from something other than $a > b$?

Happily, we can. We know that $c \geq d$. Multiplying both sides of it by b produces $bc \geq bd$. We can use the properties of inequalities given in the math review appendix (Appendix A) to combine $ac > bc$ with $bc \geq bd$

and show that the conclusion of $ac > bd$ is true.

Having discovered how to show the truth of the conclusion, we can write up the proof as a tightened version of the above discussion.

Proof (Direct): We are given that $a > b$, and that $c > 0$. Multiplying both sides of $a > b$ by c produces $ac > bc$.

We are also given that $c \geq d$ and $b > 0$. Multiplying both sides of $c \geq d$ by b produces $bc \geq bd$.

By the properties of inequalities, together $ac > bc$ and $bc \geq bd$ tell us that $ac > bd$.

Therefore, $ac > bd$ when $a > b > 0$, $c \geq d > 0$, where $a, b, c, d \in \mathbb{R}$.

Example 80 shows how easily a proof writer can encounter a dead-end in his or her reasoning. Textbook and research paper authors rarely reveal their dead-ends in their final proofs, making it seem that they created the proofs correctly on the first try. As this example has shown, even a short proof can be the result of much ‘wasted’ work.

Our next example will give us a reason to introduce a lemma. Before that, it will require us to do some reasoning about the meaning of the conjecture.

Example 81:

Problem: Prove that the \sqrt{x} being even is sufficient to show the necessity of x ’s evenness, given that $x \in \mathbb{R}$.

Solution: Are you tempted to say, “Yeah, I can solve this problem by finding the guy who wrote that and making him put it into English!”? We sympathize. But, Chapter 1 saves us from committing a satisfying but unnecessary act of violence: We know what ‘sufficient’ and ‘neces-

sary' mean when they are used correctly. Here, we are given that \sqrt{x} is even, and we need to show that x is even.

What does ' \sqrt{x} is even' give us? We know that only integers can be even, and that the only way for a square root to be an integer is for the argument to be a perfect square.

Let's make this easier to think about by letting $\sqrt{x} = y$. Because we are allowed to assume that \sqrt{x} is even, we can assume that y is even. Now let's consider x . $x = \sqrt{x} \cdot \sqrt{x} = y \cdot y$. If we can show that $y \cdot y$ is even, we will have shown that x is even, and our proof will be complete.

Showing that $y \cdot y$ is even requires a separate proof. Because we will be using the theorem that results from this second proof to complete our original proof, the evenness of $y \cdot y$ will be a lemma to our main proof.

That's all fine, but let's not get ahead of ourselves: Is $y \cdot y$ always even? We are given that y is even, meaning we can represent it as twice some integer (say, twice w). $y \cdot y = 2w \cdot 2w = 4w^2 = 2(2w^2)$. Because we can express $y \cdot y$ as twice an integer, $y \cdot y$ is even. And, because $x = y \cdot y$, x is even, completing the original proof.

Time to write it all up. We will start by proving the lemma (let's call it Lemma 81), but we will toss in a twist to the conjecture to keep it interesting.

Conjecture: The square of any even number is also even.

Proof (Direct): Let y represent any even number. Being even, $y = 2w$, $w \in \mathbb{Z}$. $y^2 = (2w)^2 = 4w^2 = 2(2w^2)$. This shows that y^2 is an even number.

Therefore, the square of any even number is also even.

The 'twist' in the above conjecture is that it isn't in an 'if-then' form. We had to re-write it that way, at least mentally, to know the hypothesis and conclusion: *If y is an even number, then y^2 is also even.* English-to-logic

conversions just don't go away.

There's something else to notice in that proof: We didn't restrict w to being a non-negative integer. First, the conjecture isn't limited, meaning that our proof shouldn't be limited. Second, negative integers are either odd or even, just as the positives (and zero) are. It requires no extra effort to prove the conjecture for all evens, so why not?

At last, we can prove the original conjecture, which was: The \sqrt{x} being even is sufficient to show the necessity of x 's evenness, given that $x \in \mathbb{R}$.

Proof (Direct): We are given that \sqrt{x} is even. It follows from the definition of evenness that \sqrt{x} is an integer. Let $y = \sqrt{x}$, $y \in \mathbb{Z}$. By the definition of square root, the square of the square root of a real number is the same real number. Thus, $x = (\sqrt{x})^2 = y^2$.

Lemma 81 shows that the square of any even number is also even. By that lemma and the fact that y is even, we know that y^2 , and thus x , is even, completing the argument.

Therefore, the \sqrt{x} being even is sufficient to show the necessity of x 's evenness, given that $x \in \mathbb{R}$.

If the use of the variable y in both proofs of Example 81 confuses you, feel free to use a completely different set of variables in each proof. We re-used y because we also re-used it in the explanation at the top, to highlight the connection between the original proof and the lemma.

For our final example, we will tie off a loose end. Earlier (in section 4.1) we said that sequences of logical equivalences and rule-of-inference arguments were actually proofs. Let's see how we can re-structure one of those (specifically, Example 66 of Chapter 3) as a proof.

Example 82:

Problem: Assume that if you have a dog and you drop a piece of bread

on the floor, then you won't need to pick up the bread. Further assume that you do have a dog, and you did drop a piece of bread on the floor. Prove that you didn't need to pick up the piece of bread.

Solution: We've already created this argument, so all we have to do is write up as we would a proof.

Proof (Direct): By the rule of inference known as Conjunction, we know that the conjunction of "you have a dog" and "you drop a piece of bread on the floor" is true. By applying Modus Ponens to that conjunction and the given implication, we know that you didn't need to pick up the bread.

Therefore, you didn't need to pick up the piece of bread, assuming that if you have a dog and you drop a piece of bread on the floor, then you won't need to pick up the bread; assuming that you do have a dog; and assuming that you did drop a piece of bread on the floor.

The arguments in Chapter 3 required more given information than did our previous direct proof examples, making this proof, especially the 'therefore' line, seem quite wordy. Worse, the proof wasn't especially easy to follow, as the proof's author expected us to 'see' that the rules of inference were applied correctly. That was easy enough for Conjunction, but if you hadn't seen that argument before, would you have been confident that Modus Ponens was used correctly based only on this presentation? Imagine trying to read (or write!) an explanation of an application of a messier rule of inference, such as Resolution.

When writing out an argument in English hinders comprehension more than it helps, use an alternative. Writing arguments in columns, as we did in Chapter 3, is perfectly acceptable in a proof, too — remember, those arguments are also proofs. We can wrap our original columnar argument in our usual proof trappings and create a hybrid that will be easy to follow.

Proof (Direct): Let d be ‘you have a dog,’ b be ‘you dropped a piece of bread on the floor,’ and n be ‘you need to pick up that piece of bread.’

(1)	d	[Given]
(2)	b	[Given]
(3)	$d \wedge b$	[1, 2, Conjunction]
(4)	$(d \wedge b) \rightarrow \neg n$	[Given]
(5)	$\therefore \neg n$	[3, 4, Modus Ponens]

Therefore, you didn’t need to pick up the piece of bread, assuming that if you have a dog and you drop a piece of bread on the floor, then you won’t need to pick up the bread; assuming that you do have a dog; and assuming that you did drop a piece of bread on the floor.

Writing out the ‘therefore’ line seems even more redundant that it already is, but we did so to stick to our form, and to get the reader’s mind back to the English version of the conjecture.

4.3.2 Examples of Direct Proof by Cases

Not infrequently, a conjecture’s hypothesis provides so little useful information that it is convenient to create additional information, just so that you have a starting point for your argument. One way to do this is to partition the set upon which the conjecture is constructed, and show that the conjecture holds for each partition. The description of the partitioning can often be all the additional information the argument requires.

Example 83:

Problem: Prove that $n^3 + n + 2 \in \mathbb{Z}^{\text{even}}$, where $n \in \mathbb{Z}$.

Solution: This conjecture doesn’t give us much to work with – all we know is that n can be any integer. But what does that tell us that’s

helpful? We know that integers are real numbers. We know that integers are rational. We know that the product of two integers is also an integer. We know lots about integers, but nothing that seems useful for this conjecture.

Then again, maybe we do know something of value: We know that every integer is either even or odd. If we can show that $n^3 + n + 2$ is even when n is even *and* when n is odd, we will have shown that $n^3 + n + 2$ is even for all integers. Sounds like a job for ... proof by cases!

Proof (Direct): Any integer value is either even or odd. We will show that $n^3 + n + 2$ is even either way.

Case 1: Let n be even. $n = 2k, k \in \mathbb{Z}$. $n^3 + n + 2 = (2k)^3 + (2k) + 2 = 8k^3 + 2k + 2 = 2(4k^3 + k + 1)$. This demonstrates that $n^3 + n + 2$ is even when n is even.

Case 2: Let n be odd. $n = 2j + 1, j \in \mathbb{Z}$. $n^3 + n + 2 = (2j + 1)^3 + (2j + 1) + 2 = (8j^3 + 12j^2 + 6j + 1) + (2j + 1) + 2 = 8j^3 + 12j^2 + 8j + 4 = 2(4j^3 + 6j^2 + 4j + 2)$. This demonstrates that $n^3 + n + 2$ is even when n is odd.

Therefore, $n^3 + n + 2 \in \mathbb{Z}^{\text{even}}$, where $n \in \mathbb{Z}$.

To avoid confusion, we used a different temporary variable in each case. You can make a case¹² that using the same variable for each part is like using ‘i’ as a loop-control variable in two loops within the same computer program, but we think that keeping the temporary variables unique is a good idea in proofs.

In a proof by cases, each case is a mini-proof, basically a lemma. We could treat them as lemmas, proving them individually as we did with the lemma in Example 81. When the lemmas are as closely related as they are in a proof by cases, it’s common practice to prove them all within the original proof.

We left the label on the proof as “Direct” because, even though we proved the conjecture using more than one case, it’s still a direct proof: We assumed

¹²Sorry ...

p (n is an integer) and showed q ($n^3 + n + 2$ is even). If you like, you can embellish the label to be “Direct with Cases” or “Direct, Cases.” We will just say “Direct” because the reader will be able to see at a glance that such proofs include cases.

Partitioning is a useful tool for understanding large sets. Example 83 made use of the even/odd partitioning of integers. Another way to partition integers: $\{\mathbb{Z}^-, 0, \mathbb{Z}^+\}$.¹³ Real numbers can be partitioned into the rationals and the irrationals. Characters in the basic Latin alphabet can be partitioned by case (upper and lower).

Consider truth tables. Each row of a truth table represents a unique assignment of truth values to variables. That is, a truth table is a partitioning of the possible assignments in which each assignment is a partition of cardinality one. Looked at another way, a truth table is a proof by cases.

Example 84:

Problem: Prove that $(p \vee q) \vee \bar{p}$ is a tautology.

Solution: We used a truth table to demonstrate this in Chapter 1. We could re-write the truth table in English to have each case be its own short paragraph, or we could just wrap the trappings of a proof around the truth table.

¹³That’s right – zero is neither positive nor negative. Some numeric representations, such as the IEEE 754 floating-point standard, do distinguish between +0 and –0 because doing so is sometimes scientifically useful.

Proof (Direct): There are four possible assignments of boolean values to the variables p and q . The following table shows the evaluation of $(p \vee q) \vee \bar{p}$ on all four:

	p	q	$p \vee q$	\bar{p}	$(p \vee q) \vee \bar{p}$
<i>(Case 1:)</i>	T	T	T	F	T
<i>(Case 2:)</i>	T	F	T	F	T
<i>(Case 3:)</i>	F	T	T	T	T
<i>(Case 4:)</i>	F	F	F	T	T

No matter how true and false are assigned to the variables, the expression evaluates to true.

Therefore, $(p \vee q) \vee \bar{p}$ is a tautology.

We added the “*(Case 1:)*,” “*(Case 2:)*,” etc., just to highlight how cases are used in a truth table. In general, labeling the rows of a truth table in that fashion is not necessary.

As we’ve seen multiple times, most recently in Example 84, it’s entirely possible for a conjecture to have multiple variables. If we can partition the domain of one variable (as we did in Example 83), we can certainly do it for more than one. However, there is a cost: The number of cases that need to be considered increases, making it more likely that one of them is considered more than once, or worse, overlooked entirely.

Example 85:

Problem: Prove that if gh is odd, then g and h are both odd.

Solution: You should recognize the situation: We aren’t given a hypothesis with much useful information (and we only know how to write direct proofs), so we need to manufacture additional information.

This conjecture deals with odd numbers, making us think that partitioning g and h into evens and odds might be a good approach to try. Because we have two variables and two partitions for each, there are at

most four cases to consider: Both g and h are even, both are odd, g is even and h is odd, and g is odd and h is even. However, multiplication is commutative, meaning that the last two cases are logically identical – both represent the product of an even with an odd – leaving us with only three cases to consider. Of those, only the ones with gh being odd interest us, given the conjecture’s hypothesis.

Proof (Direct): Both g and h can be odd or even. We need to consider three cases: Both g and h are even, both are odd, or one is even and the other is odd.

Case 1: Let both g and h be even. $g = 2a$ and $h = 2b$, where $a, b \in \mathbb{Z}$. $gh = (2a)(2b) = 4ab = 2(2ab)$, showing that, in this case, gh is even. As the conjecture’s hypothesis is that gh is odd, this case is irrelevant to the proof and can be ignored.

Case 2: Let both g and h be odd. $g = 2c+1$ and $h = 2d+1$, where $c, d \in \mathbb{Z}$. $gh = (2c+1)(2d+1) = 4cd+2c+2d+1 = 2(2cd+c+d)+1$, showing that gh is odd and thus fits our conjecture. For this case, the conjecture holds (that is, when gh is odd, g and h are odd).

Case 3: One of the variables is even and the other is odd. WLOG, let g be even and h be odd. $g = 2e$ and $h = 2f+1$, where $e, f \in \mathbb{Z}$. $gh = (2e)(2f+1) = 4ef+2e = 2(2ef+e)$, showing that gh is even. As with Case 1, this case can be ignored.

Of the three possible ways to assign odd and even integers to g and h , only one fits the conjecture’s hypothesis. In that case, the conclusion is true.

Therefore, if gh is odd, then g and h are both odd.

‘WLOG’ is an acronym¹⁴ for the phrase “without loss of generality,” a common way to tell the reader that an arbitrary choice can be made without violating the logic of the argument. In this proof, one of the variables needed to be even and the other odd. It makes no difference to the argument which is which, so we picked one of the two possibilities

wlog

and used ‘WLOG’ to let the reader know that our specific choice didn’t matter to the proof.

The proof in Example 85 got the job done, but it seems wasteful. We had to consider three cases to discover that only one of them mattered. This observation might cause you to think, “Maybe there’s an easier way to prove this ...” There is, but we have yet to cover it. Don’t worry; we will, in Example 96 of Chapter 5.

4.3.3 Detecting Poor Proofs

Seeing nothing but correct proofs can leave the impression that all proofs are correct. Not true, of course. The history of mathematics includes many examples of proposed proofs that turned out to be invalid arguments when they were examined closely^{15 16}.

Here are three examples of poor proofs. Each demonstrates a common mistake made in proof development. Hopefully, having seen these, you will be less likely to make similar mistakes when writing your own proofs.

Example 86:

Problem: Prove that if $5n + 4$ is even, then n is even, $n \in \mathbb{Z}$.

‘Solution’: Consider this attempt at a direct proof:

‘Proof’ (Direct): Assume that $n \in \mathbb{Z}^{\text{even}}$. $n = 2k$, $k \in \mathbb{Z}$.
 $5n + 4 = 5(2k) + 4 = 10k + 4 = 2(5k + 2)$, which matches the given information that $5n + 4$ is even.

Therefore, if $5n + 4$ is even, then n is even.

¹⁴A point of pedantry: ‘WLOG’ is more accurately an *initialism*, but *acronym* is also acceptable. Be aware that, if you start using ‘initialism’ in everyday communication, you’ll never make it as a politician. You should probably avoid ‘pedantry,’ too.

¹⁵Wikipedia has a list: http://en.wikipedia.org/wiki/List_of_incomplete_proofs

¹⁶Wikipedia is conveniently ignoring the reality that approximately 99.995% of poor proofs were written by students on homework assignments or exams and discovered to be invalid by incredulous teachers who used to have full heads of thick, lustrous hair.

The reasoning behind this proof is perfect, except for one huge problem: **It starts by assuming that the conclusion is true!** Proofs exist to demonstrate the truth of conclusions, not to accept them as being true. The conjecture that this proof proves is the converse of the given conjecture. That is: *If n is even, then $5n + 4$ is even.* As we learned in Chapter 1, $p \rightarrow q \not\equiv q \rightarrow p$. The truth of the converse says nothing about the truth of the original.

The lesson: **Never assume that the conclusion is true.**

A poorly constructed proof doesn't tell us much about the conjecture. Unfortunately, even attempting a direct proof with the correct hypothesis can lead to a bad proof, as the next example shows.

Example 87:

Problem: Prove that if $5n + 4$ is even, then n is even, $n \in \mathbb{Z}$.

'Solution': Having learned the lesson of Example 86, we try again:

'Proof' (Direct): Assume that $5n + 4$ is even. $5n + 4 = 2k$, $k \in \mathbb{Z}$. Solving for n , we find that $n = \frac{2k-4}{5}$, which is ... ummmm, wow, not even always an integer, so it can't always be even ... can it?

Maybe the Deity of Partial Credit will smile upon me if I write ...

Therefore, if $5n + 4$ is even, then n is even.

Indeed, $\frac{2k-4}{5}$ isn't always an integer. We started the proof correctly, and still couldn't prove it. That means that the conjecture is false ... right? Not necessarily. The problem with this proof is that it doesn't make use of a key piece of given information: n is an integer. This conjecture is true; it can be proven with a direct proof that uses a little more creativity, such as was used in Example 85.

The lesson: **Heed the advice in Section 4.2.2.**

By the way: The Deity of Partial Credit might give you some partial credit for remembering to supply the proper conclusion to the proof, but that doesn't mean the proof itself will be worth much. The Deity of Partial Credit, kind as she/he/it may be, still has standards.

Our last example of a poor proof is a variation on a classic.

Example 88:

Problem: Lots of people have trouble writing their sevens and their twos clearly enough to make them distinct. This isn't really a problem at all, because, as the following proof shows, $7 = 2!$ Be as sloppy as you like!

'Proof' (Direct): Seven times two is fourteen, and let's make it negative to show negative numbers a little love.

$$\begin{array}{ll}
 -14 = -14 & \text{[Any integer equals itself]} \\
 49 - 63 = 4 - 18 & \text{[Strange expressions for } -14\text{]} \\
 49 - 63 + \frac{81}{4} = 4 - 18 + \frac{81}{4} & \text{[Add } \frac{81}{4} \text{ to each side]} \\
 (7 - \frac{9}{2})^2 = (2 - \frac{9}{2})^2 & \text{[Factoring (reverse FOIL)]} \\
 7 - \frac{9}{2} = 2 - \frac{9}{2} & \text{[Take square root of both sides]} \\
 7 = 2 & \text{[Add } \frac{9}{2} \text{ to both sides]}
 \end{array}$$

Therefore, $7 = 2$.

Does seven really equal two? Is the number line not to be believed? Is reasoned society doomed to collapse?

Solution: No, no, and probably.¹⁷ There's something wrong with this proof, the same problem that many other 'proofs' of impossible conclusions share: The square root step. In this 'proof,' it is claimed that $\sqrt{(2 - \frac{9}{2})^2} = 2 - \frac{9}{2}$. What most people forget about square roots is that $\sqrt{x^2} = \pm x = |x|$. That is, $x \cdot x = -x \cdot -x = x^2$. This proof quietly drops that plus-minus sign (\pm , \LaTeX : `\pm`), a hidden step that breaks the validity of the argument. If you were to finish the argument correctly, you'd get a very boring, but correct, conclusion:

$$\begin{aligned}
 (7 - \frac{9}{2})^2 &= (2 - \frac{9}{2})^2 && \text{[From the above 'proof']} \\
 |7 - \frac{9}{2}| &= |2 - \frac{9}{2}| && \text{[Correct square roots]} \\
 |\frac{5}{2}| &= |-\frac{5}{2}| \\
 \frac{5}{2} &= \frac{5}{2} && \text{[No surprise there]}
 \end{aligned}$$

The lesson: **Check each step of proposed proofs carefully.**

Square roots, logarithms, and division by zero are common sources of logical errors in ‘proofs.’ Whenever you see someone claim that they can prove that $0 = 1$, $1 = 2$, $1 = -1$, or the like, check the proof for a step involving one of those constructs. Chances are, that’s where you will find the logical error.

4.4 Disproving Conjectures

disproof

When a conjecture ‘smells’ false, or despite your best efforts you can’t create its proof, you should spend some time trying to demonstrate that it is false. Such demonstrations are known as *disproofs*. Disproofs do prove that conjectures are false, but you won’t often see a disproof written up as a formal proof.¹⁸

4.4.1 Find a Counter-Example

counter-example

By far the most common way to show that a conjecture is false is to find a *counter-example*, a specific assignment of legal values to variables that demonstrates that the conjecture is false. Sometimes a counter-example is easy to find (they are often found at the low and high ends of domains), but often a remarkable amount of searching is required.

Example 89:

Problem: Prove or disprove: 2^n is even for all $n \in \mathbb{Z}^*$.

¹⁷Hopefully, with the power of well-written proofs, we can delay that collapse for a while longer. Everyone listens to reason ... don’t they?

¹⁸Actually, you rarely see disproofs at all. Although studying failures can be very educational, who wants to become famous for creating a failure? Some disproven conjectures have lingered as cautionary examples. Wikipedia has a small collection: http://en.wikipedia.org/wiki/Category:Disproved_conjectures.

Solution: If you start by listing 2, 4, 8, 16, 32, etc., you will soon conclude that the conjecture must be true, because each number is double the previous, and the first is even. But look at the domain more carefully: \mathbb{Z}^* includes zero, and $2^0 = 1$, which certainly isn't even. Thus, the counter-example 2^0 shows that the conjecture is not true.

To write this as an answer on a homework or exam, something like this would be fine:

Disproof (Counter-Example): Let $n = 0$. $n \in \mathbb{Z}^*$, but $2^0 = 1$ is not even. The conjecture is not true.

That's right – no fancy argument, just a clear statement of an example from the desired domain that 'breaks' the conjecture.

Example 90:

Problem: Prove or disprove this conjecture:¹⁹ If $2^k \equiv 2 \pmod{k}$, then k is prime, $k \geq 2$, $k \in \mathbb{Z}$.

Solution: At first glance, this conjecture seems to be based on too many coincidences to be true – the mod keeps increasing by one but the congruences keep occurring. If you start looking for a counter-example, you might find yourself believing in coincidences, because it does seem that only when $2^k \equiv 2 \pmod{k}$ is k prime: $2^2 \equiv 2 \pmod{2}$, $2^3 \equiv 2 \pmod{3}$, $2^4 \not\equiv 2 \pmod{4}$, $2^5 \equiv 2 \pmod{5}$, $2^6 \not\equiv 2 \pmod{6}$, $2^7 \equiv 2 \pmod{7}$, etc.

The conjecture is false, but it takes quite a bit more snooping to discover a counter-example: $2^{341} \equiv 2 \pmod{341}$. 341 looks like it might be prime, but it isn't: $341 = 11 \cdot 31$. You'd definitely want to use a computer to verify this counter-example: 2^{341} is over 100 decimal digits long!

¹⁹This is half of a biimplication conjecture, known as the Chinese Hypothesis, that never really existed. It appears to have originated as a mis-translation of a math text, rather than as a serious conjecture. The other half (if k is prime, then $2^k \equiv 2 \pmod{k}$) is an instance of Fermat's Little Theorem (not to be confused with Fermat's Last Theorem), and is true.

4.4.2 Prove the Negation

The second way to show that a conjecture is false is to prove that its negation is true. Frequently, this is similar to finding a counter-example, but the two approaches are not the same.

Imagine that you're asked to prove a conjecture of the form $p \rightarrow q$. You try and try, but can't do it, and so begin to suspect that it isn't true. You spend time looking for a counter-example, but can't find one. Instead, you try to prove the negation ($\neg(p \rightarrow q)$) is true, which would show that $p \rightarrow q$ is false. $\neg(p \rightarrow q) \equiv p \wedge \neg q$, meaning that you have to show that $p \wedge \neg q$ is true. As there's no hypothesis, a direct proof would give you nothing with which to work, and, unless you've read ahead, direct proof is the only technique you know. Even with those other techniques, proving $\neg(p \rightarrow q)$ probably won't be easy.

All is not lost – proving $\neg(p \rightarrow q)$ may still be a reasonable option if you think about what $p \rightarrow q$ represents. Back at the start of section 4.2.3 we mentioned that in this chapter the $p \rightarrow q$ notation was a stand-in for the more complete $\forall x(P(x) \rightarrow Q(x)), x \in D$ quantified predicate notation. In the following example, that quantified notation will be very helpful.

Example 91:

Problem: Disprove the conjecture of Example 89 (2^n is even for all $n \in \mathbb{Z}^*$) again, this time without using a counter-example.

Solution: Let $E(n) : 2^n$ is even, $n \in \mathbb{Z}^*$, and note that our conjecture does not have an explicit hypothesis. That is, the conjecture can be expressed as $\forall n (\mathbf{T} \rightarrow E(n))$, with the same domain. $\mathbf{T} \rightarrow E(n) \equiv E(n)$ (by the Law of the True Antecedent), allowing us to represent the conjecture as $\forall n E(n)$.

To verify that $\forall n E(n)$ is false without using a counter-example, we need to prove that its negation ($\neg \forall n E(n)$) is true. By Generalized De Morgan's Laws, $\neg \forall n E(n) \equiv \exists n \neg E(n)$. In conversational English: There's a non-negative integer that makes 2^n odd.

To prove an existential conjecture, we just need to find one member of the domain that makes the conjecture true. We already know the member: $n = 0$. All that remains is to write this up more compactly.

Disproof (Proof of the Negation): The conjecture can be expressed as $\forall n E(n)$, $n \in \mathbb{Z}^*$, where $E(n)$ represents ‘ 2^n is even.’ Its negation is $\exists n \neg E(n)$ (by Generalized De Morgan’s). This negated conjecture is true, as the following proof shows:

Proof (Direct): Consider $E(0)$. $0 \in \mathbb{Z}^*$ and $2^0 = 1$. Therefore, $\exists n \neg E(n)$ is true.

Because it is possible for 2^n to be odd, the claim that 2^n is even for all $n \in \mathbb{Z}^*$ is false.

Please note that $n = 0$ is not a counter-example, because we aren’t ‘countering’ the original conjecture with it. Rather, $n = 0$ proves the negation of the conjecture. Because the conjecture’s negation is an existential statement, all we needed was an example to prove it. The proof of the negation disproves the conjecture.

By the way, our little proof of the truth of $\exists n \neg E(n)$ is a simple example of a constructive proof (see section 4.2.3).

Having seen Example 91, it’s not hard to imagine this prove-the-negation approach working on existential conjectures that we suspect are false. Imagine you’re asked to show that $\exists x A(x)$ is true. All you need is one example, but try as you might, you can’t find one. Suspecting that the conjecture is false, you look to prove its negation. By Generalized De Morgan’s Laws, $\neg \exists x A(x) \equiv \forall x \neg A(x)$. The negation is universally quantified, just as are most of the conjectures we usually need to prove, meaning that a common proof technique (such as direct proof) might be just what we need.

Example 92:

Problem: Prove or disprove: There exists an odd multiple of four.

Solution: This conjecture is clearly false, but let’s pretend that it’s more of a challenge.

Because this is an existential conjecture, to prove it we only need one example of an odd multiple of four to show that that conjecture is true. Enumeration of multiples of four ($\dots - 8, -4, 0, 4, 8, \dots$) doesn't help us find one, making us think that the conjecture may be false.

To show that the conjecture is false, we can either find a counter-example or prove the conjecture's negation. Finding a counter-example for an existential conjecture isn't easy, so we'll try to prove the negation. Let $O(x) : x$ is an odd multiple of four.²⁰ In logic notation, our conjecture is $\exists x O(x)$, where $x \in \mathbb{Z}$. We can express its negation as a universal quantification: $\neg \exists x O(x) \equiv \forall x \neg O(x)$. In English: If x is a multiple of four, it's even. We can prove that easily.

Because the negation of the original conjecture is true, the conjecture must be false. The following write-up is more terse than was that of Example 91.

Disproof (Proof of the Negation): The given conjecture (an odd multiple of four exists) is false because its negation (all multiples of four are even) is true, as the following argument shows:

Proof (Direct): Let x be a multiple of four. $x = 4k$, $k \in \mathbb{Z}$.
 $x = 4k = 2(2k)$. Therefore, all multiples of four are even.

Disproofs, even those containing proofs, don't usually include much explanation.

These disproof techniques are independent of proof techniques. We placed the disproof section in this chapter because it makes sense to discuss disproofs soon after proofs are introduced, not because they can only be used with conjectures you first tried to prove with a direct proof.

²⁰If you said to yourself, "Hey, that predicate should be expressed as multiple predicates!", you get an imaginary gold star. If you said that aloud in a public place, try to act like you don't care what other people think. To keep this example short(er), we're cheating on the predicate. Writing it out as multiple predicates, performing the negation, and rewriting the result in English should produce the result given above. But why take our word for it?

Chapter 5

Indirect (“Contra”) Proofs

With direct proofs covered (see the previous chapter), we can consider two proof techniques that are variations of direct proof: Proof by Contraposition (a slight variation) and Proof by Contradiction (a not-as-slight variation). These are sometimes known as ‘indirect’ proof techniques because they are, well, not direct.¹

5.1 Proof by Contraposition

Sometimes, a conjecture’s hypothesis doesn’t provide much useful information with which to start a direct proof. We’ve seen that, with a little creativity, we can sometimes extract enough information to make a direct proof possible (e.g., Example 85 in Chapter 4). When you are faced with a conjecture whose conclusion seems to be a better source of information than does the hypothesis, a *proof by contraposition* might be a better choice of proof technique.

proof by contraposition

The name “proof by contraposition” completely reveals the difference between it and a direct proof: Instead of proving the given conjecture as-is, we prove its contrapositive. This technique is still logically valid because, as we learned in Chapter 1, an implication and its contrapositive are equivalent: $p \rightarrow q \equiv \neg q \rightarrow \neg p$. And so, in a proof by contraposition:

To prove $p \rightarrow q$: Assume $\neg q$, show $\neg p$.

Other than that, everything about a proof by contraposition is the same as for a direct proof. To make the reader’s life a little easier, start such

¹Proof techniques just don’t have superhero-esque origin stories.

proofs with “Proof (by Contraposition):” or “Proof (Contrapositive):”, but otherwise, you already know what to do (and what not to do!).

indirect proof

Proof by contraposition is sometimes referred to as a form of *indirect proof*, although that term is most frequently applied to proof by contradiction (see Section 5.2). The term fits here because we aren’t simply assuming p and showing q . Proofs by contradiction are even less direct, which is probably why most people reserve the term to describe that technique.

Because proofs by contraposition are very similar to direct proofs, we will give just two examples, enough to highlight the utility of the technique.

Example 93:

Problem: Prove that if $a - b$ is an irrational number, then a is irrational or b is irrational, where $a, b \in \mathbb{R}$.

Solution: To use a direct proof here, we need to know something useful about irrational numbers as a starting point for the argument. We know that any real number that isn’t expressible as a ratio of integers is irrational, but that doesn’t seem like much of a starting point for a direct proof. However, it can be useful in a proof by contraposition, because using contraposition means introducing negations, and a real number that is not irrational is rational – and something we can easily represent.

In a proof by contraposition, we will need to assume the negation of the original conclusion and show the negation of the original hypothesis. Let’s start with the original conclusion: “ a is irrational or b is irrational”. The disjunction appears to be inclusive (there are no linguistic clues to the contrary). By one of De Morgan’s laws, its negation is “ a is rational and b is rational”. We can work with that. Our new conclusion, the negation of the original hypothesis, is “ $a - b$ is rational”. That also seems manageable.

Actually, it’s more than just manageable; it’s quite straight-forward, as is the entire proof: We will represent a and b as ratios of integers and use basic algebra to demonstrate that their difference is also a ratio of integers.

Proof (Contraposition): We are assuming that a and b are both rational numbers. We need to show that their difference $(a - b)$ is also a rational number.

Rational numbers can be expressed as ratios of integers. Let $a = \frac{c}{d}$ and $b = \frac{e}{f}$, where $c, d, e, f \in \mathbb{Z}$. $a - b = \frac{c}{d} - \frac{e}{f} = \frac{cf}{df} - \frac{de}{df} = \frac{cf-de}{df}$, which is a ratio of integers. Thus, $a - b$ is a rational number.

Therefore, if $a - b$ is an irrational number, then a is irrational or b is irrational, where $a, b \in \mathbb{R}$.

The value of stating the assumed information is greater in contrapositive proofs than in direct proofs, because, having had to rewrite the hypotheses and conclusions, mistaking the old for the new is a potential problem. Writing out the contrapositive's hypothesis and conclusion can help prevent that problem from occurring. We won't usually state the new assumption and conclusion as plainly as we did here, but we do recommend the practice.

Please notice that we concluded the proof by restating the original conjecture, not its contrapositive equivalent. The reason for this is that we were asked to show the truth of the original conjecture. It is appropriate to end by stating that we have shown what we were asked to show.

The next example shows a problem that can occur if one is too eager to attempt a direct proof, and how trying a different proof technique can avoid the problem.

Example 94:

Problem: Prove that if $y^2 + y \leq xy + x$, then $y \leq x$, $x, y \in \mathbb{R}$.

Solution: If your mind is open to contrapositives, you'll look at this conjecture and see that it fits that form very nicely. If you're still locked into direct proofs, you might look at the hypothesis and notice that $y + 1$ can be factored from both sides, leading to this straight-forward but in-

valid argument:

‘Proof’ (Direct):

$$\begin{array}{rcl} y^2 + y & \leq & xy + x & \text{[Given]} \\ y(y + 1) & \leq & x(y + 1) & \text{[Factoring]} \\ y & \leq & x & \text{[Divide both sides by } (y + 1)\text{]} \end{array}$$

Therefore, $y \leq x$.

Do you see the error in the reasoning? If not, spend a few minutes thinking about it before moving onto the next paragraph.

There are actually two problems, one more immediate than the other. The immediate problem is with the division. x and y , we were told, can be any real numbers. -1 is a real number, and when $y = -1$, moving from the second line to the third is a division by zero.

The less-immediate problem got overlooked in our rush to choose a proof technique: Multiplying (or dividing) both sides of an inequality by a negative number changes its direction. This conjecture is not true for nearly all negative reals.

Time to fix these problems. So that we have something provable, let’s change the domain to the positive reals (that is, $x, y \in \mathbb{R}^+$).² The new domain eliminates the division by zero concern, meaning that we could use the direct proof. We’ll do it with contraposition anyway, so that you can compare the techniques.

Proof (Contraposition): Assume that $y > x$. We need to show that $y^2 + y > xy + x$.

$$\begin{array}{rcl} y & > & x & \text{[Given]} \\ y(y + 1) & > & x(y + 1) & \text{[Multiply both sides by } (y + 1)\text{]} \\ y^2 + y & > & xy + x & \text{[Multiply through]} \end{array}$$

Therefore, if $y^2 + y \leq xy + x$, then $y \leq x$, $x, y \in \mathbb{R}^+$.

Using a proof by contraposition allows us to start with a simple hypothesis and build toward the conclusion. This ‘simple-to-complex’ progression is often the most straight-forward way to develop an argument.

Final note: Are you thinking that we should be assuming $y \geq x$ instead of $y > x$? If so, you’re forgetting that the negation of \leq is $>$.

5.1.1 Disproofs and Contraposition

Just as there’s no special connection between direct proofs and disproofs, there’s also nothing special about disproving conjectures that initially appeared to be good candidates for proof by contraposition. The techniques presented in the previous chapter (Chapter 4) are still the ones to use.

5.2 Proof by Contradiction

In Chapter 3 we presented example dialog that mentioned ‘reductive’ reasoning. We deferred discussion of it to this chapter because well-structured reductive reasoning is also known as *proof by contradiction*. The idea is easy to state but harder to explain: We assume that the conclusion is false and reason until we reach a logical contradiction.

proof by contradiction

At first read, that statement might seem like a big pile of shhhh...aving cream,³ but it is entirely logical. Here’s how it turns out to be a valid argument.

Proof by contraposition works because the contrapositive of the conjecture is logically equivalent to the conjecture. It’s tempting to look through our tables of logical equivalences from Chapter 1 to see if there are any others that look promising. The Law of Implication seems simple enough, but it turns an implication into an inclusive-OR. There are three ways for an inclusive-OR to be true (that’s three cases to have to prove), plus we lose the conclusion. Simple, but not very helpful.

²Does it feel like we’re cheating by changing the problem? If this were a homework or exam, of course we wouldn’t change it; rather, we’d ask if there’s a typo or if a disproof is acceptable. We’re trying to make a point about thinking before doing, and to demonstrate proof by contradiction. Having learned about disproofs in the last chapter, you should be able to disprove this conjecture with its original domain. Give it a try!

³Be nice and clean! Shave every day and you’ll always look keen! (Apologies to Benny Bell, author of the 1946 song “Shaving Cream.”)

Another promising equivalence is $p \rightarrow q \equiv (p \wedge \neg q) \rightarrow \mathbf{F}$, known as *reductio ad absurdum*. There’s no inclusive-OR to worry about, and we still have a conclusion . . . but it’s just ‘false.’ How can we reason toward ‘false?’ We learned two ways in Chapter 1: Use a truth table that shows all inputs evaluating to ‘false,’ or create a logical equivalence argument that ends at ‘false.’ Either way, we demonstrate a contradiction.⁴ In a more free-form proof, like those of this chapter, we can accomplish the same thing by reasoning until we discover a result (say, \bar{x}) that is the opposite of something we already know (x). Both \bar{x} and x cannot be true. Assuming that we started with the correct compound hypotheses ($p \wedge \neg q$), and that our reasoning was logical, arriving at a contradiction (that is, arriving at \mathbf{F}) means that the original conjecture must be true, by *reductio ad absurdum*.

Here’s another way to look at how a proof by contradiction works. Start with a basic conjecture: $p \rightarrow q$. We believe it to be true, otherwise we wouldn’t be trying to prove it. That means we believe $\neg(p \rightarrow q)$ to be false. Because $\neg(p \rightarrow q) \equiv \neg(\neg p \vee q) \equiv \neg\neg p \wedge \neg q \equiv p \wedge \neg q$, we must also believe $p \wedge \neg q$ to be false. If we reason starting from $p \wedge \neg q$ and encounter a contradiction, as long as our reasoning was good, the only part of the argument that could have been false was the assumption, and therefore its opposite – the given conjecture $p \rightarrow q$ – must be true.

Still not really buying it? We understand; this isn’t easy for most people to follow the first time, or the second, or maybe even the nineteenth. Keep thinking about it; if you followed the logic behind proof by contraposition, you can get your mind around proof by contradiction, too. At a minimum, we’re confident that you understand why proof by contradiction is also known as indirect proof!

In the meantime, take a look at that assumption again: $p \wedge \neg q$. To be allowed to assume both p and $\neg q$, imagine that we made a deal with the Deity of Arguments.⁵ The Deity of Arguments was sympathetic to our need for more given information, and was willing to help, but, needing to save face with the other deities, required something in return: We had to sacrifice the conclusion in exchange for more given information. Having accepted this deal with the deity, our fate is to reason blindly until we stumble upon some sort of contradiction – we have no idea when or where it will appear.⁶

⁴Contradiction . . . proof by contradiction . . . coincidence? Yeah, right.

⁵Yes, a relative of the Deity of Partial Credit; naturally, they’re both from the same pantheon: The one from which no big-budget action movie will ever be made.

⁶For the sake of your grade on a future quiz or exam, if asked to explain the logical justification for a proof by contradiction, don’t answer “The Deity of Arguments gave it to us!” We gave you two lovely, serious explanations; answer using one of those instead.

For many people, proof by contradiction's lack of a clear target makes applying it a significant challenge. Other people really take to it, to the point that proof by contradiction is their first choice whenever a proof is needed. If you find that proof by contradiction really 'sings' to you, try to remember to use it only when it makes sense to do so.

That's a key question: When does it make sense to use a proof by contradiction? Direct proofs are useful when p gives enough information. Proofs by contraposition are useful when $\neg q$ gives enough. If neither p nor $\neg q$ give enough by themselves, perhaps their combination $(p \wedge \neg q)$ does. That's when a proof by contradiction may be the best choice.

In summary:

To prove $p \rightarrow q$: Assume $p \wedge \neg q$, show a contradiction.

To show how and when to use a proof by contradiction, we have three examples. The first one is pretty simple ... maybe *too* simple.

Example 95:

Problem: Prove that if $a \% 3 = 1$ and $b \% 3 = 2$, then $3 \mid (a + b)$.

Solution: First things first: We need to remind ourselves what the notation is trying to tell us. $a \% 3 = 1$ means that a is one more than a multiple of three. Similarly, $b \% 3 = 2$ means b is two more (or one less) than a multiple of three. $3 \mid (a + b)$ means that $a + b$ is exactly a multiple of three.

To prove this by contradiction, we assume $a \% 3 = 1$, $b \% 3 = 2$, and $3 \nmid (a + b)$ (the negation of $3 \mid (a + b)$). $3 \nmid (a + b)$ means that $a + b$ isn't a multiple of three; that is, $(a + b) \% 3$ is either one or two.

If you're thinking, "Wait; 'either one or two?'" Doing this by contradiction seems to be making this harder, not easier!", you're forgiven. As usual, we chose this example to demonstrate the proof technique and to make a point.

Proof (Contradiction): Assume that $a \% 3 = 1$, $b \% 3 = 2$, and $3 \nmid (a + b)$. Because a is one more than a multiple of three, we can represent a with $3k + 1$, where $k \in \mathbb{Z}$. Similarly, we will represent b with $3j + 2$, where $j \in \mathbb{Z}$.

$a + b = (3k + 1) + (3j + 2) = 3k + 3j + 3 = 3(k + j + 1)$. This shows that $a + b$ is a multiple of three; that is, $3 \mid (a + b)$. However, we assumed that $3 \nmid (a + b)$. No value can both be and not be a multiple of three; this is a contradiction.

Therefore, if $a \% 3 = 1$ and $b \% 3 = 2$, then $3 \mid (a + b)$.

Although our discussion ahead of the proof made it seem that things were going to get messy (“ $(a + b) \% 3$ is either one or two”), we didn’t need to reach that level of detail in the proof. Sometimes, the preparation is worse than the proof.

Speaking of messes, here’s the point we want to make about proving the conjecture of Example 95 using contradiction: We really don’t need an indirect proof to do it; a direct proof will work just fine. (For practice, create one!) Some snooty proof experts will tell you that it’s not appropriate to use contradiction when the resulting contradiction is with one of the givens. Unfortunately, it can be hard to see that that will occur before you write the proof. We aren’t that snooty; we’re happy to accept the contradiction whenever and wherever we find it.

Example 96:

Problem: One of our example conjectures in Chapter 4 (“if gh is odd, then g and h are both odd” from Example 85) required us to consider three cases when we did it with a direct proof. Would doing it with a proof by contradiction be easier?

Solution: This is the sort of question that’s good to ask yourself when you’re working on a homework assignment (but not-so-good to ponder during an exam!). You’ve completed a proof, and you could turn it in

... but that annoying voice in the back corner of your brain is telling you that you can do better. Let's humor that voice and see if we can do better with a proof by contradiction.

Using contradiction, we get to assume that " gh is odd" is true, and also that the negation of " g and h are both odd" is true. But what is that negation? It's tempting just to flip 'odd' to 'even,' but that's not the correct negation. " g and h are both odd" is a short-cut way of saying " g is odd and h is odd." To negate that, we need to haul out one of De Morgan's Laws. The resulting negation is " g is even or h is even." Thus, we can assume g is even and h is odd, or g is odd and h is even, or that both g and h are even. As with Example 95, this is starting to sound like a lot of work. Maybe it will be, maybe it won't; there's one sure way to find out: Try it!

Proof (Contradiction): Assume that gh is odd, and that g is even or h is even (or both). If only one variable is even, it doesn't matter which one, because we are going to create the product gh and multiplication is commutative. If both are even, we can again assume that either one is even. WLOG, let g be even and equal to $2k$, $k \in \mathbb{Z}$. $gh = (2k)h = 2(kh)$, by associativity of multiplication. This shows that gh must be even, a contradiction of our assumption that gh is odd.

Therefore, if gh is odd, then g and h are both odd.

This is much shorter than the direct proof we created for this conjecture. It's so short that, by comparison, you might be worried that something's wrong with it. There isn't; the brevity can be attributed to the use of a more appropriate proof technique.

Actually, it could be quite a bit shorter. In many textbooks, authors would skip the justification ahead of the 'WLOG,' leaving that for the readers to puzzle out on their own. In a class in which you are learning to write proofs, be safe and include the justification. Now, if we wanted to make the proof longer,⁷ we could have considered all three cases separately, much as we did in the direct proof. Thanks to our knowledge of multiplication, there was no need to do that.

Example 96 shows how choosing the appropriate proof technique can save time. Even so, it isn't a 'classic' proof by contradiction, because it contradicted a piece of assumed information. The next example is even better, because the conjecture would be very difficult to prove with another technique.

Example 97:

Problem: Prove that $\log_3 4$ is irrational.

Solution: This is another conjecture not in 'if – then' form. There's a good reason for that: There isn't a specific hypothesis, just the desired conclusion. For such conjectures, a proof by contradiction is often a good choice, because it allows us to use (the negation of) the only available piece of information as our starting point.

We know that a real number that is not irrational is rational. We will represent $\log_3 4$ with a fraction, and use our knowledge of logarithms to take us to a contradiction.

⁷Why?!? For the love of the Deity of Arguments, why?

Proof (Contradiction): Assume that $\log_3 4$ is rational. Let $\log_3 4 = \frac{n}{d}$, where $n, d \in \mathbb{Z}^+$. By the relationship between logarithms and exponents, we can rewrite this as $3^{n/d} = 4$. Raising both sides to the power of d gives $(3^{n/d})^d = 4^d$, which means $3^{(n/d) \cdot d} = 4^d$, or $3^n = 4^d$.

Observe that $\log_3 4$ is more than one (because $\log_3 3 = 1$ and \log is an increasing function) and less than two ($\log_3 9 = 2$). This means that $n > d > 1$, and thus $n - 1$ and $d - 1$ are integers that are ≥ 1 .

$3^n = 3^{n-1} \cdot 3$, which is an odd times an odd. We learned, in Example 78 of Chapter 4, that the product of two odds is odd. $4^d = 4^{d-1} \cdot 4$, which is an even times an even. The product of two evens must be even ($(2k)(2j) = 2(2kj)$). Because 3^n is odd and 4^d is even, $3^n \neq 4^d$. This contradicts the earlier observation that $3^n = 4^d$.

Therefore, $\log_3 4$ is irrational.

We have several things to say about the proof of Example 97.

- This proof is a good example of how ‘classic’ contradiction proofs work: We reasoned for a while and discovered something ($3^n = 4^d$). Then, we reasoned some more on a different fact and discovered a contradictory piece of information ($3^n \neq 4^d$). Neither of those were given to us; we had to uncover them ourselves.
- You probably noticed that we snuck⁸ in a mini-lemma to show that the product of two evens is even. It would be more proper to have a separate lemma to show this, but as we’re now quite familiar with manipulating evens and odds, we decided to use a short-cut.

⁸‘Sneaked’ is considered to be better for formal writing than is ‘snuck.’ Since when has this book been an exemplar of formal writing?

If you’re wondering why we didn’t just reference the lemma from Example 81 in Chapter 4, note that it specifically covers only the case of an even number times itself, not the case of an even times a potentially different even. They are closely related, but are not the same. It would be appropriate to say that the evenness of an even times itself is a corollary of the evenness of an even times an even, but our examples weren’t set up in that order.

- The proof’s little digression into the characteristics of $n - 1$ and $d - 1$ is the sort of discussion that most proofs would assume the reader could figure out on their own. We decided to include it because (a) when exponents are negative, we leave the realm of integers (and therefore of evens and odds), and (b) showing that $d - 1 \geq 1$ meant that we didn’t have to worry about $4^0 \cdot 4$ (an odd times an even).
- Tired of evens and odds? Want a different way to reach a contradiction? We could have used this approach for the last half of the proof: The only factors of 3^n are powers of 3, and the only factors of 4^d are powers of 4. Because n and d are greater than zero, there’s no way for 3^n to equal 4^d .
- Are you wondering if we could have used a proof by contraposition instead? Yes, we could have, but we would have created a proof by contradiction wearing a proof by contraposition costume. Here’s what we mean: As we know, in a proof by contraposition of $p \rightarrow q$, we assume $\neg q$ and show $\neg p$. We weren’t given a hypothesis, which means $p \equiv \mathbf{T}$ (we could have written the conjecture as “if true, then $\log_3 4$ is irrational”). In a proof by contraposition, then, we’d get to assume that $\log_3 4$ is rational (just like the proof by contradiction) and would need to show false (again, just like the proof by contradiction). Might as well just do a proof by contradiction!

5.3 Proving Biconditional Conjectures

Most of our conjectures have been (or could be) expressed as “for all” implications. So far we’ve ignored another kind of “for all” conjecture that can also be expressed as implications: Biconditionals. Happily, proving a biconditional does not require another proof technique. Unhappily, proving a biconditional requires that we construct two proofs.

We hope you remember that a biconditional is defined in terms of implication and AND: $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$. This equivalence tells us what we must do to prove a biconditional:

To prove $p \leftrightarrow q$: Prove both $p \rightarrow q$ and $q \rightarrow p$.

Be aware that you may “mix-n-match” proof techniques when proving a biconditional; that is, you may use a different technique for the “if” half than you used for the “only if” half. Of course, using the same technique for both halves is fine.

To construct a proof of a biconditional conjecture, we have two choices:

1. Write separate, stand-alone proofs of $p \rightarrow q$ and $q \rightarrow p$, and refer to them as lemmas in the (very short) proof of $p \leftrightarrow q$.
2. Include the proofs of $p \rightarrow q$ and $q \rightarrow p$ as cases within the proof of $p \leftrightarrow q$.

We have two examples of biconditional proofs, one for each option.

Example 98:

Problem: Prove that s and t are odd iff st is odd.

Solution: We have already done most of the work for this proof. Example 78 proved “if x and y are odd, then xy is odd,” and Example 85 proved the “if gh is odd, the g and h are odd.” All we need to do is reference those results as lemmas within our proof of the biconditional.

Proof (Direct): In Example 78, we proved the “only if” half (s and t are odd only if st is odd). In Example 85, we proved the “if” half (s and t are odd if st is odd). Together, these lemmas complete the proof of the biconditional.

Therefore, s and t are odd iff st is odd.

Of course, much of the time we need to prove both halves. The next example shows how to handle this situation.

Example 99:

Problem: Prove that $a^2 + 2a + b^2 - 6b + 10 = 0$ iff $a = -1$ and $b = 3$.

Solution: The “if” half of this is easily proven – we just plug in the given values for a and b and verify that the result is zero.

The “only if” direction requires more imagination. It’s tempting to do a little re-arranging ($a^2 + 2a + b^2 - 6b + 10 = a(a + 2) + b(b - 6) + 10$), but that doesn’t seem helpful (how will that tell us what a and b must be?). Instead, we might look at $a^2 + 2a$ and $b^2 - 6b$ and think about completing the squares by finding a way to divide up the 10 between them. This occurs to us because the form $x^2 + yx + z$ is familiar to us, and because specific values for a and b remind us of roots of quadratics. The source of $a^2 + 2a$ could be $(a + 1)^2$: $(a + 1)^2 = a^2 + 2a + 1$. That leaves 9 for $b^2 - 6b$, and happily $b^2 - 6b + 9 = (b - 3)^2$.

We’ve learned that $a^2 + 2a + b^2 - 6b + 10 = (a + 1)^2 + (b - 3)^2$. Now we need a convincing argument that the only way for $(a + 1)^2 + (b - 3)^2$ to equal zero is for a to be -1 and for b to be 3 . We know that the square of any real will be non-negative, which means that the only way to get a sum of zero is $0 + 0$. The only way for $(a + 1)^2$ to equal zero is for a to be -1 , and the only way for $(b - 3)^2$ to equal zero is for b to be 3 . We’ve got our argument.

Proof (Cases): To prove the biconditional we need to prove the two component conjectures individually.

Case 1: We use a direct proof to show that $a^2 + 2a + b^2 - 6b + 10 = 0$ if $a = -1$ and $b = 3$.

Replacing a and b with the given values, we find that $(-1)^2 + 2(-1) + (3)^2 - 6(3) + 10 = 1 - 2 + 9 - 18 + 10 = 0$. Thus, $a^2 + 2a + b^2 - 6b + 10 = 0$ if $a = -1$ and $b = 3$.

Case 2: We use a direct proof to show that $a^2 + 2a + b^2 - 6b + 10 = 0$ only if $a = -1$ and $b = 3$.

$a^2 + 2a + b^2 - 6b + 10 = a^2 + 2a + 1 + b^2 - 6b + 9 = (a + 1)^2 + (b - 3)^2$. As x^2 is ≥ 0 for any real number x , the only way for $(a + 1)^2 + (b - 3)^2$ to equal 0 is for $(a + 1)^2 = 0$ and $(b - 3)^2 = 0$. The roots of these quadratics are -1 and 3 , respectively, establishing that $a = -1$ and $b = 3$. Thus, $a^2 + 2a + b^2 - 6b + 10 = 0$ only if $a = -1$ and $b = 3$.

Therefore, $a^2 + 2a + b^2 - 6b + 10 = 0$ iff $a = -1$ and $b = 3$.

In a biconditional proof, our preferred “Proof (*technique*)” label can be more awkward than helpful, because we could use one proof technique for the “if” and another for the “only if.” As each case is a separate proof, it’s reasonable to mention the technique at the start of each case, and to say at the top that we’re doing a proof by cases, as we did here. But, as we used direct proofs in both cases, we could have used our usual format instead.

5.4 Summary of Direct and Indirect Proof Techniques

Figure 5.1 summarizes the three major proof techniques covered in the past two chapters.

The algorithm given in Figure 5.2 offers suggestions for choosing a proof technique. This algorithm is not meant to be inflexible; it has value in that it can help novice proof-writers decide how to begin. Remember, false starts

Proof Technique	(Assume) Hypothesis	(Show) Conclusion
Direct	p	q
Contraposition	$\neg q$	$\neg p$
Contradiction	$p \wedge \neg q$	A Contradiction

Figure 5.1: The three major proof techniques, their starting points, and their destinations.

<p>If the conjecture ($p \rightarrow q$) appears to be true,</p> <p style="padding-left: 40px;">If p provides useful information, Try a direct proof</p> <p style="padding-left: 40px;">Otherwise, if $\neg q$ provides useful information, Try a proof by contraposition</p> <p style="padding-left: 40px;">Otherwise, Try a proof by contradiction</p> <p>Otherwise,</p> <p style="padding-left: 40px;">If the conjecture is universally quantified, Try disproving it with a counter-example</p> <p style="padding-left: 40px;">Otherwise, Try disproving it by proving its negation</p>
--

Figure 5.2: An initial proof technique selection algorithm.

are common in proof-writing. What appears to be a good starting point for a proof may not be upon closer examination. Also, remember that not all proofs are of conjectures of the $p \rightarrow q$ form. For instance, the conjecture might be existential, for which a single confirming example will prove that it is true.

Chapter 6

Additional Set Concepts

Appendix A presents the basics of sets, including notations, operators, and visualizations with Venn diagrams. This chapter assumes that you know that material. If you are feeling a lack of confidence in your knowledge of, for example, set builder notation or the set difference operator, please review that section before continuing.

6.1 What is So Important about Sets?

A good working knowledge of sets is essential in the study of discrete structures. We will define relations, and soon after, functions, in terms of sets. Relational database management systems (RDBMSes) are based on set theory. The concept of finite probability is defined by the ratio of two set cardinalities. If you know data structures such as arrays and linked lists, you likely recognize that they are representations of collections – often sets – of values.

Beyond this class, you may find yourself studying topics in theoretical computer science. Many of those ideas are defined using sets.

Example 100:

Problems in computing that can be solved by performing a quantity of operations that is related polynomially to the size of the problem are known as *polynomial-time* (P) problems. Many sorting algorithms have this property. For example, insertion sort's quantity of operations performed is proportional to n^2 in the worst case, where n is the number of values being sorted.

Problems whose solutions can be verified (just checked for correctness, not necessarily discovered) in polynomial time are known as NP (**non-deterministic polynomial time**) problems. Again using sorting as an example, we can easily check that a list of values is sorted using a quantity of operations that can be described by a linear function. Because linear functions are polynomial, sorting is an NP problem as well as a P problem.

Are all problems in both sets? That is, are the two sets equal? No one knows; it's arguably the most important open question in computer science. Computer scientists and mathematicians are pretty sure these sets are *not* equal; we just haven't been able to prove it (or disprove it).

Why, then, are we pretty sure? Because we know of hundreds of problems for which no efficient solution (that is, no polynomial-time solution) is known. One example is the *partitioning problem*. Consider the set of integers $\{1, 3, 4, 5, 6, 9, 12\}$. Can these values be divided into two subsets such that all of the values are included and the sums of the values in each subset are equal? The answer is 'yes,' both $\{1, 3, 4, 12\}$ and $\{5, 6, 9\}$ have values whose sums are 20. This probably doesn't sound like a very hard problem, and it isn't when the quantity of values is very small, because we can easily enumerate all possible subsets. But imagine that you had to find two such subsets from a set of cardinality 500. The only known algorithm that is guaranteed to find a solution, if one exists, is the algorithm that enumerates all possible subsets. As we will see when we cover counting, the number of possible subset pairs is exponential, not polynomial, in the cardinality of the original set. For that set of cardinality 500, you'd have to examine a ridiculous quantity of subsets.¹ The lack of a better algorithm probably means that this is a hard problem.

6.1.1 Sets vs. Logic

Set theory, like proof theory, is frequently considered to be part of mathematical logic because set theory is built on a foundation of logic. As such, much

¹Specifically, $2^{500-1} - 1 = 1, 636, 695, 303, 948, 070, 935, 006, 594, 848, 413, 799, 576, 108, 321, 023, 021, 532, 394, 741, 645, 684, 048, 066, 898, 202, 337, 277, 441, 635, 046, 162, 952, 078, 575, 443, 342, 063, 780, 035, 504, 608, 628, 272, 942, 696, 526, 664, 263, 794, 687$. See what happens when you waste time reading footnotes?

of what we will learn about sets has direct parallels with the material we have already covered in Chapter 1. In fact, in this chapter we will spend a fair amount of time explaining how set operators can be defined using logic, and how set identities can be proven using logic.

Thus, it would be incorrect to say that set theory and first-order logic are competitors. They are two systems of expression and reasoning that are closely related. If you are coming into this chapter feeling confident that its content will be easier to follow than the logic chapters because sets already make sense to you, great! Hopefully, the connections we will make between sets and logic will help you understand logic even better, without destroying that warm, fuzzy feeling you have for sets.

6.1.2 Limitations of Sets

Set theory, like logic, doesn't provide the tools necessary to describe everything we might like to describe. This admission shouldn't cause you to lose faith in mathematics in general and computer science in particular. It's clear that the foundations of these fields of study are useful, because computers do work, we are able to launch satellites into orbit, etc. But, just as screwdrivers and hammers are both useful tools for different tasks, set theory is most useful for what it is designed to describe.

Where does set theory start to lose its usefulness? The next example exposes a classic difficulty in basic (a.k.a. fundamental, naive) set theory that is also a good example of a *paradox*, a statement that contradicts itself, usually in a mind-boggling way.

paradox

Example 101:

Consider a men's football team with one player who is a barber. The coach insists that every man be clean-shaven; that is, no player has a moustache or a beard. The barber happily shaves all of the players — but only those players — who do not shave themselves. Who shaves the barber?

That doesn't sound hard to answer: He shaves himself, of course. But consider the conditions of the question again. The barber isn't allowed to shave himself; remember, he shaves the players who do not shave themselves. Thus, the barber must shave himself (he can't go to the barber; he *is* the barber!) but cannot shave himself (the barber shaves only those



Figure 6.1: “My brain hurts!” Credit: Monty Python’s Flying Circus, Season 3, Episode 6 (“The War Against Pornography”), “Gumby Brain Specialist.”

players who do not shave themselves).

This is known in set theory as the Barber Paradox, and is the standard example used to illustrate the more formal Russell’s Paradox, raised by logician Bertrand Russell in 1901 as a demonstration of a problem with basic set theory. We know that we can have a set of sets; that is, a set can be a member of a set. Consider the idea of a set that is not a member of itself. Now let’s create a collection of such sets: Let S be a set of the sets that are not members of themselves. Is S a member of itself? If it is not (that is, if $S \notin S$), then by its definition S must be $\in S$. But if S is a member of itself ($S \in S$), the definition is violated. The realization that S must both be and not be a member of itself is the paradox. In logic: $S \in S \leftrightarrow S \notin S$, where $S = \{s \mid s \notin s\}$.

How’s your brain doing? Happily, there’s a solution to Russell’s Paradox: Adopt a more restrictive view of sets, such as an ‘axiomatic’ set theory. Essentially, an *axiomatic set theory* is one that adds a group of axioms (‘ground

axiomatic set theory

rules’) to the basic set ideas so that problems like Russell’s Paradox can no longer arise. For our purposes, knowing that there is a solution to such problems is enough.²

6.2 Logical Representation of Basic Set Operators

Expressing set operators using logical notation may seem to be a purely intellectual exercise. After all, we can use the operators just fine given their descriptions in English. Without the logic, though, proving set properties that use those operators requires exhaustive enumerations, much like the truth tables we used to prove logical identities. By expressing set operators in logic, we can employ the logical identities we’ve already proved (or could prove if necessary), hopefully saving time and effort.

6.2.1 Logical Representations of Union, Intersection, Difference, and Complement

All four of the set operators union (\cup), intersection (\cap), difference ($-$), and complement (\square) accept a set or two as operands and produce a set as their result. To represent them in logic, then, we need a notation that allows us to describe their behavior with logic but produce a set. Fortunately, we already have one: Set-builder notation (see [A.3.1](#)).

Let’s start with union. In [Appendix A](#), we described the union of two sets as being “a binary operator that combines the elements from the two operand sets into a single result set.” Thus, an element is in the result if it is found in the first operand set or the second operand set. Because an element found in both operand sets is included in the result, we need an inclusive OR. That’s all we need to know to express union logically:

$$A \cup B = \{c \mid c \in A \vee c \in B\}$$

Intersection can be expressed almost exactly the same way. To be in the result, an element must be in the first operand set and the second:

$$D \cap E = \{f \mid f \in D \wedge f \in E\}$$

Remembering which logical operator pairs with each of these set operators is easy. Operators that open in the same direction go together: \cap goes with \wedge (both open to the bottom) and \cup goes with \vee (both open to the top).

Difference requires a little more thought to express in logic. Difference ‘keeps’ all of the elements of the first operand set, unless the element is also

²If you would like to read about a specific example of an axiomatic theory of sets, look up “ZFC” (Zermelo-Fraenkel set theory with the Axiom of Choice).

found in the second set. That is, an element is in the result set when it is a member of the first set and not a member of the second:

$$G - H = \{i \mid i \in G \wedge i \notin H\}$$

Complement is the only unary operator of this group, which makes it the easiest to express in logic. The complement of a set is the set of all of the other elements in the universe; that is, everything not in the operand set:

$$\bar{J} = \{k \mid k \notin J\}$$

At this point, you may be wondering if set membership itself can be – should be! – expressed logically, too. After all, it can easily be viewed as a predicate. In practice, we think of set membership as a concept of sets, rather than an operation on a set. As such, we don’t consider it to be an operator, though we use it like one.

6.2.2 A New Set Operator: Cartesian Product

A very useful set operator, Cartesian Product pairs up elements of its operand sets. Before we can define Cartesian Product, we need to understand how to combine set elements into an ordered pair.

ordered pair

Definition 27: Ordered Pair

An *ordered pair* (denoted “ (a, b) ”) is a two-element pairing of set elements in which position matters.

The word “ordered” is important, as Example 102 demonstrates.

Example 102:

Let $S = \{4, 6, 2\}$. $(2, 4)$ is an ordered pair of elements from S .

$(4, 2)$ is a different ordered pair of A ’s elements. That is, $(2, 4) \neq (4, 2)$. This distinguishes ordered pairs from sets of cardinality two, where order doesn’t matter: $\{2, 4\} = \{4, 2\}$.

Now we can define Cartesian Product in terms of ordered pairs.

Definition 28: Cartesian Product

The *Cartesian Product* of two sets L and M , denoted $L \times M$ (`\times`), is the set of all possible ordered pairs (l, m) such that $l \in L$ and $m \in M$.

In logic: $L \times M = \{(l, m) \mid l \in L \wedge m \in M\}$

cartesian product

This definition is limited to just two operand sets. We can generalize the idea to work with more than two operands, and will soon. However, to start, we'll stay within this definition.

Example 103:

Problem: Compute $\{K, Q\} \times \{\clubsuit, \diamond, \heartsuit, \spadesuit\}$.

Solution: We need to pair each element of the first set with each element of the second, in that order, to form all of the ordered pairs of the result set. The result: $\{(K, \clubsuit), (K, \diamond), (K, \heartsuit), (K, \spadesuit), (Q, \clubsuit), (Q, \diamond), (Q, \heartsuit), (Q, \spadesuit)\}$

Note that this shows us a way to define the common French deck of 52 playing cards as a set of ordered pairs: $\{A, K, Q, J, 10, 9, 8, 7, 6, 5, 4, 3, 2\} \times \{\clubsuit, \diamond, \heartsuit, \spadesuit\}$.

When enumerating the result of a Cartesian Product manually, it is important to be systematic, so that no ordered pairs are missed or are duplicated. Counting the quantity of ordered pairs generated and comparing that sum against the fact that $|L \times M| = |L| \cdot |M|$ can detect simple errors. Continuing Example 103, we can verify that our French deck must consist of $|\{A, K, Q, J, 10, 9, 8, 7, 6, 5, 4, 3, 2\}| \cdot |\{\clubsuit, \diamond, \heartsuit, \spadesuit\}| = 13 \cdot 4 = 52$ ordered pairs (a.k.a. cards).

Example 104:

Problem: Compute $\emptyset \times \{1, 2\}$.

Solution: To create an ordered pair, we need one element from each

operand set. As the empty set has no elements, there can be no ordered pairs. Thus, $\emptyset \times \{1, 2\} = \emptyset$.

As mentioned above, although we won't have a lot of need for it in this book, computing Cartesian Products of more than two sets, or, equivalently, of sets of ordered pairs, is a straight-forward extension of the two-set definition. The results are sets of n -tuples (ordered lists of n elements each).

Example 105:

Problem: Compute $\{a, b\} \times \{\infty\} \times \{3, 5\}$.

Solution: The result is $\{(a, \infty, 3), (a, \infty, 5), (b, \infty, 3), (b, \infty, 5)\}$. Let's see how to compute this one Cartesian Product at a time.

We work left to right. The result of the first operator is as expected: $\{a, b\} \times \{\infty\} = \{(a, \infty), (b, \infty)\}$.

Before computing the second operator's result, it's helpful to think of $\{3, 5\}$ as being a set of 1-tuples: $\{(3), (5)\}$. That's what it really is; we don't ordinarily include the parentheses because there aren't multiple values for them to group together. But, adding the parentheses makes it easier for us to see that we are creating the Cartesian Product of a set of 2-tuples with a set of 1-tuples, which will form the set of 3-tuples listed above: $\{(a, \infty, 3), (a, \infty, 5), (b, \infty, 3), (b, \infty, 5)\}$.

In general, the idea of Example 105 is called an n -ary Cartesian Product.

n-ary cartesian product

Definition 29: n-ary Cartesian Product

The n -ary Cartesian Product of n sets S_1 through S_n , $S_1 \times S_2 \times \dots \times S_n$, produces the set of n -tuples $\{(s_1, s_2, \dots, s_n) \mid s_i \in S_i\}$.

The quantity of n -tuples in the resulting set is the product of the cardinalities of the n operand sets: $|S_1 \times S_2 \times \dots \times S_n| = |S_1| \cdot |S_2| \cdot \dots \cdot |S_n|$.

You’re probably wondering: “Why isn’t the result of Example 105 written as $\{((a, \infty), 3), ((a, \infty), 5), ((b, \infty), 3), ((b, \infty), 5)\}$?” That is another way to view things, but it’s not how the generalization of Cartesian Product is defined. This is a practical choice: In a later chapter, we’ll see that n -tuples (and n -ary Cartesian Products) are used by relational database systems to store (and process) data. This isn’t to say that the concept of ordered n -tuples that contain other ordered n -tuples doesn’t have uses; it does (for example, see nested lists in the programming language Scheme), but they aren’t as common in set theory.

6.3 Some Useful Set Concepts

Appendix A introduced the idea of set membership. Earlier in this chapter we said we’ll consider set membership to be a concept rather than an operator, even though it could be viewed as an operator. There are other set concepts that must be understood in order to use sets, and much of the rest of this book, effectively. In this section we introduce them. We won’t call any of these concepts operators, either, although cases could be made for many of them.

6.3.1 Subsets

Imagine a class of students. Some of them will end the term with the highest grade allowed by the school. That group may be considered a *subset* of the original set of all of the students in the class, because it contains only students that are members of that set of students. Sounds simple enough, but what if no student earns that highest grade? Is that set (the empty set) considered a subset of the class? And what if all of the students earn that grade? Is the class a subset of itself? Such extreme cases are why there are two definitions of ‘subset’ for us to learn.

Definition 30: Subset

Set N is a *subset* of set O (denoted $N \subseteq O$, `\subseteq`) when every element of N is also an element of O . Equivalently, O is a *superset* of N ($O \supseteq N$, `\supseteq`).

In logic: $N \subseteq O \equiv \forall p (p \in N \rightarrow p \in O), p \in \mathcal{U}$

subset

This definition deserves some explanation. First, notice that the logical expression for subset doesn't use set builder notation. This is because the 'result' of subset is a logical value (true or false) instead of a set: Either N is a subset of O or it isn't. This explains why we use \equiv instead of $=$; sets can be equal but logical expressions are equivalent.

Time to start considering those extreme cases. The definition doesn't directly address them, but we can infer how they are handled. Let's start with the situation in which $N = O$. Is $N \subseteq O$? That is, is a set a subset of itself? Clearly, yes. The definition says that N is a subset of O when all of N 's content is found within O . This is certainly the case when the sets have the same content.

Now consider the other extreme, when $|N| = 0$ (that is, $N = \emptyset$). Is $\emptyset \subseteq O$ true? Yes! The empty set has no elements. Vacuously, all of its elements are found within O . For this reason, the empty set is considered to be a subset of any set, even itself.

We'll see some examples after we introduce the other subset definition: *proper subset*.

proper subset

Definition 31: Proper Subset

Set Q is a *proper subset* of set R (denoted $Q \subset R$, `\subset`) when $Q \subseteq R$ but $Q \neq R$. Equivalently, R is a *proper superset* of Q ($R \supset Q$, `\supset`).

In logic: $Q \subset R \equiv \forall s (s \in Q \rightarrow s \in R) \wedge \exists t (t \notin Q \wedge t \in R)$, where $s, t \in \mathcal{U}$

(Be aware that some people use the symbol \subsetneq (`\subsetneq`) for proper subset.)

The difference in meaning between \subseteq and \subset is simple (a set is a subset of itself but not a proper subset of itself), but expressing that difference logically takes some work. To show that two sets Q and R are not equal, we say that R has an element that Q does not. We say it that way in this definition because $|Q| < |R|$ when $Q \subset R$.

People sometimes have trouble keeping the meanings of \subseteq and \subset straight. This observation usually helps: Think of the symbol \subseteq as the result of the merging of the symbols \subset and $=$. This justifies the inclusion of the single

horizontal line in the symbol for subset, and reminds us that the idea of set equality is included in the definition of subset.

Example 106:

The following table shows the similarities of and differences between the subset operators:

	S	T	$S \subseteq T?$	$S \subset T?$
(a)	$\{1, 2\}$	$\{1, 2, 3\}$	true	true
(b)	$\{4, 5\}$	$\{5\}$	false	false
(c)	$\{6\}$	$\{6\}$	true	false
(d)	\emptyset	$\{7\}$	true	true
(e)	\emptyset	\emptyset	true	false
(f)	$\{\emptyset\}$	$\{\emptyset, 8\}$	true	true

The examples of line (d) follow from the fact that the empty set is considered to be a subset (and proper subset) of any non-empty set. Line (e) mirrors line (c); as with any other set, the empty set is a subset but not a proper subset of itself. Line (f) is included to make the point that sets can contain other sets, even the empty set, as elements, and when they do, the subset definitions don't change. That is, in line (f), you could replace the two occurrences of \emptyset with another set, say $\{9\}$, and the results would be the same.

6.3.2 Set Equality

Set equality sounds like a concept that's so basic that it should have been covered back in Appendix A. We saved it for this chapter because its definition can be so easily expressed using the idea of subset.

Definition 32: Set Equality

Sets V and W are equal (denoted $V = W$) iff $V \subseteq W$ and $W \subseteq V$.

set equality

We aren't providing the logical version of equality because it's a straightforward application of the logical construction of subset.

Example 107:

Let $S = \{7, 15, 16\}$ and $T = \{16, 7, 15\}$. Applying the set equality definition, we see that $S = T$ because both $S \subseteq T$ and $T \subseteq S$ are true.

We can reach the same conclusion by inspecting the set elements. Despite the different orderings of the elements, S and T have the same cardinality and contain the same elements. The subset-based definition is more generally useful than is the inspection approach. For example, it will serve as the foundation of set equality proofs by cases in Section 6.4.2.

Example 108:

Problem: Let $A = \emptyset$ and $B = \{\emptyset\}$. Does $A = B$?

Solution: No, because these are different sets with different cardinalities: $|A| = 0$ (the empty set has no elements) and $|B| = 1$ (the empty set is the only element of B).

6.3.3 Power Sets

A companion idea to subset is that of *power set*.

power set

Definition 33: Power Set

The *power set* of a set S (denoted $\mathcal{P}(S)$, `\mathcal{P}`), is the set of all of S 's subsets.

There are a variety of symbols that are used to represent a power set, but this calligraphic 'P' is the most commonly used today.

Remember that the empty set is a subset of any set, making the empty set an element of any set's power set. Also remember that the definition of power set is based on the definition of subset, not of proper subset, meaning that the set itself is another member of the power set.

Example 109:

Problem: Let $A = \{e, f, g\}$. What is $\mathcal{P}(A)$?

Solution: Computing power sets is one of those activities that attracts silly mistakes like a kitchen floor attracts the buttered side of toast. To reduce the chance of an error, be systematic when listing the subsets: Start with all of the subsets of size 0, then list those of size 1, etc., ending with those of size $|A|$.

There is just one subset of size 0: The empty set. There are $|A| = 3$ subsets of size 1: $\{e\}$, $\{f\}$, and $\{g\}$. There are also three subsets of size 2: $\{e, f\}$, $\{e, g\}$, and $\{f, g\}$. Finally, the set itself is the only subset of size 3. Thus, $\mathcal{P}(A) = \{\emptyset, \{e\}, \{f\}, \{g\}, \{e, f\}, \{e, g\}, \{f, g\}, \{e, f, g\}\}$.

A tidbit of useful power set information: $|\mathcal{P}(S)| = 2^{|S|}$. Combined with a systematic listing of the subsets by size, knowing this can help you with power set problems that might seem baffling at first glance.

Example 110:

Problem: Let $E = \{\emptyset, \{\emptyset\}\}$. What is $\mathcal{P}(E)$?

Solution: Let's start by using what we just learned: $|\mathcal{P}(E)| = 2^{|E|} = 2^2 = 4$. Now we just have to identify those four subsets.

Two are easy: All sets have the empty set and itself as subsets. We just need the other two, which each must be of size 1, as we've covered size 0 and size 2. Each of these two subsets must contain one of the individual elements of E . We're done: $\mathcal{P}(E) = \{\emptyset, \{\emptyset\}, \{\{\emptyset\}\}, \{\emptyset, \{\emptyset\}\}\}$.

If you're wondering how to prove that $|\mathcal{P}(S)| = 2^{|S|}$, great! See the counting chapter for the proof.

6.3.4 Disjoint Sets

Sets that have no common elements are *disjoint*.

disjoint sets

Definition 34: Disjoint Sets

Sets S and T are *disjoint sets* when $S \cap T = \emptyset$.

There is no recognized symbol for disjoint sets.

Example 111:

Problem: Explain why the sets $\{i, k, m, o, q\}$ and $\{r, p, n, l, i\}$ are not disjoint.

Solution: By the definition of disjoint sets, for these sets to be disjoint the intersections of these sets must be the empty set. These sets have the element i in common; that is, $\{i, k, m, o, q\} \cap \{r, p, n, l, i\} = \{i\} \neq \emptyset$. Thus, these sets are not disjoint.

Example 112:

Problem: Consider the set $C = \{a, e\}$. How many subsets of the universe $\mathcal{U} = \{a, e, i, o, u\}$ are disjoint with C ?

Solution: We need to figure out how many subsets there are of \mathcal{U} that do not contain a or e . This is the same as asking how many subsets there are of $\{i, o, u\}$. We already know that answer, because that collection of subsets is the power set of $\{i, o, u\}$, and its cardinality is $2^{|\{i, o, u\}|} = 8$. There are 8 subsets of $\{a, e, i, o, u\}$ that are disjoint with C .

Example 112 is a good demonstration of how complex problems can be made much less so by looking at the problem from a different point of view. We could find the answer by listing all of the subsets of a five-element universe (all 32 of them) and counting those that do not contain a or e . A little thought

saved a lot of work. We will see this problem-solving technique frequently in the counting chapter. (And yes, this example is a counting problem!)

The idea of disjoint sets can be easily extended to more than a pair of sets by computing all of the pairwise intersections of the sets and verifying that all of those intersections are empty.

Example 113:

Problem: Are the sets $\{11, 15\}$, $\{19\}$, $\{10, 13, 14, 16\}$, and $\{12, 15, 18\}$ all disjoint from one another?

Solution: No, because there is one pair of sets (the first and the fourth) that share 15.

6.3.5 Partitions

We need to make a brief detour into the ideas of *generalized union* and *generalized intersection*, because we need generalized union to conveniently define the idea of a partition.

generalized union
generalized intersection

We know that the symbol \cup is used to represent the union of two sets. But what if we have a large group of sets and want to represent the unioning of all of them? Instead of creating a massive expression $(A \cup B \cup C \cup \dots)$, we can create a short-cut in the same way that we use Σ to represent a large summation. That is:

$$\bigcup_{i=1}^n S_i = S_1 \cup S_2 \cup \dots \cup S_n.$$

(`\bigcup\limits` in \LaTeX , or just `\bigcup` for a more compact version.) Similarly, we can use \bigcap (`\bigcap` in \LaTeX) to represent a lengthy intersection expression. End of detour!

Partitions of a set are closely related to the idea of disjoint sets, as the definition makes clear.

Definition 35: Partition

partition

Non-empty sets S_1, S_2, \dots, S_n form a *partition* of a base set B when $\bigcup_{i=1}^n S_i = B$ and $S_j \cap S_k = \emptyset$ for all j, k such that $1 \leq j, k \leq n$.

That is, a group of sets is a partition of B if (a) each set is non-empty, (b) each element of B is found in one of the sets, and (c) all of the sets of the group are disjoint from one another.

Example 114:

Problem: Together, are the set of odd integers (\mathbb{Z}^{odd}) and the set of even integers (\mathbb{Z}^{even}) a partition of \mathbb{Z} ?

Solution: Yes! All integers are either odd or even (yes, zero, too; it's even), and no integer is both. Because \mathbb{Z}^{odd} and \mathbb{Z}^{even} are not empty, every integer is either $\in \mathbb{Z}^{\text{odd}}$ or $\in \mathbb{Z}^{\text{even}}$ (not both), and $\mathbb{Z}^{\text{odd}} \cup \mathbb{Z}^{\text{even}} = \mathbb{Z}$, \mathbb{Z}^{odd} and \mathbb{Z}^{even} form a partition of \mathbb{Z} .

Let's revisit Example 113's sets from the perspective of a partition.

Example 115:

Problem: Collectively, are the sets $\{11, 15\}$, $\{19\}$, $\{10, 13, 14, 16\}$, and $\{12, 15, 18\}$ a partition of the set $B = \{i \mid 10 \leq i \leq 19, i \in \mathbb{Z}\}$?

Solution: (This is going to sound familiar ...) No, because there is one pair of sets (the first and the fourth) that share 15. For the group to be a partition, each element must be a member of exactly one set.

Let's drop the second occurrence of 15 and try it again.

Example 116:

Problem: Are the sets $\{11, 15\}$, $\{19\}$, $\{10, 13, 14, 16\}$, and $\{12, 18\}$ a partition of the set $B = \{i \mid 10 \leq i \leq 19, i \in \mathbb{Z}\}$?

Solution: Again, no, but for a different reason. For the collection of sets to be a partition, each value of the base set B must be included in one of the sets. $|B| = 10$ but the sum of the cardinalities of the partition sets is just 9. The element 17 is missing.

6.4 Set Identities

The connection between logic and sets extends to their collections of identities. In this section, we will see that these identities are very similar (which is good, as it means that they are easier to remember and to apply). Just as logical identities (see Section 1.5.2) are useful in proving the equivalence of logical expressions, set identities are useful in proving set equivalences. We can also ‘jump’ between set and logic notations to prove the set identities, should we feel the need.³

6.4.1 Common Set Identities

As we did with the logical identities back in Chapter 1, we have divided the set identities into separate tables by the set operators they use. Our collection of set identities isn’t nearly as extensive as is our lists of logical identities. That’s not to imply that set identities are less important; rather, we don’t have as many operators. We don’t have corresponding set operators for exclusive-OR, implication, and bimplication.⁴ We do have set difference and Cartesian Product, but the latter makes sets of ordered pairs, not of basic elements, with the result that it doesn’t ‘mingle’ well with the others.

Table 16: Some Set Identities using Union (\cup) and Intersection (\cap)

³And we will!

⁴Before those of you with set theory backgrounds — you know who you are — get too worked up about this claim, remember that this chapter didn’t cover symmetric difference, and we aren’t considering subset and proper subset to be operators. Still need to get lathered up? Please address all complaints to `/dev/null`.

(a)	$S \cap S = S$ $S \cup S = S$	Idempotency
(b)	$S \cap \emptyset = \emptyset$ $S \cup \mathcal{U} = \mathcal{U}$	Domination
(c)	$S \cap \mathcal{U} = S$ $S \cup \emptyset = S$	Identity
(d)	$S \cap T = T \cap S$ $S \cup T = T \cup S$	Commutativity
(e)	$S \cap (T \cap W) = (S \cap T) \cap W$ $S \cup (T \cup W) = (S \cup T) \cup W$	Associativity
(f)	$S \cap (T \cup W) = (S \cap T) \cup (S \cap W)$ $S \cup (T \cap W) = (S \cup T) \cap (S \cup W)$	Distributivity
(g)	$S \cap (S \cup T) = S$ $S \cup (S \cap T) = S$	Absorption Laws

Table 17: Some More Set Identities (adding Complement ($\bar{}$))

(a)	$\overline{\overline{S}} = S$	Dbl. Complement (Involution)
(b)	$S \cap \overline{S} = \emptyset$ $S \cup \overline{S} = \mathcal{U}$	Complement Laws
(c)	$\overline{S \cap T} = \overline{S} \cup \overline{T}$ $\overline{S \cup T} = \overline{S} \cap \overline{T}$	De Morgan's Laws

Table 18: Still More Set Identities (adding Difference ($-$))

(a)	$S - T = S \cap \overline{T}$	Definition of Difference
(b)	$S - T = S - (S \cap T)$	
(c)	$S - S = \emptyset$	
(d)	$S - \emptyset = S$	
(e)	$\emptyset - S = \emptyset$	
(f)	$S - \mathcal{U} = \emptyset$	
(g)	$\mathcal{U} - S = \overline{S}$	

6.4.2 Proving Set Expressions

You probably remember that, in Chapter 1, we demonstrated how to prove the truth of logical equivalences. At the time, we didn't know what a 'proof' was,⁵ but now we know that what we were doing was constructing straight-forward direct proofs. We will do the same sort of thing to prove set expressions.

Let's ease into these proofs by starting with an easy one. On rare occasions, we get lucky and are asked to prove something that's covered by a definition.

Example 117:

Problem: Prove or disprove: $A \subseteq \mathcal{P}(A)$, where A is a set.

Solution: Of course, we already know that a set's power set contains the set itself, but we can't hold our noses in the air, sniff derisively, and walk stiffly away, in search of a conjecture that is up to our standards. We need to prove it if we can, and we definitely can.

Proof (Direct): By definition, the power set of a set S is the set containing all of S 's subsets. Because every set is a subset of itself (according to the definition of subset), each set is an element of its own power set.

Therefore, $A \subseteq \mathcal{P}(A)$.

⁵Ah, those happier days of our youth . . .

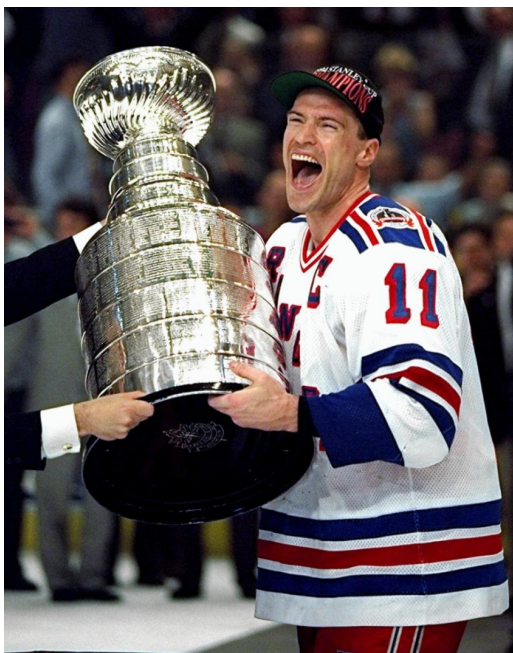


Figure 6.2: Mark Messier, Hall of Fame hockey player and now namesake of all inelegant proofs. (Credit: NY Daily News)

We hope you enjoyed Example 117; the rest of the examples will be much messier.⁶

Example 118:

Problem: Prove or disprove: If $A \subset B$ and $B \subseteq C$, then $A \subset C$.

Solution: As always, it's a good idea to check that the conjecture is worth trying to prove. A quick Venn diagram (see Figure 6.3) suggests that the conjecture holds.

To create a formal proof, we will make use of our old friend, logic. We

⁶Yes, this word is the entire justification for Figure 6.2. We should be ashamed, but aren't.

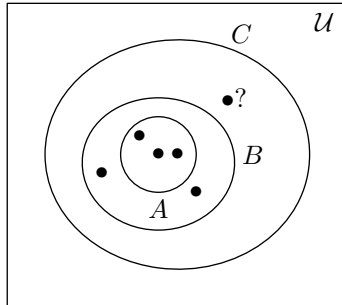


Figure 6.3: Yes, when $A \subset B$ and $B \subseteq C$, it seems that $A \subset C$ is true. (See Example 118.)

know how to express \subset and \subseteq logically, and we know lots of logical equivalences and several common rules of inference. We'll just put some of them to work. Here's the complete proof; a dissection follows.

Proof (Direct): Assume that the elements represented by d and e , below, are members of the same universe as are the elements of the sets A , B , and C .

(1)	$A \subset B$	[Given]
(2)	$B \subseteq C$	[Given]
(3)	$\forall d(d \in A \rightarrow d \in B) \wedge \exists e(e \notin A \wedge e \in B)$	[1, Def. of \subset]
(4)	$\forall d(d \in B \rightarrow d \in C)$	[2, Def. of \subseteq]
(5)	$\forall d[(d \in A \rightarrow d \in B) \wedge (d \in B \rightarrow d \in C)] \wedge \exists e(e \notin A \wedge e \in B)$	[3, 4, Conjunction]
(6)	$\forall d[(d \in A \rightarrow d \in B) \wedge (d \in B \rightarrow d \in C)] \wedge \exists e(e \notin A \wedge e \in C)$	[5, Modus Ponens]
(7)	$\forall d(d \in A \rightarrow d \in C) \wedge \exists e(e \notin A \wedge e \in C)$	[6, Hypo. Syll.]
(8)	$\therefore A \subset C$	[7, Def. of \subset]

Therefore, if $A \subset B$ and $B \subseteq C$, then $A \subset C$.

We have two useful pieces of given information, making a direct proof seem like a good choice of proof technique. As it does for logical proofs, a tabular format works well for set proofs, although, as you can see, the lines can get a little long.

We begin with the givens (lines 1 and 2), and express them logically (lines 3 and 4). We did not include “ $d, e \in \mathcal{U}$ ” on lines 3 through 7 in the interest of space, instead opting to pre-declare them on the first line of the proof block.

The proof gets more interesting in line 5, where we combine the universal quantifications from lines 3 and 4. We can do this because a universal quantification is just that: Universal. Both $d \in A \rightarrow d \in B$ and $d \in B \rightarrow d \in C$ are assumed true for all elements, which we can say in one statement rather than two. We aren’t required to combine them in order to complete the proof, but doing so allows us to show that such a merger is possible.

Line 6 differs from Line 5 by just one character: The last B is replaced by a C . Here’s how we can justify making that replacement. We know, from Line 5, that there’s an element in B ($\exists e(e \in B)$). We also know that if $e \in B$, then $e \in C$, by the universal quantification portion of Line 5. By Modus Ponens, we know that $e \in C$ follows.

At this point, the odds are good that you have at least one of these two questions: “Why do we have to make that one-letter change?”, and “ $e \in B$ is still assumed true, so why isn’t it still in the expression of Line 6?” One answer covers both questions. We know our destination: $A \subset C$. To get there, we need to be able to assume that $\exists e(e \notin A \wedge e \in C)$. $e \in B$ is indeed still assumed to be true, but it’s no longer useful to us, so we dropped it. Keeping it just adds unnecessary clutter to our already well-cluttered proof.

Line 7 neatly follows from Line 6, thanks to Hypothetical Syllogism on $d \in A \rightarrow d \in B$ and $d \in B \rightarrow d \in C$. The resulting expression is exactly the logical form of $A \subset C$. Making sure that the reader recognizes that completes the argument.

A common type of set proof requires showing that two set expressions S and T are equal; that is, that they describe the same set of elements. Here are two possible approaches to this type of conjecture:

1. Follow the definition of set equality: Prove that both $S \subseteq T$ and $T \subseteq S$ are true, using the logical expression for \subseteq .

2. Express one side in terms of set builder notation and logical operators, prove, and convert back to set notation.

The latter is usually less work, but the former is good to know, because it also shows how to prove subset (and proper subset) expressions.

To make the two approaches easier to compare and contrast, we will apply them both to the same set expression. This expression looks like it should be straight-forward to prove, and it is ... if you choose the right approach. We will start with the subset definition approach.

Example 119:

Problem: Prove or disprove: $S - S = \emptyset$.

Solution: Applying the definition of subset naturally leads to a two-part proof, one for each ‘direction.’

Proof (Direct): By the definition of set equality, if $S - S \subseteq \emptyset$ and $\emptyset \subseteq S - S$, then $S - S = \emptyset$. We will prove both parts. Throughout, assume that $y, z \in \mathcal{U}$.

Case 1: Consider $S - S \subseteq \emptyset$.

$$\begin{array}{ll}
 (1) & S - S \subseteq \emptyset = \forall z(z \in S - S \rightarrow z \in \emptyset) & [\text{Def. of } \subseteq] \\
 (2) & = \forall z(z \in S - S \rightarrow \mathbf{F}) & [\text{Nothing } \in \emptyset] \\
 (3) & = \forall z(z \notin S - S) & [\text{Law of False Cons. }] \\
 (4) & = \forall z(z \notin \{y \mid y \in S \wedge y \notin S\}) & [\text{Def. of Set Diff. }] \\
 (5) & = \forall z(z \notin \{y \mid \mathbf{F}\}) & [\text{Negation Laws }] \\
 (6) & = \forall z(z \notin \emptyset) & [\text{Meaning of } \emptyset] \\
 (7) & = \mathbf{T} & [\text{Still nothing } \in \emptyset!]
 \end{array}$$

Case 2: Consider $\emptyset \subseteq S - S$.

$$\begin{array}{ll}
 (1) & \emptyset \subseteq S - S = \forall z(z \in \emptyset \rightarrow z \in S - S) & [\text{Def. of } \subseteq] \\
 (2) & = \forall z(\mathbf{F} \rightarrow z \in S - S) & [\text{Nothing } \in \emptyset] \\
 (3) & = \forall z(\mathbf{T}) & [\text{Def. of } \rightarrow] \\
 (4) & = \mathbf{T} & [\text{Tautology }]
 \end{array}$$

Therefore, $S - S = \emptyset$.

Several of the steps of this proof rely on knowledge of the empty set. For example, line 2 of Case 1 follows from the realization that the empty set contains no elements, meaning that $z \in \emptyset$ must be false. A less clear example from Case 1 is the step from line 5 to line 6. $\{y \mid \mathbf{F}\}$ looks cryptic, but can be thought of as a trivial case for set builder notation. Elements are in the set ‘such that false.’ That is, the condition for membership can never be true, and as no element can pass the inclusion test, the set must be empty.

Case 2 is much less complex. The trickiest part is interpreting $\forall z(\mathbf{T})$, but doing so is easier if you think about truth tables and tautologies. In a truth table, we can see that a logical expression is a tautology when the last column contains only values of ‘true.’ Similarly, $\forall z(\mathbf{T})$ says that the condition is true for all elements of the universe. That’s as true as it gets. Note that we could have included the same step in Case 1 ($\dots \forall z(z \notin \emptyset) = \forall z(\mathbf{T}) = \mathbf{T}$), but doing so wouldn’t have clarified the argument.

If you didn’t get too consumed by the details of the first case of the proof of Example 119, you may have noticed something interesting about the first case: Hidden inside is a proof of the original conjecture! Example 119’s proof is fine, but it’s far longer than the ‘convert to logic’ alternative, which is the version hidden in the first case. Here’s that approach, as a stand-alone proof.

Example 120:

Problem: Prove or disprove: $S - S = \emptyset$.

Solution: As we’ve already created this approach as part of Example 119, we’ll get straight to the proof.

Proof (Direct): Assume that $y \in \mathcal{U}$.

$$\begin{array}{ll} (1) & S - S = \{y \mid y \in S \wedge y \notin S\} \quad [\text{Def. of Set Difference}] \\ (2) & = \{y \mid \mathbf{F}\} \quad [\text{Negation Laws}] \\ (3) & = \emptyset \quad [\text{Meaning of } \emptyset] \end{array}$$

Therefore, $S - S = \emptyset$.

Short and sweet! This is clearly the technique to use to prove this conjecture, but Example 119 was much more educational.

These set proof examples have introduced a few new equalities and equivalences that are handy to remember when writing set proofs. Table 19 has a more complete list.

Table 19: Foundational Set Equalities and Equivalences

(a)	$\forall z(\mathbf{T}) \equiv \mathbf{T}$	Tautology
(b)	$\forall z(\mathbf{F}) \equiv \mathbf{F}$	Contradiction
(c)	$\{z \mid \mathbf{T}\} = \mathcal{U}$	“Such that true” means all elements are in the set
(d)	$\{z \mid \mathbf{F}\} = \emptyset$	“Such that false” means no elements are in the set

A reminder: Proof-writing is often a slow process that involves a lot of ‘wasted’ work, work that never appears in the final version that you show the world. That can be frustrating, but remember that this happens to everyone who writes proofs, paints, repairs cars, breathes, etc. The more proofs you write, the better you will get at selecting an initial proof approach, and the faster your brain will dig up useful answers to all of the “OK, *now* what do I do?” questions.

6.5 Choosing a Set Representation

A set can be viewed as an unordered list without duplicate elements. This naturally leads computer programmers to think of arrays, linked lists, trees, and hash tables as possible data structures for set representations.

An alternative way to look at the problem is to consider the operations that need to be performed on sets. This is the subject of one of our favorite programming mantras: “Choose the representation that best supports the operations.” We are now very familiar with the typical set operations (and operation-like characteristics such as subset). The question: Do any of those classic linear data structures do a good job supporting set operations?

As this isn’t a data structures book, you won’t be subjected to an exhaustive analysis. Instead, we will give a quick answer: Not really. OK, we can be a little less quick. All of the data structures mentioned at the top of this section can be used, but they all have their own advantages and disadvantages, including operation efficiency (having an ordered representation makes some operations more efficient to execute) and space (references stored in linked list and tree nodes add storage overhead).

Example 121:

Java, starting with version 1.2, has offered a `Set` interface with none of the basic set operators ... at least not by the names we know them. For example, `addAll()` and `removeAll()` are essentially union and difference, respectively, while `contains()` can be used to construct intersection.

This interface is supported by classes such as `HashSet` and `TreeSet`, which, as their names suggest, use a hash table and a tree (as of this writing a Red-Black tree, a kind of balanced binary search tree) as their representations. With this arrangement, Java allows programmers to select a representation that best fits their program’s needs.

bit vector

A space-saving alternative representation for a set, one that also does a remarkably good job supporting set operators, is a *bit vector*. Essentially, a bit vector is just an array of bits. To use one as a set representation, we assign each member of the universe a position in the vector. If the bit is ‘0’, the set doesn’t contain the corresponding element; if it is ‘1’, the set does contain it.

Example 122:

Consider $\mathcal{U} = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. This means that we need a bit vector of nine bits in length. Let the left-most bit represent 1, the next 2,

etc. Using that mapping, here are some examples of sets and their corresponding bit vector representations.

Set	Representation
$\{3, 4\}$	001100000
$\{1, 3, 5, 7, 9\}$	101010101
\emptyset	000000000
\mathcal{U}	111111111

Are you concerned that, by using a bit vector, we are imposing an ordering on the elements of \mathcal{U} ? We aren't changing the definition of a set; the ordering of the set elements is still irrelevant. $\{3, 4\}$ and $\{4, 3\}$ are still the same set. Both sets just have the same representation when viewed as a bit vector, which, as they are the same set, makes a lot of sense. Thus, the concept and the representation need not possess exactly the same characteristics. What matters is how well the representation supports the concept and its operations.

The second advantage of using bit vectors to represent sets is that most set operations can be performed very efficiently. Computer central processing units (CPUs) are designed to perform a relatively small collection of basic operations very efficiently. Those operations almost always include bit-wise logical operations such as AND and OR. A 64-bit CPU, for example, can AND or OR together two 64-bit values, which means we can perform set intersection and union operations on sets from a 64-element universe with a single CPU instruction. 64 elements may not seem like a lot, but that's more than the quantities of upper- and lower-case letters in the modern Latin alphabet (26 each), and the quantity of Roman digits (10), combined. Table 20 shows how the four basic set operations can be performed on bit vectors.

Table 20: Set and Corresponding Bit-wise Logical Operators

Set Operators	Bit-wise Logical Operators
$A \cup B$	$A \vee B$
$A \cap B$	$A \wedge B$
\overline{A}	\overline{A}
$A - B (= A \cap \overline{B})$	$A \wedge \overline{B}$

The fact that $A - B = A \cap \overline{B}$ was introduced in Appendix A. If you are curious how to perform these bit-wise logical operators in C or Java programs, see Table 6 in Chapter 1.

Example 123:

Again let $U = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Also assume that $O = \{1, 3, 5, 7, 9\}$, $E = \{2, 4, 6, 8\}$, and $C = \{4, 6, 8, 9\}$. As bit vectors, based on the given element ordering of U , $O = 101010101$, $E = 010101010$, and $C = 000101011$.

The following tables demonstrate each of the four set operators. Union, intersection, and complement are straight-forward. Difference, because it is a compound operator, requires an additional step.

Union:

$$\begin{array}{r} O \\ \cup E \\ \hline U \end{array} \Rightarrow \begin{array}{r} \{1, 3, 5, 7, 9\} \\ \cup \\ \{2, 4, 6, 8\} \\ \hline \{1, 2, 3, 4, 5, 6, 7, 8, 9\} \end{array} \Rightarrow \begin{array}{r} 101010101 \\ \vee \\ 010101010 \\ \hline 111111111 \end{array}$$

Intersection:


$$\begin{array}{r} C \\ \cap E \\ \hline \{4, 6, 8\} \end{array} \Rightarrow \begin{array}{r} \{4, 6, 8, 9\} \\ \cap \\ \{2, 4, 6, 8\} \\ \hline \{4, 6, 8\} \end{array} \Rightarrow \begin{array}{r} 000101011 \\ \wedge \\ 010101010 \\ \hline 000101010 \end{array}$$

Complement:

$$\begin{array}{r} \overline{O} \\ E \end{array} \Rightarrow \begin{array}{r} \overline{\{1, 3, 5, 7, 9\}} \\ \{2, 4, 6, 8\} \end{array} \Rightarrow \begin{array}{r} \overline{101010101} \\ 010101010 \end{array}$$

Difference:

$$\begin{array}{r} E \\ - C \\ \hline \{2\} \end{array} \Rightarrow \begin{array}{r} E \\ \cap \overline{C} \\ \hline \{2\} \end{array} \Rightarrow \begin{array}{r} \{2, 4, 6, 8\} \\ \cap \\ \overline{\{4, 6, 8, 9\}} \\ \hline \{2\} \end{array} \Rightarrow \begin{array}{r} 010101010 \\ \wedge \\ 111010100 \\ \hline 010000000 \end{array}$$



Bit vectors seem to be a great representation; they have both space efficiency (just 1 bit of storage per element) and operation efficiency (operators correspond to CPU instructions). Unfortunately, if we look more closely, problems appear, including:

1. We need to keep track of the mapping between the bit positions and the elements of the sets,
2. Not all operators are bit-vector-friendly (e.g., Cartesian Product), and
3. Subsets of infinite sets cannot be represented.

Choosing a data representation means accepting tradeoffs. For example, if we use a linked list as a set representation, we can store the set elements directly (no mapping to bits needed) and we can store a subset of an infinite set, but storing the sets will require much more space (to maintain the list structure) and processing the operators will be considerably more expensive (the lists must be traversed). Good software developers consider such issues before settling on a representation ...and they (usually grudgingly!) switch to a new representation when one is needed.

Chapter 7

Matrices

The word “matrix” has a variety of definitions.¹ Not surprisingly, we are interested in a definition used in mathematics:

Definition 36: Matrix

A *matrix* is an n -dimensional collection of values, $n \in \mathbb{Z}^+$.

matrix

Figure 7.1 shows examples of small one-dimensional and two-dimensional matrices. We stop at two because comprehensible two-dimensional representations of three-dimensional matrices are hard to create. Happily, for this book, one- and two-dimensional matrices are all that we will need.

7.1 The Utility of Matrices

The ACM/IEEE Computer Society produces curriculum guidelines for Computer Science education, to help secondary and higher-education institutions ensure that they are graduating students that possess a common core of basic knowledge of the discipline. The section on Discrete Structures in the 2013 guidelines² does not mention the words ‘matrix’ or ‘matrices’ at all. So why, then, does this book devote a chapter to the topic?

¹Such as: The area that produces the cells that become a fingernail is called the *matrix*. Makes you think of the Matrix movie franchise in a new way, doesn’t it?

²ACM/IEEE-CS Joint Task Force on Computing Curricula, ACM Press and IEEE Computer Society Press, December 2013. DOI: <http://dx.doi.org/10.1145/2534860>

$$\left[\begin{array}{cccc} 2 & 1729 & 87539319 & \end{array} \right] \quad \left[\begin{array}{cccc} 9 & -2 & 0 & 5 \\ 18 & 1 & -6 & 7 \end{array} \right]$$

Figure 7.1: One- and Two-Dimensional Matrices of Integers

There are several reasons to provide the basics of matrices in a discrete structures book. For example:

- Chapter 8 covers a type of set known as a relation. There are multiple useful ways to represent relations, one of which is a matrix representation.
- Another representation of relations is a data structure known as a graph. Graphs, a generalization of tree data structures, are far more useful than that one application. Matrices are used to represent graphs within programs.
- Speaking of programs: If you are studying discrete structures, you have almost certainly written a program that used an array data structure. Many definitions of matrices define them in terms of arrays, which can be confusing because the two words are often assumed to be synonyms. An n -dimensional array and an n -dimensional matrix are essentially the same thing.
- In computer graphics, matrices are used for a variety of purposes, including 2D projections, affine transformations, and texturing. We will present one computer graphics example later in this chapter (see Section 7.5).
- Cryptography, the study of ways to encode messages such that only the intended recipient can easily decode them, relies heavily on matrices. The Advanced Encryption Standard (AES), for example, is based on a sequence of matrix operations.

In short, a good working knowledge of matrices and their common operations is essential to understand many topics in Computer Science, most immediately the concepts of relations and graphs covered later in this book.

$$\left[\begin{array}{ccc} 2 & 1729 & 87539319 \end{array} \right] \quad \left(\begin{array}{ccc} 1 & 3 & 5 \end{array} \right) \quad \left[\begin{array}{c} 1 \\ 3 \\ 6 \end{array} \right] \quad \left(\begin{array}{c} 0 \\ 1 \\ 4 \end{array} \right)$$

Figure 7.2: Examples of Row and Column Vectors with Brackets and Parentheses

7.2 Matrix Fundamentals

7.2.1 Matrix Notations and Sizes

Figure 7.1 demonstrates how one- and two-dimensional matrices are often drawn. The collections of values are bounded on the left and right with brackets (a.k.a. square brackets), with the tiny bracket ‘tips’ on the tops and bottoms of the symbols serving to limit the content vertically. Some people prefer to use parentheses (a.k.a. round brackets) instead of brackets.

One-dimensional matrices can be presented as *row matrices* (a.k.a. *row vectors*) or as *column matrices* (a.k.a. *column vectors*). The difference is more than just a matter of taste, as we will see later in this chapter. Either way, we still use brackets or parentheses to delimit their content, as shown in Figure 7.2.

Just as we labeled predicates, propositions and sets, we label matrices, usually with upper-case letters, as in:

$$T = \left[\begin{array}{ccc} 2 & 1729 & 87539319 \end{array} \right]$$

To reference the values within a one-dimensional matrix, we use the lower-case version of the matrix’s label and append a subscript, with the first value (left-most for row matrices or top-most for column matrices) indexed with one.³ Thus, in T , $t_1 = 2$, $t_2 = 1729$, and $t_3 = 87539319$.

In two-dimensional matrices, we need two subscripts. By convention, the first index is the row index, the second is the column index, and element m_{11} is in the upper-left corner. In our two-dimensional matrix from Figure 7.1:

³I know, I know. You’re a proud, card-carrying computer programmer, and know that, for reasons of efficiency, most programming languages use zero-based indexing for arrays and lists. Mathematics is just slightly older than computer science, so mathematicians got to define matrix indexing. To facilitate communication, everyone, even the proudest computer scientist, uses one-based indexing when discussing matrices.

$$M = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \end{matrix} & \begin{bmatrix} 9 & -2 & 0 & 5 \\ 18 & 1 & -6 & 7 \end{bmatrix} \end{matrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \end{bmatrix}$$

$m_{11} = 9$ and $m_{24} = 7$. Should a matrix have more than nine rows or columns, commas are added to separate the row and column values, as in $m_{32,15} = 7$.

We follow the same row–column ordering to define the size of a two–dimensional matrix. M has two rows and four columns, and so is a 2×4 (read: “two by four”) matrix.

What about the size of a one–dimensional matrix? We can’t only say that T , above, is a matrix of length three, because that alone doesn’t distinguish between a row vector of length three and a column vector of length three. The problem is solved by thinking of a vector as either a rather flat, or a rather thin, two–dimensional matrix. Thus, T , a row vector, is a 1×3 matrix. A column vector of n values is described as having a size of $n \times 1$. Once the type of vector has been established, we can use a simplified notation for element references. That is, t_1 references the same element as does t_{11} , t_2 references the same element as does t_{12} , etc. Because we already know that T is a row vector, the single-subscript notation is all that we need.

7.2.2 Basic Matrix Definitions (and One Operation)

We need to present a few definitions (one of which is really an operation) before we encounter the numeric matrix operations that will be useful later in the book. The first one is easy:

square matrix

Definition 37: Square Matrix

A *square matrix* is a two–dimensional matrix in which the number of rows equals the number of columns.

Square matrices are frequently used in computing. For that reason, you will see many definitions of matrix representations and operations that require operands to be square. Note that the next definition is not one of them!

$$\begin{array}{ccc}
 T = \begin{bmatrix} 2 & 1729 & 87539319 \end{bmatrix} & U = \begin{bmatrix} 2 \\ 1729 \\ 87539319 \end{bmatrix} & V = \begin{bmatrix} 12 & 23 \\ 34 & 45 \\ 56 & 67 \end{bmatrix} \\
 \\
 W = \begin{bmatrix} 12 & 23 \\ 34 & 45 \end{bmatrix} & X = \begin{bmatrix} 12 & 23 \\ 43 & 45 \\ 56 & 67 \end{bmatrix} & Y = \begin{bmatrix} 2 \\ 1729 \\ 87539319 \end{bmatrix}
 \end{array}$$

Figure 7.3: Six Integer Matrices for Example 124.

Definition 38: Matrix Equality

Matrices A and B are *equal* (denoted $A = B$) iff they share the same dimensions **and** all pairs of corresponding elements are equal.

*matrix equality***Example 124:**

Consider the matrices in Figure 7.3. T and U have the same content, but they have different sizes. Thus, $T \neq U$.

V and W match in all four of their corresponding elements ($v_{11} = w_{11}$, $v_{12} = w_{12}$, etc.) but their sizes are not the same, so $V \neq W$.

V and X have the same size (3×2) but not all corresponding pairs of values match ($v_{21} \neq w_{21}$, for example), so $V \neq X$.

There are two matrices that are equal: $U = Y$. Both are 3×1 column vectors, and all three corresponding pairs of values are equal. Sure, the two columns of values are formatted differently, but that is not a problem.

Note that only W in Figure 7.3 is an example of a square matrix.

Does Definition 38 allow for us to say that a matrix is equal to itself? For example, referring again to Figure 7.3, can we say that $W = W$? Yes! W

is the same size as itself, and all of its corresponding pairs of values match. Thus, $W = W$.

Time for the operation masquerading as a matrix definition.

transposition

Definition 39: Transposition

The transposition of an $m \times n$ matrix A is the $n \times m$ matrix A^T in which the rows of A become the columns of A^T .

Example 125:

Consider this matrix S :

$$S = \begin{bmatrix} 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \end{bmatrix}$$

To construct S^T , write the elements of the first row vertically. Do the same with the second row's values, writing them vertically just to the right of the column you formed from the first row's content. Do the same with the content of the third row. The result is the transpose of S :

$$S^T = \begin{bmatrix} 1 & 2 & 3 \\ 5 & 6 & 7 \\ 9 & 10 & 11 \\ 13 & 14 & 15 \end{bmatrix}$$

Example 126:

Can vectors be transposed? Yes! Refer back to matrices T and U in Figure 7.3. $U = T^T$ and $T = U^T$. Because $U = Y$, we can also say that $Y = T^T$ and $T = Y^T$.

A generalization of Example 126 is worth remembering: Given a matrix A , the transpose of the transpose of A equals A . In notation, $(A^T)^T = A$.

Example 127:

There's a well-known visualization of matrix transposition that is worth knowing because, once you learn it, you are unlikely to forget how to transpose a matrix.

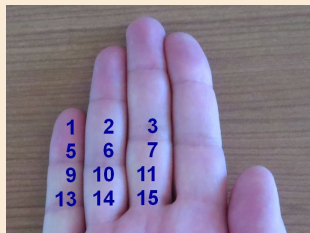
Imagine a right human hand, held up so that you see the back of it with its fingers pointing to the left. If you're lacking in imagination, consult this photograph:



More imagination: Think of the little finger as the first row of the matrix to be transposed. The (in this case ringless) ring finger represents the second row, etc.

$$\begin{bmatrix} 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \end{bmatrix} \implies \begin{array}{cccc} 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \end{array}$$

Now imagine the hand flipped over, with the palm facing you and the fingers pointing up, as in this picture:⁴



Now the little finger represents the first column of the matrix, with the ring finger the second column. That is, the flipped hand represents

the transposition of the matrix represented by the first view of the hand.

The next definition of this subsection makes use of all three of the preceding definitions.

matrix symmetry

Definition 40: Matrix Symmetry

Matrix A is *symmetric* iff $A = A^T$.

Definition 40 clearly uses matrix equality and matrix transposition, but it doesn't say anything about A needing to be a square matrix. We can infer this detail from the definition. We know that equal matrices need to have matching quantities of rows and columns, and that transposition swaps rows and columns. Thus, for A and A^T to be equal, A 's quantities of rows and columns must be the same. In other words, any symmetric matrix must be a square matrix.

Example 128:

Consider this matrix:

$$R = \begin{bmatrix} -2 & 4 & 1 & 9 \\ 4 & 13 & -6 & 21 \\ 1 & -6 & 0 & 4 \\ 9 & 21 & 4 & 10 \end{bmatrix}$$

Because R 's first row and first column contain the same values, as do R 's second row and second column, etc., R^T must equal R . By Definition 40, R is symmetric (and square!).

⁴Speaking of imagination: How did the owner of this hand not make a fortune as a hand model? Maybe this book will be the hand's big break. A hot-shot movie producer will be reading the book, trying to decide whether or not to option it as a major summer block-buster motion picture. ("Who should I cast as matrix S ?") Suddenly, she sees this hand. "Forget the movie! I need this hand for a proctology commercial! The silly number tattoos don't matter; after all, we only need the index finger."

Example 129:

In the matrix of capital letters Q , below, some letters are missing. In order for the completed matrix Q' to be symmetric, which letters must be placed in the labeled locations?

$$Q = \begin{bmatrix} w & y & [1] & y & m \\ y & a & b & p & [2] \\ f & b & x & o & z \\ [3] & p & o & [4] & a \\ m & c & z & a & s \end{bmatrix}$$

Before rushing to examine the missing values, check the size of Q . If Q is not square, it can never equal its transpose, meaning that the missing values are irrelevant. Happily, Q is square — 5×5 .

Location [1] is the third value of the first row. For Q to be symmetric, [1] must match the third value of the first column, which is f . We need to put an f in place of [1].

Do you prefer starting with columns? No problem! Location [2] is the second value of the fifth column. The second value of the fifth row is c , so we replace [2] with c to make the fifth column symmetric with the fifth row.

Let's use our indexing syntax to do [3]. [3] is located at q_{41} . To be symmetric, [3] must match q_{14} , which holds a y .

Finally, we have to replace [4]. It is located at q_{44} , which means that we have to replace [4] with ... itself! Because there's no letter at q_{44} , we can make Q' symmetric using any letter we wish to use; why not d ?

Here's the resulting matrix Q' :

$$Q' = \begin{bmatrix} w & y & f & y & m \\ y & a & b & p & c \\ f & b & x & o & z \\ y & p & o & d & a \\ m & c & z & a & s \end{bmatrix}$$

$Q' = (Q')^T$; it is symmetric.

Before the final symmetry example, one more matrix definition.

main diagonal

Definition 41: Main Diagonal

The *main diagonal* of an $n \times n$ matrix A consists of the elements a_{11} , a_{22} , \dots , a_{nn} .

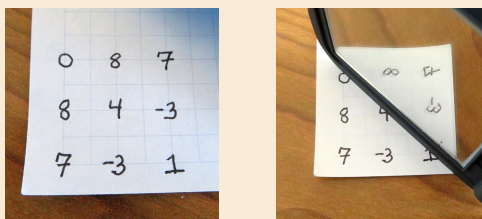
That is, the main diagonal runs from the upper-left corner to the lower-right corner of the matrix. Onto the example!

Example 130:

Did you enjoy the transposition-as-hand-flips idea? If so, you'll probably like this matrix symmetry visualization idea, too.

Imagine that you have a small rectangular mirror and a square matrix. Stand the mirror on one edge along the main diagonal so that you can see the values below the main diagonal reflected in the mirror. If the values in the matrix in the locations above the main diagonal match those in the reflection, the matrix is symmetric.

For instance, here's a 3×3 matrix that just happens to be a symmetric matrix, and the same matrix with a mirror positioned upon it as described above:



Yes, you have to realize that the infinity symbol is a reflected '8', that the accented 'w' is '-3', and the reflected '7' is ... well, whatever that symbol is, but apart from that, the reflected values match the existing values below the main diagonal.

Are you wondering about the values on the main diagonal? Don't worry about them! As we saw in Example 129, the main diagonal's values stay right where they are when the transposition is created. Because they only have to match themselves, main diagonal values cannot break symmetry.

7.3 Numeric Matrix Operations

Matrix transposition, covered in Section 7.2.2, is a matrix operation, but one that can be applied to any matrix, no matter the content. When the content of a matrix consists of real numbers, we can do a few more operations, the last of which is more than a bit of work to evaluate.

7.3.1 Matrix Addition and Subtraction

Matrix addition is a simple and commonly-used operation. If you have ever written a computer program that filled two arrays with data and added the content together, element by element, to fill a third array, you've performed matrix addition.

Definition 42: Matrix Addition

The sum of two numeric $n \times m$ matrices A and B is the $n \times m$ matrix C such that $c_{ij} = a_{ij} + b_{ij}$. (Also known as *Matrix Sum*.)

matrix addition

The definition of matrix subtraction is only slightly different.

Definition 43: Matrix Subtraction

The difference of two numeric $n \times m$ matrices A and B is the $n \times m$ matrix C such that $c_{ij} = a_{ij} - b_{ij}$. (Also known as *Matrix Difference*.)

matrix subtraction

Example 131:

Let:

$$O = \begin{bmatrix} 0 & 9 \\ -4 & 2 \\ 4 & 5 \\ -2 & 1 \end{bmatrix} \quad P = \begin{bmatrix} 4 & -8 \\ 1 & 8 \\ -4 & 2 \\ 0 & 1 \end{bmatrix}$$

O and P are both 4×2 in size, and contain integers. Thus, we can

add their corresponding pairs of values to compute the matrix addition $O + P$:

$$O + P = \begin{bmatrix} 0 + 4 & 9 + (-8) \\ -4 + 1 & 2 + 8 \\ 4 + (-4) & 5 + 2 \\ -2 + 0 & 1 + 1 \end{bmatrix} = \begin{bmatrix} 4 & 1 \\ -3 & 10 \\ 7 & 0 \\ -2 & 2 \end{bmatrix}$$

Perhaps not surprisingly, the sum $P + O$ is the same:

$$P + O = \begin{bmatrix} 4 + 0 & -8 + 9 \\ 1 + (-4) & 8 + 2 \\ -4 + 4 & 2 + 5 \\ 0 + (-2) & 1 + 1 \end{bmatrix} = \begin{bmatrix} 4 & 1 \\ -3 & 10 \\ 7 & 0 \\ -2 & 2 \end{bmatrix}$$

Example 131 suggests (but does not prove!) that matrix addition is commutative, which it is. Matrix addition is commutative because each element of the resulting matrix is formed from the addition of the corresponding elements of the operand matrices, and addition of real values is commutative.

Example 132:

Matrix addition is commutative, but what about matrix difference? The answer to that question depends on the answer to this question: Is basic subtraction commutative? That is, does $a - b = b - a$ for all values of a and b ?

Remember, to disprove a conjecture, we only need one counter-example. Let's try $a = o_{42} = 1$ and $b = p_{42} = 1$ from the matrices O and P in Example 131. $a - b = 1 - 1 = 0$ and $b - a = 1 - 1 = 0$. In this case $a - b = b - a$, which means that this is not a counter-example. But, this case is the specific situation in which $a = b$. We need a little more variety.

Let's try an $a \neq b$ example by using $a = o_{11} = 0$ and $b = p_{11} = 4$. $a - b = 0 - 4 = -4$, but $b - a = 4 - 0 = 4$. Because $a - b \neq b - a$, we have our counter-example. It follows that matrix difference is not commutative.

7.3.2 Scalar Multiplication

Having just covered matrix difference, we can reveal a minor secret: We did not need to cover it. Just as we can accomplish the subtraction of two real numbers by multiplying the second by -1 and adding ($a - b = a + (-1 \cdot b)$), we can perform matrix difference by performing a scalar multiplication on the second matrix and adding.

Before we can define scalar multiplication, we need to explain what a ‘scalar’ is.

Definition 44: Scalar

A scalar is a real number.

scalar

Yes, very exciting. Also somewhat superficial, but good enough for our needs.⁵

Definition 45: Scalar Multiplication

The multiplication of a scalar d and an $n \times m$ numeric matrix A , written dA , is the $n \times m$ matrix B such that $b_{ij} = d \cdot a_{ij}$.

scalar multiplication

In plain English: To perform a scalar multiplication, we multiply every element of the given matrix by the given scalar.

An alternate term for ‘scalar multiplication’ is ‘scalar product,’ but that term is more commonly used as an alternate name for ‘dot product,’ which is why we are not going to use the phrase ‘scalar product’ in this book.

Example 133:

Consider matrix O from Example 132, and the scalar 3:

⁵The term *scalar* has several meanings in math and science. For example, a scalar is, to mathematicians more generally, an element of any *field*, where a field is a set of values for which the operations of addition, subtraction, multiplication, and division are defined. For example, the set \mathbb{Z} can be called the field of integer values. Another example: In computer programming, ‘scalar’ is an older term for a variable.

$$3O = 3 \cdot O = 3 \cdot \begin{bmatrix} 0 & 9 \\ -4 & 2 \\ 4 & 5 \\ -2 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 27 \\ -12 & 6 \\ 12 & 15 \\ -6 & 3 \end{bmatrix}$$

Example 134:

In Section 7.3.1, we explained, but did not demonstrate, matrix difference. Here's how to do matrix difference with scalar multiplication, using matrices O and P from Example 132:

$$\begin{aligned} O - P &= O + (-1P) \\ &= \begin{bmatrix} 0 & 9 \\ -4 & 2 \\ 4 & 5 \\ -2 & 1 \end{bmatrix} + \left(-1 \cdot \begin{bmatrix} 4 & -8 \\ 1 & 8 \\ -4 & 2 \\ 0 & 1 \end{bmatrix} \right) \\ &= \begin{bmatrix} 0 & 9 \\ -4 & 2 \\ 4 & 5 \\ -2 & 1 \end{bmatrix} + \begin{bmatrix} -4 & 8 \\ -1 & -8 \\ 4 & -2 \\ 0 & -1 \end{bmatrix} \\ &= \begin{bmatrix} -4 & 17 \\ -5 & -6 \\ 8 & 3 \\ -2 & 0 \end{bmatrix} \end{aligned}$$

7.3.3 Matrix Multiplication (a.k.a. Matrix Product)

Matrix multiplication, the first time you see it, may appear to be someone's idea of a joke. It can be difficult to see, from the definition, how there can be a practical application of such a complex collection of seemingly arbitrary operations. So, let's start with a practical application.

Example 135:

Sports leagues often need to rank their teams for playoff seedings, or just bragging rights. The obvious starting point is each team's total number of victories, but how can we 'break' ties?

Consider the Letters League, with teams Alpha, Beta, Gamma, and Delta. Each team has played two games, with the following results. In the first pair of games, Alpha defeated Gamma and Beta defeated Delta. In the second pair, Alpha defeated Delta and Gamma defeated Beta. We can represent these results in a *domination matrix*, D . d_{rc} is '1' if team r defeated team c , and is '0' otherwise. D is a square matrix (4×4 in this case), with the rows and columns labeled identically:

$$D = \begin{array}{c} \\ \\ \\ \end{array} \begin{array}{c} \text{Alpha} \\ \text{Beta} \\ \text{Gamma} \\ \text{Delta} \end{array} \begin{array}{c} \text{Alpha} \\ \text{Beta} \\ \text{Gamma} \\ \text{Delta} \end{array} \begin{array}{c} \text{Wins} \\ \\ \\ \end{array}$$

$$= \begin{array}{c} \text{Alpha} \\ \text{Beta} \\ \text{Gamma} \\ \text{Delta} \end{array} \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 2 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Adding up the rows gives us the number of victories: Alpha has won two games, and Beta and Gamma one game each. (Adding the columns gives the number of losses.) Beta and Gamma are tied with one victory each. But are they equally talented? What additional information can we use to distinguish them?

Besides telling us direct wins and losses, the domination matrix can also tell us *transitive* results. That is, we know that Alpha defeated Gamma, and then Gamma defeated Beta. By transitivity, we could reason that Alpha is a better team than Beta, even though Alpha and Beta have not played directly against one another.

We can also represent these transitive results (which are called *two-step* results, with the direct results called *one-step*) using a similar matrix

which, for reasons of foreshadowing, we will name D^2 :

$$D^2 = \begin{array}{c} \text{Alpha} \\ \text{Beta} \\ \text{Gamma} \\ \text{Delta} \end{array} \begin{array}{c} \text{Alpha} \\ \text{Beta} \\ \text{Gamma} \\ \text{Delta} \end{array} \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \rightarrow \begin{array}{c} \text{Two-Step} \\ \text{Results} \end{array} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

The ‘1’ in the top row is the example we just used: Alpha is two-step dominant over Beta because Alpha defeated Gamma and Gamma defeated Beta. The other ‘1’ in D^2 comes from Gamma defeating Beta and Beta defeating Delta.

Adding the values across the rows of D^2 gives the numbers of two-step results. Adding the one-step totals to the two-step totals distinguishes Beta and Gamma: Gamma’s total is two, while Beta’s is still one:

$$\text{One-Step} + \text{Two-Step} = \begin{array}{c} \text{One-Step} \\ \text{Results} \end{array} \begin{bmatrix} 2 \\ 1 \\ 1 \\ 0 \end{bmatrix} + \begin{array}{c} \text{Two-Step} \\ \text{Results} \end{array} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{array}{c} \text{Totals} \\ \text{Alpha} \\ \text{Beta} \\ \text{Gamma} \\ \text{Delta} \end{array} \begin{bmatrix} 3 \\ 1 \\ 2 \\ 0 \end{bmatrix}$$

You may well be asking: “OK, great, but what does this have to do with *multiplying* matrices?” Filling D^2 with two-step results wasn’t too hard to do by inspection, because we have just four teams and four games played. Any thoughts as to an operation that we might be able to use to automate that process to handle more teams and more games? We will graciously give you one guess.

(We will tell you if your guess is correct, we promise. We will revisit this example later in this chapter, in Example 149.)

Matrix addition and scalar multiplication were straight-forward; matrix multiplication is not.

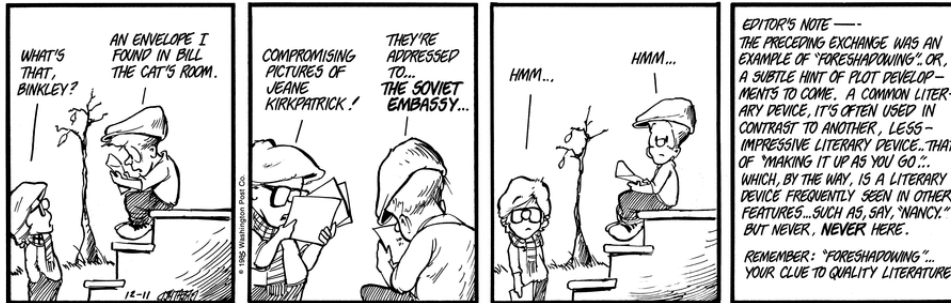


Figure 7.4: Foreshadowing — also your guide to quality textbook writing?
 Credit: Berkeley Breathed, “Bloom County”, December 11, 1985.

Definition 46: Matrix Multiplication

The product of an $m \times n$ matrix A and an $n \times o$ matrix B is an $m \times o$ matrix $C = A \cdot B = AB$ in which $c_{ij} = \sum_{k=1}^n (a_{ik} \cdot b_{kj})$. (Also known as *Matrix Product*.)

matrix multiplication

For a definition that is not even two lines long, we have a lot to explain.

First, take a close look at the sizes of the operand matrices A and B . They have a letter (n) in common, and that is not an accident. We can perform the product of matrices D and E , in that order (that is, with D on the left of E), only when D 's quantity of columns equals E 's quantity of rows.

Second, look at the size of the resulting matrix, C . Its quantity of rows (m) matches the quantity of rows of A and its quantity of columns (o) matches the quantity of columns of B . Again, this is not an accident.

Putting these two size observations together gives us the two preliminary steps of performing a matrix multiplication:

1. Verify that the quantity of columns of the first/left matrix equals the quantity of rows of the second/right matrix. If they do not match, a matrix multiplication cannot be performed on the matrices in this order.
2. The size of the resulting matrix will have the same quantity of rows as does the first/left matrix, and the same quantity of columns as does the second/right matrix.

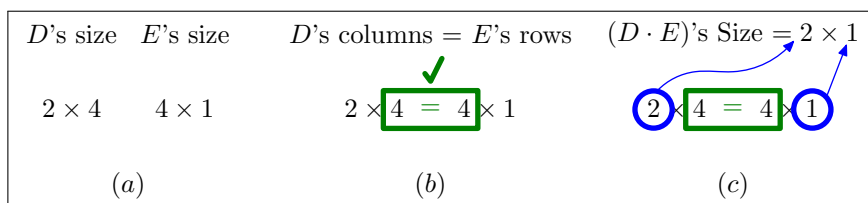


Figure 7.5: Checking Matrix Sizes before Matrix Multiplication

When written out, the steps are pretty dry. We can show these steps visually instead, as Example 136 and Figure 7.5 demonstrate.

Example 136:

Let D and E be the following matrices:

$$D = \begin{bmatrix} 3 & 5 & -1 & 0 \\ 0 & 2 & 3 & -2 \end{bmatrix} \quad E = \begin{bmatrix} 1 \\ 0 \\ -4 \\ 2 \end{bmatrix}$$

D 's size is 2×4 , and E 's is 4×1 . By the definition of matrix multiplication, in order to perform $D \cdot E$, the number of columns of D (4) must equal the number of rows of E (4). The remaining two values, the rows of D (2) and the columns of E (1) give us the size of their product: 2×1 .

Figure 7.5 shows a visualization that can help you remember the needed relationships. Begin, as shown in part (a), by writing down the sizes of the matrices D and E , side by side, with the size of the first/left matrix on the left. This places the number of columns of the left matrix next to the number of rows of the right matrix, making it easy to remember that this is the pair of values that must be equal (part (b)). The remaining pair of values provide the size of the resulting matrix, as shown in part (c).

Determining the that matrix product can be computed, and how large the resulting matrix will be, is most of the text of the Definition 46. Unfortunately, the hard part remains: Computing the product. The notation is short and

sweet in the definition (“ $c_{ij} = \sum_{k=1}^n (a_{ik} \cdot b_{kj})$ ”), but what does that expression require us to do?

Start with c_{ij} . The subscripts tell us the location within the result matrix whose value we are trying to compute: row i , column j . The summation $\left(\sum_{k=1}^n\right)$ has the other two variables, n and k . We already know what n represents: The quantity of columns of the first/left matrix, A , as well as the quantity of rows of the second/right matrix, B . k starts at 1 and is incremented through n . For each value of k , we find the values at locations a_{ik} and b_{kj} and multiply them. Adding up those n products produces the value to be placed at c_{ij} . Yes, computing a matrix product can require a fair amount of arithmetic, but not difficult arithmetic, merely tedious and error-prone arithmetic.⁶

Example 137:

Let’s finish what we started in Example 136. Here are D and E again, with place-holders for the values of the resulting $D \cdot E$ matrix, which we will call F :

$$D = \begin{bmatrix} 3 & 5 & -1 & 0 \\ 0 & 2 & 3 & -2 \end{bmatrix} \quad E = \begin{bmatrix} 1 \\ 0 \\ -4 \\ 2 \end{bmatrix} \quad D \cdot E = F = \begin{bmatrix} \square \\ \square \end{bmatrix}$$

Let’s start with f_{11} . With $i = j = 1$, and D ’s quantity of columns being 4, the summation becomes $f_{11} = \sum_{k=1}^4 (d_{1k} \cdot e_{k1})$. Expanding the

⁶We cannot understand how ‘tedious’ and ‘error-prone’ are not wildly popular names for children. They aren’t even gender-specific! Yeah, it might be hard to know which name to give to your first child, but you can always swap names later.

summation makes our task plain:

$$\begin{aligned}
 f_{11} &= \sum_{k=1}^4 (d_{1k} \cdot e_{k1}) \\
 &= (d_{11} \cdot e_{11}) + (d_{12} \cdot e_{21}) + (d_{13} \cdot e_{31}) + (d_{14} \cdot e_{41}) \\
 &= (3 \cdot 1) + (5 \cdot 0) + (-1 \cdot -4) + (0 \cdot 2) \\
 &= 3 + 0 + 4 + 0 \\
 &= 7
 \end{aligned}$$

Notice that what we are doing is ‘walking’, left to right, across the first row of D while simultaneously ‘sliding’, top to bottom, down the first column of E . This is always what you will need to do to compute the value at row i and column j of the result matrix: Walk across the i -th row of the first/left matrix, and slide down the j -th column of the second/right matrix.

All that remains is for us to repeat this for the other location, f_{21} :

$$\begin{aligned}
 f_{21} &= \sum_{k=1}^4 (d_{2k} \cdot e_{k1}) \\
 &= (d_{21} \cdot e_{11}) + (d_{22} \cdot e_{21}) + (d_{23} \cdot e_{31}) + (d_{24} \cdot e_{41}) \\
 &= (0 \cdot 1) + (2 \cdot 0) + (3 \cdot -4) + (-2 \cdot 2) \\
 &= 0 + 0 + (-12) + (-4) \\
 &= -16
 \end{aligned}$$

And so:

$$\begin{bmatrix} 3 & 5 & -1 & 0 \\ 0 & 2 & 3 & -2 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ -4 \\ 2 \end{bmatrix} = \begin{bmatrix} 7 \\ -16 \end{bmatrix}$$

Before we leave D and E , a final question: Can we compute the matrix product $E \cdot D$? E is 4×1 in size, but D has 2 rows. $1 \neq 2$, and so $E \cdot D$ cannot be computed. It follows that, in general, matrix multiplication is not commutative.

Example 138:

Are you worried that you will have trouble remembering to walk across the rows of the first matrix and slide down the columns of the second, rather than the reverse? Perhaps you're still a bit worried about checking the matrix sizes correctly? Positioning the operand matrices in an unusual way can help with both concerns.

Consider the matrices G and H and the eventual matrix product J , positioned as we've been positioning matrices:⁷

$$G = \begin{bmatrix} 3 & 2 \\ 0 & -2 \\ 1 & 0 \end{bmatrix} \quad H = \begin{bmatrix} 1 & 3 \\ 3 & 4 \end{bmatrix} \quad G \cdot H = J = \begin{bmatrix} \square & \square \\ \square & \square \\ \square & \square \end{bmatrix}$$

By moving H up, and shifting J to be next to G and beneath H , we produce:

$$H = \begin{bmatrix} 1 & 3 \\ 3 & 4 \end{bmatrix}$$

$$G = \begin{bmatrix} 3 & 2 \\ 0 & -2 \\ 1 & 0 \end{bmatrix} \quad \begin{bmatrix} \square & \square \\ \square & \square \\ \square & \square \end{bmatrix} = J = G \cdot H$$

This arrangement shows that J fits perfectly next to G and beneath H because it has the same quantity of rows as does G , and the same quantity of columns as does H — exactly our sizing relationships. Also notice that the rows of G and the columns of H , if you imagine extending them, intersect at the boxes within J . If you select a particular box from J , the row of G and the column of H that intersect at that box are the row/column that we will walk-across/slide-down to produce the value that belongs in that box. Be aware that we have to check separately that the quantity of columns of G equals the number of rows of H .

What's the content of J ? We encourage you to work that out yourself!

⁷“Wait; G , H , ... J ? What have you got against I ?” Calm down; we like I just fine. A special family of matrices has the name I , so using it here to name a different matrix could be confusing. Consumed by curiosity about I ? Jump ahead to Section 7.4.3.

7.4 Matrix Powers and the Identity Matrix

7.4.1 Matrix Powers

Consider multiplying a square matrix by another square matrix. The only way that this can be done is if both operand matrices are the same size (because the quantity of rows of the first/left square matrix must equal the quantity of columns of the second/right square matrix). The resulting matrix will be of the same size as the operand matrices.

Let's take this a bit further. If we want to multiply a matrix by itself, the matrix must be square, and the resulting matrix will be a square matrix of the same size. We can multiply the result by the original again, and repeat this as many times as we wish. It's similar to multiplying a real number by itself multiple times to raise the number to an integer power. For example, $2 \cdot 2 = 2^2 = 4$, $(2 \cdot 2) \cdot 2 = 2^2 \cdot 2 = 2^3 = 8$, etc. Doing repeated matrix multiplication with square numeric matrices forms *matrix powers* of the matrix.

matrix power

Definition 47: Matrix Power

The n^{th} matrix power of an $m \times m$ numeric matrix A , denoted A^n , is the result of $n - 1$ successive matrix products of A .

Example 139:

What is K^2 , if $K = \begin{bmatrix} 0 & 1 \\ 1 & 2 \end{bmatrix}$?

Because we already know how to perform matrix multiplication, computing K^2 is straight-forward.

$$\begin{aligned}
 K^2 &= K \cdot K \\
 &= \begin{bmatrix} 0 & 1 \\ 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 \\ 1 & 2 \end{bmatrix} \\
 &= \begin{bmatrix} 1 & 2 \\ 2 & 5 \end{bmatrix}
 \end{aligned}$$

Example 140:

K^2 wasn't hard, because it didn't matter which copy of K was on the left — $K = K$! But to create K^3 , do we compute $(K \cdot K) \cdot K$ or $K \cdot (K \cdot K)$?

The answer is: It doesn't matter! From Example 139, we know what K^2 contains. Let's compute both $K^2 \cdot K$ and $K \cdot K^2$:

$$\begin{aligned}
 K^2 \cdot K &= \begin{bmatrix} 1 & 2 \\ 2 & 5 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 2 & 5 \\ 5 & 12 \end{bmatrix} \\
 K \cdot K^2 &= \begin{bmatrix} 0 & 1 \\ 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 \\ 2 & 5 \end{bmatrix} = \begin{bmatrix} 2 & 5 \\ 5 & 12 \end{bmatrix}
 \end{aligned}$$

That might seem to be a little ... suspicious. Sure, this worked out, but K is symmetric; maybe that's a special case. We could try it again with a non-symmetric matrix, but perhaps that would be a matrix that we cooked up to make both orderings work. To convince people that the ordering of the matrix products really does not matter, what we need is more generality, such as we would use in a proof.

Example 141:

Problem: Prove that A^3 , the third matrix power of an 2×2 numeric matrix A , is equal to both $A^2 \cdot A$ and $A \cdot A^2$.

Solution: Let's get something out of the way immediately: We demonstrated, in Section 7.3.3, that matrix multiplication is not generally com-

mutative. However, our current problem is not a general situation; rather, it is specific to matrix powers, in which there is only one given matrix. Further, Example 140 gives us some reason to believe that matrix multiplication might actually be commutative for matrix powers. Given that, attempting a proof is worth the trouble.

We know that the resulting matrix A^3 will be a 2×2 matrix. What we need to show is that the four values that comprise A^3 are the same for both $A^2 \cdot A$ and $A \cdot A^2$. Because the matrices we need to compute are all 2×2 , and because the power is just 3, we can simply multiply all four values for each of the two products and verify that they are the same expressions. Hard? Not really. Tedious and error-prone? Definitely. Good thing we're going to do the dirty work for you!

Conjecture: If A is a 2×2 numeric matrix, then $A^3 = A^2 \cdot A = A \cdot A^2$.

Proof (Direct): We are given that A is a 2×2 numeric matrix.

Let $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$, where $a, b, c, d \in \mathbb{R}$.

$$\begin{aligned} A^2 &= \begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{bmatrix} a & b \\ c & d \end{bmatrix} \\ &= \begin{bmatrix} a^2 + bc & ab + bd \\ ac + cd & bc + d^2 \end{bmatrix} \end{aligned}$$

$$\begin{aligned} A^2 \cdot A &= \begin{bmatrix} a^2 + bc & ab + bd \\ ac + cd & bc + d^2 \end{bmatrix} \cdot \begin{bmatrix} a & b \\ c & d \end{bmatrix} \\ &= \begin{bmatrix} (a^3 + abc) + (abc + bcd) & (a^2b + b^2c) + (abd + bd^2) \\ (a^2c + acd) + (bc^2 + cd^2) & (abc + bcd) + (bcd + d^3) \end{bmatrix} \end{aligned}$$

$$\begin{aligned} A \cdot A^2 &= \begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{bmatrix} a^2 + bc & ab + bd \\ ac + cd & bc + d^2 \end{bmatrix} \\ &= \begin{bmatrix} (a^3 + abc) + (abc + bcd) & (a^2b + abd) + (b^2c + bd^2) \\ (a^2c + bc^2) + (acd + cd^2) & (abc + bcd) + (bcd + d^3) \end{bmatrix} \end{aligned}$$

The expressions in all four locations of both results are equivalent (due to commutativity of addition of real numbers).

Therefore, $A^3 = A^2 \cdot A = A \cdot A^2$.

A note about the added parentheses: We added them to the result matrices to organize the expressions a bit; they aren't mathematically necessary.

In the proof, we explained what we were doing in terms of commutativity.

We can also look at it in terms of associativity: $A^3 = AAA = (AA)A = A(AA)$. Associativity of matrix multiplication holds for any three numeric matrices A (of size $i \times j$), $B(j \times k)$, and $C(k \times l)$, not just for matrix powers: $A(BC) = (AB)C$. For that matter, matrix multiplication is also distributive over matrix addition, when the sizes are appropriate: $D(E + F) = DE + DF$.

Are you wondering how the proof can be generalized to A^4 , to A^5 , and eventually to A^m , where m is any positive integer power? We can do that by applying the definition of matrix multiplication more directly. We can also do it inductively ... but we haven't covered inductive proofs yet. We soon will; stay tuned!

7.4.2 The Cost of Matrix Multiplication

How much effort is required to multiply two matrices? That's not an easy question to answer. Complicating factors include whether or not code is being parallelized, how memory is being managed, and which matrix multiplication algorithm is used.⁸ We can ignore those complications and still learn something useful about how to multiply matrices using the definition's approach.

When analyzing the execution cost of an algorithm, individual operations are typically ignored in favor of an approximate category result. For example, sequentially searching a list is a linear operation, meaning that the work required is described as a linear function of n , the quantity of items in the list, rather than other function of n (e.g., logarithmic or quadratic). Different implementations of sequential search can have different linear functions that describe the number of operations performed, such as $3n + 2$ or $5n + 6$. Because those are linear functions, we say that sequential search belongs to the "linear" category of algorithm efficiencies.

In this section, we will be more detailed. We know that matrix multiplications require many real-number multiplications and additions. But how many? Let's find out.

The Cost of Multiplying Two Matrices

We defined two matrices D and E in Example 136:

⁸Yes, there are algorithms other than the one described by the matrix multiplication definition, all created to require less effort. Two examples: The definition's 'naive' approach can be parallelized. Strassen's algorithm exchanges multiplications for additions, but an advantage is gained only for matrices with hundreds of rows/columns.

$$D = \begin{bmatrix} 3 & 5 & -1 & 0 \\ 0 & 2 & 3 & -2 \end{bmatrix} \quad E = \begin{bmatrix} 1 \\ 0 \\ -4 \\ 2 \end{bmatrix}$$

Because the size of $F = D \cdot E$ is 2×1 , we need to compute two intermediate results, f_{11} and f_{21} . But, as we perform the same operations to compute each of them (albeit on different groups of values), we can figure the numbers of multiplications and additions for any one element f_{ij} and double it to get the total for the entire matrix product.

Consider f_{21} . Expanding the definition's summation expression, as we did in Example 137, shows us the required quantities of operations:

$$f_{21} = \sum_{k=1}^4 (d_{2k} \cdot e_{k1}) = (d_{21} \cdot e_{11}) + (d_{22} \cdot e_{21}) + (d_{23} \cdot e_{31}) + (d_{24} \cdot e_{41})$$

Computing f_{21} requires four multiplications and three additions. Thus, computing the content of F requires eight multiplications and six additions. Hardly a lot of either, but of course the numbers will grow as sizes of the matrices grow. We need to generalize our observations so that we can determine the work required for any pair of matrices.

To make what follows easier to understand, let's re-introduce the variables we used in the matrix product definition. Assume that we are computing $C = A \cdot B$. Let A be $m \times n$ and let B be $n \times o$, which means C is $m \times o$.

Both the quantities of multiplications and additions are determined by n , which is both the quantity of columns of A , and the number of rows of B . Specifically, for each element c_{ij} , we perform n multiplications of elements of A and B , and $n - 1$ additions to sum those products. There are $m \cdot o$ values in C to compute. Thus, there are mno multiplications and $m(n - 1)o$ additions performed in the computation of C .

Example 142:

Question: How many total multiplications and additions are required to compute AB using the definition when A is 10×12 and B is 12×6 ?

Answer: Using our m , n , and o variables, $m = 10$, $n = 12$, and $o = 6$.

There are $mno = 10 \cdot 12 \cdot 6 = 720$ multiplications and $m(n-1)o = 10 \cdot 11 \cdot 6 = 660$ additions, for a total of $720 + 660 = 1380$ operations.

Are you wondering how costly multiplications and additions of real numbers are to perform? There's no easy answer to that question, because different CPUs (Central Processing Units) implement addition and multiplication operations differently. Very roughly, floating-point multiplications are twice as expensive as are floating-point additions and subtractions.

The Cost of Multiplying Three Matrices

At the end of Section 7.4.1, we learned that matrix multiplication is associative; that is, $ABC = (AB)C = A(BC)$. This means that we can compute the matrix product ABC either by computing AB and then multiplying (on the right) by C , or by computing BC and multiplying (on the left) by A . At first glance, you might believe that the choice doesn't matter — both give you the correct answer. But do they both require the same amount of effort to produce that answer? Let's find out.

Example 143:

We will again assume, as we did in Example 142, that A is 10×12 and B is 12×6 . Further, assume that our new third matrix, C , is 6×8 .

From Example 142, we know what it costs to compute the 10×6 matrix AB . Computing the 10×8 matrix $(AB)C$ requires an additional $10 \cdot 6 \cdot 8 = 480$ multiplications and $10 \cdot 5 \cdot 8 = 400$ additions, which is 880 operations. Add in the 1380 operations needed to compute AB , and the grand total for $(AB)C$ is 2260 operations.

We need to deal with $A(BC)$ from scratch. BC requires $12 \cdot 6 \cdot 8 = 576$ multiplications and $12 \cdot 5 \cdot 8 = 480$ additions. $A(BC)$ requires $10 \cdot 12 \cdot 8 = 960$ multiplications and $10 \cdot 11 \cdot 8 = 880$ additions. The grand total is 2896.

The difference is a (probably) surprising 636 (28%) more operations to compute $A(BC)$ than are needed to compute $(AB)C$. Keep in mind that this result is for matrices of these three sizes. The math will work out differently for other matrix sizes.

The take-away message: If you need to compute a sequence of matrix multiplications just once, the associativity you choose is likely to make a

difference, but probably not enough of a difference to be a concern. However, if you need to perform the same product repeatedly (for example, because the contents of the matrices change), figuring out the most efficient product sequence is likely to be worth the effort.⁹

Example 144:

Huge matrices are not uncommon. The SuiteSpace Matrix Collection,¹⁰ as of this writing, includes a dataset of U.S. patent citations covering just utility patents granted during the 37 years 1963 through 1999. The matrix representing it is $3,774,768 \times 3,774,768$. Using the matrix multiplication definition's approach, computing its second matrix power would require 53,786,191,549,544,312,832 multiplications and a mere 53,786,177,300,670,859,008 additions, for a total of more than 10^{20} operations.

7.4.3 Multiplicative Identity and the Identity Matrix

You might remember learning about additive and multiplicative identities in other classes. A value a is the *additive identity* of a set of values if adding a to another value of the set does not change the value. That is, $a + x = x$. We hope you know that $a = 0$ is the additive identity for the set of real numbers. Similarly, a value m is the *multiplicative identity* of a set when $m \cdot x = x$. Considering the set of reals again, its multiplicative identity is $m = 1$.

Because addition and multiplication are defined for matrices, it makes sense for us to think about additive and multiplicative identities for numeric matrices. The $m \times n$ additive identity matrix only contains zeroes. Of course, we have to choose m and n to allow the matrix addition to be performed. Practically, additive identity matrices are not very interesting. Multiplicative identity matrices, on the other hand, are rather useful.

A multiplicative identity matrix (always called I) must be sized so that the matrix multiplication with another matrix A , $A \cdot I$, is possible, and must have content such that the result equals A .

⁹Fun fact: The number of ways to associate an n matrix multiplication sequence is the $(n - 1)^{\text{st}}$ Catalan number. For example, there are $C_3 = 5$ ways to associate a multiplication sequence of $n = 4$ matrices: $W(X(YZ))$, $W((XY)Z)$, $(W(XY))Z$, $((WX)Y)Z$, and $(WX)(YZ)$.

¹⁰Look for "SNAP/cit-Patents" at <https://sparse.tamu.edu/>

Let's consider size first. If A is $m \times n$ in size, to compute $A \cdot I$, I must have n rows. For the result to equal A , I must also have n columns. Thus, I is a square matrix of size $n \times n$. Notationally, we represent this matrix as I_n .

But what about $I \cdot A$? We want that to equal A , too. No problem; if we want I on the left of A , we need to use I_m to make the matrix multiplication work. That is: When the size of A is $m \times n$, multiplying on the left of A with I_m , or multiplying on the right with I_n , will make the sizes work out correctly.

Now for the content: What must I_m and I_n contain so that the products $I_m \cdot A$ and $A \cdot I_n$ both produce the original matrix A 's content? We can figure this out for ourselves by thinking about how a matrix product is produced. Let A be 2×2 , and consider $A \cdot I_2 = A$. Let's expand the matrices and stack them to make the products easier to see:

$$I_2 = \begin{bmatrix} i_{11} & i_{12} \\ i_{21} & i_{22} \end{bmatrix}$$

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} a'_{11} & a'_{12} \\ a'_{21} & a'_{22} \end{bmatrix} = A \cdot I_2 = A$$

Consider just the upper-left value of the product (a'_{11}). From the definition of matrix multiplication, we know that $a'_{11} = (a_{11} \cdot i_{11}) + (a_{12} \cdot i_{21})$. To end up with $a'_{11} = a_{11}$, We need to eliminate all of the other values on the right-hand side. Using multiplicative identity, i_{11} must be 1. To eliminate a_{12} , we need to multiply by zero, so $i_{21} = 0$.¹¹

To complete the first row, a'_{12} must equal $(a_{11} \cdot i_{12}) + (a_{12} \cdot i_{22})$. By similar reasoning, $i_{12} = 0$ and $i_{22} = 1$, giving us $I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. We need to retain the values of the second row of A in exactly the same way. Happily, this content for I_2 will do that job, too. You can check for yourself that this content also works for $I_2 \cdot A$, as well as for other sizes of A that have more rows of A (when computing $A \cdot I_2$) or more columns of A (for $I_2 \cdot A$).

¹¹The property $a \cdot 0 = 0 \cdot a = 0$ has at least two names, both of them quite dull: The Multiplication Property of Zero, and the Zero Property of Multiplication. We propose that it be renamed "Multiplicative Oblivion", which is significantly more exciting. "Multiplicative Domination" would work, too, but asking logical equivalences to share "Domination" with multiplication doesn't feel right.



Figure 7.6: If only Merlin had thought to consign Morgana to Mathematical Oblivion instead. Credits: John Boorman’s “Excalibur”; <https://memegenerator.net>.

We can make I any size we need. In general:

$$I_n = \begin{matrix} & \begin{matrix} 1 & 2 & \dots & n \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ \vdots \\ n \end{matrix} & \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix} \end{matrix}$$

Having provided so much explanation, the actual definition is old news:

Definition 48: Identity Matrices

Identity Matrices (denoted I_n) are $n \times n$ matrices populated with 1 along the main diagonal and 0 elsewhere.

identity matrices

Time to talk about the utility of the identity matrix, starting with matrix powers. We now know that $A^3 = A \cdot A \cdot A$, and $A^2 = A \cdot A$. It follows that $A^1 = A$. But what, if anything, does A^0 represent? Right: I !

Here are two reasons that this makes sense. First, when r is a real number, we know that $r^1 = r$ and $r^0 = 1$. $A^1 = A$ and $A^0 = I$ parallels these basic

powers of real numbers. Second, think about writing a few lines of code to compute a running product. Traditionally, we would declare a variable to hold the product, and would use a loop to multiply values into this variable. But what should be the initial value of this variable? It can't be zero, because of "Mathematical Oblivion",¹² but one works nicely. The first time through the loop, we multiply the variable's value of one by the first value that forms the running product, and the result is that value. With that starting point, multiplying in the rest of the values will work as desired.¹³ Similarly, when needing to compute a running product of matrices, we can initialize our running product matrix to hold I and multiply the first matrix from the collection into it.

Why would we need to have a running product of a collection of matrices? Computing a matrix power (see Section 7.4.1) is one reason. A second appears when performing object transformations in computer graphics, as explained in Section 7.5.

7.5 Matrix Operations in Orthographic Projections

7.5.1 Our Problem: Design a Shed . . . or Maybe a Dog House

Imagine that you want to build, in the back left corner of your yard, a storage shed that is 12 feet wide and 12 feet deep, with walls 9 feet high and a gable roof that rises to 12 feet at the ridgeline. To help yourself visualize the shed, you sketch out a diagram, with the origin at the corner of your yard, as shown in Figure 7.7 (a).

This diagram is a two-dimensional representation of a three-dimensional object. In drafting, this kind of representation is known as an *orthographic projection*.¹⁴

Having drawn your shed to show three sides, next you decide to draw only the front view of the shed; that is, how the front of the shed would look if you stood in the middle of the road, directly in front of the shed, and looked

¹²You read the previous footnote, didn't you?

¹³Yes, we could initialize the product variable to the first value of our collection and start the 'product-ing' with the second value, but that wouldn't help us make our point, and nothing is more important than whatever point we are currently trying to make . . . whether or not we can remember what it was ten seconds from now.

¹⁴Specifically, an axonometric orthographic projection. We're pretty sure. It sounds impressive, anyway. We are confident that it is not a *perspective projection*, because there are no vanishing points.

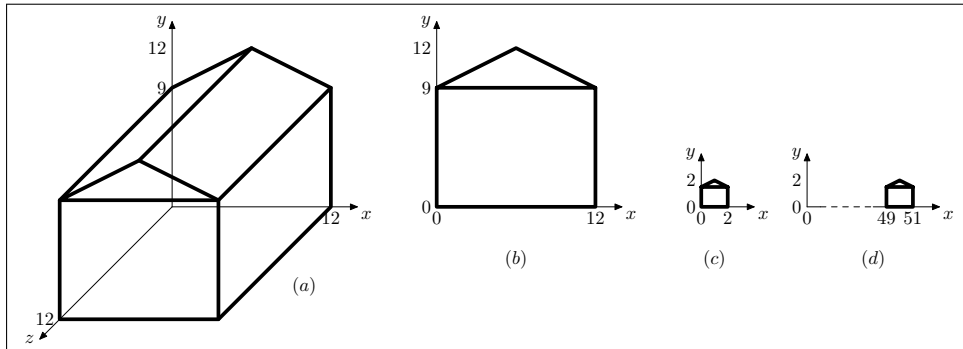


Figure 7.7: Turning a Shed into a Dog House in Three Easy Steps.

toward your back yard. You would see only the front of the shed, not the sides, the roof, or the back. This is shown in Figure 7.7 (b). You can think of this as ‘flattening’ the shed against an imaginary wall along your back lot line, much as you might flatten an aluminum can by standing it upright on the ground and stomping on it with your foot. This is also an example of an orthographic projection, but is more straight-forward because the projection plane — the imaginary wall against which we are flattening the shed — is the same as one of the three coordinate planes (in this case, the xy plane).¹⁵

Time for a plot twist: You start pricing materials and realize that your vision greatly exceeds your budget. What will you do? Quit and look like an incompetent fool to your friends and neighbors? No chance! You regroup and get to work on the design of a dog house for your faithful companion. A shed design quickly becomes a dog house design with just a little *scaling* (a.k.a. resizing) of the dimensions. After measuring your puzzled dog, you decide to make the dog house one-sixth the width of the shed, and one-fourth as tall and as deep. Figure 7.7 (c) shows the scaled front view.

Your intellectual reputation has been saved! But is the back left corner of the yard the right place for the dog house? If you win the lottery¹⁶ and revive your shed dreams, you’ll still want it in the back corner. With that in mind, you decide to place the dog house in the middle of the back of the yard. Your lot is 100 feet wide, so the middle of the house will be 50 feet from the

¹⁵We are intentionally not giving precise definitions of these projections, because we want to get back to matrices and matrix operations before the sun flames out. Please forgive us, American Design Drafting Association!

¹⁶Reality check: You are highly unlikely to win the lottery’s grand prize. How extremely highly unlikely? We will cover that in a later chapter. In the meantime, just take our word for it: Ridiculously extremely highly unlikely.

back corners. This *translation* of the dog house from one location to another is depicted in Figure 7.7 (d).

7.5.2 Combining The Operations

These three steps (‘flattening’ the shed, scaling it, and translating it) can be done in separate steps, but they can be combined into one step. Let (x, y, z) be a point on our original shed diagram (Figure 7.7 (a)). To create Figure 7.7 (b), the orthographic projection (‘flattening’) simply moves all of the points to the xy -plane, which can be done by simply setting the z components of the points to zero. Scaling, to create Figure 7.7 (c), is done by multiplying the x and y components by their scaling factors s_x and s_y . In this case, $s_x = 1/6$ and $s_y = 1/4$. The final step, the translation to create Figure 7.7 (d), is accomplished with addition (or subtraction) of translation amounts t_x and t_y . Here, we want to slide along the x -axis to the right by 49 feet but leave all of the y components unchanged. Thus, our translation amounts are $t_x = 49$ and $t_y = 0$. Combined, we produce formulae we can use to transform every (x, y, z) point that we used to describe the shed into points (x', y', z') that describe the dog house’s front view:

$$\begin{aligned}x' &= s_x \cdot x + t_x = (1/6)x + 49 \\y' &= s_y \cdot y + t_y = (1/4)y \\z' &= 0\end{aligned}$$

For example, consider the upper front right corner of the shed diagram. It is described by the point $(12, 9, 12)$. Applying the expressions above, we find that the corresponding corner of the dog house is located at $(51, 2^{1/4}, 0)$.

That was pretty easy ...and not a matrix in sight! Unfortunately, this matrix-free approach has limitations as the applications become more complex. First, that was the transformation of one point; we have to translate all of the points that describe the object. Second, the shed / dog house is just one object; what if we have to work with several different objects, as in an architectural drawing of an office building? Third, perhaps we need to transform several objects many times per second to achieve smooth simulation motion, as in a real-time video game.

When speed is critical, we can try to reduce the number of operations that need to be performed. One way to do that is to combine transformation operations into one or more matrices. Modern computers go a step further and off-load those matrix operations to one or more graphics processing units — GPUs — often found on graphics cards added to general-purpose computers.

To turn our projection–scaling–translation operation sequence into matrix operations, we need to start by representing a point as a matrix. We could use a row vector or a column vector. Traditionally, a column vector is used, so we’ll follow that convention. Thus:

$$(x, y, z) \implies \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Time to consider the operations. We will start with the orthographic projection. In our example, the projection retained the x and y components of each point and replaced the z components with zeroes. We can perform this with matrix multiplication and a 3×3 matrix that is almost an identity matrix:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x \\ y \\ 0 \end{bmatrix}$$

A lot of work, isn’t it, just to copy x and y and replace z with zero? This can’t be worth the trouble! It wouldn’t be worth the trouble if this were all that we needed to do ... but it isn’t.

Scaling requires multiplying x and y by s_x and s_y , respectively. We can do that with the same almost–identity matrix, by replacing the ones with s_x and s_y :

$$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \frac{1}{6} & 0 & 0 \\ 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \frac{1}{6}x \\ \frac{1}{4}y \\ 0 \end{bmatrix}$$

Notice that we are performing both the ‘flattening’ and the scaling with this one matrix — two transformations in one! This works because the product of the two matrices is just this scaling matrix:

$$\begin{bmatrix} \frac{1}{6} & 0 & 0 \\ 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{6} & 0 & 0 \\ 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

The final operation, the translation, cannot be done by modifying this combined transformation matrix. But, we can do with matrix addition:

$$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ 0 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \frac{1}{6} & 0 & 0 \\ 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} 49 \\ 0 \\ 0 \end{bmatrix}$$

Working out these matrix operations will produce the expressions for x' , y' , and z' from earlier: $x' = s_x \cdot x + t_x = (1/6)x + 49$, $y' = s_y \cdot y + t_y = (1/4)y$, and $z' = 0$.

Needing to perform the translation with a matrix addition is unsatisfying, after having been able to combine the ‘flattening’ with the scaling into a single matrix. It is possible to combine all three operations into a single matrix multiplication, but not with 3×3 matrices — 4×4 matrices, combined with *homogeneous coordinates*, are necessary. If you’d like to know how homogeneous coordinates work, consult an introductory computer graphics textbook. With the matrix background you now possess, homogeneous coordinates, not to mention other types of transformations, will be easier to understand.

7.6 Logical Matrices

Some of the matrices that we have used in this chapter, identity matrices in particular, contain only two values: Zero and one. Does having just two values to work with make you think about false, true, and logical operations? Whether it did or not, that’s where this section will take us.

Logical matrices, a.k.a. “(0, 1)–matrices”, are so named because their content is drawn from the domain consisting of just the two symbols ‘0’ and ‘1’. We do not interpret the symbols as integers; instead, we treat ‘0’ as ‘false’ and ‘1’ as ‘true’. Such matrices have many applications. We will present one here, along with three operations that are reminiscent of matrix operations that we introduced earlier in this chapter. Related applications will appear in a later chapter.

7.6.1 Using Logical Matrices to Represent Subsets of Cartesian Products

Quick review: A Cartesian Product of two sets A and B is the set of all possible ordered pairs (a, b) of the elements of A and B .

Example 145:

Let $A = B = \{a, b\}$. Then $A \times B = \{(a, a), (a, b), (b, a), (b, b)\}$.

We know that Cartesian Products are sets, which is why we have been representing Cartesian Products with set notation. Logical matrices are another possible representation. Location M_{ab} in the matrix is set to 1 when the ordered pair (a, b) is an element of $A \times B$.

Example 146:

Continuing from Example 145, because $A \times B$ contains all possible ordered pairs, every location of M contains a 1:

$$M = \begin{array}{cc} & \begin{array}{cc} a & b \end{array} \\ \begin{array}{c} a \\ b \end{array} & \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \end{array}$$

Each ‘1’ tells us that a particular ordered pair exists in the set. For example, the lower-left ‘1’ means that the ordered pair (b, a) is in the set.

Now consider two subsets of $A \times B$. Let $S = \{(a, b), (b, b)\}$ and $T = \{(a, b), (b, a)\}$. Represented as logical matrices with the same labels:

$$S = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \quad T = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

7.6.2 ‘Meet’ and ‘Join’

In Section 7.3.1 we introduced matrix addition and matrix subtraction, which applied addition and subtraction to corresponding pairs of matrix elements. Similarly, with logical matrices, we can apply logical AND (‘ \wedge ’) and inclusive OR (‘ \vee ’) to pairs of same-sized $(0, 1)$ -matrices. The names of these operations are ‘meet’ and ‘join’, respectively.¹⁷

¹⁷Don’t blame us for these names! We didn’t come up with them. ‘Meet’ and ‘join’ are also used as names for related but more complex operations on lattices, but we have yet to learn whom to blame for the names.

'meet'

Definition 49: 'Meet'

The 'meet' of two logical $n \times m$ matrices A and B is the $n \times m$ matrix C such that $c_{ij} = a_{ij} \wedge b_{ij}$.

'join'

Definition 50: 'Join'

The 'join' of two logical $n \times m$ matrices A and B is the $n \times m$ matrix C such that $c_{ij} = a_{ij} \vee b_{ij}$.

Performing 'meet' and 'join' is no more complex than is performing matrix addition. What can be complex is remembering which operation performs ANDing and which ORing. Here's the memory aid we use: The words 'or' and 'join' both contain the letter 'o'.¹⁸

Believe it or not, 'meet' and 'join' do have practical value, as the next example demonstrates.

Example 147:

Continuing from Example 146, the 'meet' and 'join' of the logical matrices S and T are:

$$S \text{ 'meet' } T = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \text{ 'meet' } \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$S \text{ 'join' } T = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \text{ 'join' } \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

The corresponding set representations of these results are $\{(a, b)\}$ and $\{(a, b), (b, a), (b, b)\}$. So what? Consider this: The intersection of the sets S and T from Example 146 is the set $\{(a, b)\}$ and their union is the set $\{(a, b), (b, a), (b, b)\}$. That is, $S \cap T$ on subsets of Cartesian Products, and S 'meet' T on logical matrices representing the same subsets, are actually

¹⁸Yeah, we know; it's not a great aid, but it works for us.

performing the same operation. The same is true of $S \cup T$ and S ‘join’ T .

With a little reflection, this shouldn’t be very surprising — we already know about the connections between logical operators and set operators from the previous chapter. We will learn much more about subsets of Cartesian Products when we cover relations in Chapter 8.

7.6.3 Logical Matrix Product

Do you remember matrix multiplication from Section 7.3.3? We hope so, because if you do, *logical matrix products* (a.k.a. *boolean products*) will be a lot easier to understand! Their definitions are very similar; we merely replace additions with inclusive ORs and multiplications with logical ANDs, and introduce new symbols \odot (`\odot`) and \bigvee (`\bigvee`).

Definition 51: Logical Matrix Product

The logical matrix product of an $m \times n$ logical matrix A and an $n \times o$ logical matrix B is an $m \times o$ logical matrix $C = A \odot B$ in which $c_{ij} = \bigvee_{k=1}^n (a_{ik} \wedge b_{kj})$.
(Also known as *Boolean Product*.)

logical matrix product

The \bigvee symbol tells us that we are to OR together all of the conjunctions we produce for each application of the expression. That is, just as \sum tells us to add up all of the terms of the given sequence, \bigvee says to OR up all of the terms of the given sequence. For a logical matrix product, we have a sequence of conjunctions to OR together.

Example 148:

Let L be a 3×3 logical matrix and M a 3×2 logical matrix. Before computing the result of $N = L \odot M$, let’s start by stacking the matrices

as we have done for matrix multiplication:

$$M = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \square & \square \\ \square & \square \\ \square & \square \end{bmatrix} = L \odot M = N$$

Consider n_{11} . If we were computing $L \cdot M$, the expression would be $n_{11} = (0 \cdot 1) + (1 \cdot 1) + (0 \cdot 0)$. To produce $L \odot M$ instead, we replace the additions with inclusive ORs and the multiplications with ANDs:

$$\begin{aligned} n_{11} &= (l_{11} \wedge m_{11}) \vee (l_{12} \wedge m_{21}) \vee (l_{13} \wedge m_{31}) \\ &= (0 \wedge 1) \vee (1 \wedge 1) \vee (0 \wedge 0) \\ &= 0 \vee 1 \vee 0 \\ &= 1 \end{aligned}$$

Because most people are less comfortable with logical operators than they are with arithmetic operators, silly mistakes are common when computing these products manually. Please compute the rest of N 's content on your own. If you list the resulting values row by row, pretend that the six values are bits of a binary value, and convert it to octal, you should get 53_8 .¹⁹ For example, following this conversion process, L becomes $010110101_2 = 265_8$.

Example 149:

Finally, it is time to complete Example 135! At the end of that example, we strongly hinted that matrix multiplication was the way to compute the D^2 matrix from which we computed the two-step column vector, and it is. Thanks to what we have covered between these examples, we now know that “ D^2 ” isn’t just a name; it represents the second power of D , the matrix product $D \cdot D$.

¹⁹Forgotten how to convert from Base 2 to Base 8? See Section A.11 in Appendix A. Yeah, we could just give you the answer, but would you work it out for yourself if we did? That’s what we thought; see Figure 7.8.

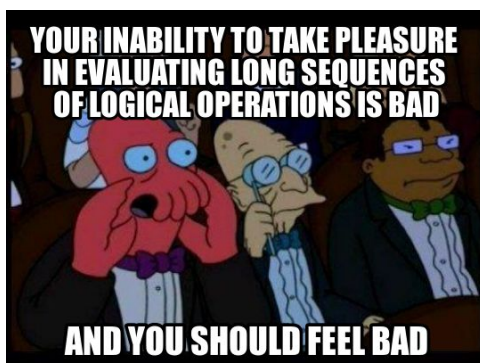


Figure 7.8: Honestly, if you enjoyed it, we'd be concerned. Credits: “Futurama” episode 4ACV18 (“The Devil’s Hands Are Idle Playthings”); <https://memedad.com>.

A deeper question: How does that matrix multiplication produce the two-step results? A follow-up question: Can we use a logical matrix product to compute the same result?

We will answer both questions in this example. But first, for convenience, here are D and D^2 , but with the actual (lower-case) Greek letters instead of their English names, to save space and to help you learn to recognize them:

$$D = \begin{array}{c} \alpha \\ \beta \\ \gamma \\ \delta \end{array} \begin{array}{cccc} \alpha & \beta & \gamma & \delta \\ \left[\begin{array}{cccc} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right] \end{array} \quad D^2 = \begin{array}{c} \alpha \\ \beta \\ \gamma \\ \delta \end{array} \begin{array}{cccc} \alpha & \beta & \gamma & \delta \\ \left[\begin{array}{cccc} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{array} \right] \end{array}$$

Consider the 1 at row α and column β of D^2 , and recall that it represents the transitive result that team Alpha (α) could be considered to be ‘better’ than team Beta (β) because, looking at D , Alpha beat Gamma and Gamma beat Beta.

In Example 135, that 1 value in D^2 was discovered inspecting the content of D . We want to learn why it can also be produced by matrix multiplication. For clarity, here is the expanded expression for the location row α column β of D^2 , and its evaluation:

$$\begin{aligned}
 d_{\alpha\beta}^2 &= (d_{\alpha\alpha} \cdot d_{\alpha\beta}) + (d_{\alpha\beta} \cdot d_{\beta\beta}) + (d_{\alpha\gamma} \cdot d_{\gamma\beta}) + (d_{\alpha\delta} \cdot d_{\delta\beta}) \\
 &= (0 \cdot 0) + (0 \cdot 0) + (1 \cdot 1) + (1 \cdot 0) \\
 &= 0 + 0 + 1 + 0 \\
 &= 1
 \end{aligned}$$

Notice which of the four products produced the 1: $d_{\alpha\gamma} \cdot d_{\gamma\beta}$. Computing these four products requires us to examine all four ways of ‘connecting’ Alpha to Beta through the four teams. Looked at another way, each value d_{ij} represents a win (1) or a loss (0) when i played j . The only way that $d_{\alpha\beta}^2$ can be 1 is for both $d_{\alpha x}$ and $d_{x\beta}$ to be 1 for at least one team x . In this case, $x = \gamma$. Because the matrix product is examining all ways that Alpha could be transitively (a.k.a. two-steps) better than Beta, if a way exists, the matrix product will find it. Here there was one way: Alpha beat Gamma, and Gamma beat Beta.

The complete matrix power D^2 contains the two-step results for all possible pairings of teams. As we know, each result is due to four products, but some of those products will always be zero in this situation. For example, consider the term $d_{\alpha\alpha} \cdot d_{\alpha\beta}$ from above. No intrasquad games (that is, a game in which a team plays itself) will count in the standings, meaning that $d_{\alpha\alpha}$ will always be zero. In the full four-product expression, only the last two terms could possibly be non-zero, because there are only two teams to consider other than Alpha and Beta.

This has an important implication: Depending on the game outcomes, with four teams playing each other at most once, it is possible for $d_{\alpha\beta}^2$ to evaluate to zero, one, or two. Using the simple “one-step plus two-step” tie-breaking approach shown in Example 135, the two-step results could contribute a significant amount to the sum, perhaps more than you feel the two-step data is worth as a way to distinguish teams. One way to limit the two-step contribution is to cap the sum; for example, if the two-step amount is greater than one, you could just add one.

7.6.4 Logical Matrix Powers

Because the definitions of matrix multiplication and logical matrix multiplication are so similar in construction, the fact that we have a concept known

as *logical matrix powers* (a.k.a. *boolean powers*) that corresponds to matrix powers should not be much of a surprise.

Definition 52: Logical Matrix Powers

The n^{th} logical matrix power of an $m \times m$ matrix L , denoted $L^{[n]}$, is the $m \times m$ logical matrix resulting from $n - 1$ successive logical matrix products. (Also known as *Boolean Powers*.)

logical matrix powers

Notice that we add square brackets around the exponent, to distinguish the notations for matrix powers from logical matrix powers. Another detail that we need to mention, for completeness: $L^{[0]} = I$.

Example 150:

At the end of Example 149, we suggested capping the magnitude of the two-step results as a way to keep them from contributing too much to the “one-step plus two-step” sum. Computing a logical matrix power instead of a matrix power will do this automatically.

Let’s extend the game example by another set of games: Alpha continues its domination of the league by defeating Beta, and Gamma defeats Delta to keep Delta winless. D is now:

$$D = \begin{array}{c} \alpha \\ \beta \\ \gamma \\ \delta \end{array} \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

So that we can compare them, here are D^2 and $D^{[2]}$. The difference should be easy to identify:

$$D^2 = \begin{array}{c} \alpha \quad \beta \quad \gamma \quad \delta \\ \alpha \begin{bmatrix} 0 & 1 & 1 & 2 \end{bmatrix} \\ \beta \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix} \\ \gamma \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \\ \delta \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \quad D^{[2]} = \begin{array}{c} \alpha \quad \beta \quad \gamma \quad \delta \\ \alpha \begin{bmatrix} 0 & 1 & 1 & 1 \end{bmatrix} \\ \beta \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix} \\ \gamma \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \\ \delta \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

D^2 reveals that Alpha is transitively ‘better’ than Delta in two ways: Alpha defeating Beta and Beta defeating Delta is the first, and Alpha defeating Gamma and Gamma defeating Delta is the second. Because D^2 is computed with addition, we get a sum of two:

$$\begin{aligned} d_{\alpha\delta}^2 &= (d_{\alpha\alpha} \cdot d_{\alpha\delta}) + (d_{\alpha\beta} \cdot d_{\beta\delta}) + (d_{\alpha\gamma} \cdot d_{\gamma\delta}) + (d_{\alpha\delta} \cdot d_{\delta\delta}) \\ &= (0 \cdot 1) + (1 \cdot 1) + (1 \cdot 1) + (1 \cdot 0) \\ &= 0 + 1 + 1 + 0 \\ &= 2 \end{aligned}$$

The content of a logical matrix power is naturally capped at one because, now matter how many times we inclusively-OR with ‘true’, the result is nothing more than ‘true’:²⁰

$$\begin{aligned} d_{\alpha\delta}^{[2]} &= (d_{\alpha\alpha} \wedge d_{\alpha\delta}) \vee (d_{\alpha\beta} \wedge d_{\beta\delta}) \vee (d_{\alpha\gamma} \wedge d_{\gamma\delta}) \vee (d_{\alpha\delta} \wedge d_{\delta\delta}) \\ &= (0 \wedge 1) \vee (1 \wedge 1) \vee (1 \wedge 1) \vee (1 \wedge 0) \\ &= 0 \vee 1 \vee 1 \vee 0 \\ &= 1 \end{aligned}$$

In Example 150, you may have noticed that, with this extra pair of games in D , there are no ties — each team has a unique number of victories — making the computation of the two-step wins unnecessary . . . if all we needed them for was to break ties. Looking beyond ties, ranking teams and estimating

²⁰Logically, nothing is more true than ‘true’, but one might look beyond logic: “Fairy tales are more than true: not because they tell us that dragons exist, but because they tell us that dragons can be beaten.” Neil Gaiman, in *Coraline*, attributed the quote to G. K. Chesterton’s *Tremendous Trifles*, but Chesterton never wrote it.

differences in measures of skill in various sports are important for reasons such as tournament seeding and sports wagering. Some of the ranking schemes created for these purposes include transitive results as contributing data.

Chapter 8

Relations

As a reader of English prose, the word ‘relation’ might cause you to think about your relatives — parents, children, spouse, etc. — because you are *related* to them biologically or legally. This mental association is useful in this chapter, too, because we can represent such connections, and many others, with *relations*.

Before reading on, we need to ask: How confident do you feel about sets in general, and Cartesian Products in particular? The Math Review appendix and the Additional Set Concepts chapter can help prepare you for this chapter’s content. For that matter, reviewing the Matrices chapter wouldn’t hurt, either.

8.1 (Binary) Relations

The simplest variety of relation is a *binary relation* (a.k.a. *dyadic*, *2-ary* or *2-ary* relation), and is so named because such relations are sets of pairs of elements. If you’re guessing that this is where Cartesian Products come in to the story, you’re right.

Definition 53: (Binary) Relation

A *(binary) relation* from a set X (the *domain*) to a set Y (the *codomain*) is a subset of the Cartesian Product of X and Y ($X \times Y$).

(binary) relation

In general, a binary relation has two *base sets*, the ‘from’ base set (the domain) and the ‘to’ base set (the codomain). In many relations, the domain

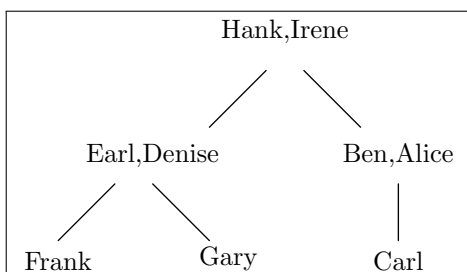


Figure 8.1: A Family Tree

and codomain are the same set, simply called the base set of the relation. In that situation, the wording “on set W ” is usually used instead of “from set W to set W .” The phrases have the same meaning; the former is just less awkward to write or speak.

Example 151:

Figure 8.1 shows a simple family tree. Hank and Irene have two children, Denise and Ben. Denise and Earl also have two children, Frank and Gary, while Ben and Alice have one child, Carl. The set of family members is $M = \{\text{Alice, Ben, Carl, Denise, Earl, Frank, Gary, Hank, Irene}\}$.

We can define a binary parent relation, P , on the set M (that is, from M to M) by creating a set of ordered pairs such that the left member of the pair is a parent and the right member is a child of that parent. That is, P will contain the ordered pair $(\text{Hank}, \text{Denise})$ because Hank is a parent of Denise. Here is the complete relation P :

$$P = \{(\text{Hank}, \text{Denise}), (\text{Hank}, \text{Ben}), (\text{Irene}, \text{Denise}), (\text{Irene}, \text{Ben}), (\text{Earl}, \text{Frank}), (\text{Earl}, \text{Gary}), (\text{Denise}, \text{Frank}), (\text{Denise}, \text{Gary}), (\text{Ben}, \text{Carl}), (\text{Alice}, \text{Carl})\}.$$

How do we know that P is a relation on M ? Easy! P is a subset of the Cartesian Product $M \times M$ — that’s all we need for P to be a binary relation. There are many other subsets of $M \times M$ that are also binary relations,¹ but none of them represent the family tree of Figure 8.1 using (parent,child) ordered pairs.

When elements of domains and codomains are paired in a relation, we say that they are *related* within the context of the relation.

Definition 54: Related

When L is a relation and $(x, y) \in L$, x is said to be *related* to y and is denoted “ $x L y$ ”.

related

If you think that this “ $x L y$ ” notation looks strange now, wait until you see it in an example!

Example 152:

In Example 151, Hank is the parent of Denise. One way to represent this fact is by using set membership notation: $(\text{Hank}, \text{Denise}) \in P$. This is perfectly fine, as P is a set and $(\text{Hank}, \text{Denise})$ is an element of P . Another way is to use the notation introduced by the definition of related: $\text{Hank } P \text{ Denise}$. (Told you it would look strange!)

The “ $x L y$ ” notation looks a little less odd with numbers.

Example 153:

Let $R = \{(x, y) \mid x \% y = 2\}$, where $x, y \in \{2, 4, 6, 8, 10\}$. For which pairs of values is $x R y$ true?

We need values x and y where $y > 2$ and x is two more than a multiple of y . (When $y = 2$, two more than a multiple of y is another multiple of y , and so $x \% y = 0$.) For the given domain, $6 R 4$, $10 R 4$, $8 R 6$, and $10 R 8$ are all true, and so $R = \{(6, 4), (10, 4), (8, 6), (10, 8)\}$.

¹How many others? The total number of possible ordered pairs formed from our set M of family members is $|M \times M| = |M| \cdot |M| = 9 \cdot 9 = 81$. Next, think back to power sets, which is where we learned that there are $2^{|S|}$ possible subsets of a set S . Here, there are 2^{81} possible subsets of $M \times M$, and thus 2^{81} possible relations, of which P is just one. So, there are $2^{81} - 1 = 2, 417, 851, 639, 229, 258, 349, 412, 351$ other relations.

Let's consider a slight variation on Example 153:

Example 154:

Let $R = \{(x, y) \mid x \% y = 1\}$, where $x, y \in \{2, 4, 6, 8, 10\}$. For which pairs of values is $x R y$ true?

None of them! Our domain has only multiples of two, meaning that we can produce only remainders that are multiples of two,² and so a remainder of one is impossible. Thus, $R = \emptyset$.

To take the example a step further: When $R = \emptyset$, is R still a relation? Yes! The empty set is a subset of any set, which means that R is a subset of the Cartesian Product of $\{2, 4, 6, 8, 10\}$ with itself, meeting the conditions of the definition of 'relation.'

This book will not use the “ $x L y$ ” notation very often, but other authors do use it, which is why you need to be familiar with it.

8.2 Creating Relations from Relations

We have seen that the idea of a relation is rather straight-forward, and that we can use set-builder notation to describe the Cartesian Product subsets that represent our situations. Another approach to the creation of relations is to build new relations from existing ones, rather than defining them directly from one or more base sets.

There are a variety of ways to build relations from other relations. One way is through the application of set operators such as union, intersection, and difference. Relations are sets, after all. But are the results of these operators actually relations on the same base set(s)? Are there other ways to build relations from existing relations? And, why would we want to build relations in these ways? This section will provide some answers to these questions.

²Yes, zero is a multiple of two. The product of any integer with x produces a multiple of x . $0 \cdot 2 = 0$, so zero is a multiple of two.

8.2.1 Relations from Relations using Set Operators

If you don't remember how the set operators union (\cup), intersection (\cap), difference ($-$), and complement ($\bar{\square}$) operate, we suggest you review the set sub-section of Appendix A (Math Review) before reading further.

Example 155:

Consider the relation $C = D \times D$, where $D = \{4, 5, 6\}$. That is, $C = \{(4, 4), (4, 5), (4, 6), (5, 4), (5, 5), (5, 6), (6, 4), (6, 5), (6, 6)\}$. Let:

$$EQ = \{(x, y) \mid x = y\} = \{(4, 4), (5, 5), (6, 6)\}, \text{ and}$$

$$LT = \{(x, y) \mid x < y\} = \{(4, 5), (4, 6), (5, 6)\},$$

where $x, y \in D$ for both. (Yes, normally we use single-letter set identifiers, but EQ for 'equals' and LT for 'less than' are more meaningful names.)

Questions: Is $EQ \cup LT$ a relation on D ? If so, what does it represent?

Yes, the union of two sets on D must also be a relation on D . Here's why. All that the union of two relations does is collect the ordered pairs of each into a new third relation (ignoring any duplicates, of course). The ordered pairs are not changed in this process. As a consequence, the resulting relation's content must be a collection of ordered pairs on D , and thus, as a subset of $D \times D$, a relation on D .

As for what $EQ \cup LT$ represents, we hope you have already figured it out. If we collect together all of the pairs representing 'equal to' and those representing 'less than,' the result is the set of ordered pairs representing 'less than or equal to.' We'll label it as LE :

$$LE = \{(x, y) \mid x \leq y\} = \{(4, 4), (4, 5), (4, 6), (5, 5), (5, 6), (6, 6)\}$$

because we'll use it in the next example.

The reasoning for why the union of two relations on a set is also a relation on the same set holds for intersection and difference, too: We're forming

relations from ordered pairs provided by the given relations. Example 156 demonstrates the utility of those two operators.

Example 156:

We'll continue where Example 155 stopped. Let's define one more relation, GE :

$$GE = \{(x, y) \mid x \geq y\} = \{(4, 4), (5, 4), (5, 5), (6, 4), (6, 5), (6, 6)\}$$

What do the relations $LE \cap GE$ and $LE - GE$ represent?

Intersection retains the ordered pairs that are in both of the given relations. The common concept between \leq and \geq is equality, which is why $LE \cap GE = \{(4, 4), (5, 5), (6, 6)\} = EQ$.

We can think of $LE - GE$ as starting with the content of LE , from which we 'remove' the ordered pairs that are also found in GE . That is, we remove the (x, x) pairs. Removing the pairs that represent equality from LE leaves only those that represent 'less than': $LE - GE = LT$.

Set complement needs a little more justification, because it doesn't directly re-use the content of the input relations as union, intersection, and difference do. Recall that the complement of a set S is the set of all of the elements of the universe that are not in S . When S is a relation, \bar{S} is all of the ordered pairs from the universe (for our running example, the universe is C , the Cartesian Product) that are not in S . Stated in terms of our example: $\bar{S} = C - S = (D \times D) - S$. This expression shows that, because complement is defined in terms of set difference, we are actually still re-using ordered pairs; we just had to dig a little bit to uncover that fact.

Example 157:

Recall that $EQ = \{(4, 4), (5, 5), (6, 6)\}$ (from Example 155). What is \overline{EQ} ?

EQ contains all of the ordered pairs representing equality. If we remove equality from the universe, what remains is inequality: $\overline{EQ} = \mathcal{U} - EQ =$

$\{(4, 5), (4, 6), (5, 4), (5, 6), (6, 4), (6, 5)\}$.

There's one more set operator that we haven't discussed in the context of creating relations from relations: Cartesian Product. We left it for the end for a good reason: The Cartesian Product of two binary relations is not another binary relation; it's a *4-ary*, or *tetradic*, relation. That is, its content isn't ordered pairs, it's ordered quadruples. Because the result isn't a binary relation, we have no use for it ... at the moment. The idea of computing Cartesian Products of relations has definite value. We'll revisit this idea in Section 8.2.4.

4-ary relation

8.2.2 Relations from Relations using Inversion

Return with us now to those thrilling days of ³ ... Chapter 1, when we learned that the converse of the implication $p \rightarrow q$ is the implication $q \rightarrow p$ — we simply exchanged the positions of the antecedent and the consequent. Applying this idea of exchanging positions to the elements within the ordered pairs of a relation creates the *converse relation*, also known as the *inverse relation*.⁴

Definition 55: Inverse Relation

The inverse (a.k.a. converse) of a relation R from set A to set B is the relation $R^{-1} = \{(b, a) \mid (a, b) \in R\}$ from B to A , where $a \in A$ and $b \in B$.

inverse relation

Con conversationally, R^{-1} has R 's ordered pairs, but the content of each pair is in the opposite order.

³... yesteryear! That was part of the usual opening to "The Lone Ranger" radio program.

⁴Given what we know of the terms 'inverse' and 'converse' in logic, this naming reuse is unfortunate, but only a little. In logic, inverse and converse are different adjustments to $p \rightarrow q$, but the inverse ($\neg p \rightarrow \neg q$) and converse ($q \rightarrow p$) are logically equivalent, giving some support to the idea that the terms 'inverse relation' and 'converse relation' can have the same meaning. As if two names for one idea isn't enough, there are others, including 'reciprocal,' 'opposite,' 'dual,' and 'transpose.' We'll stick with 'inverse.' As for other notations ... don't ask.

Example 158:

Example 151 defined the parent relation, P , that matched the content of the family tree in Figure 8.1. What does the inverse relation P^{-1} represent?

Recall that:

$$P = \{(\text{Hank}, \text{Denise}), (\text{Hank}, \text{Ben}), (\text{Irene}, \text{Denise}), (\text{Irene}, \text{Ben}), (\text{Earl}, \text{Frank}), (\text{Earl}, \text{Gary}), (\text{Denise}, \text{Frank}), (\text{Denise}, \text{Gary}), (\text{Ben}, \text{Carl}), (\text{Alice}, \text{Carl})\}$$

on the set $M = \{\text{Alice}, \text{Ben}, \text{Carl}, \text{Denise}, \text{Earl}, \text{Frank}, \text{Gary}, \text{Hank}, \text{Irene}\}$. Swapping the elements of the ordered pairs creates a relation that defines the child relation $C = \{(x, y) \mid x \text{ is a child of } y\}$ on M . That is:

$$P^{-1} = C = \{(\text{Denise}, \text{Hank}), (\text{Ben}, \text{Hank}), (\text{Denise}, \text{Irene}), (\text{Ben}, \text{Irene}), (\text{Frank}, \text{Earl}), (\text{Gary}, \text{Earl}), (\text{Frank}, \text{Denise}), (\text{Gary}, \text{Denise}), (\text{Carl}, \text{Ben}), (\text{Carl}, \text{Alice})\}$$

In Example 158, the relation is on a set. Having the domain and codomain be the same set is necessary for R^{-1} to also be a relation on the same base set, but is not a requirement for the result to be called an inverse relation.

8.2.3 Relations from Relations using Relational Composition

The definition of relational composition is a bit mind-numbing. So, rather than start this third way to build relations from relations with the definition, we'll start with an example. Don't worry; we'll bore you with the definition soon!

Example 159:

In Miss Othmar's class, she records the traditional A–B–C–D–F grades of her students, but highlights them in her grade book with different colors. This makes for a two-step grading process: Record a letter-grade for each student, and highlight each student's name with the letter-grade's color. Each step has a corresponding relation.

The first step, that of recording the grade for each student, creates the relation G . Here is the content of G , for a subset of four of Miss Othmar's students:

$$G = \{(Amos,D),(Jane,B),(Levi,D),(Ruth,A)\}$$

For the colors, she uses a red–yellow–blue diverging palette, which we will label with K to avoid confusion with the grade ‘C’:

$$K = \{(A,blue),(B,sky),(C,yellow),(D,orange),(F,red)\}$$

(‘Sky’ is a light blue color.) When she is done, Miss Othmar looks at the highlighted names and realizes that her two–step process has created a new relation M from a merging of G and K that pairs student names with colors:

$$M = \{(Amos,orange),(Jane,sky),(Levi,orange),(Ruth,blue)\}$$

A diagram can help make sense of the connections between the ordered pairs in G and K . Consider Figure 8.2, starting with the $(Amos,D)$ pair, and note the arrow pointing at Amos. Follow that arrow as it leaves $(Amos,D)$ from the D and enters the $(D,orange)$ pair at its D . Finally, it leaves from ‘orange’ (as does another arrow). This sequence of arrows shows the connection that forms the ordered pair $(Amos,orange)$ in the new relation. All of the other ordered pair matches have their own sequences of arrows. Note that reusing ordered pairs, as this example does with $(D,orange)$, is fine.

The ‘merging’ of two relations to create a third, as demonstrated in Example 159, is called *relational composition*. Besides needing two existing relations, we need the ‘to’ set of one relation to be the same as the ‘from’ set of the other. This enables an overlapping of the ordered pairs that makes the composition possible. In Example 159, the shared set is the set of letter grades.

Now that you have seen a practical example of relational composition, the definition should be significantly less mind–numbing than it would have otherwise been. Even so, you’ll still want to read it carefully!

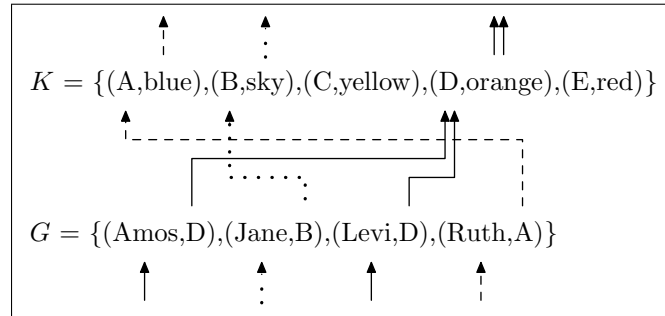


Figure 8.2: Relational Composition: Names to Grades to Colors.

*relational composition***Definition 56: Relational Composition**

Let G be a relation from set A to set B , and let F be a relation from the same set B to set C . The relational composition of relations F and G , denoted $F \circ G$ (\LaTeX: \circ)⁵, is the relation of ordered pairs (a, c) such that $(a, b) \in G$ and $(b, c) \in F$, where $a \in A$, $b \in B$, and $c \in C$.

Example 160:

At Watson–Deer Elementary School’s Mid–Morning Recess games, the playground monitors award gold, silver, bronze, and pewter medals (well, certificates; they’re cheaper) for first through fourth places, respectively.⁶ In this month’s jumprope endurance challenge, first place went to Lua, second was shared by Eve and Abe after an unfortunate collision, and fourth went to Kai. How can we create a relation that pairs the ‘medals’ with the students?

If you haven’t already finished sarcastically muttering “Duh, using relational composition, maybe?”, we’re disappointed. Let’s do it. We have two relations, M for the ‘medals’ and P for the places:

$$M = \{(\text{gold}, \text{first}), (\text{silver}, \text{second}), (\text{bronze}, \text{third}), (\text{pewter}, \text{fourth})\}$$

functional composition

⁵Does this notation look familiar? It’s also used for *functional composition*, which you’ve probably encountered before. We will cover it in the next chapter, where we will compare it to relational composition.

$$P = \{(first, Lua), (second, Eve), (second, Abe), (fourth, Kai)\}$$

Next, we need to order the relations correctly. Is it $M \circ P$ or $P \circ M$? According to the definition, the shared element must be the second member of the ordered pairs of the second relation listed. (In the definition, G is listed second in “ $F \circ G$,” and G ’s ordered pairs have the shared values — the b ’s — second.) Here, the place names are shared by both relations, and they appear second in M ’s ordered pairs. Thus, M needs to be second in the notation, making the correct ordering $P \circ M$.

Now to create $P \circ M$ ’s ordered pairs. We take each of the ordered pairs of the second relation (M) in turn, matching them with the corresponding ordering pair(s) from P to form the (medal, student) pairs we want:

$$P \circ M = \{(gold, Lua), (silver, Eve), (silver, Abe), (pewter, Kai)\}.$$

Figure 8.3 shows the visualization. Note that in this example, we use (silver, second) twice, because we have two elements of P that start with ‘second.’ Also note that we do not have any arrows running through (bronze, third) at all, because there wasn’t a third place finisher (due to the tie for second).

Are you curious as to why our diagrams are constructed to have the arrows pointing bottom-to-top instead of top-to-bottom? We did this to better match how people are likely to write the relations. Given a composition $F \circ G$, it’s natural to write the content of F first and that of G second. Doing this means that the arrows will go bottom-up.

8.2.4 Other Means of Creating Relations from Relations

Time to pick up where Section 8.2.1 feared to tread: Using Cartesian Product on relations. In that section we mentioned that the result of the Cartesian Product of two binary relations creates a 4-ary relation. While such a relation isn’t useful in a discussion of binary relations, it is more generally useful. Here’s how.

⁶“Pewter?” Yes, pewter! It’s an alloy of tin, copper, and other metals. The U.S. Figure Skating Championships started awarding pewter medals to fourth place finishers in 1959. It goes without saying that the idea hasn’t caught on with other sports, but it works well with our example.

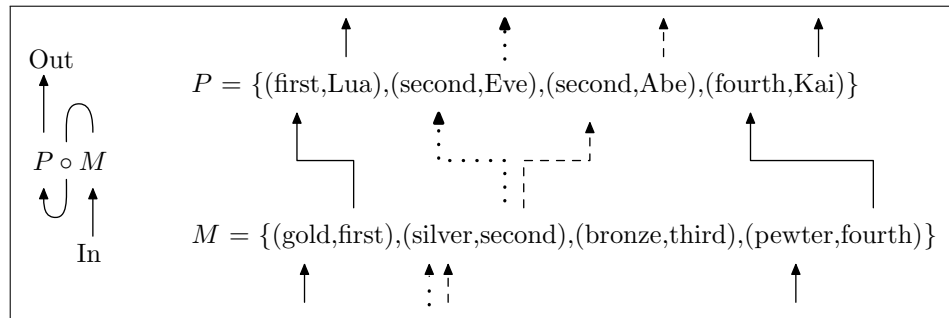


Figure 8.3: Relational Composition: Medals to Places to Students.

When people and organizations need to store a lot of information about an endeavor, such a business, they are likely to use a software product known as a *DBMS* (short for Database Management System) to both store the information and provide convenient access to it. Many DBMSes organize the information entrusted to them using the *relational model*, which is so named because the stored information is arranged into relations. Relations are good at storing basic types of information, such as numbers and words, but not so good with more complex types, such as songs and movies. Many current relational DBMSes have been augmented to support such types, but other, non-relational, types of DBMSes exist, too.

Example 161 demonstrates the utility of Cartesian Product to a relational DBMS.

Example 161:

Professor Plum has two relations that describe his small graduate seminar. The first, the class roster, has the ID, name, and email of each of the three registered students. The second, a seating chart, pairs each student with a chair in the room. As sets, we would write them like this:

$$R = \{(21, Boddy, core), (52, Peacock, peahen), (68, Mustard, mustum)\}$$

$$S = \{(A05, 52), (B19, 21)\}$$

... but when viewed as relations in a DBMS, they would be structured as tables with meaningful names and labels:

*dbms**relational model*

Roster			Seating	
SID	Name	Email	Chair	SID
21	Boddy	core	A05	52
52	Peacock	peahen	B19	21
68	Mustard	mustum		

Apparently, Mustard was absent the day the professor made the seating chart, but the DBMS won't be bothered by this.

The professor would like to confirm with each student their chosen seat, but the seat info is in one relation and the email addresses are in another. Cartesian Product to the rescue! Using that operator, the DBMS can pair up all of the roster information with all of the seating information in a 5-ary relation:

Roster \times Seating				
SID ₁	Name	Email	Chair	SID ₂
21	Boddy	core	A05	52
21	Boddy	core	B19	21
52	Peacock	peahen	A05	52
52	Peacock	peahen	B19	21
68	Mustard	mustum	A05	52
68	Mustard	mustum	B19	21

The result contains all possible associations of the ordered triples from Roster and the ordered pairs from Seating. The table rows that the professor needs, shown in bold in the table, are the two in which the SIDs are equal (because they are the associations that combine the information belonging to the same students).

Example 161 stopped before Professor Plum was satisfied — how is the professor to get just the subset of rows with equal SIDs from the Cartesian Product? And how can Mustard be identified as needing to receive a “see me” email to get a seat? The basic group of set operators does not include operators for these purposes, but relational DBMSes provide them. The *select operator* can answer the first question. The combination of Cartesian Product with

select operator

join operator
outer join

select is so commonly used in relational DBMSes that another operator, the *join operator*, was created to perform both actions with a single command. A variation known as *outer join* can be used to help answer the second question.

Curious to know more about relational DBMSes, their connections to sets, and what else they can do? Database classes are commonly offered by Computer Science departments and business schools. Talk with an academic advisor . . . today!

8.3 Graph Representations of Relations

So far, our only way to represent a relation is as a set of ordered pairs. That fits the definition of ‘relation’ neatly, but it isn’t always a convenient representation to use when we need to identify characteristics of relations, which we will need to do soon.

A representation of relations that is far more visual is one based on a data structure known as a *graph*. You may have already learned a few things about graphs; if so, this sub-section will be (mostly?) review. If not, don’t worry; although graphs is a large topic, we only need a small subset for our purposes.

8.3.1 Graphs

The connection between graphs and relations is made clear by the definition.

graph

Definition 57: Graph

A graph $G = (V, E)$, where V is a finite, non-empty set of vertices and E is a binary relation on V .

By this definition, a graph isn’t so much a way to visualize relations as it is a slight variation on the definition of binary relations. So much for graphs being deep and mysterious!⁷

Nothing in our definition of graphs tells us what a graph looks like. Easily rectified: Think of a vertex as a dot on a piece of paper. An ordered pair is a pair of dots with an arrow (in graph terminology, an *edge*) drawn from the left member of the pair to the right member of the pair. When we use

edge

⁷Yeah, so, OK: *Basic* graphs aren’t deep and mysterious. Dig a little deeper, and they can get pretty complex. Happily, basic graphs are all that we need in this chapter.

an arrow to indicate ordering of a pair of vertices, the graph is known as a *directed graph*, or *digraph* for short.

digraph

You might be wondering: May a vertex have an edge that points back to itself? Yes! This type of edge is known as a *self-loop*. We will soon see that self-loops are very helpful for recognizing one of our upcoming relation properties.

self-loop

You might also be wondering: The definition explicitly says that the relation is on V . Does this mean that “from — to” relations cannot have graph representations? No! We can handle such relations with a simple adjustment: Allow V to be the union of the relation’s ‘from’ and ‘to’ sets.

This is all we need to know about graphs to create diagrams that visualize relations!

8.3.2 Using Graphs to Represent Relations

In the near future (the next section!), we will use graphs to help us understand a variety of properties of relations. For the graphs to be most useful, they need more than vertices and edges — they also need clarity, and, well, a touch of aesthetics. Aesthetics is the study and appreciation of beauty, and is applicable here because the ‘attractiveness’ of our graphical representations of relations will add to their clarity.⁸ Rather than trying to explain, we will demonstrate with a couple of examples.

Example 162:

In Example 153, our base set was $\{2, 4, 6, 8, 10\}$. When creating a graph of an “on” relation, the base set is also our collection of vertices, meaning that the visualization of the graph has five dots. As a set, the relation R consists of four ordered pairs: $\{(6, 4), (10, 4), (8, 6), (10, 8)\}$. Thus, our visualization also has four edges (arrows).

Figure 8.4 shows three different graphs, all of which represent R . The first version, 8.4(a), has two sets of vertices, one for the domain and one for the codomain. This isn’t a good representation for this example, because relation R is an “on” relation, not a “from one set, to a second

⁸“Mathematics, rightly viewed, possesses not only truth, but supreme beauty.” — Bertrand Russell. The standard for our figures in this book is less lofty: “Well, my eyes ain’t bleedin’!”

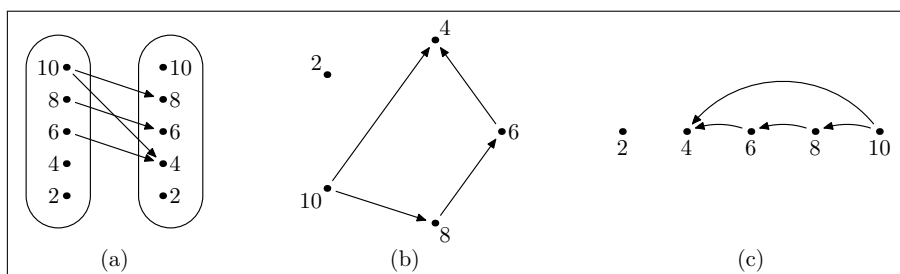


Figure 8.4: Three Graphs of the Relation of Example 153

set” relation. By representing the single base set $\{2, 4, 6, 8, 10\}$ of R as two separate groups, we are improperly implying that the domain and codomain sets are different sets.

Figure 8.4(b) has just one set of vertices, arranged in a circle. This addresses our concern about Figure 8.4(a), and also ‘de-clutters’ the visualization. Figure 8.4(c) opts for a linear arrangement of the vertices, providing a more compact graph.

For relation R , either (b) or (c) is an acceptable graph; which one is ‘better’ is mostly a matter of taste.

Example 163:

In the English alphabet, the distinction between consonants and vowels is based on whether or not the human vocal tract is restricted when producing the sound.

Of the first five letters of the English (a.k.a. Basic Latin) alphabet, ‘A’ and ‘E’ are vowels; the rest (‘B’, ‘C’, and ‘D’) are consonants. The set S of the (letter,sound-type) pairs is a relation from the set of the first five letters to the set of speech sound types: $S = \{(A,vowel),(B,consonant), (C,consonant),(D,consonant),(E,vowel)\}$.

We could just scatter the seven vertices (the five letters plus ‘consonant’ and ‘vowel’) and start drawing arrows, but, as Figure 8.5(a) shows, the result can be disorganized.

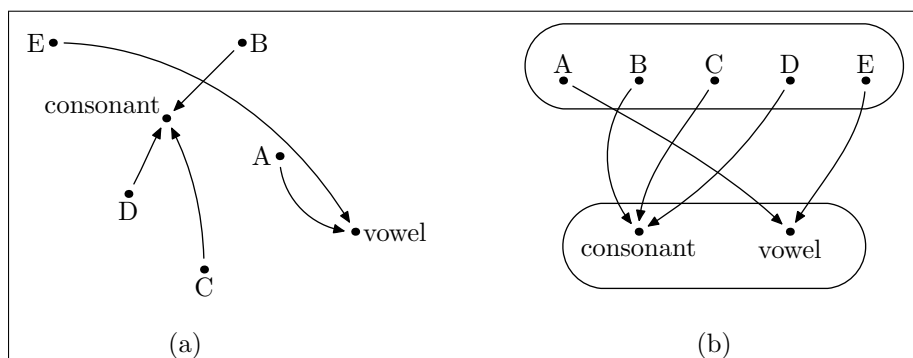


Figure 8.5: Two Graphs of the Relation of Example 163.

When the domain and codomain are separate sets, in the interest of clarity, we will group the domain and codomain elements separately. Doing so will make it easier for us to answer basic questions at a glance, such as “Do all of the letters have an associated sound type?”, and “How many of the letters are vowels?”. One such graph is shown in Figure 8.5(b).

Example 164:

The family tree diagram (Figure 8.1) can be re-drawn as a digraph, as shown in Figure 8.6. Typically, family trees do not include explicit arrows, because the orderings are well-understood. Including the extra lines and arrowheads only adds clutter, not clarity.

8.4 Four Properties of Binary Relations

Fair warning: This section will present a few concepts that are likely to be confusing and maybe even headache-inducing, at which point you may wonder, “Why do I have to learn this???”. Hopefully, at the same time you ask that question, you will remember this paragraph. Individually, most of these properties may not appear to be helpful. Together, though, they are used to define additional properties that have practical applications in computer science. Please stay focused; your patience will be rewarded.

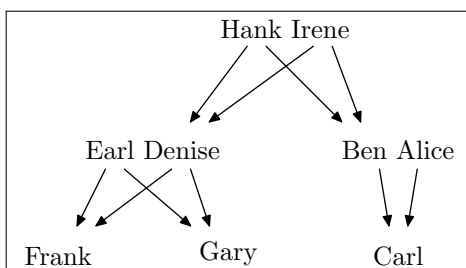


Figure 8.6: The family tree of Figure 8.1 re-drawn as a digraph.

Something else to know before we dive in: All four of the following properties apply only to relations defined “on” a set. This means, for example, that there is little value in asking if relation S from Example 163 possesses any of these properties, because the answer will always be ‘no.’

8.4.1 Reflexivity

Reflexivity is the easiest of the four, both to define and to understand, making it a good property to examine first. This will not be a short sub-section, though, as we will take advantage of reflexivity’s relative simplicity to make several points that have value for the following properties, too.

In English, ‘reflexive’ is an adjective used to describe something that is pointed back toward its own beginning. For example, a song about songs is reflexive. Another: A dog chasing its own tail is playing reflexively. With those examples in mind, the definition of reflexivity as it applies to relations is easy to understand and remember.

Reflexive Relations

In conversational English, when a relation is reflexive, each member of the base set is related to itself. More formally:

reflexivity

Definition 58: Reflexivity

A (binary) relation R on a set A is *reflexive* if $(a, a) \in R, \forall a \in A$.

Example 165:

Consider the relation $N = \{(x, y) \mid x \text{ and } y \text{ have the same first name}\}$ on the set $M = \{\text{Michael Douglas}\}$. Is N even a relation on M ? If it is, is it a reflexive relation on M ?

Let's start by showing all of the ordered pairs in N — this won't take long:

$$N = \{(\text{Michael Douglas}, \text{Michael Douglas})\}$$

For N to be a relation on M , it must be a subset of $M \times M$. As defined above, $N \subseteq M \times M$. Because every set is a subset of itself, yes, N is a subset of $M \times M$, and so N is a relation on M .

Now for reflexivity: For N to be a reflexive relation on M , it must pass the test supplied by the definition of reflexivity. The definition says that N must contain the ordered pair (x, x) for each $x \in M$. Because M contains only one name, the ordered pair that N must contain to be reflexive is $(\text{Michael Douglas}, \text{Michael Douglas})$, and it does. Thus, N is reflexive.

Example 165 was about as small an example of a reflexive relation as we can create, but we can go smaller. Even better, going smaller is worth our time.

Example 166:

Somewhere around the end of the movie “The Ant-Man and the Wasp,” Ant-Man's support team, well, takes a powder.⁹ We can represent members of the non-existent support team with the set $T = \emptyset$. Is the Cartesian Product $T \times T$ a reflexive relation?

Was your first thought a quick ‘no’? We don't blame you for thinking that, but the answer is actually ‘yes.’ $T \times T$ is a relation on T . It's also true that $T \times T = \emptyset$, making it the empty relation. (Remember, any set is a subset of itself, and the empty set *is* a set.)

How can the empty relation be reflexive? Vacuously! The definition

is conditional: A (binary) relation R is reflexive **IF** $(a, a) \in R, \forall a \in A$. For every element t of T , we must have the ordered pair (t, t) in the reflexive relation. There are no elements in T , so there can be no ordered pairs. Thus, there are no ordered pairs that must appear in the relation in order for it to be reflexive. This is why our empty relation vacuously satisfies the definition.

We defined vacuous truth in Chapter 1 and explained vacuous proof in Chapter 4, but this was our first opportunity to use the concept. More are coming!

Example 167:

Two-letter words are really ordered pairs. For example, “to” is an English word, but “ot” is not — order matters.

Let W be a relation on the set $L = \{a, e, h\}$ such that $(x, y) \in W$ when xy (the concatenation of x and y) is an English word. According to the 6th edition of “The Official Scrabble Players Dictionary,” six two-letter words can be formed from the letters in L : ‘aa’, ‘ae’, ‘ah’, ‘eh’, ‘ha’, and ‘he’. Thus, $W = \{(a, a), (a, e), (a, h), (e, h), (h, a), (h, e)\}$. Is W reflexive?

$(a, a) \in W$, but $(e, e) \notin W$, and so W is not reflexive. There is no reason to check (h, h) — as soon as we know that one reflexive ordered pair is missing, we know that the relation cannot be reflexive.

Recognizing Reflexivity in Graphs of Relations

Checking for reflexivity in a set of ordered pairs isn’t very difficult, so long as the base set of the relation isn’t too large. Checking for reflexivity in a graph is even easier.

Figure 8.7 contains graphs of Examples 165 and 167. Take a few moments to examine them, keeping in mind that the first is reflexive and the second is not. The relevant difference is found in the self-loops: Reflexive relations

⁹The phrase “to take a powder” is slang for leaving in a hurry. If you don’t already see the connection to the movie, go watch it . . . after you’ve also watched a pile of prequels!

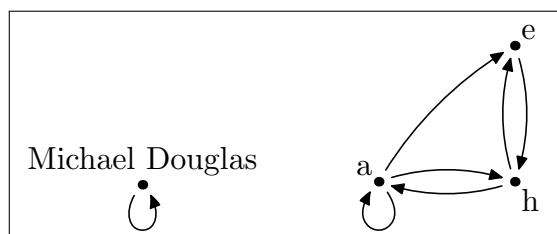


Figure 8.7: Graphs of the Relations of Examples 165 (left) and 167.

have self-loops on all vertices. Or, relations that are not reflexive are missing at least one self-loop.

The self-loops, at least how we have drawn them, look a bit like earrings. This mental image is a nice way to remember reflexivity in graphs. People usually feel that a missing earring is bad news. Similarly, a missing earring is bad news for reflexivity.

8.4.2 Symmetry

You may be familiar with the word *symmetry* in the context of bilateral (a.k.a. reflection) symmetry or radial (a.k.a. rotational) symmetry in biology. When discussing relations, symmetry has a very different meaning.

Symmetric Relations

Definition 59: Symmetry

A (binary) relation S on set B is *symmetric* when $(b, c) \in S$ iff $(c, b) \in S$.

symmetry

In English, this definition tells us that when an ordered pair (x, y) is in a symmetric relation, the ordered pair (y, x) is also in the relation.

Appropriately, relational symmetry is defined in terms of a logical operator (biimplication) whose truth table column is horizontally symmetric: The top half of the column $\begin{pmatrix} T \\ F \end{pmatrix}$ is a mirror-image of the bottom half $\begin{pmatrix} F \\ T \end{pmatrix}$.

Examples are just as helpful with symmetry as they are with reflexivity.

Example 168:

Vivian, Willa, and Ximena have registered as a team for an intramural 3–on–3 basketball tournament. Let $T = \{(x, y) \mid x \text{ and } y \text{ are teammates}\}$ on the set of players $P = \{\text{Vivian, Willa, Ximena}\}$. Is T symmetric?

To be symmetric, $(x, y) \in T$ if and only if $(y, x) \in T$. It is clear that that logical statement holds in this example, because any two people in the set P are teammates, regardless of the order in which we list them. For instance, if $(\text{Vivian, Willa}) \in T$ then $(\text{Willa, Vivian}) \in T$, and if $(\text{Willa, Vivian}) \in T$ then $(\text{Vivian, Willa}) \in T$. This is the case for any pair of teammates drawn from P . Therefore, yes, T is symmetric.

As long as this example is fresh, let's ask (and answer) a review question about it: Is T reflexive?

The answer depends on the definition of “teammate.” Is a person his or her own teammate? We hope you'll agree that the answer is ‘no.’ That is, $(\text{Vivian, Vivian}) \notin T$, which is enough to prove that T is not reflexive.

Example 169:

We used a two–letter word example (Example 167) to help explain reflexivity. Recall that the relation is $W = \{(a, a), (a, e), (a, h), (e, h), (h, a), (h, e)\}$ on the set $W = \{a, e, h\}$. Is W symmetric?

We can answer this via inspection of the ordered pairs in W . All that we need to do is consider each ordered pair $(x, y) \in W$ and verify that $(y, x) \in W$. If that is always true, W is symmetric.

Let's start with (a, a) . If $(a, a) \in W$, then it's clear that the ordered pair with the content in reverse order (that is, (a, a)) will also be in the relation, because it's the same ordered pair! This means that we can ignore all such ordered pairs (that is, the reflexive ordered pairs) when testing for symmetry.

The next ordered pair is (a, e) . The reversed pair, (e, a) , is not in W . This means that “if $(a, e) \in W$ then $(e, a) \in W$ ” is false, which also means that “ $(a, e) \in W$ iff $(e, a) \in W$ ” is false. Because the iff is always true in a symmetric relation, we know that W is not symmetric, without having to test any more pairs of ordered pairs.

Note that W exhibits symmetry with the pairs (a, h) and (h, a) , as well as with the pairs (e, h) and (h, e) . That’s two of the three that we need for symmetry, but that’s not good enough – we need three out of three.

Example 170:

Example 165 introduced the relation $N = \{(\text{Michael Douglas}, \text{Michael Douglas})\}$ on the set $\{(\text{Michael Douglas})\}$. N is reflexive, but is it symmetric?

Let’s conduct the same test that we used in Example 169: For each ordered pair $(x, y) \in N$, is $(y, x) \in N$? If that’s always true, then N is symmetric. N only has one ordered pair, making this test easy to perform. $(\text{Michael Douglas}, \text{Michael Douglas}) \in N$, so we need the reversed ordered pair, $(\text{Michael Douglas}, \text{Michael Douglas})$, to also be in N . Because it is the same ordered pair, yes, it is in N . As that is the only ordered pair, and that ordered pair passed the test, then yes, N is symmetric.

Here’s another way to see that N is symmetric. N ’s only ordered pair is a reflexive ordered pair — N has no ordered pairs of the form (x, y) where $x \neq y$. Without at least one such ordered pair, we cannot violate the definition of symmetry. If a relation cannot violate the definition, the relation must satisfy the definition. By that reasoning, N is (vacuously) symmetric.

Recognizing Symmetry in Graphs of Relations

Detecting reflexivity in the graph of a relation was pretty easy. Happily, detecting symmetry in graphs is almost as easy ... if we draw aesthetically pleasing graphs!

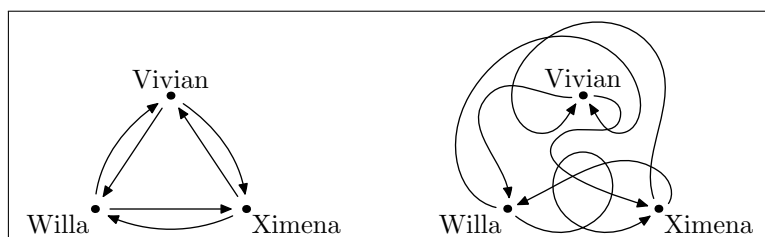


Figure 8.8: Two Graphs of the Relation of Example 168.

When a relation is symmetric, its graph will have only “back and forth” edges. That is, when a symmetric relation has an edge from vertex v to vertex w , it will also have an edge from w back to v . In a clearly-drawn graph, a pair of vertices with only one of the two edges will be easily identified. Note that self-loops are inherently “back and forth” all by themselves, so we can ignore them whether they exist or not.

Example 171:

Figure 8.8 shows two versions of graphs of the 3-on-3 basketball team example (Example 168). The graph on the left shows the pairs of edges plainly, making the symmetry of the relation easy to recognize. The graph on the right ... doesn't. Remember, good visualizations make the important details clear.

8.4.3 Antisymmetry

This is the most important detail to know about antisymmetry:

Antisymmetry is **not** the opposite of symmetry!

Puzzled? We sympathize. In English, *anti* is a prefix meaning “opposite,” which leads people to believe that an *antisymmetric* relation is one that is *not* symmetric. As we will see, there is a lot of middle ground between a symmetric relation and an antisymmetric relation, but we will also see that some relations are both symmetric and antisymmetric.

Can you handle a little more confusion? There's another property of relations called *asymmetry* that is also not the opposite of symmetry, even though

the prefix *a* means “not.”¹⁰ Good news! We won’t be defining asymmetry for a simple reason: This book doesn’t need it.

Antisymmetric Relations

If antisymmetry isn’t the opposite of symmetry, then what does it mean? Thanks for asking!

Definition 60: Antisymmetry

A (binary) relation A on set D is antisymmetric if $(d, e) \in A$ and $d \neq e$, then $(e, d) \notin A$, $\forall d, e \in D$.

antisymmetry

Con conversationally: In an antisymmetric relation, distinct elements are paired together only once. As with symmetry, the self-loop (a.k.a. reflexive) edges don’t matter to antisymmetry; they can exist or not.

Example 172:

Drew and Faustina keep track of the relative ages of their offspring with the relation $O = \{(x, y) \mid x \text{ is older than } y\}$ on their set of children $C = \{\text{Poseidon, Zoe, Nikita}\}$. Assuming that Poseidon is 7, newborn Zoe is 0, and future child Nikita is -2 (what can we say; Drew and Faustina have a plan), is the relation O antisymmetric?

As a set of ordered pairs, $O = \{(\text{Poseidon, Zoe}), (\text{Poseidon, Nikita}), (\text{Zoe, Nikita})\}$. For a relation to be antisymmetric, it cannot contain an ordered pair (y, x) if it already contains the ordered pair (x, y) , when x and y are distinct. Here, that means O cannot have (Zoe, Poseidon) , cannot have $(\text{Nikita, Poseidon})$, and also cannot have (Nikita, Zoe) . O contains none of these pairs, thus O is antisymmetric.

¹⁰Makes you wonder what people were ~~dr~~thinking when they named these properties, doesn’t it? They have to be related to whomever decided that *flammable* and *inflammable* should have the same meaning. (Before you write us: We already know why they have the same meaning. Save your word-origin wisdom for someone on Reddit who really needs it.)

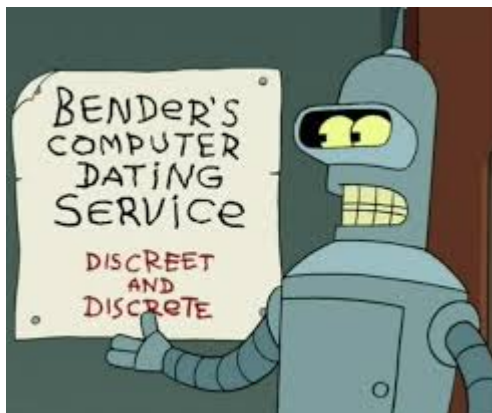


Figure 8.9: How Drew and Faustina met. Credit: “Futurama” episode 2ACV07 (“Put Your Head on My Shoulders”).

Example 173:

In Example 167 and Example 169, we learned that the two-letter word relation is not reflexive and also not symmetric. Is it or is it not antisymmetric?

Recall that $W = \{(a, a), (a, e), (a, h), (e, h), (h, a), (h, e)\}$ on $L = \{a, e, h\}$. We can test W an ordered pair at a time, checking each for the ‘matching’ (reversed) ordered pair that would destroy W ’s dream of antisymmetry. (a, a) is a self-loop, which the definition of antisymmetry tells us to ignore. The next pair is (a, e) . If $(e, a) \in W$, we know W is not antisymmetric and can stop. But, $(e, a) \notin W$, so we continue. Next is (a, h) . This time the reversed ordered pair (h, a) is in W . Because a and h are paired together twice in the relation, W is not antisymmetric.

Combined, Examples 169 and 173 demonstrate that relations can easily be both not symmetric and not antisymmetric. We mentioned at the start of this sub-section that a relation can be both symmetric and antisymmetric. The next example demonstrates that we’ve already seen one such relation.

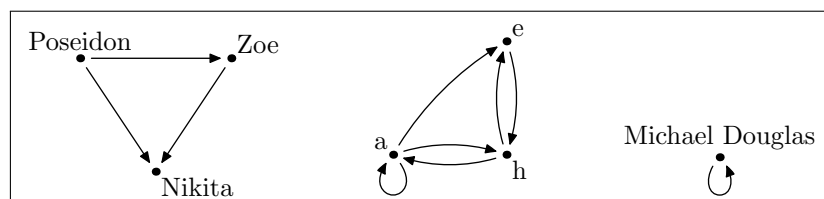


Figure 8.10: Graphs of the Relations of Examples 172, 173, and 174.

Example 174:

Back in Example 165, we introduced the relation $N = \{(\text{Michael Douglas}, \text{Michael Douglas})\}$ on the set $M = \{\text{Michael Douglas}\}$ and showed that it is reflexive. In Example 170, we showed that it was also symmetric, albeit vacuously. Is N also antisymmetric?

We already spoiled the answer to that question: Yes, N is also antisymmetric. Why? For the same reason that N is symmetric: N doesn't violate the definition of antisymmetry, which, like the definition of symmetry, is conditional. The key part of the definition: "[...] if $(d, e) \in A$ and $d \neq e$, then $(e, d) \notin A$ [...]" Relation N does not contain any such (d, e) ordered pairs, meaning that the antecedent of the 'if' is false, which by definition makes the conditional expression true. No ordered pairs in N violate the definition of antisymmetry, and so, vacuously, N is antisymmetric.

Recognizing Antisymmetry in Graphs of Relations

Antisymmetry is just as easily detected in a well-drawn graph as is symmetry. The difference is that, instead of making sure we have only "back and forth" pairs of edges, we need to make certain that there are no "back and forth" edges. That is, we want to see only single edges between pairs of vertices.

Figure 8.10 contains the graphs of all three of the relations used in this sub-section. As with reflexivity and symmetry, only a quick glance is required to verify the presence or absence of antisymmetry. The left graph clearly has no "back and forth" arrows (and so its relation is antisymmetric), the center graph has two such pairs of edges (either of which violates the definition of antisymmetry), and the right graph obviously has no worrisome edges (mean-

ing that its relation is antisymmetric because the relation does not even try to violate the definition of antisymmetry).

8.4.4 Transitivity

The bad news: While our fourth relation property, transitivity, is easy to explain, it is harder to detect within relations than are the other three properties. The good news: Transitivity is also the property that you are most likely to have seen previously. For example, if I tell you that $a = b$ and that $b = c$, you know that a and c must also be equal. That's transitivity of equality. What you probably did not know before reading this sentence is that even the transitivity of equality is based on . . . you guessed it, relations.¹¹

Transitive Relations

As promised, transitivity's definition is straight-forward.

transitivity

Definition 61: Transitivity

A relation T on set F is transitive if $(f, g) \in T$ and $(g, h) \in T$, then $(f, h) \in T$, $\forall f, g, h \in F$.

Example 175:

Consider the stack of blocks in Figure 8.11, and the concept of “above,” as in blocks P and I are both above block L.

Let relation $A = \{(b, c) \mid b \text{ is above } c \text{ in Figure 8.11}\}$ on the set of blocks $B = \{E, I, L, P\}$. Is A transitive?

To answer this question, having A 's set of ordered pairs will help. In alphabetical order, $A = \{(I, E), (I, L), (L, E), (P, E), (P, I), (P, L)\}$.

The definition of transitivity tells that we only need to worry about pairs

¹¹Transitivity in grammars is distinct from transitivity in relations. A transitive verb requires a direct object to be grammatically correct (e.g., *She aced the class.*), while an intransitive verb does not (*He slept.*).

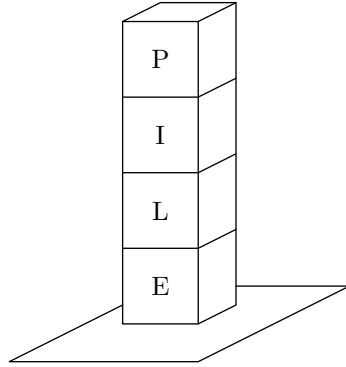


Figure 8.11: Example 175's stack of blocks.

of ordered pairs that share the ‘overlapping’ value (e.g., g in (f, g) and (g, h)). In a transitive relation, the pairs of the form (f, h) must also be in the relation. Our task is to identify all of the $(f, g) - (g, h)$ pairs in the relation and check that the pair (f, h) is also in the relation.

Helpful advice: **Be systematic!** If you randomly look around for overlapping pairs, you are likely to both find some pairs twice and skip some pairs entirely. The former is just wasted work, but the latter could cause you to report an incorrect answer.

Here's our system: Moving left to right, consider each ordered pair in the relation as our leading (f, g) pair. Scan (also left to right) through the list of ordered pairs, finding all that start with g , and create all of the $(f, g) - (g, h)$ pairs. For each such pair of pairs, search for the pair (f, h) in the relation. If an (f, h) is not in the relation, we can stop, because the relation cannot be transitive when an (f, h) ordered pair is missing. But if all of the (f, h) ordered pairs are found, the relation is transitive.

Making a table to keep track of all of the pairs of pairs is worthwhile. The first (leftmost) ordered pair in A is (I, E) , so we start our table's first row with it:

(f, g)	(g, h)	(f, h)	Present?
(I, E)			

Now we check A for ordered pairs that have E as the left value. In our diagram, the E block is not above any other blocks. Thus, there can be no ordered pairs in A of the form (E, \square) . If you're hoping that this means we're done ... sorry. Remember, the definition is conditional: **IF** (f, g) and (g, h) are in T , then so must (f, h) . Having just (f, g) doesn't allow that condition to be tested, and an untested condition cannot fail. We must continue our search.

The next ordered pair in A is (I, L) . This time, scanning A reveals an overlapping (g, h) pair: (L, E) . For A to be transitive, it must also contain the (f, h) pair, which is (I, E) . A quick search reveals that, yes, $(I, E) \in A$. (Or we can remember that we just examined it in the first row ...) This allows us to complete the second row of the table:

(f, g)	(g, h)	(f, h)	Present?
(I, E)	—	—	—
(I, L)	(L, E)	(I, E)	Yes!

Got the idea? We hope so, because if we continue at this pace, this example will require many more pages to finish. Let's skip ahead to this state of our table:

(f, g)	(g, h)	(f, h)	Present?
(I, E)	—	—	—
(I, L)	(L, E)	(I, E)	Yes!
(L, E)	—	—	—
(P, E)	—	—	—
(P, I)	(I, E)	(P, E)	Yes!

We're pausing here to make this point: We found one $(f, g) - (g, h)$ pair that begins with (P, I) , but there is another! Block I is above both blocks E and L ; we need to check both of their corresponding ordered pairs. The final table:

(f, g)	(g, h)	(f, h)	Present?
(I, E)	—	—	—
(I, L)	(L, E)	(I, E)	Yes!
(L, E)	—	—	—
(P, E)	—	—	—
(P, I)	(I, E)	(P, E)	Yes!
(P, I)	(I, L)	(P, L)	Yes!
(P, L)	(L, E)	(P, E)	Yes!

Thanks to systematism,¹² we have found all four pairs of overlapping ordered pairs, and found that A contains all of the necessary (f, h) ordered pairs that need to accompany them. A is transitive!

Example 175 required more work than did any of our previous relation property examples. As we warned, checking transitivity is straight-forward but often requires more effort than does checking the three relation properties we've already covered.

Example 176:

Time for another visit to the two-letter word example (Example 167). The relation, as a set of ordered pairs, is $W = \{(a, a), (a, e), (a, h), (e, h), (h, a), (h, e)\}$ on the set $W = \{a, e, h\}$. Is W transitive?

Using the table approach from Example 175, we begin with the ordered pair (a, a) , which is a reflexive ordered pair. We don't need to spend any time finding $(f, g) - (g, h)$ matches for such ordered pairs; here's why. Whether a reflexive ordered pair is the (f, g) pair or the (g, h) pair, the other pair will also be the (f, h) pair. For example, $(a, a) - (a, e)$ needs (a, e) to be in the relation, and $(1, 2) - (2, 2)$ would need $(1, 2)$. In both cases, the relation needs an ordered pair that it obviously already possesses. We can't totally ignore such pairs, though; stay tuned.

In this example, (a, a) is the only ordered pair that doesn't need a row in our table, but every little bit helps. So, we start our table with (a, e) :

¹²It is so a word! *Systematism* is a noun meaning "following a method or system." On this we can agree: It's an uninspiring superpower.



Figure 8.12: “Sati! Come here, darling. Leave the poor man in peace.” “But, Papa, he’s Keanu Reeves, not Michael Douglas!” “Even more reason ...”
Credit: Warner Bros, “The Matrix Revolutions.”

(f, g)	(g, h)	(f, h)	Present?
(a, e)	(e, h)	(a, h)	Yes!
(a, h)	(h, a)	(a, a)	Yes!

This is why we couldn’t totally ignore (a, a) : We needed it to test the definition’s condition. Onward:

(f, g)	(g, h)	(f, h)	Present?
(a, e)	(e, h)	(a, h)	Yes!
(a, h)	(h, a)	(a, a)	Yes!
(a, h)	(h, e)	(a, e)	Yes!
(e, h)	(h, a)	(e, a)	No!

W was looking good for transitivity for a while, but we soon discovered a missing (f, h) ordered pair. One is all that we need to show that W is not transitive. (If you complete the table, you’ll find that there are more missing ordered pairs.)

We bet that you can guess what our last transitivity example is going to be.

Example 177:

Ah, Michael Douglas; we can’t seem to leave the poor man in peace.

Example 165's relation is $N = \{(\text{Michael Douglas}, \text{Michael Douglas})\}$ on $\{(\text{Michael Douglas})\}$. We've explained that N is reflexive, symmetric, and antisymmetric. Is it also transitive?

Having seen our explanations for symmetry and antisymmetry on this example, you're probably expecting the answer to be, "Yes, vacuously!" We'll give partial credit for that answer, for being both right and wrong. Read on!

As we covered in Example 175, we need a $(f, g) - (g, h)$ pairing of ordered pairs to define a (f, h) ordered pair that needs to be missing for the relation to be declared 'not transitive.' N technically does have such a pairing: $(\text{Michael Douglas}, \text{Michael Douglas}) - (\text{Michael Douglas}, \text{Michael Douglas})$. We already know that pairings using reflexive ordered pairs cannot cause the definition to be violated. Still, if we were to complete the table anyway (and abbreviate in the interest of space) . . .

(f, g)	(g, h)	(f, h)	Present?
(M. D., M. D.)	(M. D., M. D.)	(M. D., M. D.)	Yes!

. . . we would confirm that N does not violate the definition, and so is transitive. But, because we were actually able to test the condition using N 's ordered pair, saying that N 's satisfaction of the definition is vacuous is incorrect.

Recognizing Transitivity in Graphs of Relations

As you read the transitivity examples, you were probably wishing that we would hurry up and get to this sub-section, where the secret to visualizing transitivity would be revealed. Sorry to say, there is no such secret.

We'll get straight to the point: Graphs are not nearly as helpful for visualizing transitivity as they are for visualizing reflexivity, symmetry, and antisymmetry. For very simple relations, yes, a graph can clearly show transitivity. But, even for graphs with just a handful of edges, identifying transitivity can be tricky.

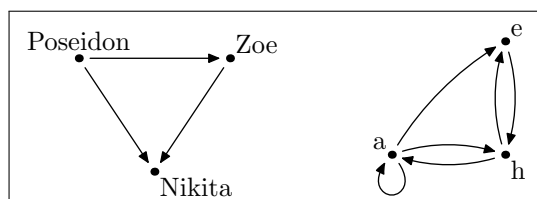


Figure 8.13: Repeats the Graphs of the Relations of Examples 172 and 167.

Example 178:

Example 172 introduced Drew and Faustina’s family and the relation O that contains ordered pairs of their children by age. The graph of O is reproduced in Figure 8.13 (left figure). We can tell at a glance that O is transitive. No need to exhaustively examine a list of ordered pairs this time!

Example 179:

The graph of our two-letter word example (the right figure in Figure 8.13) contains just six edges, yet it is not easy to see that the relation the graph depicts is not transitive. Yes, the fact that we would need the (e, a) edge to accompany the edges (e, h) and (h, a) is fairly clear. But how clear is it that we also need (h, h) and (e, e) ? Sure, we could sit down and work through all of the pairs of edges, but that’s no different than working through all of the pairs of ordered pairs; remember, the edges are just representations of the ordered pairs.

The take-away message: The systematic approach is still the approach to take to verify that a relation is or is not transitive. Well ... unless there’s another relation representation that is more helpful?

8.5 Matrix Representations of Relations

We’ve just seen (last section!) that digraph representations of graphs are a nice way to visualize all of our four graph properties except transitivity. Two-dimensional matrices (Chapter 7) are another useful representation for

relations, with a couple of new benefits. One, nearly all programming languages have matrix operation libraries available. This means that, if we can figure out how to identify relation properties within a matrix representation, we can have a computer do the dirty work of property-checking for us. Two, we can detect transitivity within matrices using matrix multiplication. True, multiplying matrices is a pain — but not if we politely ask a computer to do it for us!

There is a small catch: While we can use matrices to represent any binary relation, matrix representations are most useful for ‘on’ relations. Happily, our four relation properties require ‘on’ relations.

8.5.1 Representing Binary Relations Using Two-Dimensional Matrices

Old news: Binary relations are sets of ordered pairs. The first element of each ordered pair is found in the ‘from’ base set, while the second element is found in the ‘to’ base set. To create a corresponding 2-D matrix, we line up all of the elements of the ‘from’ set, and use them to label the rows of the matrix. Separately, we do the same with the elements of the ‘to’ set, and use them to label the columns.

The content of the matrix M representing the relation R consists of the values 0 and 1. Element m_{rc} is set to 1 if and only if $(r, c) \in R$, and element m_{rc} is set to 0 if and only if $(r, c) \notin R$.

Example 180:

Consider again Example 163, which presented a ‘from-to’ relation example of letters and two sound types (consonants and vowels). The ‘from’ base set was $\{A,B,C,D,E\}$, the ‘to’ base set was $\{\text{consonant},\text{vowel}\}$, and the relation was $S = \{(A,\text{vowel}), (B,\text{consonant}), (C,\text{consonant}), (D,\text{consonant}), (E,\text{vowel})\}$. Using those orderings of the elements in the ‘from’ and ‘to’ sets, our matrix representation’s content is:

$$M = \begin{array}{c} A \\ B \\ C \\ D \\ E \end{array} \begin{array}{cc} \textit{consonant} & \textit{vowel} \\ \left[\begin{array}{cc} 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{array} \right] \end{array}$$

The matrix representation of S is quite useful. Of course, we can easily see that, for example, $(A, \textit{vowel}) \in S$ and that $(A, \textit{consonant})$ is not, but we can do more. If we add up the numbers in each row, we learn that all of the sums are one, meaning that all of the letters must be vowels or consonants, but not both. The sums of the columns tell us that there are three consonants and two vowels.

Relation S in Example 180 uses different base sets for its domain and codomain, making it what we've been calling a 'from-to' relation rather than an 'on' relation. Our relation properties (reflexivity, etc.) can be used only with 'on' relations. One implication of this restriction is that our matrices will be square. Another is that we will label our rows and columns with the same set of values. We need to follow an important rule when doing that labeling: *List the values in the same order for both the row labeling and the column labeling!* You can choose any ordering of the base set values that you like; you just have to like it enough to use it for both row and column labeling. Use that ordering left to right on the columns, and top-down on the rows.

8.5.2 Recognizing Reflexivity in Matrices of Relations

A quick review of Definition 58: Relation R on a set A is reflexive if $(a, a) \in R$, $\forall a \in A$. In order for a matrix representation to be useful in helping us (or a computer) recognize reflexivity, the collection of ones in the matrix that represent the (a, a) ordered pairs need to stand out relative to all of the other ones in the matrix. Happily, they do.

Example 181:

Back in Example 155, we constructed the relation $LE = \{(4, 4), (4, 5), (4, 6), (5, 5), (5, 6), (6, 6)\}$ on $D = \{4, 5, 6\}$, but haven't yet used it as an example

for any of our relation properties. Time to change that! Is LE reflexive?

Here are two possible matrix representations of LE :

$$M_1 = \begin{array}{c} 4 \\ 5 \\ 6 \end{array} \begin{array}{c} 4 \\ 5 \\ 6 \end{array} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \quad M_2 = \begin{array}{c} 6 \\ 4 \\ 5 \end{array} \begin{array}{c} 6 \\ 4 \\ 5 \end{array} \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

$M_1 \neq M_2$ because we used different label orderings for each. But notice where the (a, a) ordered pairs are found, regardless of the ordering: Down the main diagonal (upper-left to lower-right) of the matrix. In LE , all three of the reflexive ordered pairs are present, which is why all three of the main diagonal elements are ones. These matrices demonstrate that when the main diagonal of the matrix representation is all ones, the relation is reflexive.

A small detail of Example 181: The elements not on the main diagonal are irrelevant as far as reflexivity is concerned.

8.5.3 Recognizing Symmetry in Matrices of Relations

In Chapter 7, we learned the definition of matrix symmetry. Informally: When we swap the columns and rows of a square matrix, if the original and resulting matrices are equal, the matrix is symmetric.

Swapping the rows and columns means swapping the row and column indices of the matrix elements. That is, element m_{rc} in the original matrix will become element m_{cr} in the new matrix, and vice-versa. Push this fact on your mental stack; we'll come back to in very soon.

Back to relations. A relation is symmetric when every ordered pair (s, t) in the relation is accompanied by its reversed ordered pair (t, s) . In matrix form, elements m_{st} and m_{ts} are both one.

Time to pop your stack, put these ideas side-by-side, and ask the big question: If the matrix representation of a relation is a symmetric matrix, must that relation be a symmetric relation? Sounds like it's time for ... a proof!¹³

¹³We apologize profusely for the number of pages that have passed between proofs. We will work hard to increase the proof-to-page ratio going forward because ... OK, so no one asked for it. But we'll try anyway.

Example 182:

Problem: Prove or disprove: If the matrix representation M of relation R is a symmetric matrix, then R is a symmetric relation.

Solution: Thanks to our definitions of matrix and relation symmetry, the proof is straight-forward. We need to show the connection between matrix and relation symmetry.

Proof (Direct): Recall that, by the definition of a symmetric matrix, all of the $m_{rc} - m_{cr}$ pairs must have matching values.

Assume that M is the matrix representation of a relation R on a base set S , and assume that M is symmetric. Consider the elements m_{st} and m_{ts} of M , where $s, t \in S$. Because M is symmetric, $m_{st} = m_{ts} = 1$ or $m_{st} = m_{ts} = 0$. If the former, both of the ordered pairs (s, t) and (t, s) are elements of R , satisfying the definition of a symmetric relation. If the latter, neither ordered pair is in R , vacuously satisfying the definition.

Therefore, if the matrix representation M of relation R is a symmetric matrix, then R is a symmetric relation.

The converse is also true, making this an “if and only if” theorem.

Because relation symmetry corresponds to matrix symmetry so nicely, all of our knowledge of symmetry from matrices can be applied to relations to determine their symmetry, as the next example demonstrates.

Example 183:

Brothers Percival, Quinn, and Reginald are being quiet . . . too quiet for their mother’s taste.

Mom: “Boys, what are you doing?”

Reggie: “I’m playing Global Thermonuclear War with Percy.”

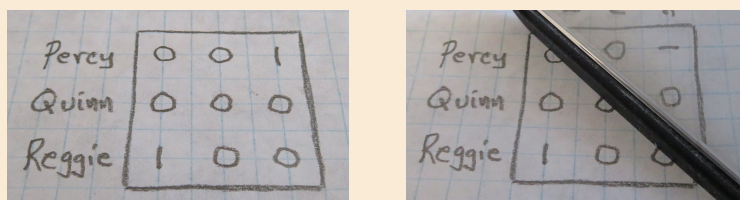
Percy: “And I’m playing Global Thermonuclear War with Reggie.”

Quinn: “I’m reading ‘Slaughterhouse-Five’ again.”

Mom: “What lovely symmetry!”

Is their mother right? Are the boys’ current interactions symmetric?

We can check by (a) inspecting ordered pairs in the interaction relation, (b) drawing a graph of the relation and looking for the “back and forth” edges, or (c) creating the relation’s matrix representation and perching a mirror on the main diagonal. Let’s risk seven years’ bad luck and go with (c)!



The content of the upper-right corner of the matrix, as shown in the left picture, matches that of the reflected mirror-universe version, demonstrating that the interaction relation is symmetric. As usual, Mom’s right.

A thought: Should we have (Quinn,Quinn) in the relation? Probably not, unless Quinn reads aloud to himself. But, as far as symmetry is concerned, either is fine — remember that in symmetry, the reflexive ordered pairs don’t matter.

8.5.4 Recognizing Antisymmetry in Matrices of Relations

If you understood how to recognize symmetry in a matrix representation of a relation, and you haven’t already forgotten what antisymmetry is, you probably already know how to detect antisymmetry: None of the corresponding lower-left and upper-right values can match.

Example 184:

Question: Is the ‘PILE’ relation from Example 175 antisymmetric?

Answer: Seems like it ought to be; how can two cubes be above each



Figure 8.14: Logically, the mirror universe matrix should have a sinister mustache and goatee. Credit: Star Trek, “Mirror, Mirror,” Season 2, Episode 4.

other? But let’s be sure. The relation in that example was named A , so let’s call our matrix representation M_A :

$$M_A = \begin{array}{c} E \\ I \\ L \\ P \end{array} \begin{array}{cccc} E & I & L & P \\ \left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{array} \right] \end{array}$$

We’re going to make you imagine the mirror this time. There are six corresponding pairs of values above and below the main diagonal to check, and they are all different. For antisymmetry, this is good; A is antisymmetric.

8.5.5 Recognizing Transitivity in Matrices of Relations

We promised that representing relations with matrices would solve our transitivity recognition problem, and it will . . . if you’re a computer, or can somehow multiply two matrices rapidly in your head. No matter who or what does the

math, you should understand why matrix multiplication can be used to check for transitivity in a relation.

Before we get to ‘why it works,’ let’s cover the ‘how to do it.’ Start by creating a matrix representation M_R of the relation R . Next, multiply M_R by itself via matrix multiplication. If the same set of elements that were zero in M_R are still zero in M_R^2 , then R is transitive. Otherwise, R is not transitive.

Example 185:

Question: Is $R = \{(A,A),(A,B),(B,B),(C,A),(C,C)\}$ a transitive relation?

Answer: R is not transitive. We can verify this using the exhaustive “if (f, g) and (g, h) , then (f, h) ” pairs of ordered pairs approach we demonstrated in Example 175. Specifically, R is not transitive because it contains (C,A) and (A,B) , but not (C,B) .

Of course, this example exists to show how to get that answer using M_R , which is:

$$M_R = \begin{array}{c} A \\ B \\ C \end{array} \begin{array}{ccc} A & B & C \\ \left[\begin{array}{ccc} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{array} \right] \end{array}$$

Note that there are four elements that are zero in M_R . Now take a look at M_R^2 :

$$M_R^2 = \begin{array}{ccc} \left[\begin{array}{ccc} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{array} \right] \cdot \left[\begin{array}{ccc} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{array} \right] = \begin{array}{c} A \\ B \\ C \end{array} \begin{array}{ccc} A & B & C \\ \left[\begin{array}{ccc} 1 & 2 & 0 \\ 0 & 1 & 0 \\ 2 & \mathbf{1} & 1 \end{array} \right] \end{array}$$

M_R^2 has only three elements that are zero. The one in red, the one that is no longer zero, is — you guessed it — (C,B) ’s.

But ... why is the element of the missing ordered pair no longer zero? And, why *only* that element? To reach the answers to those questions, we will ask another question: Why is the ordered pair (C,B) needed to

make R transitive? Phrased another way: What is sufficient to make (C,B) necessary for transitivity?

You might remember that we answered that question just ahead of this example: Having both (C,A) and (A,B) in R is sufficient to make (C,B) necessary for transitivity to hold. But other pairs of ordered pairs are also sufficient: (C,B) and (B,B) , as well as (C,C) and (C,B) . All three fit the $(f,g) - (g,h)$ pattern that defines transitivity.

Another question before we can answer the original questions: How does matrix multiplication ‘find’ those potentially useful pairs of ordered pairs? Exhaustively! Consider how the ‘1’ at row C and column B came to be. We evaluated this expression (we’ve dropped the relation label for clarity):

$$\begin{aligned} m_{CB}^2 &= m_{CA} \cdot m_{AB} + m_{CB} \cdot m_{BB} + m_{CC} \cdot m_{CB} \\ &= 1 \cdot 1 + 0 \cdot 1 + 1 \cdot 0 \\ &= 1 + 0 + 0 \\ &= 1 \end{aligned}$$

(This expression is just the expansion of the general summation that defines matrix multiplication. If you don’t remember that summation, hop back to Section 7.3.3 for an explanation.)

There are a couple of things to notice about this expression, within the context of transitivity. First, the subscripts of the elements in the products in the first line correspond exactly to the collection of sufficient pairs of ordered pairs that we identified earlier. In effect, the products are testing whether or not the three $(f,g) - (g,h)$ pairs exist in R . If one does, its product is one; otherwise, it is zero. Second, should more than one pair of the ordered pairs exist in R , the sum will be greater than one. This is why we talk about zero and non-zero elements rather than zero and one. (More on this later!)

With those additional questions answered, we can return to the original questions (finally!). We now know why m_{CB} is no longer zero. Only m_{CB} became non-zero because, of all of the potentially ‘sufficient’ pairs of ordered pairs in R , only the $(C,A) - (A,B)$ pair of ordered pairs exists. Each zero in M_R^2 represents a potentially ‘necessary’ ordered pair

that turned out to not be necessary, due to the lack ‘sufficient’ pairs of ordered pairs in R . The matrix multiplication, simply because of the way matrix multiplication is defined, conveniently tested all of them for us and reported the ‘nothing to see here’ results as zeros.

Bonus information: You probably noticed that two values of two appear in M_R^2 . You might be wondering if there’s a difference between a value of one and a value of two. There is! The value represents the number of ‘sufficients’ that exist in R to make that ordered pair necessary. For example, consider m_{AB} . There are three possible pairs of ordered pairs that are sufficient to make (A,B) necessary for transitivity, and R has two of them: (A,A) — (A,B) and (A,B) — (B,B).

Continuing that thought: M_R always consists of just zeros and ones, meaning that we can view it as a logical (i.e., (0, 1)) matrix. You might see where this observation is leading, and the answer is: Yes, we can use logical matrix product in place of matrix multiplication to test for transitivity. Of course, unlike $M_R \cdot M_R$, the result of $M_R \odot M_R$ will contain only zeros and ones.

Example 186:

We already know that relation A from Example 175 (the ‘PILE’ example) is transitive. Let’s verify it using a logical matrix product.

From Example 184, we know the content of M_A . We just need to compute $M_A^{[2]}$:

$$M_A^{[2]} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix} \odot \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix} = \begin{matrix} E & I & L & P \\ \begin{matrix} 0 \\ 1 \\ 0 \\ 1 \end{matrix} & \begin{matrix} 0 \\ 0 \\ 0 \\ 0 \end{matrix} & \begin{matrix} 0 \\ 0 \\ 0 \\ 1 \end{matrix} & \begin{matrix} 0 \\ 0 \\ 0 \\ 0 \end{matrix} \end{matrix}$$

M_A has 10 zeros. Those same elements are all still zero in $M_A^{[2]}$, verifying that A is transitive.

Do the ‘extra’ zeros in $M_A^{[2]}$ bother you? Don’t let them. $M_A^{[2]}$ is not an accumulation of values of both M_A and $M_A \odot M_A$. Instead, think of it as a collection of all of the (f, h) pairs produced by the exhaustive $(f, g) - (g, h)$ testing for transitivity. Some of the ordered pairs in A ((I,L), for instance) aren’t required to be in A by any $(f, g) - (g, h)$ pairs of ordered pairs, and so they aren’t represented in $M_A^{[2]}$. $M_A^{[2]}$ tells us what must be present in A for it to be transitive, and (I,L) isn’t needed. That’s why extra zeros aren’t a problem, but missing zeros are.

8.6 Equivalence Relations

Feeling a little burnt out on relation properties? Sorry, but we have a few more to go. There is a potential bright side to that news: The remaining properties are (mostly) defined in terms of properties that we already know. All you need to do is remember which of the old properties are needed by each of the new ones. More good news: We’ll have a system to help you remember them!

8.6.1 Motivating Equivalence Relations

Do you know an object-oriented (OO) programming language, such as Java, C++, or Python? Such languages are used by programmers to create software systems in which objects encapsulate the data describing, and operations that manipulate, the concepts upon which such systems are built. For example, if we wanted to create a program that plays Klondike solitaire, we would likely represent each of the 52 playing cards with an object, and each of the seven piles of the tableau with seven additional objects.

To ensure that the human player follows the rules of the game, our program would need to check that cards are played legally. In Klondike solitaire, when a black-suited eight is atop a pile, only a red-suited seven can be played upon it. To perform this check, the program needs to be able to compare two card objects to verify that the new card is one less than the top card, and that the colors of the suits are different.

To help programmers test objects against each other, the language Java allows programmers to mark them as being *comparable*. In Java 12’s documentation for comparability¹⁴ are these words:

¹⁴Specifically, this is in the Java 12 API for the `Comparable` interface. See: <https://docs.oracle.com/en/java/javase/12/docs/api/java.base/java/lang/Comparable.html>

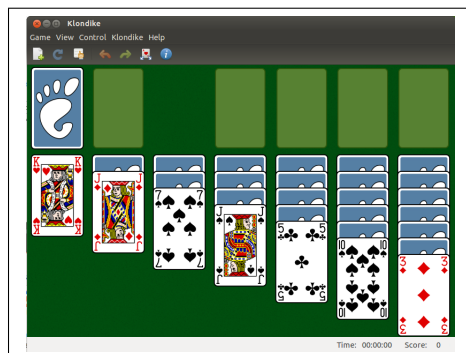


Figure 8.15: Gnome AisleRiot’s Klondike Solitaire interface.

For the mathematically inclined,¹⁵ the *relation* that defines the natural ordering on a given class C is:

$$\{(x, y) \text{ such that } x.\text{compareTo}(y) \leq 0\}.$$

The *quotient* for this total order is:

$$\{(x, y) \text{ such that } x.\text{compareTo}(y) == 0\}.$$

It follows immediately from the contract for `compareTo` that the quotient is an *equivalence relation* on C , and that the natural ordering is a *total order* on C .

(In Java, the ‘factories’ that create objects are called *classes*.) Note that comparability is defined in terms of relations, and that the concept of a quotient is an example of an equivalence relation. Just another reminder that the material we’re covering is useful in computer science.

If you’re wondering what *quotients* and *total orders* are, keep reading. Quotients of relations will be explained soon, in Section 8.6.3, while total orders are the last of our relation properties and will be covered in Section 8.8.

¹⁵That’s us! Just by reading this, you are henceforth “mathematically inclined.” Congratulations! Knowledge of the secret handshake will be revealed in due time.

8.6.2 Equivalence Relations

After all of that setup, the actual definition is likely to be a bit anticlimactic:

equivalence relation

Definition 62: Equivalence Relation

A relation E on set A is an equivalence relation if E is reflexive, symmetric, and transitive.

That is the definition, but there is more to the story. Some implications of that combination of relation properties will be raised by the following examples. Going into them, you need to know that the first three letters of the Greek alphabet are *alpha* (upper case: A , lower case: α), *beta* (B , β), and *gamma* (Γ , γ).

Example 187:

Question: Let $S = \{(x,y) \mid x \text{ and } y \text{ are the same letter, regardless of case}\}$ on the set $U = \{A,B,\Gamma\}$ (note that $U \neq \mathcal{U}$, the universe). Is S an equivalence relation?

Answer: Let's start with a little review. To be an equivalence relation on U , S must first be ...? Right: A relation on U . Represented as a set of ordered pairs, $S = \{(A,A),(B,B),(\Gamma,\Gamma)\}$. S is clearly a subset of $U \times U$, making it a relation on U .

To be an equivalence relation, S must be reflexive, symmetric, and transitive. Let's consider each in turn.

Reflexive: This is very easy to verify. The only three ordered pairs in S are the three reflexive ordered pairs that S needs to be a reflexive relation.

Symmetric: Do we have $(y,x) \in S$ for each $(x,y) \in S$? We clearly do, because in all of our ordered pairs, $x = y$. Or, we can remember that the reflexive ordered pairs can be ignored for symmetry, leaving us with no other ordered pairs, making the relation vacuously symmetric.

Transitive: The only $(f, g) - (g, h)$ pairs of ordered pairs in S are those where $f = g = h$, and so all necessary (f, h) ordered pairs are present, too.

Yes, S is an equivalence relation.

It is not difficult to believe that a relation that only pairs three letters with themselves is an equivalence relation; $A=A$, $B=B$, and $\Gamma = \Gamma$ all seem very equivalent. But what if we add the lower-case letters?

Example 188:

Question: We have S and U as in Example 187, let $L = \{\alpha, \beta, \gamma\}$, and change the base relation of S to $L \cup U$. Is S still an equivalence relation?

Answer: First, remember that S is defined to say that letters are the same regardless of case. With the new, larger base set, the content of S is also larger: $S = \{(A, A), (A, \alpha), (\alpha, A), (\alpha, \alpha), (B, B), (B, \beta), (\beta, B), (\beta, \beta), (\Gamma, \Gamma), (\Gamma, \gamma), (\gamma, \Gamma), (\gamma, \gamma)\}$ S is clearly still a relation on $L \cup U$, so let's get to the good stuff, starting with the matrix representation of S , M_S :

$$M_S = \begin{matrix} & \begin{matrix} A & \alpha & B & \beta & \Gamma & \gamma \end{matrix} \\ \begin{matrix} A \\ \alpha \\ B \\ \beta \\ \Gamma \\ \gamma \end{matrix} & \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{matrix}$$

Reflexive: The main diagonal of M_S contains only ones, showing that S is reflexive.

Symmetric: Ignoring the reflexive ordered pairs, we only have the ordered pairs $(A, \alpha), (\alpha, A), (B, \beta), (\beta, B), (\Gamma, \gamma),$ and $(\gamma, \Gamma)\}$ to check. As every (x, y) has the corresponding (y, x) right next to it, S is easily verified to be symmetric.

Transitive: There is no joy like the joy of manually computing M_S^2

on a 6×6 matrix. Due to the quantity and location of the zeros in the matrix, computing M_S^2 really isn't very hard. The result:

$$M_S^2 = \begin{array}{c} A \\ \alpha \\ B \\ \beta \\ \Gamma \\ \gamma \end{array} \begin{array}{c} A \\ \alpha \\ B \\ \beta \\ \Gamma \\ \gamma \end{array} \begin{bmatrix} 2 & 2 & 0 & 0 & 0 & 0 \\ 2 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 2 & 0 & 0 \\ 0 & 0 & 2 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 0 & 2 & 2 \end{bmatrix}$$

All of the zeros of M_S are still zeros in M_S^2 . S is transitive.

Thus, S on $L \cup U$ is also an equivalence relation.

Are you thinking, “OK, S satisfied the definition, but A and α are *not* the same letter! How can S be an equivalence relation when A and α are not equivalent?” Leave the definition aside for a moment, and think about how we used set builder notation to describe S : “ x and y are the same letter, regardless of case.” As case doesn't matter, A and α are equal, according to the instructions used to create S .

8.6.3 Equivalence Classes

You probably noticed that the non-zero values in the M_S matrix of Example 188 seem to be in three clusters along the main diagonal. Those clusters are prominent due to the way the base set values are ordered. Still, the groupings suggest that the alphas have a affinity for one another, as do the betas and the gammas. Those sets of associated elements inspires the concept of an *equivalence class*.

equivalence class

Definition 63: Equivalence Class

Let E be an equivalence relation on set B , with $b \in B$. b 's *equivalence class*, denoted $[b]$, is the set $\{c \mid (b, c) \in E\}$, where $c \in B$.

In plain(er?) English, the equivalence class of b is the set of all of the values c that appear on the right of b in ordered pairs of E .

Example 189:

Question: What is $[\Gamma]$, using the equivalence relation S from Example 188?

Answer: Let's answer this using the set of ordered pairs representation of S , and then using M_S .

There are two ordered pairs in S that have Γ on the left side: (Γ, Γ) and (Γ, γ) . All we have to do to create the equivalence class $[\Gamma]$ is build a set consisting of the two right-side values of those ordered pairs. $[\Gamma] = \{\Gamma, \gamma\}$.

Equivalence classes are even easier to see within matrix representations, because all of the ordered pairs that begin with a particular element will be non-zero values within that element's row of the matrix. The column labels of those non-zero values are the elements of the equivalence class of the row label. In M_S of Example 188, there are two non-zero values in Γ 's row. Their column labels are Γ and γ . It follows that $[\Gamma] = \{\Gamma, \gamma\}$.

Remember that only equivalence relations have equivalence classes.

The set of all of an equivalence relation's equivalence classes is known as the *quotient* of the relation's base set.

Definition 64: Quotient

Again let E be an equivalence relation on set B . B 's *quotient* with respect to E , denoted B/E , is the set of all of E 's equivalence classes.

quotient

Together, all of the equivalence classes of a relation “divide up” (form a partition of) the relation's base set. Looked at this way, the term ‘quotient,’ and its division-like notation, makes some sense.

Example 190:

There are three equivalence classes of relation S on the base set $L \cup U$ in Example 188: $[A] = [\alpha] = \{A, \alpha\}$, $[B] = [\beta] = \{B, \beta\}$, and $[\Gamma] = [\gamma] = \{\Gamma, \gamma\}$. The quotient of $L \cup U$, $(L \cup U)/S$, is $\{\{A, \alpha\}, \{B, \beta\}, \{\Gamma, \gamma\}\}$.

8.7 Partial Orders

An activity we hope you experience at least twice a day: Teeth-brushing. Imagine that you are standing just outside of your bathroom. Your not-too-old toothbrush and your non-empty tube of toothpaste are both sitting on the counter. Before brushing can happen, you need to walk into the bathroom, you must grab hold of the toothbrush, you must grab the toothpaste tube, and you need to apply a blob of toothpaste to the brush. Here are two sequences of these actions that can result in your teeth being brushed with toothpaste:¹⁶

Sequence #1	Sequence #2
Walk in	Walk in
Grab toothbrush	Grab toothpaste
Grab toothpaste	Apply paste to brush
Apply paste to brush	Grab toothbrush
Brush teeth	Brush teeth

Some of these actions need to be performed in a specific order (for example, grabbing the toothpaste before applying it to the brush), while others can be performed in either order (such as, we can grab either the brush or the tube first) and our goal of clean(er) teeth can be achieved.

Figure 8.16 shows the immediate ordering relationships between the five actions, as a digraph. Note that, although entering the bathroom (“Walk In”) is ordered before “Brush Teeth,” there is no edge directly connecting them. This is because the ordering is transitively implied by the edges that already exist. However, there are no edges, direct or implied, between the actions of grabbing the toothpaste tube and grabbing the toothbrush. Thus, some pairs of actions are ordered and some are not. Such a relation is known as a *partially ordered relation*, or, more compactly, a *partial order*.

¹⁶Yeah, Sequence #2 assumes that your toothbrush is pretty stable on the countertop. Imagine you can afford nice, heavy, solid gold toothbrush handles . . .

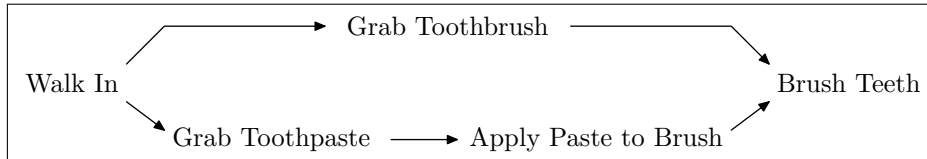


Figure 8.16: Orderings of the Teeth-brushing Actions

Just as we have two types of disjunction, we have two kinds of partial order. We'll define both because both are useful, but we'll do so one at a time.

8.7.1 Reflexive Partial Orders

The name “reflexive partial order” is a big clue as to one of the three relation properties that combine to define it. We dropped another clue above, when we commented on transitivity implying unshown edges in Figure 8.16. That leaves one relation property, which is antisymmetry. That’s right: The only difference between an equivalence relation and a reflexive partial order is the “anti” in front of “symmetric.”

Definition 65: Reflexive Partial Order

A relation R on set P is a reflexive partial order if R is reflexive, *antisymmetric*, and transitive.

reflexive partial order

An alternate name for this kind of partial order is “weak partial order.” A third is “non-strict partial order,” which hints at the name of our yet-to-come second partial order variety.

Our first reflexive/weak/non-strict partial order example is the classic: Less than or equal to.

Example 191:

Question: Let $L = \{(i, j) \mid i \leq j\}$ on the set $P = \{11, 13, 17, 19\}$. Is L a reflexive partial order?

Answer: To be a reflexive partial order, L must be reflexive, antisym-

metric, and transitive.

Reflexive: As any integer is \leq itself, L will contain $(11,11)$, $(13,13)$, etc., and is therefore reflexive.

Antisymmetric: Quick review: To be antisymmetric, a relation that contains (p, q) cannot also contain (q, p) when $p \neq q$. L has several such ordered pairs, including $(11,13)$ and $(13,17)$. But, the reversed pairs $(13,11)$ and $(17,13)$ cannot be in L because $13 \not\leq 11$ and $17 \not\leq 13$. The same is true of all other such pairs of elements of P , telling us that L is antisymmetric.

Transitive: Let's be exhaustive this time, but not *too* exhaustive. We'll make our lives easier by ignoring the (p, p) ordered pairs, as Example 176 taught us we can do. That leaves us with just four $(f, g) - (g, h)$ pairs of ordered pairs to check. As all needed (f, h) pairs are present, L is transitive:

(f, g)	(g, h)	(f, h)	Present?
$(11,13)$	$(13,17)$	$(11,17)$	Yes!
$(11,13)$	$(13,19)$	$(11,19)$	" ¹⁷
$(11,17)$	$(17,19)$	$(11,19)$	"
$(13,17)$	$(17,19)$	$(13,19)$	"

Yes, L is a reflexive partial order.

A question before we leave this example: Is L also an equivalence relation? To check L for equivalence, we only need to consider symmetry, as we've already verified that L is reflexive and transitive. L is not symmetric, because, for example, we have $(11, 13) \in L$ but not $(13, 11)$. So, no, L is not also an equivalence relation.

*Poset*¹⁸ is a term associated with partial orders. It is an abbreviation of

¹⁷This is a *ditto mark*, and indicates that the text on the previous line is repeated on this line. Ditto is also the name of a Pokémon and of one of the twins (repeated ... get it?) in the long-running "Hi and Lois" newspaper comic strip.

¹⁸To us, this sounds like the name of some near-future, gotta-have child's toy, like a Furby or a Tickle Me Elmo. "Mom! Dad! Get your kids a Poset this holiday season! It's adorable **and** relational!" (Fine print: Posets are both flammable and inflammable. Earplugs

the phrase “partially ordered set,” and is the name of the base set of a reflexive partial order.

Definition 66: Poset

A *poset* (partially ordered set) is the base set B of a reflexively partially ordered relation R , and is denoted by the ordered pair (B, R) .

poset

Example 192:

Question: Does relation L from Example 191 have a poset? If so, what is it?

Answer: That example showed that L is a reflexive partial order, which means that L does have a poset, and that poset is its base set, P .

Beyond the notation given in the poset definition, there’s a bit more poset notation that you should understand, and learning it soon after a less–than–or–equal–to example is the best time.

Say that R is a reflexively partially ordered relation on B , with $x, y \in B$ and $(x, y) \in R$. When discussing reflexive partial orders, instead of writing “ $(x, y) \in R$,” people will shorten it to “ $x \preceq y$ ” (L^AT_EX: `\preceq`). Notice that the symbol ‘ \preceq ’ is very similar to the more common ‘ \leq ’ symbol, but with more artistic flair. And, because ‘ \preceq ’ represents the recursively partially ordered relation, people will often use the symbol ‘ \preceq ’ to also be the relation’s name, instead of using a letter, making the poset notation (B, \preceq) .

Still with us? We hope so, but there’s more to the story. Instead of tracking down the correct symbol, people will just use ‘ \leq ’ instead, resulting in the notation (B, \leq) and leading people new to posets into thinking that posets are only for less–than–or–equal–to relations. Which raises a question: Why not pick a totally new symbol instead of one that looks like ‘ \leq ’? The reason is that partial orders based on ‘ \leq ’ are the classic examples, which, unfortunately, caused ‘ \leq ’ to be adopted as the symbol for all reflexive partial

recommended. Requires 13 AAAAA batteries, not included nor available in stores. Manufacturer not responsible for abuse of the good/evil switch.)

orders, whether or not they are based on that operator. People who use ‘ \preceq ’ are both sticking with tradition while trying to be (slightly) distinct from ‘ \leq ’.

In summary: When you see the notation (B, \preceq) or (B, \leq) , you are being told that B is a poset on a reflexive partial order named ‘ \preceq ’ or ‘ \leq ’, and should be aware that the partial order may or may not involve the less-than-or-equal-to operator.

Example 193:

Question: We began this section with two sequences of teeth-brushing operations. Is that example a reflexive partial order?

Answer: Figure 8.16 showed a graph representation of that example’s relation. As a set of ordered pairs, it’s $T = \{(\text{Walk In, Grab Toothbrush}), (\text{Walk In, Grab Toothpaste}), (\text{Grab Toothpaste, Apply Paste to Brush}), (\text{Grab Toothbrush, Brush Teeth}), (\text{Apply Paste to Brush, Brush Teeth})\}$.

As the discussion with the figure made clear, the graph, and thus the relation T , does not include all of the edges necessary for T to be transitive. This means that T cannot be a reflexive partial order.

Even if we added all of the needed ordered pairs to achieve transitivity, T still wouldn’t be reflexive, and so still would not be a reflexive partial order.

A follow-up question: Is the base set of T a poset? No. Remember that the ‘po’ stands for ‘partially ordered.’ For a base set to be a poset, it must be associated with a reflexive partial order, which T is not. The same base set could be a poset for a different relation, but not for T .

8.7.2 Irreflexive Partial Orders

If you hear a mathematician say that something is a “partial order,” odds are that they mean it’s a reflexive partial order. However, the teeth-brushing example (with the additional ordered pairs for transitivity) certainly feels as though we could call it a partial order. Insisting on reflexivity works fine for situations such as less-than-or-equal-to, but excludes other relations that we might practically know to be ‘partial orders,’ too.

This is why we also have *irreflexive partial orders*. Because we only use the concept of irreflexivity for this one kind of partial order, we didn't include the term alongside reflexivity, symmetry, and the rest. But, as we can't define irreflexive partial order without it, ...

Definition 67: Irreflexivity

A relation E on set B is *irreflexive* iff no element of B is related to itself in E .

irreflexivity

Irreflexivity is at the opposite extreme of reflexivity: To be reflexive, a relation must have *all* (a, a) ordered pairs; to be irreflexive, a relation must have *no* (a, a) ordered pairs.

The parallels between reflexivity/irreflexivity and symmetry/antisymmetry are strong. Just as it is possible for a relation to be both symmetric and antisymmetric, it's possible for a relation to be both reflexive and irreflexive. Similarly, just as there are many relations that are neither symmetric nor antisymmetric, there are also many relations that are neither reflexive nor irreflexive. Finally, just as “antisymmetric” is not the same as “not symmetric,” “irreflexive” is not the same as “not reflexive.”

Now that that's out of the way, we can return to partial orders.

Definition 68: Irreflexive Partial Order

A relation I on set B is an *irreflexive partial order* if it is *irreflexive*, antisymmetric, and transitive.

irreflexive partial order

Just as we only added the prefix “anti” to the definition of equivalence relation to create the definition of reflexive partial order, we only added the prefix “ir” to the definition of reflexive partial order to get that of irreflexive partial order.

An alternate name for “irreflexive partial order” is “strict partial order.” Many people prefer “weak” and “strict” to the longer “reflexive” and “irreflexive,” but we like the full names because they perfectly describe the only difference between the two concepts.

Example 194:

Question: Is the teeth-brushing example's relation an irreflexive partial order?

Answer: No, at least not in its original form. Remember, relation T from Example 193 is not transitive.

This time, let's actually add the additional ordered pairs to T that the graph of Figure 8.16 did not include. We'll call the augmented relation T' (read: T-prime), and will abbreviate the action names to make the length more manageable.

The abbreviated original ordered pairs of T are (WI,GTB), (WI,GTP), (GTP,APB), (GTB,BT), and (APB,BT). Transitivity requires the additions of (WI,APB), (GTP,BT), and (WI,BT).¹⁹ Combined, we have $T' = \{(WI,GTB), (WI,GTP), (GTP,APB), (GTB,BT), (APB,BT), (WI,APB), (GTP,BT), (WI,BT)\}$. For practice, feel free to verify the transitivity of T' yourself.

Having achieved transitivity, we need to check irreflexivity and antisymmetry, too. T' is irreflexive, because it contains no (a, a) pairs. T' is also antisymmetric because no $(d, e) - (e, d)$ pairs of ordered pairs exist. T' is therefore an irreflexive partial order.

8.7.3 Remembering the Definitions of Equivalence Relation, Reflexive Partial Order, and Irreflexive Partial Order

We promised to present a way for you to easily remember the definitions of equivalence relation and of the two partial orders. We aim to keep our promises!

First, all three definitions are built of three concepts. Second, all three include transitivity. Third, in the order in which we presented them, each definition is a slightly different version of the previous definition. If you can remember the equivalence relation definition and the two adjustments, you'll be able to reconstruct the partial order definitions when you need them.

¹⁹Bonus knowledge! Because T' is the smallest transitive relation such that $T \subseteq T'$, T' is known as the *transitive closure* of T .



Figure 8.17: Where’s Anti Ir? Well, she was . . . um . . . indisposed.²² Credit: Metro–Goldwyn–Mayer, “Wizard of Oz.”

The equivalence relation definition can be remembered for consisting of only the three ‘positive’ relation properties: Reflexivity, Symmetry, and Transitivity. (We think of them as being ‘positive’ because their names do not include any of the English ‘not’ prefixes.)

Now for the two adjustments. You’ve probably seen the movie “The Wizard of Oz,” or perhaps read the original book, “The Wonderful Wizard of Oz.” The last line of the movie is spoken by Dorothy: “Oh, Auntie Em, there’s no place like home!” Imagine that Auntie Em has a sister, Irma. It almost goes without saying that Dorothy would call her . . . Anti Ir.²⁰ Those are the two ‘negative’ prefixes that we need to add, one at a time, in that order, to the equivalence order definition to create the two partial order definitions. You need to remember to which property names to add which prefixes, but that’s not hard, either; ‘antireflexive’ and ‘irsymmetric’ just sound weird.²¹

²⁰We don’t care how loudly this made you groan; we refuse to apologize for it. Also, although you might expect Anti Ir to be very, very evil because her name is two negatives, the two cancel out, and so Dorothy loves her just as much as Auntie Em. Isn’t that sweet?

²¹We hate to have to mention this, but . . . ‘anti–reflexive’ is actually used an alternate name for irreflexivity by some people. On the bright side, ‘irsymmetric’ doesn’t seem to have a meaning . . . so far.

²²You know, like Chekov was during the entire “Space Seed” episode of “Star Trek.” Also, color was illegal in Kansas until 1967.

8.8 Total Orders

Remember the mention of the phrase ‘total order’ back in Section 8.6.1? It was in the quote from Java’s documentation of the `Comparable` interface, alongside ‘equivalence relation.’ Sounds like they might be connected, doesn’t it? They are, though not directly, at least as viewed through their definitions.

Just based on their names, you might assume that a partial order is an incomplete total order. That is, maybe you get a partial order when you start arranging something but get bored and don’t complete the job, while a total order is what you’d have produced if you’d finished what you’d started. Actually, that’s not too far off!

Before we can define what a total order is, we need to define a familiar term.

comparable

Definition 69: Comparable

Let R be a reflexive partial order on the set W , and let $w, x \in W$. w and x are *comparable* when either $w \preceq x$ or $x \preceq w$.

Note that the definition doesn’t need to include “...but not both” in its wording, because reflexive partial orders are, by definition, antisymmetric.

Example 195:

Question: Let $D = \{(x, y) \mid x \text{ divides } y\}$, where $x, y \in \{1, 2\}$. Are 1 and 2 comparable, in D ?

Answer: To answer this question properly, we need to remember that comparability depends on the relation being a reflexive partial order. Let’s begin there: Is D a reflexive partial order?

For convenience, let’s show D as a set of ordered pairs: $D = \{(1, 1), (1, 2), (2, 2)\}$. To be a reflexive partial order, a relation must be reflexive, antisymmetric, and transitive. D is reflexive (it contains $(1, 1)$ and $(2, 2)$), antisymmetric (it has $(1, 2)$ but not $(2, 1)$), and transitive ($(1, 1) - (1, 2) \rightsquigarrow (1, 2)$ and $(1, 2) - (2, 2) \rightsquigarrow (1, 2)$), making it a reflexive partial order.

Now we can worry about 1 and 2. For these values to be comparable, D must contain exactly one of $(1, 2)$ or $(2, 1)$. It contains just the former, so, yes, 1 and 2 are comparable in D .

With comparability explained, we can define *total order*.

Definition 70: Total Order

Let T be a reflexive partial order on the set S , and let $r, s \in S$. T is a *total order* when each pair of elements r and s are comparable. (Or, T is a total order when it is antisymmetric, transitive, and each pair of elements r and s are comparable.)

total order

Note that the definition is really two versions. Either is correct. The first version is just one step beyond a reflexive partial order, making it handy if you already know that about the relation. The second version follows the same three-property pattern as the partial order definitions. The reason that the second version doesn't include reflexivity is that comparability requires that all of the reflexive ordered pairs be present.

Example 196:

Question: In Example 191, we demonstrated that the relation $L = \{(i, j) \mid i \leq j\}$ on the set $P = \{11, 13, 17, 19\}$ is a reflexive partial order. Is L also a total order?

Answer: Because we already know L to be a reflexive partial order, all that we need to demonstrate is that all pairs of elements in P are comparable in L . Let's consider 17 as an example. The four ordered pairs in L that include 17 are $(11, 17)$, $(13, 17)$, $(17, 17)$, and $(17, 19)$, demonstrating that all elements of P are comparable with 17 (including 17 itself). Because we can also identify the corresponding ordered pairs for all three of the other elements of P , L is a total order.

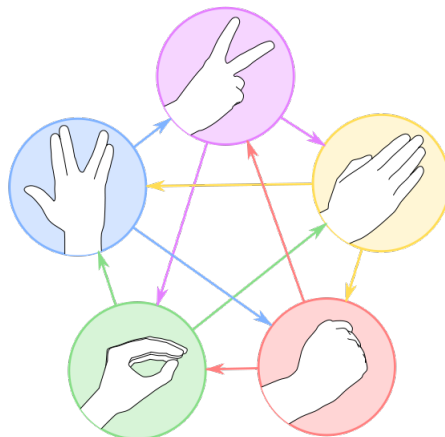


Figure 8.18: A graph of Rock–Paper–Scissors–Lizard–Spock. Clockwise from the top: Scissors, paper, rock, lizard, Spock. Credit: Wikimedia Commons.

Example 197:

The hand game Rock–Paper–Scissors (a.k.a. Roshambo) is a popular way to resolve a dispute with another person. It has variants that date back centuries. Each person simultaneously uses one of their hands to make one of those three shapes. If the first person makes a rock and the second scissiors, the first person wins, because rock crushes scissiors. Similarly, scissiors cut paper, and paper covers rock. If both people make the same shape, they try again until one wins.

A popular modern extension is Rock–Paper–Scissors–Lizard–Spock, created by Sam Kass and Karen Bryla. Figure 8.18 shows a graph of the game’s “who defeats whom” relation. As described by Kass: “Scissiors cuts Paper covers Rock crushes Lizard poisons Spock smashes Scissiors decapitates Lizard eats Paper disproves Spock vaporizes Rock crushes Scissiors.”²³ What Kass (and every graph we’ve ever seen of this version) didn’t bother to include are the ties (e.g., rock ties rock), which can certainly occur. In ordered pairs notation, including the ties, the game’s relation is: $R = \{(\text{scissiors}, \text{paper}), (\text{paper}, \text{rock}), (\text{rock}, \text{lizard}), (\text{lizard}, \text{spock}), (\text{spock}, \text{scissiors}), (\text{scissiors}, \text{lizard}), (\text{lizard}, \text{paper}), (\text{pa}$

per, spock), (spock, rock), (rock, scissors), (rock, rock), (paper, paper), (scissors, scissors), (lizard, lizard), (spock, spock)} on the set $B = \{\text{rock, paper, scissors, lizard, spock}\}$.

Question: Is R a total order?

Answer: To show that the answer is ‘yes’ (which it seems it must be for the game to work), let’s use the second version of the total order definition this time. We have three tasks: Show antisymmetry, transitivity, and, finally, comparability of every pair of elements a and b of B within R , as (a, b) or (b, a) .

Task #1: Is R antisymmetric? This takes a bit of work, but with a little patience we can verify that no non-reflexive (a, b) pair in R has a (b, a) pair. (If one did, we couldn’t determine who wins.) Yes, R is antisymmetric.

Task #2 is our old nemesis, transitivity, which ... trips us up! Here’s an example to demonstrate: R contains (rock, scissors) and (scissors, paper). To be transitive, R must also contain (rock, paper), but it doesn’t. Because R is not transitive, the answer to the question is ‘no:’ R is not a total order.

Does this result bother you? Are you concerned that the game is broken because its relation isn’t totally ordered? Don’t let it worry you. The game needs antisymmetry and comparability to work, but it doesn’t need transitivity, because the game doesn’t need $(f, g) - (g, h)$ sequences of ordered pairs. (If you examine R , you’ll find that B meets the “ $w \preceq x$ or $x \preceq w$ ” requirement, meaning that the game meets the comparability requirement.) You can think of the game’s components as being ordered enough for its needs, but not *totally* ordered. For the same reason, the original Rock–Paper–Scissors isn’t based on a total ordering, either.

²³Source: <http://www.samkass.com/theories/RPSSL.html>

Appendix A

Math Review

This book does not expect that the reader has any more mathematical background than college algebra. That is, you should already be comfortable with algebraic expressions, polynomials, exponents and logarithms, functions, etc. For some of you, much time has passed since you learned this material. To ensure that you can handle discrete structures, reading this appendix is recommended. Note that some college algebra topics are covered in the main body of this book; that is, you won't find all of them here.

A.1 Fractions

Fractions do not appear often in discrete structures, but often enough that reviewing – and practicing – the basics is a good idea. Students lose many points by performing manual manipulations of fractions carelessly.

Definition 71: Common Fraction

A *common fraction*¹ is the quotient of two integers x (the *numerator*) and y (the *denominator*), written x/y or $\frac{x}{y}$, where the denominator is not zero.

($\text{\LaTeX: \frac{x}{y}}$ produces $\frac{x}{y}$.)

common fraction

In the form x/y , the $/$ is commonly known as a *forward slash*, but may also be known as a *solidus*, *virgule*, or even *separatrix*,² depending on the context

¹Common fractions are also known as *vulgar fractions*. No kidding!

²Still not kidding. Who says appendices have to be boring?

in which it is used. The little horizontal line in $\frac{x}{y}$ is a *vinculum*.³

Table 21: Common Fraction Equalities

$\frac{x}{z} + \frac{y}{z} = \frac{x+y}{z}$	$\frac{x}{w} + \frac{y}{z} = \frac{xz+wy}{wz}$
$\frac{x}{z} - \frac{y}{z} = \frac{x-y}{z}$	$\frac{x}{w} - \frac{y}{z} = \frac{xz-wy}{wz}$
$\left(\frac{x}{z}\right)\left(\frac{y}{z}\right) = \frac{xy}{z^2}$	$\left(\frac{x}{w}\right)\left(\frac{y}{z}\right) = \frac{xy}{wz}$
$\frac{\frac{x}{z}}{\frac{y}{z}} = \left[\left(\frac{x}{z}\right)\left(\frac{z}{y}\right) = \right] \frac{x}{y}$	$\frac{\frac{x}{w}}{\frac{y}{z}} = \frac{xz}{wy}$
(a) Equal Denominators	(b) Unequal Denominators

Table 21 shows the basics of fraction addition, subtraction, multiplication, and division.

Example 198 demonstrates several fraction operations, including removing common factors to reduce the result to lowest terms.

Example 198:

Problem: Evaluate $\frac{\frac{1}{6} + \frac{3}{8}}{\frac{5}{12}} - \frac{4}{5}$ and, if necessary, reduce the result.

$$\begin{aligned} \text{Solution: } \frac{\frac{1}{6} + \frac{3}{8}}{\frac{5}{12}} - \frac{4}{5} &= \frac{\frac{8}{48} + \frac{18}{48}}{\frac{5}{12}} - \frac{4}{5} = \frac{\frac{26}{48}}{\frac{5}{12}} - \frac{4}{5} = \frac{\frac{26}{20}}{\frac{48}{48}} - \frac{4}{5} = \left(\frac{26}{48}\right)\left(\frac{48}{20}\right) - \frac{4}{5} \\ &= \frac{26}{20} - \frac{4}{5} = \frac{26}{20} - \frac{16}{20} = \frac{10}{20} = \frac{1}{2} \end{aligned}$$

A.2 Rational Numbers

Fractions consisting of integers lead us directly to numbers that either can, or cannot, be expressed as common fractions.

³Vinculum' was also used as the name of a processing unit in a Borg vessel in *Star Trek: Voyager*.

Definition 72: Rational Number

A value that can be expressed as the ratio of two integers is a *rational number*.

rational number

Notice that the word “ratio” appears in both the definition and within the word “rational.” This makes the definition easy to remember.

How can we determine whether or not decimal values, such as 3.375, are rational? If the fractional part (the part to the right of the decimal point) is a terminating sequence of digits (that is, if the fractional part does not continue indefinitely), then the entire value can be expressed as a fraction.

Example 199:

Problem: Demonstrate that the decimal value 3.375 can be expressed as a rational number.

Solution: 3.375 is a representation of the sum of two fractions: $\frac{3}{1}$ and $\frac{375}{1000}$. $\frac{3}{1} + \frac{375}{1000} = \frac{3000}{1000} + \frac{375}{1000} = \frac{3375}{1000}$. Thus, 3.375 is a rational number, because we can express it exactly by the (unreduced) fraction $\frac{3375}{1000}$.

Decimal values with repeating fractional parts (such as 17.4444... or 0.87262626...)⁴ are also rational numbers, as Example 200 illustrates.

Example 200:

Problem: Demonstrate that 0.87262626... can be represented as the ratio of two integers.

Solution: Let $x = 0.87262626\dots$. We can multiply both sides by 100, which leaves the fractional part consisting of only the repeating digits ($100x = 87.262626\dots$). Multiplying by 10,000 leaves the fractional part with the same repeating pattern: $10000x = 8726.262626\dots$. In both cases, the fractional part is representing the same value. By sub-

⁴To highlight the repeating portions of such numbers, mathematicians use an overbar (L^AT_EX: `\overline{}`) to indicate the repeating part. For example, 0.87262626... can be represented as $0.87\overline{26}$. This ‘overbar’ can also be called a vinculum.

tracting the left and right sides of the $100x$ equality from those of the $10000x$ equality, we learn that $9900x = 8639$, or $x = \frac{8639}{9900}$. That is, $0.87262626\dots = \frac{8639}{9900}$.

You may be surprised to read that 3.375 (and any so-called ‘terminating’ decimal value) is actually a repeating decimal! This is correct because we can append infinitely many zeros to it ($3.37500000\dots$) without changing its value; thus, it is technically a repeating decimal. As a practical matter, writing such values with “000...” at the end is unnecessary if we accept that, notationally, 3.375 represents $3.375000\dots$

irrational

What about values that cannot be expressed as the ratio of two integers? They are called *irrational* numbers. Examples include the values π ($3.1415926\dots$), e ($2.71828\dots$), and $\sqrt{2}$ ($1.4142\dots$). If a decimal value’s fractional part is terminating or repeating, the value is rational. Otherwise, the value is irrational.

A.3 Set Basics

Set notation and set operators are used throughout discrete structures. Students frequently know the basics before studying discrete structures, which is why we cover them in this appendix. More advanced set topics are covered in the main part of this book.

set

Definition 73: Set

A *set* is an unordered collection of unique objects.

We show the elements of a set by listing them, comma-separated, inside of ‘curly brackets’ (\LaTeX : $\{$ and $\}$), a.k.a. ‘braces.’⁵ We’ll use upper-case letters to label sets for later reference.

Example 201:

A rainbow’s colors are red, orange, yellow, green, blue, indigo, and violet.⁶ The set of a rainbow’s colors can be written as:

⁵If you can get over the association with the appliances of teenage dental torture.

$$R = \{\text{red, orange, yellow, green, blue, indigo, violet}\}.$$

Because order doesn't matter to sets, there are $7! = 5,040$ acceptable ways to list the elements of set R .

As the definition says, the elements of a set must be unique. That is, an element may appear in a specific set at most once.

Example 202:

The set of letters in the word 'letter': $S = \{l, e, t, r\}$

Note that, if you need to have a set that allows duplicates, the concept has already been invented. Such sets are known as *multisets* or *bags*.

multiset, bag

A.3.1 Set Notation

Much notation is associated with sets. Here are the basics:

- To indicate that an element is a member of a set, use \in (`\in`). Referring to Example 202, $e \in S$. To negate this, use \notin (`\notin`), as in $f \notin S$.
- A set with no members, known as the *empty set*, is represented by a set of curly brackets with nothing in-between (`{ }`) or by the symbols \emptyset (`\emptyset`) or \varnothing (`\varnothing`).⁷
- The elements of some sets are difficult (sometimes impossible!) to enumerate. To define such sets, we use *set builder* notation. The format: $\{\text{variable(s)} \mid \text{description of set membership}\}$. (`\mid` produces the vertical line.)
- The entire collection of set elements available in a given situation is known as the *universal set*, which we will represent with the symbol \mathcal{U} (`\mathcal{U}`).

empty set

set builder

universal set

⁶The first letters of which, in order, form a name famous to schoolchildren if not to history: Roy G. Biv. But we're talking sets here, not sequences!

⁷`\amssymb` is an optional package for \LaTeX that supplies additional math symbols

Example 203:

$e \in S$	Show that e is a member of set S
$f \notin S$	Show that f is not an element of S
$\{ t \mid t \text{ is a positive multiple of } 3 \}$	$\{3, 6, 9, 12, \dots\}$
$\{ \alpha \mid \alpha \text{ is a UNICODE symbol} \}$	There are 110,181 in UNICODE 6.1!
$\mathcal{U} = \{0, 1, 2, 3\}$	All possible Base 4 digits

A.3.2 Fundamental Set Operators

We will cover four set operators here. Others will be introduced in the body of this book as they are needed.

- Union (\cup , $\LaTeX: \cup$): A binary operator that combines the elements from the two operand sets into a single result set. If an element appears in both operand sets, it will appear only once in the result set.
- Intersection (\cap , $\LaTeX: \cap$): Like union, intersection is a binary operator. Unlike union, the result set contains the elements that the operand sets have in common.
- Difference ($-$, $\LaTeX: -$): In $S - T$, the result set contains those elements of S that are **not** also in T . Alternatively, think of the result set temporarily containing all elements of S . Remove from the result set all items also found in the set $S \cap T$. The remaining elements comprise $S - T$. Some people use the term *relative complement* instead of *difference*, and use \setminus ($\LaTeX: \backslash$) instead of $-$ as the symbol.
- Complement (\overline{set} , $\LaTeX: \overline{set}$): In a given setting, we know that the elements of a set are limited by the universal set. The complement of a set S , \bar{S} , is the set of all elements of the universal set that are not members of S . That is, $\bar{S} = \mathcal{U} - S$.
- Cardinality ($| set |$, $\LaTeX: \mid set \mid$): The cardinality of a set is the quantity of items in the set.

relative complement

Example 204:

Let $\mathcal{U} = \{0, 1, 3, 4, 5, 7, 8, 9\}$, $D = \{1, 3, 5, 7, 9\}$, $E = \{4, 8\}$, and $F = \{1, 4, 7, 9\}$.

- | | |
|--|------------------------------|
| (a) $E \cup F = \{1, 4, 7, 8, 9\}$ | (f) $F - D = \{4\}$ |
| (b) $D \cap F = \{1, 7, 9\}$ | (g) $E - D = \{4, 8\} (= E)$ |
| (c) $\overline{F} = \{0, 3, 5, 8\}$ | (h) $E - E = \emptyset$ |
| (d) $\overline{\emptyset} = \mathcal{U}$ | (i) $ E = 2$ |
| (e) $F \cap \overline{D} = \{4\}$ | (j) $ E - E = 0$ |

The fact that $F - D = F \cap \overline{D}$ in Example 204 is not merely a coincidence, it is also true for all pairs of sets from the same universal set. That is, for any sets A and B whose elements are drawn from the same set \mathcal{U} , $A - B = A \cap \overline{B}$. Operators, such as set difference and complement, that can be defined in terms of primitive operators are sometimes called *derived operators*.

Set operators are subject to rules of *precedence* and *associativity* as are any other operators. You are likely familiar with these concepts from the programming language(s) you know. For example, in Java multiplication has precedence over addition, telling us that, after the evaluation of the statement $a = 4 + 5 * 6$;, a receives the value 34 instead of 54. While you can find precedence and associativity suggestions for set operators, no canonical rules exist.⁸ Rather than adding to the confusion, we will simply use parentheses when necessary to order the operators in our set expressions.

derived operator
precedence
associativity

A.3.3 Set Visualization with Venn Diagrams

To make the set operators easier to understand, *Venn Diagrams* are often used. As Figure A.1 suggests, they are found on the internet nearly as frequently as they are in discrete structures books.

venn diagram

The enclosing rectangle represents the universal set. Frequently, the label \mathcal{U} is placed in the upper-right corner as a reminder. Each set of interest within the universe is represented with a labeled circle or other (usually curved) ‘blob.’ These circles are positioned to overlap in a way that produces a region for every possible relationship between the sets. Figure A.2 shows common Venn diagram representations for two, three, and four sets.

⁸Some applications avoid the issue entirely. The database language SQL handles precedence by considering all set operators to have the same precedence.

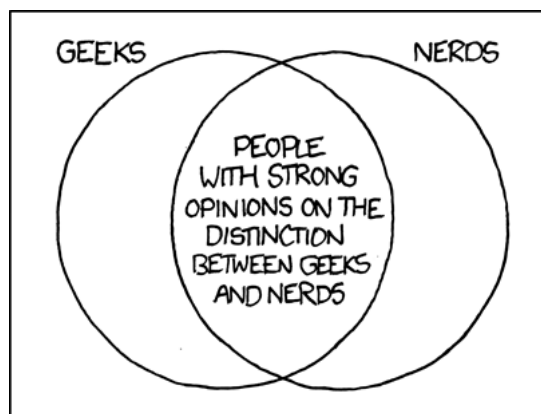


Figure A.1: Geeks and Nerds Venn Diagram (<http://xkcd.com/747/>). Credit: Randall Munroe, xkcd.com.

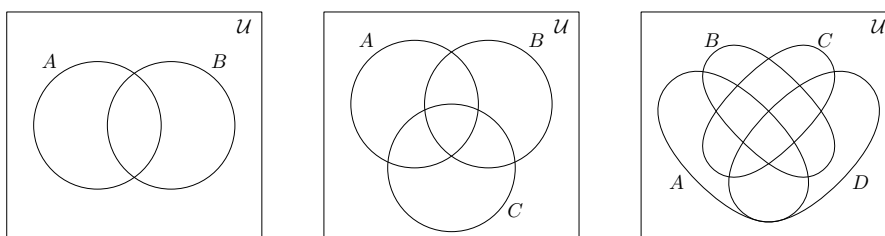


Figure A.2: Standard 2-set, 3-set, and 4-set Venn Diagrams

By shading or otherwise highlighting the appropriate regions, visualizations of the results of set operations can be created. Figure A.3 shows Venn diagram representations of the results of some of the expressions in Example 204. While the diagrams with the bold colors are eye-catching, the versions with the cross-hatching are easier to create by hand by students taking timed quizzes or exams.

A.3.4 Notations for Sets of Numbers

A reasonably well-accepted collection of labels exist that mathematicians use to represent collections of numbers. Here are the labels used in this book:

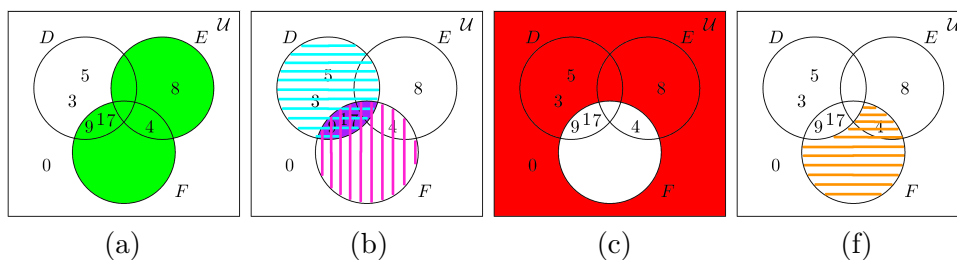


Figure A.3: Venn diagrams of (a) $E \cup F$, (b) $D \cap F$, (c) \bar{F} , and (f) $F - D$. Labels correspond to those of Example 204.

Table 22: Sets of Numbers		
Symbol	Name	Set Representation
\mathbb{Z}	All Integers	$\{\dots, -2, -1, 0, 1, 2, \dots\}$
$\mathbb{Z}^+, \mathbb{N}^+$	Positive Integers	$\{1, 2, 3, \dots\}$
$\mathbb{Z}^*, \mathbb{N}_0$	Non-negative Integers	$\{0, 1, 2, 3, \dots\}$
\mathbb{Z}^-	Negative Integers	$\{\dots, -3, -2, -1\}$
\mathbb{Z}^{even}	Even Integers	$\{\dots, -4, -2, 0, 2, 4, \dots\}$
\mathbb{Z}^{odd}	Odd Integers	$\{\dots, -3, -1, 1, 3, \dots\}$
\mathbb{Q}	Rational Numbers	$\{\frac{a}{b} \mid a, b \in \mathbb{Z} \text{ and } b \neq 0\}$
$\overline{\mathbb{Q}}$	Irrational Numbers	$\{i \mid i \notin \mathbb{Q} \text{ and } i \notin \text{Imaginary}\}$
\mathbb{R}	Real Numbers	$\{\mathbb{Q} \cup \overline{\mathbb{Q}}\}$

Some explanations are in order. \mathbb{Z} (`\mathbb{Z}`) is used for integers because the German word for “integers” is “zahlen,” and many influential mathematicians were German. (That, and several sets of numbers have names that begin with “I” in English, specifically “integer,” “irrational,” and “imaginary.”) \mathbb{Z}^{even} and \mathbb{Z}^{odd} are non-standard notations, but accepted symbols for these sets do not exist and \mathbb{Z}^{even} and \mathbb{Z}^{odd} have the advantage of being easily understood. Finally, $\overline{\mathbb{Q}}$ is as accepted a symbol for the irrational numbers as there is. We have just seen that the overline represents the complement of a set, thus $\overline{\mathbb{Q}}$ represents “not rational,” or “irrational.”

The relationships between most of these can be shown in a non-too-messy Venn diagram; see Figure A.4.

You may be wondering about the omission of natural numbers, a set represented by the symbol \mathbb{N} . The problem: What are the natural numbers, exactly? Most mathematicians now consider zero to be ‘natural,’ but others

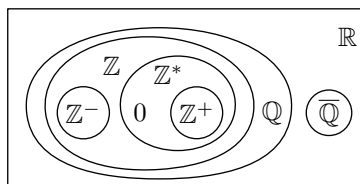


Figure A.4: A Visualization of Number Set Relationships

prefer the traditional interpretation, where a symbol isn't needed until there is something to count. To avoid confusion, we will ignore \mathbb{N} and will use \mathbb{Z}^* when we want natural numbers including zero and \mathbb{Z}^+ when zero is excluded.

A.4 Associativity, Friends and a Distant Cousin

We know that algebraic expressions consist mostly of operands (values), operators (unary as well as binary), and grouping symbols (such as parentheses). Set and logical expressions, both of which are common in discrete structures, have these components as well.

To correctly evaluate an expression, we need to know the ways in which its components can be rearranged without changing the eventual evaluation of the expression. The properties that govern these rearrangements include associativity, commutativity, and distributivity. We include transitivity in this section, too, even though it covers a rather different area than do the others.⁹

A.4.1 Associativity

The associative property covers groupings across binary operators (such as subtraction in arithmetic and intersection in sets).

associativity

Definition 74: Associativity

Assume that \triangle is a binary operator that can be applied to values a , b , and c . The operator \triangle is *associative* if $(a\triangle b)\triangle c = a\triangle(b\triangle c)$ for all acceptable values a , b , and c .

⁹It also ends in “tivity”; for us, that’s reason enough to put it here!

The symbol \triangle in Definition 74 (and the geometric shapes used in the upcoming definitions in this subsection) represents any acceptable binary operator, not a specific one. More formally, associativity is known as the *associative property* and the expression is known as the *associative law*.

Not all operators are associative, as Example 205 shows.

Example 205:

Addition over integers is an associative binary operator: $(9 + 13) + 5 = 9 + (13 + 5)$

Union over sets is also associative: $G \cup (H \cup I) = (G \cup H) \cup I$

Subtraction over integers is **not** associative: $(2 - 3) - 4 \neq 2 - (3 - 4)$.

Be aware that you cannot select just any three values to show that an operator is associative. Consider $(0 - 0) - 0 = 0 - (0 - 0)$. This is certainly true, but that relationship is not generally true. Thus, subtraction over integers is not associative.

A.4.2 Commutativity

Commutativity covers positioning of operands around binary operators.

Definition 75: Commutativity

Assume that \bowtie is a binary operator that can be applied to values d and e . The operator \bowtie is *commutative* if $d \bowtie e = e \bowtie d$ for all acceptable values d and e .

commutativity

Example 206:

Multiplication over integers is commutative: $8 * 5 = 5 * 8$

Intersection over sets is, too: $J \cap K = K \cap J$

Set difference is **not** commutative: $L - M \neq M - L$

A.4.3 Distributivity

Unlike associativity and commutativity, distributivity requires two distinct operators.

distributivity

Definition 76: Distributivity

Assume that \square and \diamond are binary operators that can be applied to values x , y , and z . \square is *distributive* over \diamond if both of these conditions hold:

- (a) $x \square (y \diamond z) = (x \square y) \diamond (x \square z)$ (\square is *left distributive* over \diamond), **and**
- (b) $(y \diamond z) \square x = (y \square x) \diamond (z \square x)$ (\square is *right distributive* over \diamond),

for all acceptable values x , y and z .

Notice that the difference between (a) and (b) in Definition 76 is due to the commutativity of the operator represented by \square . So long as \square is commutative over the values, there is no behavioral distinction between left and right distributivity.

Example 207:

With integers, multiplication is distributive over addition, because, for example, $4 * (7 + 9) = (4 * 7) + (4 * 9)$ and $(7 + 9) * 4 = (7 * 4) + (9 * 4)$ (either way, $64 = 64$).

However, addition is not distributive over multiplication: $3 + (5 * 6) \neq (3 + 5) * (3 + 6)$ ($33 \neq 72$).

A.4.4 Transitivity

As mentioned at the top of this section, transitivity does not fit well with the other properties. It is still important in discrete structures, and even though it should be known to you already, to be complete we should mention it in this appendix somewhere. This section is as good a place as any to cover it.

Definition 77: Transitivity

A binary relationship denoted by a relational operator \odot is *transitive* if, whenever $a \odot b$ and $b \odot c$ are both true, then $a \odot c$ is also true.

transitivity

Binary *relational operators* are those that compare two values. For instance, in mathematics ‘ $<$ ’ is used to indicate that a number n is less than another number m , as in $n < m$. If so, the operator evaluates to true; otherwise, false. As it happens, ‘ $<$ ’ is transitive, as demonstrated by Example 208 and detailed later on in Table 25.

*relational operator***Example 208:**

- (a) $-9 < -3$ and $-3 < 5$. By transitivity of $<$, we know that $-9 < 5$.
- (b) $(3 + 5) = 8$ and $8 = 2^3$. Because $=$ is transitive, we know that $(3 + 5) = 2^3$.

Example 209:

Consider the game Rock-Paper-Scissors.¹⁰ Paper beats rock and rock beats scissors. If the relationship “beats” were transitive, then paper would beat scissors, but of course it does not. Thus, the “beats” operator is not transitive in the context of Rock, Paper, Scissors.

A.5 Properties of Inequalities

Many people are fuzzy on what an “inequality” is, which makes the concept a good place to start. An *inequality* is a comparison between two values that can be true when the values are not equal. Thus, the relational operators \neq , $<$, and $>$ are all unquestionably inequalities. The operators \leq and \geq are also considered to be inequalities, even though $4 \leq 4$ is true (that is, even though

inequality

¹⁰See <http://en.wikipedia.org/wiki/Rock-paper-scissors> for the rules.

these operators can also be true in an equality situation, we consider them to be inequalities because they can be true when the operands differ, as in $4 \leq 5$).

Tables 23, 24, and 25 detail several properties of inequalities that you should know.

Table 23: Inequality Properties of Addition and Subtraction

For real numbers a , b , and c :

	If	Then
(a)	$a < b$	$a + c < b + c$
(b)	$a \leq b$	$a + c \leq b + c$
(c)	$a > b$	$a + c > b + c$
(d)	$a \geq b$	$a + c \geq b + c$
(e)	$a \neq b$	$a + c \neq b + c$

Note: In all of these, addition can be replaced by subtraction.

Table 24: Inequality Properties of Multiplication and Division

For real numbers a , b , and c , when $c \neq 0$ and ...

... when c is positive:

... when c is negative:

	If	Then		If	Then
(a)	$a < b$	$ac < bc$	(f)	$a < b$	$ac > bc$
(b)	$a \leq b$	$ac \leq bc$	(g)	$a \leq b$	$ac \geq bc$
(c)	$a > b$	$ac > bc$	(h)	$a > b$	$ac < bc$
(d)	$a \geq b$	$ac \geq bc$	(i)	$a \geq b$	$ac \leq bc$
(e)	$a \neq b$	$ac \neq bc$	(j)	$a \neq b$	$ac \neq bc$

Note: In all of these, multiplication can be replaced by division.

Table 25: Inequality Properties of Transitivity

For real numbers a , b , and c :

	If	Then		If	Then
(a)	$a < b$ and $b < c$	$a < c$	(e)	$a > b$ and $b > c$	$a > c$
(b)	$a \leq b$ and $b \leq c$	$a \leq c$	(f)	$a \geq b$ and $b \geq c$	$a \geq c$
(c)	$a < b$ and $b \leq c$	$a < c$	(g)	$a > b$ and $b \geq c$	$a > c$
(d)	$a \leq b$ and $b < c$	$a < c$	(h)	$a \geq b$ and $b > c$	$a > c$

Transitivity, in the context of relations, is covered in Chapter 8.

Example 210:

Let $a = 5$, $b = 5$, and $c = 8$. $a \leq b$ is true (5 is less than **or** equal to 5). $b < c$ is also true ($5 < 8$). Because we know $a \leq b$ and $b < c$, by Table 25(d) we may conclude that $a < c$ ($5 < 8$). Note that although we can also say that $5 \leq 8$ is true, stating that $5 < 8$ is more accurate, because it correctly shows the impossibility of $a = c$ being true when $a \leq b$ and $b < c$.

Example 211:

Problem: Demonstrate that $2m > p$ when $6n \geq p$ and $m > 3n$, where all variables are integers.

Solution: By Table 24(c), we can multiply both sides of $m > 3n$ by 2 to produce $2m > 6n$. Table 25(g) allows us to conclude that $2m > p$ because $2m > 6n$ and $6n \geq p$.

A.6 Summation and Product Notations

Expressing lengthy sums (such as $1 + 2 + 3 + 4 + 5 + 6 + 7 + 8$) requires a lot of writing, or some sort of abbreviation technique (perhaps $1 + 2 + \dots + 8$). Such an abbreviation could be misinterpreted. In this case, is $1 + 2 + \dots + 8$ representing $1 + 2 + 3 + 4 + 5 + 6 + 7 + 8$ or $1 + 2 + 4 + 8$ (the sum of the first four positive powers of 2)? To help avoid confusion, mathematicians use summation and product notations.

A.6.1 Summation Notation

summation notation

Summation notation provides an unambiguous way to express the idea that a long sequence of terms that follow a computational pattern should be added together. Here is an example:

$$\sum_{i=0}^3 2^i \quad (\text{\LaTeX: \sum\limits_{i=0}^3 2^i})^{11}$$

This example expresses the sum $1 + 2 + 4 + 8$. Here's how: The variable i takes on integer values, starting with 0 and ending with 3. For each of those four integers, we replace the i in 2^i with the integer and add together the results. Thus:

Example 212:

The expression $\sum_{i=0}^3 2^i$ expands to $2^0 + 2^1 + 2^2 + 2^3$, which is equal to $1 + 2 + 4 + 8 = 15$.

Each part of $\sum_{i=0}^3 2^i$ has a name:

- Σ (`\Sigma`, for a stand-alone version) is the Greek letter (capital) sigma. (The lower case sigma, σ (`\sigma`), is used elsewhere in mathematics, but not to represent summation.)
- i is the *index of summation*, or just *index* for short.

index of summation

¹¹The `\limits` command tells `\LaTeX` to format the limits above and below the sigma. Discarding it produces a more compact version ($\sum_{i=0}^3 2^i$), useful for in-line expressions.

- 0 is the *lower limit*. *lower limit*
- 3 is the *upper limit*. *upper limit*
- 2^i (`\TeX: 2^{i}`) is the *sequence*. *sequence*

A.6.2 Product Notation

Closely related to summation notation is *product notation*, which is identical in appearance except for a different Greek letter, Π (`\TeX: \Pi`), the capital version of the symbol π (`\TeX: \pi`) that represents the ratio of a circle's circumference to its diameter. *product notation*

Here's the same example as above using product instead of summation notation:

$$\prod_{i=0}^3 2^i \quad (\text{\TeX: \prod\limits_{i=0}^3 2^{i}})$$

And, of course, the terms from the sequence are multiplied instead of added.

Example 213:

Changing Σ to Π in Example 212: $\prod_{i=0}^3 2^i = 2^0 \cdot 2^1 \cdot 2^2 \cdot 2^3 = 1 \cdot 2 \cdot 4 \cdot 8 = 64$.
(`\TeX: \cdot` produces the multiplication dot.)

A.6.3 Interpreting Sequences within Sum and Product Notation

Whether using summation or product notation, be careful with the formatting of the sequence, as Example 214 shows.

Example 214:

The summation expression $\sum_{i=1}^2 \frac{4}{i} + i$ is interpreted as $\left(\sum_{i=1}^2 \frac{4}{i}\right) + i$, rather than as $\sum_{i=1}^2 \left(\frac{4}{i} + i\right)$. In most cases, the evaluation of the two versions is likely to be different. Here, $\left(\sum_{i=1}^2 \frac{4}{i}\right) + i = \left(\frac{4}{1} + \frac{4}{2}\right) + i = 6 + i$ (the trailing

i is out of the scope of the summation, and so remains untouched), while

$$\sum_{i=1}^2 \left(\frac{4}{i} + i \right) = \left(\frac{4}{1} + 1 \right) + \left(\frac{4}{2} + 2 \right) = 5 + 4 = 9.$$

A.6.4 Nesting Summation and Product Notations

Can the sequence of a summation include another summation, or even a product? Certainly!

The key to evaluating these nested notations is to work from the inside to the outside. That is, expand the inner-most summation or product first, then deal with the next inner-most, etc. Example 215 demonstrates.

Example 215:

Problem: Evaluate $\sum_{a=1}^5 \sum_{b=0}^2 (2a + b)$.

Solution: Here, $\sum_{b=0}^2 (2a + b)$ is the inner-most summation. Don't worry about the a in the sequence; it will be dealt with when we address the outer summation.

$$\sum_{b=0}^2 (2a + b) = (2a + 0) + (2a + 1) + (2a + 2) = 6a + 3$$

We have simplified the original nested summation to the single summation $\sum_{a=1}^5 (6a + 3)$, which we already know how to evaluate:

$$\begin{aligned} \sum_{a=1}^5 (6a + 3) &= (6 \cdot 1 + 3) + (6 \cdot 2 + 3) + (6 \cdot 3 + 3) + (6 \cdot 4 + 3) + (6 \cdot 5 + 3) \\ &= 9 + 15 + 21 + 27 + 33 \\ &= 105 \end{aligned}$$

Thus, $\sum_{a=1}^5 \sum_{b=0}^2 (2a + b) = 105$.

An observation on Example 215: We can simplify $\sum_{a=1}^5 (6a + 3)$ a bit before starting its evaluation. Doing so will save us some operations. Notice that $6a + 3 = 3(2a + 1)$. Because we are doing a summation, we can multiply the result of the summation by three instead of multiplying each sequence term by three. (You should recognize this as an application of distributivity; see

section A.4. That is, $\sum_{a=1}^5 (6a + 3) = \sum_{a=1}^5 3(2a + 1) = 3 \left(\sum_{a=1}^5 (2a + 1) \right)$. You can simplify the summation even further by turning the “+1” into a “+5” and by pulling out the “2”, but while these steps will simplify the summation, it makes the expression as a whole a bit of a mess. Try it and see for yourself!

To conclude this subsection, Example 216 examines the connection between nested summations and nested FOR loops in programming languages.

Example 216:

Problem: Write nested pseudocode FOR loops that computes $\sum_{a=1}^5 \sum_{b=0}^2 (2a + b)$.

Solution: Each Σ represents an iteration operation over the index of summation, starting with the lower limit and stopping after the upper limit is considered. The parallels between index-controlled loops (such as FOR loops) and summation notation are numerous, making FOR loops a natural stand-in for summation in a program. Apart from the summation-to-FOR-loop translation, we only need a variable to hold the sum being computed.

```
sum <- 0
for a from 1 through 5
  for b from 0 through 2
    sum <- sum + 2a + b
output sum
```

In Example 215, we considered the summations essentially independently, working inside-out. Computers do not treat the instructions representing nested FOR loops that way, but they could (especially if certain code optimizations are applied). We could manually apply the same inside-out process to our code example in Example 216 to eliminate the inner loop:

```
sum <- 0;
for a from 1 through 5
  sum <- sum + 6a + 3
output sum
```

$$\text{divisor (s)} \rightarrow 4 \left| \begin{array}{r} 6 \\ 27 \\ \hline 24 \\ \hline 3 \end{array} \right. \begin{array}{l} \leftarrow \text{quotient (q)} \\ \leftarrow \text{dividend (n)} \\ \leftarrow \text{remainder (r)} \end{array}$$

Figure A.5: Long Division with Component Names

A.7 Integer Division

Remember learning long division of integers in grade school? All of the numbers were integers, and most of the parts had hard-to-remember names. Figure A.5 reminds us of the parts and their names.

To save space, we can summarize the division in Figure A.5 using the representation $27/4 = 6 \text{ r} 3$, where the ‘r’ means ‘remainder.’¹² In discrete structures, the results (the quotient and the remainder) are both of use, but the remainder is of particular interest.

A.7.1 Modulo and Integer Division Operators

modulo

integer division

divides

The *modulo* operator, represented by %, the percent sign (`\%`), is a binary operator that takes the dividend and divisor and produces the remainder of the integer division.¹³ The *integer division* operator, represented by \, the back slash (`\backslash`), takes the same operands but produces the quotient. Finally, the *divides* operator, represented by | (`\mid` or `|`) also accepts the same operands, but in reverse order and returns a boolean result: True when the division results in no remainder, false otherwise. The reversing of the operands is important to remember; that is, $16 \% 2 = 0$ corresponds to $2 \mid 16$, and we would read it aloud as “2 divides 16.”

Example 217:

Some parallel examples of the modulo, integer division, and divides operators:¹⁴

¹²If you want to be old-school, you can use the \div symbol to represent division (`\div`). It’s called an *obelus* and is not, to our knowledge, a part of a Borg ship. Yet.

¹³Sometimes people refer to the modulo operator as the ‘modulus’ operator. ‘Modulus’ can mean several things, but none of them is an alternate name for modulo.

(a) $27 \% 4 = 3$	(f) $27 \setminus 4 = 6$	(k) $4 \mid 27 = \text{False}$
(b) $4 \% 4 = 0$	(g) $4 \setminus 4 = 1$	(l) $4 \mid 4 = \text{True}$
(c) $4 \% 27 = 4$	(h) $4 \setminus 27 = 0$	(m) $27 \mid 4 = \text{False}$
(d) $0 \% 5 = 0$	(i) $0 \setminus 5 = 0$	(n) $5 \mid 0 = \text{True}$
(e) $3 \% 0 = \text{Undef.}$	(j) $3 \setminus 0 = \text{Undef.}$	(o) $0 \mid 3 = \text{Undef.}$

If any of the modulo examples from Example 217 are confusing, (c) is probably the one. Think of it this way: In $27 \% 4$, we ask ourselves how many instances of the divisor (4) we can subtract from the dividend (27) before what's left (the remainder) is less than the divisor. To evaluate $4 \% 27$, we ask the same question. Because $4 < 27$, we cannot subtract any instances of 27 from 4 without the result being less than 27, and so the remainder is the original dividend: 4.

A word of caution: When writing “/”, “\”, and “|” by hand, be careful to indicate the slope (or the lack thereof!) clearly, so that the reader (often the person grading your answer) understands which operator you are using. The division symbols are yet another reason to use a scientific document processing system such as L^AT_EX to format your work whenever possible.

A.7.2 Congruences

Two integers are congruent if they produce the same remainder when divided by a given value. The following definition has the details.

Definition 78: Congruent

If $b, r \in \mathbb{Z}$ and $m \in \mathbb{Z}^+$, then b and r are *congruent* modulo m (written $b \equiv r \pmod{m}$) if and only if $b \% m = r \% m$ (or, if and only if $m \mid (b - r)$). (L^AT_EX: `\equiv` produces ‘ \equiv ’)

congruent

Some terminology to go with Definition 78: b is the *base*, r is the *remainder* or *residue*, and m is the *modulus*.¹⁵

¹⁴Why no examples with negative dividends or divisors? Because they really complicate the story, that's why! There's more than one way to handle negatives with modulo, and negatives don't often appear with modulo in computer science applications. And before you ask about floating point values: This is a *discrete* structures book! (That's the answer whenever we don't feel like dealing with real numbers, for any reason. Yes, it's very handy.)

¹⁵And here is one of those uses of ‘modulus’ now!

Example 218:

Problem: Are 35 and 108 congruent modulo 17?

Solution: As Definition 78 states, $35 \equiv 108 \pmod{17}$ if $17 \mid (35 - 108)$. $35 - 108 = -73$, and $17 \mid -73$ is false. (73 is not a multiple of 17, so 17 cannot divide -73 evenly.) Thus, 35 and 108 are not congruent modulo 17 (which can be written: $35 \not\equiv 108 \pmod{17}$). (L^AT_EX: `\not\equiv`)

If the -73 from Example 218 bothers you, you can switch the base and the residue: $35 \equiv 108 \pmod{17}$ is equivalent to $108 \equiv 35 \pmod{17}$.

Congruences are useful in many advanced areas of computer science, including hash functions and cryptography. Cryptography is hard, though; telling time is comparatively easy, and congruences apply to time, too, as Example 219 shows.

Example 219:

Problem: It is now 7:50. What time will it be in 155 minutes?

Solution: As we well know, there are 60 minutes in an hour. An integer division ($155/60 = 2 \text{ r } 35$) reveals that 155 minutes is the same as 2 hours and 35 minutes. As a congruence, $155 \equiv 35 \pmod{60}$. Adding 2:35 to 7:50 by adding hours to hours and minutes to minutes gives a time of 9:85, which still needs some work because $85 \geq 60$. Congruences to the rescue: $85 \equiv 25 \pmod{60}$. Replacing the 85 in 9:85 with the residue (25) and incrementing the hours by one produces the answer: In 155 minutes, it will be 10:25.

A.8 Evens and Odds¹⁶

When we introduce proof techniques in Chapters 4 and 5, we will start with simple examples that frequently involve basic properties of integers. Two of those properties are the ideas of ‘even’ and ‘odd’ numbers.

¹⁶You’re thinking of skipping this section, aren’t you? It’s only a page long! Read it!

Definition 79: Even

An integer n is *even* if and only if there exists an integer k such that $n = 2k$.

even

There are other equally valid ways of defining even. For example, n is even if and only if $2 \mid n$. Another: n is even if and only if $n \% 2 = 0$. A third: n is even if and only if $n \equiv 0 \pmod{2}$. Why, then, did we choose the one that needs a second variable to be defined? Because, for many of our proof examples, that form is the most useful.

Please note: Regardless of the way the concept of evenness is expressed, the integer zero is an even number. Zero is a special case when we are dividing by integers, and zero is a special case when we consider positive and negative values, but zero is not a special case with evens and odds: Zero is even.

With ‘even’ defined, we can move on to ‘odd’:

Definition 80: Odd

An integer n is *odd* if and only if there exists an integer k such that $n = 2k + 1$.

odd

All of the other ways we gave to define ‘even’ can be adjusted to define ‘odd’: n is odd if and only if $2 \nmid n$ (`\nmid`), or if and only if $n \% 2 = 1$, or if and only if $n \equiv 1 \pmod{2}$. We can even adjust the ‘ k ’ definition by adding (or subtracting) any odd integer from $2k$. For example, $n = 2k - 1001$ is just as valid as $n = 2k + 1$. The advantage of the “+1” form is that applying functions, such as squaring, to $2k + 1$ is usually much easier than applying them to an expression such as $2k - 1001$.

Finally, notice that we can partition (see Section 6.3.5) all integers into one or the other of these two subsets. That is, no integer is both even and odd, and no integer (not even zero!) is neither even nor odd.

A.9 Logarithms and Exponents

In the days before reliable calculating devices, logarithms were used to simplify otherwise tedious-to-evaluate expressions. We use them today in computer science primarily for algorithm analysis. For example, consider the binary

search algorithm. The amount of work a computer performs in carrying out that algorithm is described in terms of a logarithm of the quantity of data being searched.

The name “logarithm” frightens nearly as many students as does the word “proof”. You may find them less imposing if you know that they are just exponents written in a different way. When working with integers, we know that a sequence of products of the same value can be more tersely expressed using an exponent. For example, $3 \cdot 3 \cdot 3 \cdot 3 = 3^4 = 81$. In the expression $3^4 = 81$, 3 is the *base*, 81 is the *power*, and 4 is the *logarithm* (or *exponent*). Definition 81 gives another way to structure exponential expressions.

blp relationship

Definition 81: BLP Relationship

Assume that b , l , and p are positive real numbers. The Base–Logarithm–Power (‘BLP’)¹⁷ relationship states:

$$b^l = p \text{ if and only if } \log_b p = l \quad (\text{\LaTeX: \log_2})$$

where b is the base, l is the logarithm, and p is the power.

Example 220:

Problem: Evaluate: $\log_4 64$.

Solution: Let’s name the result of the evaluation of this expression x . That is, $\log_4 64 = x$. By BLP (Definition 81), we can rewrite $\log_4 64 = x$ as $4^x = 64$. Because $64 = 4^3$, $x = 3$, and so $\log_4 64 = 3$.

¹⁷The right-hand side of the relationship ($\log_b p = l$) is sometimes referred to as the *logarithmic function*, but if this entire biconditional relationship has a formal name, no reference I consulted uses it. Such a key idea deserves some sort of name, so we introduce Base–Logarithm–Power, or ‘BLP’ for short (and suggest the pronunciation “blip”).

Example 221:

Problem: Evaluate: $\log_x x$ when $x > 0$.

Solution: At first glance, you might think that the problem is too vaguely specified for an evaluation to be performed. However, by applying BLP, we can transform $\log_x x = y$ into $x^y = x$. As x is just a short-cut way of writing x^1 , we see that $y = 1$, and so $\log_x x = 1$. This is a useful identity to remember.

There are many exponent and logarithm identities that are good to learn. Table 26 summarizes some of the laws that we think are worth knowing.

Table 26: Laws of Exponents and Logarithms

Laws of Exponents		Laws of Logarithms	
(a)	$w^{x+y} = w^x w^y$	(f)	$\log_b(x^y) = y \log_b x$
(b)	$(w^x)^y = w^{xy}$	(g)	$\log_b(xy) = \log_b x + \log_b y$
(c)	$v^x w^x = (vw)^x$	(h)	$\log_b\left(\frac{x}{y}\right) = \log_b x - \log_b y$
(d)	$\frac{w^x}{w^y} = w^{x-y}$	(i)	$b^{\log_b x} = x$
(e)	$\frac{v^x}{w^x} = \left(\frac{v}{w}\right)^x$	(j)	$\log_a x = \frac{\log_b x}{\log_b a}$

Example 222:

Problem: Evaluate $\frac{\log_5 4 + \log_5 8}{\frac{1}{\log_2 5}}$.

Solution: We start by making the log bases the same, and simplify from there.

$\frac{\log_5 4 + \log_5 8}{\frac{1}{\log_2 5}} = \frac{\log_5 4 + \log_5 8}{\left(\frac{1}{\log_5 5}\right) \left(\frac{1}{\log_5 2}\right)}$	Table 26, Line j, with $b = 5$
$= \frac{\log_5 4 + \log_5 8}{\log_5 2}$	Result of Ex. 221, and $1/(1/y) = y$
$= \frac{\log_5 32}{\log_5 2}$	Table 26, Line g
$= \log_2 32$	Table 26, Line j
$= \log_2 2^5$	It's good to know powers of 2!
$= 5 \log_2 2$	Table 26, Line f
$= 5$	Example 221 again

To help put logarithmic functions in context, Figure A.6 shows a plot of three functions: $f(n) = 1$ (a constant function), $f(n) = \log_2 n$ (a logarithmic function commonly used in computer science), and $f(n) = n$ (a linear function), with added axis lines for $n = 1$ and $n = 2$. There are several items of interest in this plot. First, observe how slowly $\log_2 n$ grows; it is much closer to $f(n) = 1$ than to $f(n) = n$ as n becomes larger. Second, observe that $\log_2 n = 1$ when $n = 2$ (not a surprise, now that we know $\log_b b = 1$). Third, $\log_2 1 = 0$ (also not a surprise, as $b^0 = 1$ implies that $\log_b 1 = 0$). Fourth, when $n < 1$, $\log_2 n < 0$. In computer science, n usually represents the size of a problem in terms of the size of the data structure (that is, the quantity of data items stored). Thus, values of n below 1 aren't often of interest.

A.10 Working with Quadratic Equations

We have occasion to factor quadratic equations in this book. Not many occasions, but we do have them. One reason to factor them is to find their roots.

quadratic

Definition 82: Quadratic

A *quadratic* is an polynomial expression in one variable of the form $ax^2 + bx + c$, where a , b , and c are real constants with $a \neq 0$.

quadratic equation

Setting a quadratic equal to a value creates a *quadratic equation*.

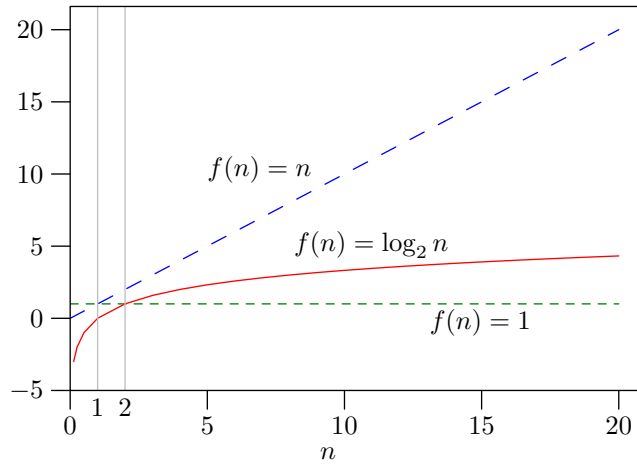


Figure A.6: A linear-axes plot of $f(n) = 1$, $f(n) = \log_2 n$, and $f(n) = n$.

A.10.1 Factoring Quadratics

First, two definitions.

Definition 83: Perfect Square

An integer n is a *perfect square* if the product of some integer m with itself equals n . Alternatively, n is a perfect square if its square root is an integer.

perfect square

Definition 84: Quadratic Formula

Given a quadratic equation $ax^2 + bx + c = 0$, the value(s) of x that make this equation true (known as the *root(s)* of the equation) are given by the evaluation of the *quadratic formula*: $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$. The expression within the square root, $b^2 - 4ac$, is the *discriminant*.

quadratic formula

There are several techniques used to factor quadratic polynomials. As this is a discrete structures book, we will limit ourselves to quadratics that can be

discriminant

factored using only integers. How do you know? If a , b , and c are integers and the evaluation of $b^2 - 4ac$ (known as the *discriminant*; see Definition 84) is a perfect square, then the roots will be integers.

Because we are sticking with integers, factoring quadratics will be comparatively easy to accomplish. Here are three techniques:

- (1) *'Guess and Check'*. Despite the inelegant name, this method does have a formal origin: Vieta's formulas.¹⁸ Easiest to use when $a = 1$, in 'guess and check' we look for values y and z such that $x^2 + bx + c = (x + y)(x + z) = x^2 + (y + z)x + yz$. Note that $y + z = b$ and $y \cdot z = c$. To apply this method, we 'guess' a pair of factors of c , and 'check' that they satisfy those two equations.
- (2) *The AC Method*. This method, which is a generalization of 'guess and check', is likely to be slower when applied by hand, but you may find it to be easier to use when $a \neq 1$. The idea comes from this equation: $ax^2 + bx + c = \frac{a^2x^2 + abx + ac}{a} = \frac{(ax+d)(ax+e)}{a}$, where d and e are factors of the product $a \cdot c$ such that $d \cdot e = a \cdot c$, and their sum equals b . As with 'guess and check', the hard part is finding the factors d and e ; after that, just substitute the values and simplify. The denominator a will divide away when the discriminant is a perfect square.
- (3) *Apply the quadratic formula*. The factors of $ax^2 + bx + c$ are a , $x - \frac{-b + \sqrt{b^2 - 4ac}}{2a}$, and $x - \frac{-b - \sqrt{b^2 - 4ac}}{2a}$. That is,

$$ax^2 + bx + c = a \left(x - \frac{-b + \sqrt{b^2 - 4ac}}{2a} \right) \left(x - \frac{-b - \sqrt{b^2 - 4ac}}{2a} \right).$$

When the discriminant is a perfect square, the square roots will evaluate to integers. This method is as general as they come, but it does rely on your memory of the quadratic formula and your ability to perform a fair amount of math by hand without making any mistakes.

Example 223:

Problem: Factor $x^2 - x - 6$ using 'guess and check'.

Solution: To match the additions in the form $ax^2 + bx + c$, we need

¹⁸wikipedia.org/wiki/Vieta's_formulas

b and c to be negative; specifically, $a = 1$, $b = -1$, and $c = -6$. In ‘guess and check’, we need to find factors of -6 . Ignoring the negation for a moment, the factors of 6 are 1, 2, 3, and 6. We need a pair of these four factors whose product is 6. The only two pairs that work are 1 & 6 and 2 & 3. To get $yz = -6$ from these factors, there are four possibilities for y and z : -1 & 6, 1 & -6 , -2 & 3, and 2 & -3 . To satisfy the other equation ($y + z = -1$), only the pair 2 & -3 works. Thus, $x^2 - x - 6 = (x + 2)(x - 3)$.

Notice that the pairings of the factors are the smallest with the largest, the second-smallest with the second-largest, etc. You do not need to consider all possible pairings of the factors, because the products of the other pairings won’t equal c .

Example 224:

Problem: Factor $6x^2 + 13x - 5$ using the AC method.

Solution: Here, $a = 6$, $b = 13$, and $c = -5$. In the AC method, we want to find factors of ac named d and e such that $\frac{(ax+d)(ax+e)}{a}$, subject to $de = ac$ and $d + e = b$. $ac = -30$, and 30’s factors are 1, 2, 3, 5, 6, 10, 15, and 30. Possible d & e pairs are 1 & 30, 2 & 15, 3 & 10, and 5 & 6, for a total of 8 pairs when negatives are added (remember, we need the product to be -30). Only the pair -2 & 15 sum to b . Substituting and simplifying:

$$\begin{aligned} \frac{(ax+d)(ax+e)}{a} &= \frac{(6x-2)(6x+15)}{6} && \text{Substitution} \\ &= \frac{(2)(3x-1)(3)(2x+5)}{6} && \text{Factor the binomials} \\ &= (3x-1)(2x+5) && \text{Divide away the 6's} \end{aligned}$$

Thus, $6x^2 + 13x - 5 = (3x - 1)(2x + 5)$.

Example 225:

Problem: Factoring $6x^2 + 13x - 5$ using the AC method was a pain, with all of those factors of 30!

Solution: You can always use the quadratic formula:

$$\begin{aligned}
 ax^2 + bx + c &= a \left(x - \frac{-b + \sqrt{b^2 - 4ac}}{2a} \right) \left(x - \frac{-b - \sqrt{b^2 - 4ac}}{2a} \right) \\
 &= 6 \left(x - \frac{-13 + \sqrt{13^2 - (4 \cdot 6 \cdot -5)}}{2 \cdot 6} \right) \left(x - \frac{-13 - \sqrt{13^2 - (4 \cdot 6 \cdot -5)}}{2 \cdot 6} \right) \\
 &= 6 \left(x - \frac{-13 + \sqrt{289}}{12} \right) \left(x - \frac{-13 - \sqrt{289}}{12} \right) \\
 &= 6 \left(x - \frac{1}{3} \right) \left(x - \frac{-5}{2} \right) \\
 &= 6 \left(\frac{3x-1}{3} \right) \left(\frac{2x+5}{2} \right) \\
 &= (3x-1)(2x+5)
 \end{aligned}$$

After trading the pair of factors of 30 for the tag-team pairs of square roots and fractions, once again $6x^2 + 13x - 5 = (3x - 1)(2x + 5)$.

A.10.2 Roots of Quadratic Equations

roots

As mentioned in Definition 84, the value(s) for x that cause $ax^2 + bx + c = 0$ to be true are known as the *root(s)* of the quadratic equation.

binomial

Having refreshed our memory of how to factor quadratics, finding the root(s) of (a.k.a. solving) a quadratic equation is easy. Each quadratic factors to two *binomials* (expressions that are the sum of two terms; e.g., $ax + d$). If either evaluates to zero, the evaluation of the entire quadratic must be zero. It follows that the roots of the quadratic equation $ax^2 + bx + c = (hx + i)(jx + k) = 0$ are simply $r_1 = -\frac{i}{h}$ and $r_2 = -\frac{k}{j}$. This shows that a quadratic equation has at most two unique roots, a situation that will occur when the discriminant is positive. A single root ($r_1 = r_2$) exists when the discriminant evaluates to zero, and no (real) roots exist otherwise.

Example 226:

Problem: Solve (that is, determine the roots of) $6x^2 + 13x - 5$.

Solution: In Examples 224 and 225, we factored this quadratic and learned that $6x^2 + 13x - 5 = (3x - 1)(2x + 5)$. To make $(3x - 1)(2x + 5) = 0$, we need $3x - 1 = 0$ or $2x + 5 = 0$. That is, either $x = \frac{1}{3}$ or $x = -\frac{5}{2}$. Thus, the roots of $6x^2 + 13x - 5 = 0$ are $r_1 = \frac{1}{3}$ and $r_2 = -\frac{5}{2}$.¹⁹

A.11 Positional Number Systems

Number systems are not core discrete mathematics topics. However, values drawn from them are often used as the subject of counting problems in discrete math, and as you will see such problems in this book, the basics of such systems are worth providing.

Fundamentally, a *number system* is any representation used to express quantities. Using pebbles to represent heads of cabbage is a number system, although not one that scales well to corporate farming. The *decimal* system we use daily is also an example of a number system, but more specifically a *positional number system*. Positional systems are useful because they have symbols that individually represent small quantities, but those same symbols can also represent much larger quantities if the symbols are grouped and ordered.

number system

decimal

Definition 85: Positional System

A *positional (number) system* represents quantities using a combination of symbols (known as *digits* or *glyphs*) and relative locations of those symbols to represent quantities abstractly. Specifically, a positional system of base b employs a set of b digits. A sequence of these digits $d_n d_{n-1} \dots d_1 d_0$ represents an integer value of base b equivalent to the decimal quantity

$$d_n \cdot b^n + d_{n-1} \cdot b^{n-1} + \dots + d_1 \cdot b^1 + d_0 \cdot b^0 = \sum_{i=0}^n d_i b^i.$$

positional system

By changing the base, we can create an infinite number of positional number systems. Happily, only a few of those are commonly encountered in computer science.

A.11.1 Decimal (Base 10)

Decimal is a good system with which to start, because we are already quite familiar with it. We all learned, a long time ago, that there are ten digits available in decimal: ‘0’, ‘1’, ‘2’, ‘3’, ‘4’, ‘5’, ‘6’, ‘7’, ‘8’, and ‘9’. As Definition 85 says, the base of the system matches the number of available digits. Thus, we know decimal’s base is ten. Each digit position in a decimal number represents

¹⁹Yeah, OK, this example’s roots aren’t integers. But they are rational!

a power of ten, and the digit in that position tells us how many of that power we include in the representation of the value.

Example 227:

681 in decimal is a representation of the sum of 600, 80, and 1. That is, $681 = 6 \cdot 10^2 + 8 \cdot 10^1 + 1 \cdot 10^0$. Beats the heck out of carrying around 681 pebbles!

A.11.2 Binary (Base 2)

Modern computers store information in primary storage as sequences of electrical charges – ‘power is present’ and ‘no power is present’. Using this two-valued system as the basis for storing data naturally led to the use of the *binary* positional number system.

As the name ‘binary’ suggests, the base is two. As the necessary quantity of digits is less than the quantity used by decimal, rather than create two new glyphs, binary simply borrows the first two glyphs from decimal: ‘0’ and ‘1’. (A single binary digit is called a *bit* ²⁰.) To distinguish binary values from decimal values, we usually append a subscript of ‘2’ at the end of a binary value, as in ‘1011011₂’. (You can append a ‘10’ to the end of decimal values if you wish, but in practice un-subscripted values are assumed to be base 10.)

Because binary, like decimal, is a positional system, we can determine the quantity being represented using the summation provided in Definition 85, as Example 228 demonstrates.

Example 228:

Problem: Determine the decimal quantity represented by the binary value 1011011₂.

Solution: Each position of a binary value represents a power of 2, starting with 2⁰ on the right and increasing the exponent by one for each position we move to the left. Writing the position values beneath the digits is sometimes helpful:

²⁰The name *bit* is just a concatenation of the letters ‘bi’ from ‘binary’ and ‘t’ from ‘digit’.

Actually, this same process works to convert decimal to any other base with one small change: Take the description above and replace each occurrence of ‘2’ by ‘base b ’. Algorithm 1 details the process. Note that a stack is used to hold the remainders and produce them in the proper order. If you do not know how a stack operates, you can replace it with an array. Store the remainders starting at index 0, and output them in reverse order.

Algorithm 26: Decimal to Base b Conversion

```

1  dividend <- decimal value
2  divisor <- base b
3  initialize stack
4  loop
5      quotient <- dividend \ divisor
6      remainder <- dividend % divisor
7      push remainder onto stack
8      dividend <- quotient
9      if dividend is 0, leave loop
10 end loop
11 pop stack content to output

```

If you need to do a base conversion manually, the hardest part is remembering the order in which to write the remainders. Remembering what the stack does for us in Algorithm 1 is one good way to get it right. Another is to look at Example 229 and think about continuing the division process longer than is necessary. Doing so will create a bunch of zeros. Only the left side of the sequence of remainders can have zeros added to it without changing the result. That is, $1_2 = 0001_2 \neq 1000_2$.

The really annoying part of the division algorithm is the need to do a lot of divisions. There is an alternative approach that works well for small decimal-to-binary conversions and can be done in your head a bit more easily than several long divisions.

Example 230:

Problem: Determine the binary representation of the decimal value 103 again ... this time with only the powers of your mind!

Solution: What we need is the correct collection of powers of 2 that sum to 103. Start by thinking of the largest power of 2 that is less than or equal to 103: 64. Write down a ‘1’, and compute $103 - 64$: 39. Now consider all of the powers of 2 from the next smaller power of 2 (32) down to and including 0, in decreasing order. For each one, if we need it to reach the remaining amount, write down a ‘1’ and subtract it to get the new remaining amount (write down a ‘0’ otherwise). Stop when the remaining amount is zero. In this example, we need 32 to get to 39; we write down a ‘1’ and compute the difference $39 - 32 = 7$. We don’t need 16 or 8 (write down two ‘0’s). At this point, it is likely obvious that we need all of the rest (4, 2, and 1) to reach 7, and so we end with three more ‘1’s. Our final collection is: 1100111.

A.11.3 Octal (Base 8)

Binary may be what computers like to use, but most human beings would rather not have to work with long strings of zeros and ones. A compromise positional number system is *octal* (base 8). This is a compromise because the representations are shorter (easier for people to remember) but also easy to convert back and forth to binary. As with binary, we can use a subset of the decimal digits for octal representations, specifically the digits 0 through 7. As octal is also a positional number system, each digit position is worth a power of 8. A difference from binary is that we have the ability to have several of any given power of 8 included in our sum; that is, 42_8 means that we have 4 8s plus 2 1s. In binary, we could only accept a power of two or ignore it.

octal

The decimal-to-octal and octal-to-decimal conversions can be done as we did them for binary (repeated multiplications or divisions by 8), or we can do them in two steps: decimal-to-binary-to-octal and octal-to-binary-to-decimal.

Example 231:

Problem: Convert 2065_8 to decimal.

Solution: We will do this in two ways: Through binary, and directly.

Option #1: Octal-to-binary-to-decimal. We start with an observation: A three-digit binary number has possible decimal values ranging from

0 (000) through 7 (111), exactly the range of the octal digits. A consequence is that we can convert octal to binary by simply expanding each octal digit to its three-digit binary representation. In this example, $2065_8 \rightarrow (010)(000)(110)(101) \rightarrow 10000110101_2$ (after dropping the extraneous leading 0). Having seen Example 228, we know what to do next: $1024 + 32 + 16 + 4 + 1 = 1077_{10}$.

Option #2: Octal-to-decimal. To accomplish this, we just apply the sum from Definition 85: $2 \cdot 8^3 + 6 \cdot 8^1 + 5 \cdot 8^0 = 1024 + 48 + 5 = 1077$.

Example 232:

Problem: Determine the octal representation of the decimal value 103.

Solution: We will also do this example in two ways.

Option #1: Decimal-to-binary-to-octal. We have already accomplished the first half of this in Example 229: $103_{10} = 1100111_2$. To convert from binary to octal, we can construct groups of three binary digits and convert them to octal digits. Note that *the grouping by threes must start on the right and move left!* That is, $1100111_2 \rightarrow (1)(100)(111) \rightarrow 147_8$.

Option #2: Decimal-to-octal. We can apply repeated divisions, as used in Example 229:

$$\begin{array}{r} \text{ R } 1 \\ 8 \overline{) 1} \text{ R } 4 \\ 8 \overline{) 12} \text{ R } 7 \\ 8 \overline{) 103} \end{array}$$

Thus, $103_{10} = 147_8$ (and also verifies the result of Option #1).

A.11.4 Hexadecimal (Base 16)

We motivated octal by presenting it as a compromise between binary and decimal. *Hexadecimal* ('hex' for short) can be thought of in the same way. The name 'hexadecimal' suggests its base: Together, 'hex' for six and 'decimal' for ten give 16 as the base, which, like base eight for octal, is a power of two.

hexadecimal

But we have a new problem: A base 16 system needs 16 digits, and we only have the ten from decimal to work with. The solution is to use the first six letters of the alphabet ('A' through 'F') to represent the values 10 through 15. Table 27 will be a handy reference until you become more familiar with the system.

Table 27: Binary, Octal, Decimal, and Hex: 0 through 15₁₀

Binary	Octal	Decimal	Hex	Binary	Octal	Decimal	Hex
0000	0	0	0	1000	10	8	8
0001	1	1	1	1001	11	9	9
0010	2	2	2	1010	12	10	A
0011	3	3	3	1011	13	11	B
0100	4	4	4	1100	14	12	C
0101	5	5	5	1101	15	13	D
0110	6	6	6	1110	16	14	E
0111	7	7	7	1111	17	15	F

Hexadecimal is more commonly used than is octal for two main reasons. First, because it has a base larger than 10, hex representations of values often require fewer digits than do the octal or the decimal representations. Second, a hex digit is representable with four binary digits, which is exactly half of a byte.²¹ Thus, we can represent the content of a byte with no more than two hex digits instead of up to three decimal or octal digits.

If you understand conversions between decimal, binary, and octal, you understand them for hexadecimal; just remember to use base 16 instead of base 8. The conversions between octal and hex are easily performed via binary.

Example 233:

Problem: Express 1352₈ in hexadecimal.

Solution: First, we convert the octal digits into three-digit binary values to get the binary equivalent: 1352₈ → (001)(011)(101)(010) → 1011101010₂. To convert from binary to hex, we can group the bits by fours (as we did

²¹Half a byte is called a *nybble*. Byte ...tiny bite ...nybble; get it? 'Nibble' is an acceptable alternate spelling.

by threes for octal). Again, we must group from right to left: $1011101010_2 \rightarrow (10)(1110)(1010) \rightarrow 2EA_{16}$.

A.11.5 Other Bases

As we hope is becoming clear from the preceding examinations of binary, octal, and hexadecimal, any integer value greater than one can be used as the base in a positional number system. Definition 85 can be used to convert any of them to decimal, and the division technique of Example 229 can convert from decimal to them.

Example 234:

Problem: Determine the decimal value of 413_5 .

Solution: By Definition 85: $4 \cdot 5^2 + 1 \cdot 5^1 + 3 \cdot 5^0 = 100 + 5 + 3 = 108_{10}$.