MITOPENCOURSEWARE

Mathematics for Computer Science

by Eric Lehman, F. Thomson Leighton, & Albert R. Meyer

CC BY NC SA

Mathematics for Computer Science

revised Monday 18th May, 2015, 01:43

Eric Lehman
Google Inc.

F Thomson Leighton
Department of Mathematics
and the Computer Science and AI Laboratory,
Massachussetts Institute of Technology;
Akamai Technologies

Albert R Meyer
Department of Electrical Engineering and Computer Science
and the Computer Science and AI Laboratory,
Massachussetts Institute of Technology

2015, Eric Lehman, F Tom Leighton, Albert R Meyer. This work is available under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 license.

Contents

IV Probability

V Recurrences

I Proofs

Introduction

This text explains how to use mathematical models and methods to analyze prob lems that arise in computer science. Proofs play a central role in this work because the authors share a belief with most mathematicians that proofs are essential for genuine understanding. Proofs also play a growing role in computer science; they are used to certify that software and hardware will *always* behave correctly, some

thing that no amount of testing can do.

Simply put, a proof is a method of establishing truth. Like beauty, "truth" some times depends on the eye of the beholder, and it should not be surprising that what constitutes a proof differs among fields. For example, in the judicial system, *legal* truth is decided by a jury based on the allowable evidence presented at trial. In the business world, *authoritative* truth is specified by a trusted person or organization, or maybe just your boss. In fields such as physics or biology, *scientific* truth is confirmed by experiment.[1] In statistics, *probable* truth is established by statistical analysis of sample data.

Philosophical proof involves careful exposition and persuasion typically based on a series of small, plausible arguments. The best example begins with "Cogito ergo sum," a Latin sentence that translates as "I think, therefore I am." This phrase comesfrom the beginning of a 17th century essay by the mathematician/philosopher,
Rene´ Descartes, and it is one of the most famous quotes in the world: do a web search for it, and you will be flooded with hits.

Deducing your existence from the fact that you're thinking about your existence is a pretty cool and persuasive-sounding idea. However, with just a few more lines

[1]Actually, only scientific *falsehood* can be demonstrated by an experiment—when the experiment fails to behave as predicted. But no amount of experiment can confirm that the *next* experiment won't fail. For this reason, scientists rarely speak of truth, but rather of *theories* that accurately predict past, and anticipated future, experiments.

of argument in this vein, Descartes goes on to conclude that there is an infinitely beneficent God. Whether or not you believe in an infinitely beneficent God, you'll probably agree that any very short "proof" of God's infinite beneficence is bound to be far-fetched. So even in masterful hands, this approach is not reliable.

Mathematics has its own specific notion of "proof."

Definition. A *mathematical proof* of a *proposition* is a chain of *logical deductions* leading to the proposition from a base set of *axioms*.

The three key ideas in this definition are highlighted: *proposition*, *logical deduc tion*, and *axiom*. Chapter 1 examines these three ideas along with some basic ways of organizing proofs. Chapter 2 introduces the Well Ordering Principle, a basic method of proof; later, Chapter 5 introduces the closely related proof method of induction.

If you're going to prove a proposition, you'd better have a precise understand ing of what the proposition means. To avoid ambiguity and uncertain definitions in ordinary language, mathematicians use language very precisely, and they often express propositions using logical formulas; these are the subject of Chapter 3.

The first three Chapters assume the reader is familiar with a few mathematical concepts like sets and functions. Chapters 4 and 7 offer a more careful look at such mathematical data types, examining in particular properties and methods for proving things about infinite sets. Chapter 6 goes on to examine recursively defined data types.

0.1 References [11],

[45], [1]

1

What is a Proof?

1.1 Propositions

Definition. A *proposition* is a statement (communication) that is either true or false.

For example, both of the following statements are propositions. The first is true, and the second is false.

Proposition 1.1.1. *2 + 3 = 5.*

Proposition 1.1.2. *1 + 1 = 3.*

Being true or false doesn't sound like much of a limitation, but it does exclude statements such as "Wherefore art thou Romeo?" and "Give me an *A*!" It also ex cludes statements whose truth varies with circumstance such as, "It's five o'clock," or "the stock market will rise tomorrow."

Unfortunately it is not always easy to decide if a proposition is true or false:

Proposition 1.1.3. *For every nonnegative integer,* n*, the value of* $n^2 C n C 41$ *is prime.*

(A *prime* is an integer greater than 1 that is not divisible by any other integer greater than 1. For example, 2, 3, 5, 7, 11, are the first five primes.) Let's try some numerical experimentation to check this proposition. Let

$$p.n/ \text{ WWD } n^2 C n C 41:^{1} \quad (1.1)$$

We begin with p.0/ D 41, which is prime; then

$$p.1/ D 43;\ p.2/ D 47;\ p.3/ D 53;\ : : : ;\ p.20/ D 461$$

are each prime. Hmmm, starts to look like a plausible claim. In fact we can keep checking through n D 39 and confirm that p.39/ D 1601 is prime.

But $p.40/ D 40^2 C 40 C 41 D 41 \ \ 41$, which is not prime. So it's not true that the expression is prime *for all* nonnegative integers. In fact, it's not hard to show that *no* polynomial with integer coefficients can map all nonnegative numbers into

[1]The symbol WWD means "equal by definition." It's always ok simply to write "=" instead of WWD, but reminding the reader that an equality holds by definition can be helpful.

prime numbers, unless it's a constant (see Problem 1.17). But the real point of this example is to show that in general, you can't check a claim about an infinite set by checking a finite set of its elements, no matter how large the finite set.

By the way, propositions like this about *all* numbers or all items of some kind are so common that there is a special notation for them. With this notation, Propo sition 1.1.3 would be

$$8n\ 2\ N\colon p.n/\ \text{is prime}: \quad (1.2)$$

Here the symbol 8 is read "for all." The symbol N stands for the set of *nonnegative integers*: 0, 1, 2, 3, . . .(ask your instructor for the complete list). The symbol "2" is read as "is a member of," or "belongs to," or simply as "is in." The period after the N is just a separator between phrases.

Here are two even more extreme examples:

Proposition 1.1.4. *[Euler's Conjecture] The equation*

$$a^4\ C\ b^4\ C\ c^4\ D\ d^{\ 4}$$

has no solution when a; b; c; d *are positive integers.*

Euler (pronounced "oiler") conjectured this in 1769. But the proposition was proved false 218 years later by Noam Elkies at a liberal arts school up Mass Ave. The solution he found was a D 95800; b D 217519; c D 414560; d D 422481. In logical notation, Euler's Conjecture could be written,

$$8a\ 2\ Z^C\ 8b\ 2\ Z^C\ 8c\ 2\ Z^C\ 8d\ 2\ Z^C\colon a^4\ C\ b^4\ C\ c^4\ ¤\ d^{\ 4}\colon$$

Here, Z^C is a symbol for the positive integers. Strings of 8's like this are usually abbreviated for easier reading:

$$8a; b; c; d\ 2\ Z^C\colon a^4\ C\ b^4\ C\ c^4\ ¤\ d^{\ 4}\colon$$

Proposition 1.1.5. $313.x^3\ C\ y^3/\ D\ z^3$ *has no solution when* x; y; z 2 Z^C.

This proposition is also false, but the smallest counterexample has more than 1000 digits!

It's worth mentioning a couple of further famous propositions whose proofs were sought for centuries before finally being discovered:

Proposition 1.1.6 (Four Color Theorem). *Every map can be colored with 4 colors so that adjacent*[2] *regions have different colors.*

[2]Two regions are adjacent only when they share a boundary segment of positive length. They are not considered to be adjacent if their boundaries meet only at a few points.

Several incorrect proofs of this theorem have been published, including one that stood for 10 years in the late 19th century before its mistake was found. A laborious proof was finally found in 1976 by mathematicians

Appel and Haken, who used a complex computer program to categorize the four-colorable maps. The program left a few thousand maps uncategorized, which were checked by hand by Haken and his assistants—among them his 15-year-old daughter.

There was reason to doubt whether this was a legitimate proof: the proof was too big to be checked without a computer. No one could guarantee that the com puter calculated correctly, nor was anyone enthusiastic about exerting the effort to recheck the four-colorings of thousands of maps that were done by hand. Two decades later a mostly intelligible proof of the Four Color Theorem was found, though a computer is still needed to check four-colorability of several hundred spe cial maps.[3]

Proposition 1.1.7 (Fermat's Last Theorem). *There are no positive integers* x, y, *and* z *such that*

$$^{n}\, x^{n} C\, y^{n} D\, z$$

for some integer n>2.

In a book he was reading around 1630, Fermat claimed to have a proof for this proposition, but not enough space in the margin to write it down. Over the years, the Theorem was proved to hold for all n up to 4,000,000, but we've seen that this shouldn't necessarily inspire confidence that it holds for *all* n. There is, after all, a clear resemblance between Fermat's Last Theorem and Euler's false Conjecture. Finally, in 1994, British mathematician Andrew Wiles gave a proof, after seven years of working in secrecy and isolation in his attic. His proof did not fit in any margin.[4]

Finally, let's mention another simply stated proposition whose truth remains un known.

Proposition 1.1.8 (*Goldbach's Conjecture*). *Every even integer greater than 2 is the sum of two primes.*

Goldbach's Conjecture dates back to 1742. It is known to hold for all numbers up to 10^{18}, but to this day, no one knows whether it's true or false.

[3]The story of the proof of the Four Color Theorem is told in a well-reviewed popular (non technical) book: "Four Colors Suffice. How the Map Problem was Solved." *Robin Wilson*. Princeton Univ. Press, 2003, 276pp. ISBN 0-691-11533-8.

[4]In fact, Wiles' original proof was wrong, but he and several collaborators used his ideas to arrive at a correct proof a year later. This story is the subject of the popular book, *Fermat's Enigma* by Simon Singh, Walker & Company, November, 1997.

For a computer scientist, some of the most important things to prove are the correctness of programs and systems—whether a program or system does what it's supposed to. Programs are notoriously buggy, and there's a growing community of researchers and practitioners trying to find ways to prove program correctness. These efforts have been successful enough in the case of CPU chips that they are now routinely used by leading chip manufacturers to prove chip correctness and avoid mistakes like the notorious Intel division bug in the 1990's.

Developing mathematical methods to verify programs and systems remains an active research area. We'll illustrate some of these methods in Chapter 5.

1.2 Predicates

A *predicate* can be understood as a proposition whose truth depends on the value of one or more variables. So “n is a perfect square” describes a predicate, since you can’t say if it’s true or false until you know what the value of the variable n happens to be. Once you know, for example, that n equals 4, the predicate becomes the true proposition “4 is a perfect square”. Remember, nothing says that the proposition has to be true: if the value of n were 5, you would get the false proposition “5 is a perfect square.”

Like other propositions, predicates are often named with a letter. Furthermore, a function-like notation is used to denote a predicate supplied with specific variable values. For example, we might use the name “P” for predicate above:

P .n/ WWD “n is a perfect square”;

and repeat the remarks above by asserting that P .4/ is true, and P .5/ is false. This notation for predicates is confusingly similar to ordinary function notation. If P is a predicate, then P .n/ is either *true* or *false*, depending on the value of n. On the other hand, if p is an ordinary function, like n^2C1, then p.n/ is a *numerical quantity*. Don’t confuse these two!

1.3 The Axiomatic Method

The standard procedure for establishing truth in mathematics was invented by Eu clid, a mathematician working in Alexandria, Egypt around 300 BC. His idea was to begin with five *assumptions* about geometry, which seemed undeniable based on direct experience. (For example, “There is a straight line segment between every

pair of points”.) Propositions like these that are simply accepted as true are called *axioms*.

Starting from these axioms, Euclid established the truth of many additional propo sitions by providing “proofs.” A *proof* is a sequence of logical deductions from axioms and previously proved statements that concludes with the proposition in question. You probably wrote many proofs in high school geometry class, and you’ll see a lot more in this text.

There are several common terms for a proposition that has been proved. The different terms hint at the role of the proposition within a larger body of work.

- Important true propositions are called *theorems*.
- A *lemma* is a preliminary proposition useful for proving later propositions.
- A *corollary* is a proposition that follows in just a few logical steps from a theorem.

These definitions are not precise. In fact, sometimes a good lemma turns

out to be far more important than the theorem it was originally used to prove.

Euclid's axiom-and-proof approach, now called the *axiomatic method*, remains the foundation for mathematics today. In fact, just a handful of axioms, called the Zermelo-Fraenkel with Choice axioms(ZFC), together with a few logical deduction rules, appear to be sufficient to derive essentially all of mathematics. We'll examine these in Chapter 7.

1.4 Our Axioms

The ZFC axioms are important in studying and justifying the foundations of math ematics, but for practical purposes, they are much too primitive. Proving theorems in ZFC is a little like writing programs in byte code instead of a full-fledged pro gramming language—by one reckoning, a formal proof in ZFC that 2 C 2 D 4 requires more than 20,000 steps! So instead of starting with ZFC, we're going to take a *huge* set of axioms as our foundation: we'll accept all familiar facts from high school math.

This will give us a quick launch, but you may find this imprecise specification of the axioms troubling at times. For example, in the midst of a proof, you may start to wonder, "Must I prove this little fact or can I take it as an axiom?" There really is no absolute answer, since what's reasonable to assume and what requires proof depends on the circumstances and the audience. A good general guideline is simply to be up front about what you're assuming.

1.4.1 Logical Deductions

Logical deductions, or *inference rules*, are used to prove new propositions using previously proved ones.

A fundamental inference rule is *modus ponens*. This rule says that a proof of P together with a proof that P IMPLIES Q is a proof of Q.

Inference rules are sometimes written in a funny notation. For example, *modus ponens* is written:

$$\text{Rule.}\ \frac{P;\ P \text{ IMPLIES } Q}{Q}$$

When the statements above the line, called the *antecedents*, are proved, then we can consider the statement below the line, called the *conclusion* or *consequent*, to also be proved.

A key requirement of an inference rule is that it must be *sound*: an assignment of truth values to the letters, P, Q, . . . , that makes all the antecedents true must also make the consequent true. So if we start off with true axioms and apply sound inference rules, everything we prove will also be true.

There are many other natural, sound inference rules, for example:

$$\text{Rule.}\ \frac{P \text{ IMPLIES } Q;\ Q \text{ IMPLIES } R}{P \text{ IMPLIES } R}$$

Rule.

IMPLIES P

On the other hand,

Non-Rule. NOT.P/ IMPLIES NOT.Q/ P IMPLIES Q

NOT.P/ IMPLIES NOT.Q/ Q

is not sound: if P is assigned T and Q is assigned F, then the antecedent is true and the consequent is not.

As with axioms, we will not be too formal about the set of legal inference rules. Each step in a proof should be clear and “logical”; in particular, you should state what previously proved facts are used to derive each new conclusion.

1.4.2 Patterns of Proof

In principle, a proof can be *any* sequence of logical deductions from axioms and previously proved statements that concludes with the proposition in question. This freedom in constructing a proof can seem overwhelming at first. How do you even *start* a proof?

Here’s the good news: many proofs follow one of a handful of standard tem plates. Each proof has it own details, of course, but these templates at least provide you with an outline to fill in. We’ll go through several of these standard patterns, pointing out the basic idea and common pitfalls and giving some examples. Many of these templates fit together; one may give you a top-level outline while others help you at the next level of detail. And we’ll show you other, more sophisticated proof techniques later on.

The recipes below are very specific at times, telling you exactly which words to write down on your piece of paper. You’re certainly free to say things your own way instead; we’re just giving you something you *could* say so that you’re never at a complete loss.

1.5 Proving an Implication

Propositions of the form “If P, then Q” are called *implications*. This implication is often rephrased as “P IMPLIES Q.”

Here are some examples:

✏ (Quadratic Formula) If $ax^2 C bx C c D 0$ and a ¤ 0, then

$$x D \frac{-b \cdot \sqrt{b^2 - 4ac}}{2a}:$$

✏ (Goldbach’s Conjecture 1.1.8 rephrased) If n is an even integer greater than 2, then n is a sum of two primes.

✏ If 0 x 2, then $-x^3 C 4x C 1>0$.

There are a couple of standard methods for proving an implication.

1.5.1 Method #1

In order to prove that P IMPLIES Q:

1. Write, “Assume P.”

2. Show that Q logically follows.

“mcs” — 2015/5/18 — 1:43 — page 12 — #20

Example

Theorem 1.5.1. *If* 0 x 2*, then* -x^3 C 4x C 1>0*.*

Before we write a proof of this theorem, we have to do some scratchwork to figure out why it is true.

The inequality certainly holds for x 0; then the left side is equal to 1 and D 1>0. As x grows, the 4x term (which is positive) initially seems to have greater magnitude than -x^3 (which is negative). For example, when x 1, we have D 4x D 4, but -x^3 D -1 only. In fact, it looks like -x^3 doesn’t begin to dominate until x>2. So it seems the -x^3 C4x part should be nonnegative for all x between 0 and 2, which would imply that -x^3 C 4x C 1 is positive.

So far, so good. But we still have to replace all those “seems like” phrases with solid, logical arguments. We can get a better handle on the critical -x^3 C 4x part by factoring it, which is not too hard:

$$-x^3 \text{ C } 4x \text{ D } x.2 - x/.2 \text{ C } x/$$

Aha! For x between 0 and 2, all of the terms on the right side are nonnegative. And a product of nonnegative terms is also nonnegative. Let’s organize this blizzard of observations into a clean proof.

Proof. Assume 0 x 2. Then x, 2-x, and 2Cx are all nonnegative. Therefore, the product of these terms is also nonnegative. Adding 1 to this product gives a positive number, so:

$$x.2 - x/.2 \text{ C } x/ \text{ C } 1>0$$

Multiplying out on the left side proves that

$$-x^3 \text{ C } 4x \text{ C } 1>0$$

as claimed. ⫪ There are a couple points here that apply to all proofs:

- ✏ You’ll often need to do some scratchwork while you’re trying to figure out the logical steps of a proof. Your scratchwork can be as disorganized as you like—full of dead-ends, strange diagrams, obscene words, whatever. But keep your scratchwork separate from your final proof, which should be clear and concise.

- ✏ Proofs typically begin with the word “Proof” and end with some sort of de limiter like ⇤ or “QED.” The only purpose for these conventions is to clarify where proofs begin and end.

“mcs” — 2015/5/18 — 1:43 — page 13 — #21

1.5.2 Method #2 - Prove the Contrapositive

An implication ("P IMPLIES Q") is logically equivalent to its

contrapositive NOT.Q/ IMPLIES NOT.P/:

Proving one is as good as proving the other, and proving the contrapositive is some times easier than proving the original statement. If so, then you can proceed as follows:

1. Write, "We prove the contrapositive:" and then state the contrapositive.

2. Proceed as in Method #1.

Example

Theorem 1.5.2. *If* r *is irrational, then* r *is also irrational.*

A number is *rational* when it equals a quotient of integers —that is, if it equals m=n for some integers m and n. If it's not rational, then it's called *irrational*. So we must show that if r is *not* a ratio of integers, then r is also *not* a ratio of integers. That's pretty convoluted! We can eliminate both *not*'s and simplify the proof by using the contrapositive instead.

Proof. We prove the contrapositive: if r is rational, then r is rational.

Assume that r is rational. Then there exist integers m and n such that:

$$r = \frac{m}{n}$$

Squaring both sides gives:

$$r^2 = \frac{m^2}{n}$$

Since m^2 and n^2 are integers, r is also rational.

1.6 Proving an "If and Only If"

Many mathematical theorems assert that two statements are logically equivalent; that is, one holds if and only if the other does. Here is an example that has been known for several thousand years:

Two triangles have the same side lengths if and only if two side lengths and the angle between those sides are the same.

The phrase "if and only if" comes up so often that it is often abbreviated "iff."

1.6.1 Method #1: Prove Each Statement Implies the Other

The statement "P IFF Q" is equivalent to the two statements "P IMPLIES Q" and "Q IMPLIES P." So you can prove an "iff" by proving *two* implications:

1. Write, "We prove P implies Q and vice-versa."

2. Write, “First, we show P implies Q.” Do this by one of the methods in Section 1.5.

3. Write, “Now, we show Q implies P.” Again, do this by one of the methods in Section 1.5.

1.6.2 Method #2: Construct a Chain of Iffs

In order to prove that P is true iff Q is true:

1. Write, “We construct a chain of if-and-only-if implications.”

2. Prove P is equivalent to a second statement which is equivalent to a third statement and so forth until you reach Q.

This method sometimes requires more ingenuity than the first, but the result can be a short, elegant proof.

Example

The *standard deviation* of a sequence of values $x_1; x_2;::::;x_n$ is defined to be: s

$$\frac{.x_1 - /^2 C .x_2 - /^2 C \ C .x_n - /^2}{n} \quad (1.3)$$

where is the average or *mean* of the values:

$$\text{WWD} \quad \frac{x_1 C x_2 C \ C x_n}{n}$$

Theorem 1.6.1. *The standard deviation of a sequence of values* $x_1;::::;x_n$ *is zero iff all the values are equal to the mean.*

For example, the standard deviation of test scores is zero if and only if everyone scored exactly the class average.

Proof. We construct a chain of “iff” implications, starting with the statement that the standard deviation (1.3) is zero:

$$s \quad \frac{.x_1 - /^2 C .x_2 - /^2 C \ C .x_n - /^2}{n} D 0: \quad (1.4)$$

Now since zero is the only number whose square root is zero, equation (1.4) holds iff

$$.x_1 - /^2 C .x_2 - /^2 C \ C .x_n - /^2 D 0: \quad (1.5)$$

Squares of real numbers are always nonnegative, so every term on the left hand side of equation (1.5) is nonnegative. This means that (1.5) holds iff

$$\text{Every term on the left hand side of (1.5) is zero.} \quad (1.6)$$

But a term $.x_i - /^2$ is zero iff $x_i D$, so (1.6) is true iff

$$\text{Every } x_i \text{ equals the mean.}$$

x

1.7 Proof by Cases

Breaking a complicated proof into cases and proving each case separately is a com mon, useful proof strategy. Here's an amusing example.

Let's agree that given any two people, either they have met or not. If every pair of people in a group has met, we'll call the group a *club*. If every pair of people in a group has not met, we'll call it a group of *strangers*.

Theorem. *Every collection of 6 people includes a club of 3 people or a group of 3 strangers.*

Proof. The proof is by case analysis[5]. Let x denote one of the six people. There are two cases:

1. Among 5 other people besides x, at least 3 have met x.
2. Among the 5 other people, at least 3 have not met x.

Now, we have to be sure that at least one of these two cases must hold,[6] but that's easy: we've split the 5 people into two groups, those who have shaken hands with x and those who have not, so one of the groups must have at least half the people. Case 1: Suppose that at least 3 people did meet x.

This case splits into two subcases:

[5]Describing your approach at the outset helps orient the reader.

[6]Part of a case analysis argument is showing that you've covered all the cases. This is often obvious, because the two cases are of the form "P" and "not P." However, the situation above is not stated quite so simply.

Case 1.1: No pair among those people met each other. Then these people are a group of at least 3 strangers. The theorem holds in this subcase.

Case 1.2: Some pair among those people have met each other. Then that pair, together with x, form a club of 3 people. So the theorem holds in this subcase.

This implies that the theorem holds in Case 1.

Case 2: Suppose that at least 3 people did not meet x.

This case also splits into two subcases:

Case 2.1: Every pair among those people met each other. Then these people are a club of at least 3 people. So the theorem holds in this subcase.

Case 2.2: Some pair among those people have not met each other. Then that pair, together with x, form a group of at least 3 strangers. So the theorem holds in this subcase.

This implies that the theorem also holds in Case 2, and therefore holds in a
ll

c a s e s . ∎

1.8 Proof by Contradiction

In a *proof by contradiction*, or *indirect proof*, you show that if a proposition were false, then some false fact would be true. Since a false fact by definition can't be true, the proposition must be true.

Proof by contradiction is *always* a viable approach. However, as the name sug gests, indirect proofs can be a little convoluted, so direct proofs are generally prefer able when they are available.

Method: In order to prove a proposition P by contradiction:

1. Write, "We use proof by contradiction."
2. Write, "Suppose P is false."
3. Deduce something known to be false (a logical contradiction).
4. Write, "This is a contradiction. Therefore, P must be true."

Example

We'll prove by contradiction that $\sqrt{2}$ is irrational. Remember that a number is *ra tional* if it is equal to a ratio of integers—for example, 3:5 D 7=2 and 0:1111 D 1=9 are rational numbers.

Theorem 1.8.1. $\sqrt{2}$ *is irrational.*

Proof. We use proof by contradiction. Suppose the claim is false, and $\sqrt{2}$ is ratio nal. Then we can write $\sqrt{2}$ as a fraction n=d in *lowest terms*.

Squaring both sides gives $2 D n^2=d^2$ and so $2d^2 D n^2$. This implies that n is a multiple of 2 (see Problems 1.10 and 1.11). Therefore n^2 must be a multiple of 4. But since $2d^2 D n^2$, we know $2d^2$ is a multiple of 4 and so d^2 is a multiple of 2. This implies that d is a multiple of 2.

So, the numerator and denominator have 2 as a common factor, which contradicts the fact that n=d is in lowest terms. Thus, $\sqrt{2}$ must be irrational. ∎

1.9 *Good* Proofs in Practice

One purpose of a proof is to establish the truth of an assertion with absolute cer tainty, and mechanically checkable proofs of enormous length or complexity can accomplish this. But humanly intelligible proofs are the

only ones that help some one understand the subject. Mathematicians generally agree that important mathe matical results can't be fully understood until their proofs are understood. That is why proofs are an important part of the curriculum.

To be understandable and helpful, more is required of a proof than just logical correctness: a good proof must also be clear. Correctness and clarity usually go together; a well-written proof is more likely to be a correct proof, since mistakes are harder to hide.

In practice, the notion of proof is a moving target. Proofs in a professional research journal are generally unintelligible to all but a few experts who know all the terminology and prior results used in the proof. Conversely, proofs in the first weeks of a beginning course like 6.042 would be regarded as tediously long-winded by a professional mathematician. In fact, what we accept as a good proof later in the term will be different from what we consider good proofs in the first couple of weeks of 6.042. But even so, we can offer some general tips on writing good proofs:

State your game plan. A good proof begins by explaining the general line of rea soning, for example, "We use case analysis" or "We argue by contradiction."

"mcs" — 2015/5/18 — 1:43 — page 18 — #26

Keep a linear flow. Sometimes proofs are written like mathematical mosaics, with juicy tidbits of independent reasoning sprinkled throughout. This is not good. The steps of an argument should follow one another in an intelligible order.

A proof is an essay, not a calculation. Many students initially write proofs the way they compute integrals. The result is a long sequence of expressions without explanation, making it very hard to follow. This is bad. A good proof usually looks like an essay with some equations thrown in. Use complete sentences.

Avoid excessive symbolism. Your reader is probably good at understanding words, but much less skilled at reading arcane mathematical symbols. Use words where you reasonably can.

Revise and simplify. Your readers will be grateful.

Introduce notation thoughtfully. Sometimes an argument can be greatly simpli fied by introducing a variable, devising a special notation, or defining a new term. But do this sparingly, since you're requiring the reader to remember all that new stuff. And remember to actually *define* the meanings of new variables, terms, or notations; don't just start using them!

Structure long proofs. Long programs are usually broken into a hierarchy of smaller procedures. Long proofs are much the same. When your proof needed facts that are easily stated, but not readily proved, those fact are best pulled out as preliminary lemmas. Also, if you are repeating essentially the same argu ment over and over, try to capture that argument in a general lemma, which you can cite repeatedly instead.

Be wary of the "obvious." When familiar or truly obvious facts are needed in a proof, it's OK to label them as such and to not prove them. But

remember that what's obvious to you may not be—and typically is not—obvious to your reader.

Most especially, don't use phrases like "clearly" or "obviously" in an attempt to bully the reader into accepting something you're having trouble proving. Also, go on the alert whenever you see one of these phrases in someone else's proof.

Finish. At some point in a proof, you'll have established all the essential facts you need. Resist the temptation to quit and leave the reader to draw the "obvious" conclusion. Instead, tie everything together yourself and explain why the original claim follows.

"mcs" — 2015/5/18 — 1:43 — page 19 — #27

Creating a good proof is a lot like creating a beautiful work of art. In fact, mathematicians often refer to really good proofs as being "elegant" or "beautiful." It takes a practice and experience to write proofs that merit such praises, but to get you started in the right direction, we will provide templates for the most useful proof techniques.

Throughout the text there are also examples of *bogus proofs*—arguments that look like proofs but aren't. Sometimes a bogus proof can reach false conclusions because of missteps or mistaken assumptions. More subtle bogus proofs reach correct conclusions, but do so in improper ways such as circular reasoning, leaping to unjustified conclusions, or saying that the hard part of the proof is "left to the reader." Learning to spot the flaws in improper proofs will hone your skills at seeing how each proof step follows logically from prior steps. It will also enable you to spot flaws in your own proofs.

The analogy between good proofs and good programs extends beyond structure. The same rigorous thinking needed for proofs is essential in the design of criti cal computer systems. When algorithms and protocols only "mostly work" due to reliance on hand-waving arguments, the results can range from problematic to catastrophic. An early example was the Therac 25, a machine that provided radia tion therapy to cancer victims, but occasionally killed them with massive overdoses due to a software race condition. A more recent (August 2004) example involved a single faulty command to a computer system used by United and American Airlines that grounded the entire fleet of both companies—and all their passengers!

It is a certainty that we'll all one day be at the mercy of critical computer systems designed by you and your classmates. So we really hope that you'll develop the ability to formulate rock-solid logical arguments that a system actually does what you think it does!

1.10 References

[11], [1], [45], [15], [19]

Problems for Section 1.1

Class Problems

Problem 1.1.

triangle, and c is the length of its hypotenuse, then

$$a^2 C b^2 D c^2:$$

This theorem is so fundamental and familiar that we generally take it for granted. But just being familiar doesn't justify calling it “obvious”—witness the fact that people have felt the need to devise different proofs of it for millllenia.[7] In this problem we'll examine a particularly simple “proof without words” of the theorem.

Here's the strategy. Suppose you are given four different colored copies of a right triangle with sides of lengths a, b, and c, along with a suitably sized square, as shown in Figure 1.1.

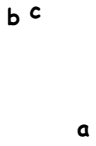

Figure 1.1 Right triangles and square.

(a) You will first arrange the square and four triangles so they form a c ↠c square. From this arrangement you will see that the square is .b a/ ↠ .b a/.

(b) You will then arrange the same shapes so they form two squares, one a ↠ a and the other b ↠ b.

You know that the area of an s ↠s square is s^2. So appealing to the

principle that *Area is Preserved by Rearranging*,

you can now conclude that $a^2 C b^2 D c^2$, as claimed.

This really is an elegant and convincing proof of the Pythagorean Theorem, but it has some worrisome features. One concern is that there might be something special

[7]Over a hundred different proofs are listed on the mathematics website http://www.cut-the knot.org/pythagoras/.

about the shape of these particular triangles and square that makes the rearranging possible—for example, suppose a D b?

(c) How would you respond to this concern?

(d) Another concern is that a number of facts about right triangles, squares and lines are being *implicitly* assumed in justifying the rearrangements into squares. Enumerate some of these assumed facts.

Problem 1.2.
What's going on here?!

$$1 = \sqrt{1} = \sqrt{1 \cdot 1} = \sqrt{-1} \cdot \sqrt{-1} = \sqrt{-1}^2 = -1.$$

(a) Precisely identify and explain the mistake(s) in this *bogus* proof. (b) Prove (correctly) that if $1 = -1$, then $2 = 1$.

(c) Every *positive* real number, r, has two square roots, one positive and the other negative. The standard convention is that the expression $\sqrt{r}$ refers to the *positive* square root of r. Assuming familiar properties of multiplication of real numbers, prove that for positive real numbers r and s,

$$\sqrt{rs} = \sqrt{r}\sqrt{s}.$$

Problem 1.3.
Identify exactly where the bugs are in each of the following bogus proofs.[8] (a) Bogus Claim: $1/8 > 1/4$.

Bogus proof.

$$\begin{aligned} 3 &> 2 \\ 3\log_{10}(1/2) &> 2\log_{10}(1/2) \\ \log_{10}(1/2)^3 &> \log_{10}(1/2)^2 \\ (1/2)^3 &> (1/2)^2; \end{aligned}$$

and the claim now follows by the rules for multiplying fractions. ∎

[8] From [44], *Twenty Years Before the Blackboard* by Michael Stueben and Diane Sandford

(b) *Bogus proof*: $1¢ = \$0.01 = (\$0.1)^2 = (10¢)^2 = 100¢ = \$1$. ∎

(c) Bogus Claim: If a and b are two equal real numbers, then $a = 0$.

Bogus proof.

$$\begin{aligned} a &= b \\ a^2 &= ab \\ a^2 - b^2 &= ab - b^2 \\ (a-b)(a+b) &= (a-b)b \\ a + b &= b \\ a &= 0. \end{aligned}$$

Problem 1.4.

It's a fact that the Arithmetic Mean is at least as large as the Geometric Mean, namely, $\sqrt{ab} \leq \frac{a+b}{2}$ for all nonnegative real numbers a and b. But there's something objectionable about the following proof of this fact. What's the objection, and how would you fix it?

Bogus proof.

$$\frac{a+b}{2} \leq \sqrt{ab}, \text{ so}$$

$$a + b \leq 2\sqrt{ab}, \text{ so}$$

$$a^2 + 2ab + b^2 \leq 4ab, \text{ so}$$

$$a^2 - 2ab + b^2 \leq 0, \text{ so}$$

$$(a-b)^2 \leq 0 \text{ which we know is true.}$$

The last statement is true because $a - b$ is a real number, and the square of a real number is never negative. This proves the claim. ⌅

Problem 1.5.

Albert announces to his class that he plans to surprise them with a quiz sometime next week.

His students first wonder if the quiz could be on Friday of next week. They reason that it can't: if Albert didn't give the quiz *before* Friday, then by midnight Thursday, they would know the quiz had to be on Friday, and so the quiz wouldn't be a surprise any more.

Next the students wonder whether Albert could give the surprise quiz Thursday. They observe that if the quiz wasn't given *before* Thursday, it would have to be given *on* the Thursday, since they already know it can't be given on Friday. But having figured that out, it wouldn't be a surprise if the quiz was on Thursday either. Similarly, the students reason that the quiz can't be on Wednesday, Tuesday, or Monday. Namely, it's impossible for Albert to give a surprise quiz next week. All the students now relax, having concluded that Albert must have been bluffing. And since no one expects the quiz, that's why, when Albert gives it on Tuesday next week, it really is a surprise!

What, if anything, do you think is wrong with the students' reasoning?

Problems for Section 1.5

Homework Problems

Problem 1.6.

Show that $\log_7 n$ is either an integer or irrational, where n is a positive integer. Use whatever familiar facts about integers and primes you need, but explicitly state such facts.

Problems for Section 1.7

Class Problems

Problem 1.7.

If we raise an irrational number to an irrational power, can the result be rational? Show that it can by considering $\sqrt{2}^{\sqrt{2}}$ and arguing by cases.

Problems for Section 1.8

Practice Problems

Problem 1.8.

Prove that for any n>0, if a^n is even, then a is even.

Hint: Contradiction.

Problem 1.9.

Prove that if a b D n, then either a or b must be $\sqrt{}$ ~~n~~, where a; b, and n are nonnegative real numbers. *Hint:* by contradiction, Section 1.8.

Problem 1.10.

Let n be a nonnegative integer.

(a) Explain why if n^2 is even—that is, a multiple of 2—then n is even.

(b) Explain why if n^2 is a multiple of 3, then n must be a multiple of 3.

Problem 1.11.

Give an example of two distinct positive integers m; n such that n^2 is a multiple of m, but n is not a multiple of m. How about having m be less than n?

Class Problems

Problem 1.12.

pthe proof of Theorem 1.8.1 that $\sqrt{2}$ is irrational? For
How far can you generalize
example, how about 3?

Problem 1.13.

Prove that $\log_4 6$ is irrational.

Problem 1.14.

Here is a different proof that $\sqrt{2}$ is irrational, taken from the American Mathemat ical Monthly, v.116, #1, Jan. 2009, p.69:

Proof. Suppose for the sake of contradiction that $\sqrt{2}$ is rational, and choose the

$\downarrow \sqrt{2} - 1\rceil$

$\downarrow \rceil q$ is a nonnegative integer. Let q^0 WWD least integer, $q>0$, such that

$\sqrt{2} - 1\ q$. Clearly $0<q^0 < q$. But an easy computation shows that $\downarrow\sqrt{2} - 1\ q^0$

$\rceil$

is a nonnegative integer, contradicting the minimality of q.

$\dashv$

(a) This proof was written for an audience of college teachers, and at this point it is a little more concise than desirable. Write out a more complete version which includes an explanation of each step.

(b) Now that you have justified the steps in this proof, do you have a preference for one of these proofs over the other? Why? Discuss these questions with your teammates for a few minutes and summarize your team's answers on your white board.

Problem 1.15.

Here is a generalization of Problem 1.12 that you may not have thought of: Lemma. *Let the coefficients of the polynomial*

$$a_0 + a_1 x + a_2 x^2 + \cdots + a_{m-1} x^{m-1} + x^m$$

be integers. Then any real root of the polynomial is either integral or irrational. (a) Explain why the Lemma immediately implies that $\sqrt[m]{k}$ is irrational whenever k is not an mth power of some integer.

(b) Carefully prove the Lemma.

You may find it helpful to appeal to:
Fact. If a prime, p, is a factor of some power of an integer, then it is a factor of that integer.

You may assume this Fact without writing down its proof, but see if you can explain why it is true.

Homework Problems

Problem 1.16.

The fact that that there are irrational numbers a; b such that a^b is rational was proved in Problem 1.7 by cases. Unfortunately, that proof was *nonconstructive*: it didn't reveal a specific pair, a; b, with this property. But

in fact, it's easy to do this: let a WWD 2 and b WWD $2\log_2 3$.

We know a^p D 2 is irrational, and a^b D 3 by definition. Finish the proof that these values for a; b work, by showing that $2\log_2 3$ is irrational.

Problem 1.17.

For n D 40, the value of polynomial p.n/ WWD n^2 C n C 41 is not prime, as noted in Section 1.1. But we could have predicted based on general principles that no nonconstant polynomial can generate only prime numbers.

In particular, let q.n/ be a polynomial with integer coefficients, and let c WWDq.0/ be the constant term of q.

(a) Verify that q.cm/ is a multiple of c for all m 2 Z.

(b) Show that if q is nonconstant and c>1, then as n ranges over the nonnegative integers, N, there are infinitely many q.n/ 2 Z that are not primes.

Hint: You may assume the familiar fact that the magnitude of any nonconstant polynomial, q.n/, grows unboundedly as n grows.

(c) Conclude that for every nonconstant polynomial, q, there must be an n 2 N such that q.n/ is not prime. *Hint:* Only one easy case remains.

Exam Problems

Problem 1.18.

Prove that $\log_9 12$ is irrational.

Problem 1.19.

Prove that $\log_{12} 18$ is irrational.

2 The Well Ordering Principle

Every *nonempty* set of *nonnegative integers* has a *smallest* element.

This statement is known as The *Well Ordering Principle*. Do you believe it? Seems sort of obvious, right? But notice how tight it is: it requires a *nonempty* set—it's false for the empty set which has *no* smallest element because it has no elements at all. And it requires a set of *nonnegative* integers—it's false for the set of *negative* integers and also false for some sets of nonnegative *rationals*—for example, the set of positive rationals. So, the Well Ordering Principle captures something special about the nonnegative integers.

While the Well Ordering Principle may seem obvious, it's hard to see offhand why it is useful. But in fact, it provides one of the most important

proof rules in discrete mathematics. In this chapter, we'll illustrate the power of this proof method with a few simple examples.

2.1 Well Ordering Proofs

We actually

$_p$have already taken the Well Ordering Principle for granted in proving that 2 is irrational. That proof assumed that for any positive integers m and n, the fraction m=n can be written in *lowest terms*, that is, in the form $m^0=n^0$ where m^0 and n^0 are positive integers with no common prime factors. How do we know this is always possible?

Suppose to the contrary that there are positive integers m and n such that the fraction m=n cannot be written in lowest terms. Now let C be the set of positive integers that are numerators of such fractions. Then m 2 C, so C is nonempty. Therefore, by Well Ordering, there must be a smallest integer, m_0 2 C. So by definition of C, there is an integer $n_0 > 0$ such that

$$\text{the fraction } \frac{m}{n_0}{}^0 \text{ cannot be written in lowest terms.}$$

This means that m_0 and n_0 must have a common prime factor, p>1. But

$$m_0 = p\, m\,;\quad n_0 D\, {}^0 = p\, n_0$$

so any way of expressing the left hand fraction in lowest terms would also work for $m_0=n_0$, which implies

$$m = p \text{ the fraction } \frac{m}{n_0}{}^0 \text{ cannot be in written in lowest terms either.}$$

So by definition of C, the numerator, $m_0=p$, is in C. But $m_0=p < m_0$, which contradicts the fact that m_0 is the smallest element of C.

Since the assumption that C is nonempty leads to a contradiction, it follows that C must be empty. That is, that there are no numerators of fractions that can't be written in lowest terms, and hence there are no such fractions at all.

We've been using the Well Ordering Principle on the sly from early on!

2.2 Template for Well Ordering Proofs

More generally, there is a standard way to use Well Ordering to prove that some property, P .n/ holds for every nonnegative integer, n. Here is a standard way to organize such a well ordering proof:

To prove that “P .n/ is true for all n 2 N” using the Well Ordering Principle: ✏ Define the set, C, of *counterexamples* to P being true.

Specifically, define C WWD fn 2 N j NOT.P .n// is trueg:

(The notation fn j Q.n/g means “the set of all elements n for which Q.n/ is true.” See Section 4.1.4.)

✏ Assume for proof by contradiction that C is nonempty.

✏ By the Well Ordering Principle, there will be a smallest element, n, in C.

✏ Reach a contradiction somehow—often by showing that P .n/ is actually true or by showing that there is another member of C that is smaller than n. This is the open-ended part of the proof task.

✏ Conclude that C must be empty, that is, no counterexamples exist. ⌅

2.2.1 Summing the Integers

Let’s use this template to prove

“mcs” — 2015/5/18 — 1:43 — page 29 — #37

Theorem 2.2.1.

$$1 C 2 C 3 C \quad C n D n.n C 1/=2 \quad (2.1)$$

for all nonnegative integers, n.

First, we’d better address a couple of ambiguous special cases before they trip us up:

✏ If n D 1, then there is only one term in the summation, and so 1 C 2 C 3 C C n is just the term 1. Don’t be misled by the appearance of 2 and 3 or by the suggestion that 1 and n are distinct terms!

✏ If n D 0, then there are no terms at all in the summation. By convention, the sum in this case is 0.

So, while the three dots notation, which is called an *ellipsis*, is convenient, you have to watch out for these special cases where the notation is misleading. In fact, whenever you see an ellipsis, you should be on the lookout to be sure you understand the pattern, watching out for the beginning and the end.

We could have eliminated the need for guessing by rewriting the left side of (2.1) with *summation notation*:

$$\sum_{i=1}^{n} i \text{ or } \sum_{1 \le i \le n} i$$

Both of these expressions denote the sum of all values taken by the expression to the right of the sigma as the variable, i, ranges from 1 to n. Both expressions make it clear what (2.1) means when n D 1. The second

expression makes it clear that when n D 0, there are no terms in the sum, though you still have to know the convention that a sum of no numbers equals 0 (the *product* of no numbers is 1, by the way).

OK, back to the proof:

Proof. By contradiction. Assume that Theorem 2.2.1 is *false*. Then, some nonneg ative integers serve as *counterexamples* to it. Let's collect them in a set:

$$C \text{ WWD } fn \in \mathbb{N} \mid 1 + 2 + 3 + \cdots + n \neq \frac{1}{2}n(n+1)g.$$

Assuming there are counterexamples, C is a nonempty set of nonnegative integers. So, by the Well Ordering Principle, C has a minimum element, which we'll call c. That is, among the nonnegative integers, c is the *smallest counterexample* to equation (2.1).

Since c is the smallest counterexample, we know that (2.1) is false for n D c but true for all nonnegative integers n<c. But (2.1) is true for n D 0, so c>0. This means c 1 is a nonnegative integer, and since it is less than c, equation (2.1) is true for c 1. That is,

$$1 + 2 + 3 + \cdots + (c-1) = \frac{1}{2}(c-1)c.$$

But then, adding c to both sides, we get

$$1 + 2 + 3 + \cdots + (c-1) + c = \frac{1}{2}(c-1)c + c = \frac{1}{2}c^2 + \frac{1}{2}c = \frac{1}{2}c(c+1),$$

which means that (2.1) does hold for c, after all! This is a contradiction, and we are done. ☐

2.3 Factoring into Primes

We've previously taken for granted the *Prime Factorization Theorem*, also known as the *Unique Factorization Theorem* and the *Fundamental Theorem of Arithmetic*, which states that every integer greater than one has a unique[1] expression as a prod uct of prime numbers. This is another of those familiar mathematical facts which are taken for granted but are not really obvious on closer inspection. We'll prove the uniqueness of prime factorization in a later chapter, but well ordering gives an easy proof that every integer greater than one can be expressed as *some* product of primes.

Theorem 2.3.1. *Every positive integer greater than one can be factored as a prod uct of primes.*

Proof. The proof is by well ordering.

Let C be the set of all integers greater than one that cannot be factored as a product of primes. We assume C is not empty and derive a contradiction.

If C is not empty, there is a least element, n 2 C, by well ordering. The n can't be prime, because a prime by itself is considered a (length one) product of primes and no such products are in C.

So n must be a product of two integers a and b where 1 < a; b < n. Since a and b are smaller than the smallest element in C, we know that a; b … C. In other words, a can be written as a product of primes $p_1 p_2 \ p_k$ and b as a product of

[1]. . . unique up to the order in which the prime factors appear

primes $q_1 \ q_l$. Therefore, n D $p_1 \ p_k q_1 \ q_l$ can be written as a product of primes, contradicting the claim that n 2 C. Our assumption that C is not empty must therefore be false. ⌅

2.4 Well Ordered Sets

A set of numbers is *well ordered* when each of its nonempty subsets has a minimum element. The Well Ordering Principle says, of course, that the set of nonnegative integers is well ordered, but so are lots of other sets, such as every finite set, or the sets rN of numbers of the form rn, where r is a positive real number and n 2 N.

Well ordering commonly comes up in computer science as a method for proving that computations won't run forever. The idea is to assign a value to the successive steps of a computation so that the values get smaller at every step. If the values are all from a well ordered set, then the computation can't run forever, because if it did, the values assigned to its successive steps would define a subset with no minimum element. You'll see several examples of this technique applied in Section 5.4 to prove that various state machines will eventually terminate.

Notice that a set may have a minimum element but not be well ordered. The set of nonnegative rational numbers is an example: it has a minimum element, zero, but it also has nonempty subsets that don't have minimum elements—the *positive* rationals, for example.

The following theorem is a tiny generalization of the Well Ordering Principle.

Theorem 2.4.1. *For any nonnegative integer,* n*, the set of integers greater than or equal to* n *is well ordered.*

This theorem is just as obvious as the Well Ordering Principle, and it would be harmless to accept it as another axiom. But repeatedly introducing axioms gets worrisome after a while, and it's worth noticing when a potential axiom can actually be proved. We can easily prove Theorem 2.4.1 using the Well Ordering Principle:

Proof. Let S be any nonempty set of integers n. Now add n to each of the elements in S; let's call this new set S C n. Now S C n is a nonempty set of *nonnegative* integers, and so by the Well Ordering Principle, it has a minimum element, m. But then it's easy to see that m n is the minimum element of S. ⌅

The definition of well ordering states that *every* subset of a well ordered set is well ordered, and this yields two convenient, immediate corollaries of Theo rem 2.4.1:

Definition 2.4.2. A *lower bound* (respectively, *upper bound*) for a set, S, of real numbers is a number, b, such that b s (respectively, b s) for every s 2 S.

Note that a lower or upper bound of set S is not required to be in the set.

Corollary 2.4.3. *Any set of integers with a lower bound is well ordered.*

Proof. A set of integers with a lower bound b 2 R will also have the integer n D bbc as a lower bound, where bbc, called the floor of b, is gotten by rounding down b to the nearest integer. So Theorem 2.4.1 implies the set is well ordered. ⌅

Corollary 2.4.4. *Any nonempty set of integers with an upper bound has a maximum element.*

Proof. Suppose a set, S, of integers has an upper bound b 2 R. Now multiply each element of S by -1; let's call this new set of elements S. Now, of course, b is a lower bound of S. So S has a minimum element m by Corollary 2.4.3. But then it's easy to see that m is the maximum element of S. ⌅

2.4.1 A Different Well Ordered Set (Optional)

Another example of a well ordered set of numbers is the set F of fractions that can be expressed in the form n=.n C 1/:

0 1 2 3 n
; ; ; ;:::; ;:::: 1 2 3 4 n C 1

The minimum element of any nonempty subset of F is simply the one with the minimum numerator when expressed in the form n=.n C 1/.

Now we can define a very different well ordered set by adding nonnegative inte gers to numbers in F. That is, we take all the numbers of the form nCf where n is a nonnegative integer and f is a number in F. Let's call this set of numbers—you guessed it—N C F. There is a simple recipe for finding the minimum number in any nonempty subset of N C F, which explains why this set is well ordered:

Lemma 2.4.5. N C F *is well ordered.*

Proof. Given any nonempty subset, S, of N C F, look at all the nonnegative inte gers, n, such that nCf is in S for some f 2 F. This is a nonempty set nonnegative integers, so by the WOP, there is a minimum one; call it n_s.

By definition of n_s, there is some f 2 F such that n_S C f is in the set S. So the set all fractions f such that n_S C f 2 S is a nonempty subset of F, and since F is well ordered, this nonempty set contains a minimum element; call it f_S. Now it easy to verify that n_S C f_S is the minimum element of S (Problem 2.14). ⌅

The set N C F is different from the earlier examples. In all the earlier examples, each element was greater than only a finite number of other

elements. In N C F, every element greater than or equal to 1 can be the first element in strictly decreas ing sequences of elements of arbitrary finite length. For example, the following decreasing sequences of elements in N C F all start with 1:

$$1; 0:$$
$$1; \tfrac{1}{}; 0: {}_{2}$$
$$1; \tfrac{2}{}; \tfrac{1}{}; 0: {}_{3\,2}$$
$$1; \tfrac{3}{}; \tfrac{2}{}; \tfrac{1}{}; 0: {}^{4\,3\,2}{}_{::}:$$

Nevertheless, since N C F is well ordered, it is impossible to find an infinite de creasing sequence of elements in N C F, because the set of elements in such a sequence would have no minimum.

Problems for Section 2.2

Practice Problems

Problem 2.1.
For practice using the Well Ordering Principle, fill in the template of an easy to prove fact: every amount of postage that can be assembled using only 10 cent and 15 cent stamps is divisible by 5.

In particular, let the notation “j j k” indicate that integer j is a divisor of integer k, and let S.n/ mean that exactly n cents postage can be assembled using only 10 and 15 cent stamps. Then the proof shows that

S.n/ IMPLIES 5 j n; for all nonnegative integers n: (2.2) Fill in

the missing portions (indicated by “. . . ”) of the following proof of (2.2). Let

C be the set of *counterexamples* to (2.2), namely

C WWD fn j :::g

Assume for the purpose of obtaining a contradiction that C is nonempty. Then by the WOP, there is a smallest number, m 2 C. This m must be positive because . . . .

But if S.m/ holds and m is positive, then S.m 10/ or S.m 15/ must hold, because . . . .

So suppose S.m 10/ holds. Then 5 j .m 10/, because. . .

But if 5 j .m 10/, then obviously 5 j m, contradicting the fact that m is a counterexample.

Next, if S.m 15/ holds, we arrive at a contradiction in the same way.

Since we get a contradiction in both cases, we conclude that. . .

which proves that (2.2) holds.

Problem 2.2.
The *Fibonacci numbers* F .0/; F .1/; F .2/; : : : are defined as follows:

F .0/ WWD 0;

F .1/ WWD 1;

F .n/ WWD F .n 1/ C F .n 2/ for n 2: (2.3)

Exactly which sentence(s) in the following bogus proof contain logical errors? Explain.

False Claim. *Every Fibonacci number is even.*

Bogus proof. Let all the variables n; m; k mentioned below be nonnegative integer valued.

1. The proof is by the WOP.
2. Let Even.n/ mean that F .n/ is even.
3. Let C be the set of counterexamples to the assertion that Even.n/ holds for all n 2 N, namely,

C WWD fn 2 N j NOT.Even.n//g:

4. We prove by contradiction that C is empty. So assume that C is not

empty. 5. By WOP, there is a least nonnegative integer, m 2 C,

6. Then m>0, since F .0/ D 0 is an even number.

7. Since m is the minimum counterexample, F .k/ is even for all k<m.

8. In particular, F .m 1/ and F .m 2/ are both even.

9. But by the defining equation (2.3), F .m/ equals the sum F .m1/CF .m2/ of two even numbers, and so it is also even.
10. That is, Even.m/ is true.
11. This contradicts the condition in the definition of m that NOT.Even.m// holds.
12. This contradition implies that C must be empty. Hence, F .n/ is even for all n 2 N.

☠

Problem 2.3.
In Chapter 2, the Well Ordering Principle was used to show that all positive rational numbers can be written in "lowest terms," that is, as a ratio of positive integers with no common factor prime factor. Below is a different proof which also arrives at this correct conclusion, but this proof is bogus. Identify every step at which the proof makes an unjustified inference.

Bogus proof. Suppose to the contrary that there was positive rational, q, such that q cannot be written in lowest terms. Now let C be the set of such rational numbers that cannot be written in lowest terms. Then q 2 C, so C is nonempty. So there must be a smallest rational, q_0 2 C. So since q_0=2 < q_0, it must be possible to express q_0=2 in lowest terms, namely,

$$q_0 \, m$$

$$2 \, D \quad (2.4)$$

$$n$$

for positive integers m; n with no common prime factor. Now we consider two cases:

Case 1: [n is odd]. Then 2m and n also have no common prime factor, and therefore

$$\downarrow m \; 2m \; \cmd$$

$$q_0 \, D \, 2 \;\; D \; {}_n$$

$$n$$

xpresses q_0 in lowest terms, a contradiction.

e

Case 2: [n is even]. Any common prime factor of m and n=2 would also be a common prime factor of m and n. Therefore m and n=2 have no common prime factor, and so m

$$q_0 \; D \; n=2$$

expresses q_0 in lowest terms, a contradiction.

Since the assumption that C is nonempty leads to a contradiction, it follows that C is empty—that is, there are no counterexamples. ⌅

Class Problems

Problem 2.4.

Use the *Well Ordering Principle* [2] to prove that

$$X^n{}_2 \, C \, 1/.2n \, C \, 1/$$

$$n.n \; {}_k$$

$$k$$

$$D : (2.5) \; {}_6 \, {}_{D0}$$

for all nonnegative integers, n.

Problem 2.5.

Use the Well Ordering Principle to prove that there is no solution over the positive integers to the equation:

$$4a^3 \, C \, 2b^3 \, D \, c^3:$$

Problem 2.6.

You are given a series of envelopes, respectively containing 1; 2; 4; : : : ; 2^m dollars. Define

> Property m: For any nonnegative integer less than 2^{mC1}, there is a selection of envelopes whose contents add up to *exactly* that number of dollars.

Use the Well Ordering Principle (WOP) to prove that Property m holds for all nonnegative integers m.

Hint: Consider two cases: first, when the target number of dollars is less than 2^m and second, when the target is at least 2^m.

Homework Problems

Problem 2.7.
Use the Well Ordering Principle to prove that any integer greater than or equal to 8 can be represented as the sum of nonnegative integer multiples of 3 and 5.

[2]Proofs by other methods such as induction or by appeal to known formulas for similar sums will not receive credit.

Problem 2.8.
Euler's Conjecture in 1769 was that there are no positive integer solutions to the equation

$$a^4 C b^4 C c^4 D d^4:$$

Integer values for a; b; c; d that do satisfy this equation were first discovered in 1986. So Euler guessed wrong, but it took more than two centuries to demonstrate his mistake.

Now let's consider Lehman's equation, similar to Euler's but with some coeffi cients:

$$8a^4 C 4b^4 C 2c^4 D d^4 \quad (2.6)$$

Prove that Lehman's equation (2.6) really does not have any positive integer solutions.

Hint: Consider the minimum value of a among all possible solutions to (2.6).

Problem 2.9.
Use the Well Ordering Principle to prove that

$$n \quad 3^{n=3} \quad (2.7)$$

for every nonnegative integer, n.

Hint: Verify (2.7) for n 4 by explicit calculation.

Exam Problems

Problem 2.10.
Except for an easily repaired omission, the following proof using the Well Ordering Principle shows that every amount of postage that can be paid exactly using only 10 cent and 15 cent stamps, is divisible by 5.

Namely, let the notation "j j k" indicate that integer j is a divisor of integer k, and let S.n/ mean that exactly n cents postage can be assembled using only 10 and 15 cent stamps. Then the proof shows that

$$S.n/ \text{ IMPLIES } 5 \text{ j } n; \text{ for all nonnegative integers n: } (2.8)$$

Fill in the missing portions (indicated by ". . . ") of the following proof of (2.8), and at the end, identify the minor mistake in the proof and how to fix

it.

Let C be the set of *counterexamples* to (2.8), namely

C WWD fn j S.n/ and NOT.5 j n/g

Assume for the purpose of obtaining a contradiction that C is nonempty. Then by the WOP, there is a smallest number, m 2 C. Then S.m10/ or S.m 15/ must hold, because the m cents postage is made from 10 and 15 cent stamps, so we remove one.

So suppose S.m 10/ holds. Then 5 j .m 10/, because. . .

But if 5 j .m 10/, then 5 j m, because. . .

contradicting the fact that m is a counterexample.

Next suppose S.m 15/ holds. Then the proof for m 10 carries over directly for m 15 to yield a contradiction in this case as well. Since we get a contradiction in both cases, we conclude that C must be empty. That is, there are no counterexamples to (2.8), which proves that (2.8) holds.

The proof makes an implicit assumption about the value of m. State the assump tion and justify it in one sentence.

Problem 2.11.
We'll use the Well Ordering Principle to prove that for every positive integer, n, the sum of the first n odd numbers is n^2, that is,

$$\sum_{i=0}^{n} (2i+1) = n^2; \quad (2.9)$$

for all n>0.

Assume to the contrary that equation (2.9) failed for some positive integer, n. Let m be the least such number.

(a) Why must there be such an m?

(b) Explain why m 2.

(c) Explain why part (b) implies that

$$\sum_{i=1}^{m} (2i+1) = 1/ C 1/ D .m 1/^2: \quad (2.10)$$

(d) What term should be added to the left hand side of (2.10) so the result equals

$$X^{m}.2.i \quad 1/ C 1/‹$$

 (e) Conclude that equation (2.9) holds for all positive

integers, n.

Problem 2.12.
Use the Well Ordering Principle (WOP) to prove that

$$2 C 4 C \ C 2n D n.n C 1/ \quad (2.11)$$

for all n>0.

Problem 2.13.
Prove by the Well Ordering Principle that for all nonnegative integers, n:

$$\checkmark_{n.n} \blacklozenge$$
$$C 1/$$
$$X^{n} {}_{2}$$
$$i^{3} \quad {}_{i} \quad D : {}_{2}$$
$$D0$$

Problems for Section 2.4

Homework Problems

Problem 2.14.
Complete the proof of Lemma 2.4.5 by showing that the number n_S C f_S is the minimum element in S.

Practice Problems

Problem 2.15.
Indicate which of the following sets of numbers have a minimum element and which are well ordered. For those that are not well ordered, give an example of a subset with no minimum element.

(a) The integers ^{p} 2.

(b) The rational numbers ^{p} 2.

(c) The set of rationals of the form 1=n where n is a positive integer.

(d) The set G of rationals of the form m=n where m; n > 0 and n g where g is a *googol*, 10^{100}.

(e) The set, F, of fractions of the form n=.n C 1/:

$$0\ 1\ 2\ 3$$
$$; ; ; ;\ldots\ldots$$
$$1\ 2\ 3\ 4$$

(f) Let W WWD N [F be the set consisting of the nonnegative integers along with all the fractions of the form n=.n C 1/. Describe a length 5 decreasing sequence of elements of W starting with 1,. . . length 50 decreasing sequence,. . . length 500.

Problem 2.16.
Use the Well Ordering Principle to prove that every finite, nonempty set of real numbers has a minimum element.

Class Problems

Problem 2.17.
Prove that a set, R, of real numbers is well ordered iff there is no infinite decreasing sequence of numbers R. In other words, there is no set of numbers r_i 2 R such that

$$r_0 > r_1 > r_2 >:::: \quad (2.12)$$

3 Logical Formulas

It is amazing that people manage to cope with all the ambiguities in the English language. Here are some sentences that illustrate the issue:

✏ "You may have cake, or you may have ice cream."

✏ "If pigs can fly, then you can understand the Chebyshev bound."

✏ "If you can solve any problem we come up with, then you get an *A* for the course."

✏ "Every American has a dream."

What *precisely* do these sentences mean? Can you have both cake and ice cream or must you choose just one dessert? Pigs can't fly, so does the second sentence say anything about your understanding the Chebyshev bound? If you can solve some problems we come up with, can you get an *A* for the course? And if you can't solve a single one of the problems, does it mean you can't get an *A*? Finally, does the last sentence imply that all Americans have the same dream—say of owning a house—or might different Americans have different dreams—say, Eric dreams of designing a killer software application, Tom of being a tennis champion, Albert of being able to sing?

Some uncertainty is tolerable in normal conversation. But when we need to formulate ideas precisely—as in mathematics and programming—the ambiguities inherent in everyday language can be a real problem. We can't hope to make an exact argument if we're not sure exactly what the statements mean. So before we start into mathematics, we need to investigate the problem of how to talk about mathematics.

To get around the ambiguity of English, mathematicians have devised a spe cial language for talking about logical relationships. This language mostly uses ordinary English words and phrases such as "or," "implies," and "for all." But mathematicians give these words precise and unambiguous definitions.

Surprisingly, in the midst of learning the language of logic, we'll come across the most important open problem in computer science—a problem whose solution could change the world.

3.1 Propositions from Propositions

In English, we can modify, combine, and relate propositions with words such as “not,” “and,” “or,” “implies,” and “if-then.” For example, we can combine three propositions into one like this:

If all humans are mortal and all Greeks are human, then all Greeks are mortal.

For the next while, we won’t be much concerned with the internals of propositions— whether they involve mathematics or Greek mortality—but rather with how propo sitions are combined and related. So, we’ll frequently use variables such as P and Q in place of specific propositions such as “All humans are mortal” and “2 C 3 D 5.” The understanding is that these *propositional variables*, like propositions, can take on only the values T (true) and F (false). Propositional variables are also called *Boolean variables* after their inventor, the nineteenth century mathematician George—you guessed it—Boole.

3.1.1 NOT, AND, and OR

Mathematicians use the words NOT, AND, and OR for operations that change or combine propositions. The precise mathematical meaning of these special words can be specified by *truth tables*. For example, if P is a proposition, then so is “NOT.P /,” and the truth value of the proposition “NOT.P /” is determined by the truth value of P according to the following truth table:

P	NOT.P /
T	F
F	T

The first row of the table indicates that when proposition P is true, the proposition “NOT.P /” is false. The second line indicates that when P is false, “NOT.P /” is true. This is probably what you would expect.

In general, a truth table indicates the true/false value of a proposition for each possible set of truth values for the variables. For example, the truth table for the proposition “P AND Q” has four lines, since there are four settings of truth values for the two variables:

P	Q	P AND Q
T	T	T
T	F	F
F	T	F
F	F	F

According to this table, the proposition “P AND Q” is true only when P and Q are both true. This is probably the way you ordinarily think about the word “and.” There is a subtlety in the truth table for “P OR Q”:

P	Q	P OR Q
T	T	T
T	F	T
F	T	T
F	F	F

The first row of this table says that "P OR Q" is true even if *both* P and Q are true. This isn't always the intended meaning of "or" in everyday speech, but this is the standard definition in mathematical writing. So if a mathematician says, "You may have cake, or you may have ice cream," he means that you *could* have both.

If you want to exclude the possibility of having both cake *and* ice cream, you should combine them with the *exclusive-or* operation, XOR:

P	Q	P XOR Q
T	T	F
T	F	T
F	T	T
F	F	F

3.1.2 IMPLIES

The combining operation with the least intuitive technical meaning is "implies." Here is its truth table, with the lines labeled so we can refer to them later.

P	Q	P IMPLIES Q	
T	T	T	(tt)
T	F	F	(tf)
F	T	T	(ft)
F	F	T	(ff)

The truth table for implications can be summarized in words as follows:

> An implication is true exactly when the if-part is false or the then-part is true.

This sentence is worth remembering; a large fraction of all mathematical statements are of the if-then form!

Let's experiment with this definition. For example, is the following proposition true or false?

"mcs" — 2015/5/18 — 1:43 — page 44 — #52

"If Goldbach's Conjecture is true, then x^2 0 for every real number x."

Now, we already mentioned that no one knows whether Goldbach's Conjecture, Proposition 1.1.8, is true or false. But that doesn't prevent you from answering the question! This proposition has the form P IMPLIES Q where the *hypothesis*, P, is "Goldbach's Conjecture is true" and the *conclusion*, Q, is "x^2 0 for every real number x." Since the conclusion is definitely true, we're on either line (tt) or line (ft) of the truth table. Either way, the proposition as a whole is *true*!

One of our original examples demonstrates an even stranger side of implications. "If pigs fly, then you can understand the Chebyshev bound."

Don't take this as an insult; we just need to figure out whether this proposition is true or false. Curiously, the answer has *nothing* to do with whether or not you can understand the Chebyshev bound. Pigs do not fly,

so we're on either line (ft) or line (ff) of the truth table. In both cases, the proposition is *true*!

In contrast, here's an example of a false implication:

"If the moon shines white, then the moon is made of white cheddar."

Yes, the moon shines white. But, no, the moon is not made of white cheddar cheese. So we're on line (tf) of the truth table, and the proposition is false.

False Hypotheses

It often bothers people when they first learn that implications which have false hypotheses are considered to be true. But implications with false hypotheses hardly ever come up in ordinary settings, so there's not much reason to be bothered by whatever truth assignment logicians and mathematicians choose to give them.

There are, of course, good reasons for the mathematical convention that implica tions are true when their hypotheses are false. An illustrative example is a system specification (see Problem 3.12) which consisted of a series of, say, a dozen rules,

if C_i: the system sensors are in condition i, then A_i: the system takes action i,

or more concisely,

$$C_i \text{ IMPLIES } A_i$$

for 1 i 12. Then the fact that the system obeys the specification would be expressed by saying that the AND

$$\text{Œ}C_1 \text{ IMPLIES } A_1\text{ç AND Œ}C_2 \text{ IMPLIES } A_2\text{ç AND AND Œ}C_{12} \text{ IMPLIES } A_{12}\text{ç} \quad (3.1)$$

of these rules was always true.

For example, suppose only conditions C_2 and C_5 are true, and the system indeed takes the specified actions A_2 and A_5. This means that in this case the system is behaving according to specification, and accordingly we want the formula (3.1) to come out true. Now the implications C_2 IMPLIES A_2 and C_5 IMPLIES A_5 are both true because both their hypotheses and their conclusions are true. But in order for (3.1) to be true, we need all the other implications with the false hypotheses C_i for i ¤ 2; 5 to be true. This is exactly what the rule for implications with false hypotheses accomplishes.

3.1.3 If and Only If

Mathematicians commonly join propositions in one additional way that doesn't arise in ordinary speech. The proposition "P if and only if Q" asserts that P and Q have the same truth value. Either both are true or both are false.

P	Q	P IFF Q
T	T	T
T	F	F
F	T	F

F F T

For example, the following if-and-only-if statement is true for every real number x:

$$x^2 \geq 0 \text{ IFF } |x| \geq 2.$$

For some values of x, *both* inequalities are true. For other values of x, *neither* inequality is true. In every case, however, the IFF proposition as a whole is true.

3.2 Propositional Logic in Computer Programs

Propositions and logical connectives arise all the time in computer programs. For example, consider the following snippet, which could be either C, C++, or Java:

```
if ( x > 0 || (x <= 0 && y > 100) )
  .
  .
  .
(further instructions)
```

Java uses the symbol || for "OR," and the symbol && for "AND." The *further instructions* are carried out only if the proposition following the word if is true. On closer inspection, this big expression is built from two simpler propositions.

Let A be the proposition that x>0, and let B be the proposition that y > 100. Then we can rewrite the condition as

$$A \text{ OR } .\text{NOT}.A/ \text{ AND } B/: \quad (3.2)$$

3.2.1 Truth Table Calculation

A truth table calculation reveals that the more complicated expression 3.2 always has the same truth value as

$$A \text{ OR } B: \quad (3.3)$$

We begin with a table with just the truth values of A and B:

A	B	A OR .NOT.A/ AND B/	A OR B
T	T		
T	F		
F	T		
F	F		

These values are enough to fill in two more columns:

A	B	A OR .NOT.A/ AND B/	A OR B
T	T	F	T
T	F	F	T
F	T	T	T
F	F	T	F

Now we have the values needed to fill in the AND column:

A	B	A OR .NOT.A/ AND B/	A OR B

T T F F T
T F F F T
F T T T T
F F T F F

and this provides the values needed to fill in the remaining column for the first OR:

A B A OR .NOT.A/ AND B/ A OR B
T T T F F T
T F T F F T
F T T T T T
F F F T F F

Expressions whose truth values always match are called *equivalent*. Since the two emphasized columns of truth values of the two expressions are the same, they are

equivalent. So we can simplify the code snippet without changing the program's behavior by replacing the complicated expression with an equivalent simpler one:

```
if ( x > 0 || y > 100 )
  ...
(further instructions)
```

The equivalence of (3.2) and (3.3) can also be confirmed reasoning by cases:

A is T. An expression of the form .T OR anything/ is equivalent to T. Since A is T both (3.2) and (3.3) in this case are of this form, so they have the same truth value, namely, T.

A is F. An expression of the form .F OR *anything*/ will have same truth value as *anything*. Since A is F, (3.3) has the same truth value as B.

An expression of the form .T AND *anything*/ is equivalent to *anything*, as is any expression of the form F OR *anything*. So in this case AOR .NOT.A/ AND B/ is equivalent to .NOT.A/ AND B/, which in turn is equivalent to B.

Therefore both (3.2) and (3.3) will have the same truth value in this case, namely, the value of B.

Simplifying logical expressions has real practical importance in computer sci ence. Expression simplification in programs like the one above can make a program easier to read and understand. Simplified programs may also run faster, since they require fewer operations. In hardware, simplifying expressions can decrease the number of logic gates on a chip because digital circuits can be described by logical formulas (see Problems 3.5 and 3.6). Minimizing the logical formulas corresponds to reducing the number of gates in the circuit. The payoff of gate minimization is potentially enormous: a chip with fewer gates is smaller, consumes less power, has a lower defect rate, and is cheaper to manufacture.

3.2.2 Cryptic Notation

Java uses symbols like “&&” and “jj” in place of AND and OR. Circuit designers use “” and “C,” and actually refer to AND as a product and OR as a sum. Mathe maticians use still other symbols, given in the table below.

Symbolic Notation

:P (alternatively, P)

English

NOT.P /

P AND Q	P ^ Q
P OR Q	P _ Q
P IMPLIES Q	P ! Q
if P then Q	P ! Q
P IFF Q	P ! Q
P XOR Q	P ˚ Q

For example, using this notatio n, “If P AND NOT.Q/, then R” would be written: .P ^ Q/ ! R:

“AND” and “OR” are easier to remember an d won’t get confused with operations on numbers.

We will often use P as an abbre viation for NOT.P /, but aside from that, we mostly stick to the words—except whe n formulas would otherwise run off the page.

The mathematical notation i s concise but cryptic. Words such as

3.3 Equivalence and Validity

and Co ntrapositives

3.3.1 Implications

Do these two sentences say the same thing?

If I am hungry, then I am grumpy.
If I am not grumpy, then I am not hungry.

We can settle the issue by recas ting both sentences in terms of propositional logic.

Let P be the proposition “I am h ungry” and Q be “I am grumpy.” The first sentence says “P IMPLIES Q” and the second says “NOT.Q/ IMPLIES NOT.P /.” Once more, we can compare these tw o statements in a truth table:

P	Q	.P IMPLIES Q/	.NOT.Q/ IMPLIES NOT.P ///
T	T	T	T
T	F	F	F
F	T	T	T
F	F	T	T

Sure enough, the highlighted columns showing the truth values of these two state ments are the same. A statement of the form "NOT.Q/ IMPLIES NOT.P /" is called

the *contrapositive* of the implication "P IMPLIES Q." The truth table shows that an implication and its contrapositive are equivalent—they are just different ways of saying the same thing.

In contrast, the *converse* of "P IMPLIES Q" is the statement "Q IMPLIES P." The converse to our example is:

If I am grumpy, then I am hungry.

This sounds like a rather different contention, and a truth table confirms this suspi cion:

P	Q	P IMPLIES Q	Q IMPLIES P
T	T	T	T
T	F	F	T
F	T	T	F
F	F	T	T

Now the highlighted columns differ in the second and third row, confirming that an implication is generally *not* equivalent to its converse.

One final relationship: an implication and its converse together are equivalent to an iff statement, specifically, to these two statements together. For example,

If I am grumpy then I am hungry, and if I am hungry then I am grumpy. are equivalent to the single statement:

I am grumpy iff I am hungry.

Once again, we can verify this with a truth table.

P	Q	.P IMPLIES Q/	AND	.Q IMPLIES P /	P IFF Q
T	T	T	T	T	T
T	F	F	F	T	F
F	T	T	F	F	F
F	F	T	T	T	T

The fourth column giving the truth values of

.P IMPLIES Q/ AND .Q IMPLIES P /

is the same as the sixth column giving the truth values of P IFF Q, which confirms that the AND of the implications is equivalent to the IFF statement.

3.3.2 Validity and Satisfiability

A *valid* formula is one which is *always* true, no matter what truth values its

vari ables may have. The simplest example is

P OR NOT.P /:

You can think about valid formulas as capturing fundamental logical truths. For example, a property of implication that we take for granted is that if one statement implies a second one, and the second one implies a third, then the first implies the third. The following valid formula confirms the truth of this property of implication.

Œ.P IMPLIES Q/ AND .Q IMPLIES R/ç IMPLIES .P IMPLIES R/:

Equivalence of formulas is really a special case of validity. Namely, statements F and G are equivalent precisely when the statement .F IFF G/ is valid. For example, the equivalence of the expressions (3.3) and (3.2) means that

.A OR B/ IFF .A OR .NOT.A/ AND B//

is valid. Of course, validity can also be viewed as an aspect of equivalence. Namely, a formula is valid iff it is equivalent to T.

A *satisfiable* formula is one which can *sometimes* be true—that is, there is some assignment of truth values to its variables that makes it true. One way satisfiabil ity comes up is when there are a collection of system specifications. The job of the system designer is to come up with a system that follows all the specs. This means that the AND of all the specs must be satisfiable or the designer's job will be impossible (see Problem 3.12).

There is also a close relationship between validity and satisfiability: a statement P is satisfiable iff its negation NOT.P / is *not* valid.

3.4 The Algebra of Propositions

3.4.1 Propositions in Normal Form

Every propositional formula is equivalent to a "sum-of-products" or *disjunctive form*. More precisely, a disjunctive form is simply an OR of AND-terms, where each AND-term is an AND of variables or negations of variables, for example,

.A AND B/ OR .A AND C /: (3.4)

You can read a disjunctive form for any propositional formula directly from its truth table. For example, the formula

A AND .B OR C / (3.5)

has truth table:

A	B	C	A AND .B OR C /
T	T	T	T
T	T	F	T
T	F	T	T
T	F	F	F
F	T	T	F
F	T	F	F

F F T F
F F F F

The formula (3.5) is true in the first row when A, B, and C are all true, that is, where A AND B AND C is true. It is also true in the second row where A AND B AND C is true, and in the third row when A AND B AND C is true, and that's all. So (3.5) is true exactly when

.A AND B AND C / OR .A AND B AND C / OR .A AND B AND C / (3.6) is

true.

Theorem 3.4.1. *[Distributive Law of* AND *over* OR*]*

A AND .B OR C / *is equivalent to* .A AND B/ OR .A AND C /:

Theorem 3.4.1 is called a *distributive law* because of its resemblance to the dis tributivity of products over sums in arithmetic.

Similarly, we have (Problem 3.10):

Theorem 3.4.2. *[Distributive Law of* OR *over* AND*]*

A OR .B AND C / *is equivalent to* .A OR B/ AND .A OR C /:

Note the contrast between Theorem 3.4.2 and arithmetic, where sums do not distribute over products.

The expression (3.6) is a disjunctive form where each AND-term is an AND of *every one* of the variables or their negations in turn. An expression of this form is called a *disjunctive normal form* (*DNF*). A DNF formula can often be simplified into a smaller disjunctive form. For example, the DNF (3.6) further simplifies to the equivalent disjunctive form (3.4) above.

Applying the same reasoning to the F entries of a truth table yields a *conjunctive form* for any formula—an AND of OR-terms in which the OR-terms are OR's only of variables or their negations. For example, formula (3.5) is false in the fourth row of its truth table (3.4.1) where A is T, B is F and C is F. But this is exactly the one row where .A OR B OR C / is F! Likewise, the (3.5) is false in the fifth row which is exactly where .A OR B OR C / is F. This means that (3.5) will be F whenever the AND of these two OR-terms is false. Continuing in this way with the OR-terms corresponding to the remaining three rows where (3.5) is false, we get a *conjunctive normal form* (*CNF*) that is equivalent to (3.5), namely,

.A OR B OR C / AND .A OR B OR C / AND .A OR B OR C /AND
.A OR B OR C / AND .A OR B OR C /

The methods above can be applied to any truth table, which implies

Theorem 3.4.3. *Every propositional formula is equivalent to both a disjunctive normal form and a conjunctive normal form.*

3.4.2 Proving Equivalences

A check of equivalence or validity by truth table runs out of steam pretty quickly: a proposition with n variables has a truth table with 2^n lines, so the effort required to check a proposition grows exponentially with the number of variables. For a proposition with just 30 variables, that's already over a

billion lines to check!

An alternative approach that *sometimes* helps is to use algebra to prove equiv alence. A lot of different operators may appear in a propositional formula, so a useful first step is to get rid of all but three: AND, OR, and NOT. This is easy because each of the operators is equivalent to a simple formula using only these three. For example, A IMPLIES B is equivalent to NOT.A/ OR B. Formulas using onlyAND, OR, and NOT for the remaining operators are left to Problem 3.13.

We list below a bunch of equivalence axioms with the symbol " ! " between equivalent formulas. These axioms are important because they are all that's needed to prove every possible equivalence. We'll start with some equivalences for AND's that look like the familiar ones for multiplication of numbers:

A AND B ! B AND A (commutativity of AND) (3.7)

.A AND B/ AND C ! A AND .B AND C / (associativity of AND) (3.8) T AND A ! A (identity for AND)

F AND A ! F (zero for AND)

Three axioms that don't directly correspond to number properties are

A AND A ! A (idempotence for AND)

A AND A ! F (contradiction for AND) (3.9) NOT.A/ ! A (double negation) (3.10)

It is associativity (3.8) that justifies writing A AND B AND C without specifying whether it is parenthesized as A AND .B AND C / or .A AND B/ AND C. Both ways of inserting parentheses yield equivalent formulas.

There are a corresponding set of equivalences for OR which we won't bother to list, except for the OR rule corresponding to contradiction for AND (3.9):

A OR A ! T (validity for OR)

Finally, there are *DeMorgan's Laws* which explain how to distribute NOT's over AND's and OR's:

NOT.A AND B/ ! A OR B (DeMorgan for AND) (3.11) NOT.A OR B/ ! A AND B (DeMorgan for OR) (3.12)

All of these axioms can be verified easily with truth tables.

These axioms are all that's needed to convert any formula to a disjunctive normal form. We can illustrate how they work by applying them to turn the negation of formula (3.5),

NOT..A AND B/ OR .A AND C //: (3.13)

into disjunctive normal form.

We start by applying DeMorgan's Law for OR (3.12) to (3.13) in order to move the NOT deeper into the formula. This gives

NOT.A AND B/ AND NOT.A AND C /:

Now applying Demorgan's Law for AND (3.11) to the two innermost AND-terms, gives

.A OR B/ AND .A OR C /: (3.14)

At this point NOT only applies to variables, and we won't need Demorgan's Laws any further.

Now we will repeatedly apply The Distributivity of AND over OR (Theorem 3.4.1) to turn (3.14) into a disjunctive form. To start, we'll distribute .A OR B/ over AND to get

..A OR B/ AND A/ OR ..A OR B/ AND C /:

Using distributivity over both AND's we get

..A AND A/ OR .B AND A// OR ..A AND C / OR .B AND C //:

By the way, we've implicitly used commutativity (3.7) here to justify distributing over an AND from the right. Now applying idempotence to remove the duplicate occurrence of A we get

.A OR .B AND A// OR ..A AND C / OR .B AND C //:

Associativity now allows dropping the parentheses around the terms being OR'd to yield the following disjunctive form for (3.13):

A OR .B AND A/ OR .A AND C / OR .B AND C /: (3.15)

The last step is to turn each of these AND-terms into a disjunctive normal form with all three variables A, B, and C. We'll illustrate how to do this for the second AND-term .B AND A/. This term needs to mention C to be in normal form. To introduce C, we use validity for OR and identity for AND to conclude that

.B AND A/ ! .B AND A/ AND .C OR C /:

Now distributing .B AND A/ over the OR yields the disjunctive normal form .B AND A AND C / OR .B AND A AND C /:

Doing the same thing to the other AND-terms in (3.15) finally gives a disjunctive normal form for (3.5):

.A AND B AND C / OR .A AND B AND C / OR
.A AND B AND C / OR .A AND B AND C / OR
.B AND A AND C / OR .B AND A AND C / OR
.A AND C AND B/ OR .A AND C AND B/ OR
.B AND C AND A/ OR .B AND C AND A/:

Using commutativity to sort the term and OR-idempotence to remove duplicates, finally yields a unique sorted DNF:

.A AND B AND C / OR
.A AND B AND C / OR
.A AND B AND C / OR
.A AND B AND C / OR
.A AND B AND C /:

This example illustrates a strategy for applying these equivalences to convert any formula into disjunctive normal form, and conversion to conjunctive normal form works similarly, which explains:

Theorem 3.4.4. *Any propositional formula can be transformed into disjunctive normal form or a conjunctive normal form using the equivalences listed above.*

What has this got to do with equivalence? That's easy: to prove that two for mulas are equivalent, convert them both to disjunctive normal form over the set of variables that appear in the terms. Then use commutativity to sort the variables and AND-terms so they all appear in some standard order. We claim the formulas are equivalent iff they have the same sorted disjunctive normal form. This is obvious if they do have the same disjunctive normal form. But conversely, the way we read off a disjunctive normal form from a truth table shows that two different sorted DNF's over the same set of variables correspond to different truth tables and hence to inequivalent formulas. This proves

Theorem 3.4.5 (Completeness of the propositional equivalence axioms). *Two propo sitional formula are equivalent iff they can be proved equivalent using the equiva lence axioms listed above.*

The benefit of the axioms is that they leave room for ingeniously applying them to prove equivalences with less effort than the truth table method. Theorem 3.4.5 then adds the reassurance that the axioms are guaranteed to prove every equiva lence, which is a great punchline for this section. But we don't want to mislead you: it's important to realize that using the strategy we gave for applying the ax ioms involves essentially the same effort it would take to construct truth tables, and there is no guarantee that applying the axioms will generally be any easier than using truth tables.

3.5 The SAT Problem

Determining whether or not a more complicated proposition is satisfiable is not so easy. How about this one?

$$.P \text{ OR } Q \text{ OR } R/ \text{ AND } .P \text{ OR } Q/ \text{ AND } .P \text{ OR } R/ \text{ AND } .R \text{ OR } Q/$$

The general problem of deciding whether a proposition is satisfiable is called *SAT*. One approach to SAT is to construct a truth table and check whether or not a T ever appears, but as with testing validity, this approach quickly bogs down for formulas with many variables because truth tables grow exponentially with the number of variables.

Is there a more efficient solution to SAT? In particular, is there some brilliant procedure that determines SAT in a number of steps that grows *polynomially*—like

n^2 or n^{14}—instead of *exponentially*—2^n—whether any given proposition of size n is satisfiable or not? No one knows. And an awful lot hangs on the answer. The general definition of an "efficient" procedure is one that runs in *polynomial time*, that is, that runs in a number of basic steps bounded

by a polynomial in s, where s is the size of an input. It turns out that an efficient solution to SAT would immediately imply efficient solutions to many other important problems involving scheduling, routing, resource allocation, and circuit verification across multiple dis ciplines including programming, algebra, finance, and political theory. This would be wonderful, but there would also be worldwide chaos. Decrypting coded mes sages would also become an easy task, so online financial transactions would be insecure and secret communications could be read by everyone. Why this would happen is explained in Section 8.12.

Of course, the situation is the same for validity checking, since you can check for validity by checking for satisfiability of a negated formula. This also explains why the simplification of formulas mentioned in Section 3.2 would be hard—validity testing is a special case of determining if a formula simplifies to T.

Recently there has been exciting progress on *SAT-solvers* for practical applica tions like digital circuit verification. These programs find satisfying assignments with amazing efficiency even for formulas with millions of variables. Unfortu nately, it's hard to predict which kind of formulas are amenable to SAT-solver meth ods, and for formulas that are *un*satisfiable, SAT-solvers generally get nowhere.

So no one has a good idea how to solve SAT in polynomial time, or how to prove that it can't be done—researchers are completely stuck. The problem of determining whether or not SAT has a polynomial time solution is known as the "P vs. NP" problem.[1] It is the outstanding unanswered question in theoretical computer science. It is also one of the seven Millenium Problems: the Clay Institute will award you $1,000,000 if you solve the P vs. NP problem.

3.6 Predicate Formulas

3.6.1 Quantifiers

The "for all" notation, 8, has already made an early appearance in Section 1.1. For example, the predicate

$$\text{“}x^2 \ 0\text{”}$$

[1]P stands for problems whose instances can be solved in time that grows polynomially with the size of the instance. NP stands for *nondeterministtic polynomial time*, but we'll leave an explanation of what that is to texts on the theory of computational complexity.

is always true when x is a real number. That is,

$$8x \ 2 \ R\text{:} \ x^2 \ 0$$

is a true statement. On the other hand, the predicate

$$\text{“}5x^2 \ 7 \ D \ 0\text{”}$$

is only sometimes true; specifically, when $x \ D \ \pm\sqrt{7=5}$. There is a "there exists" notation, 9, to indicate that a predicate is true for at least one, but not necessarily all objects. So

$$9x \ 2 \ R\text{:} \ 5x^2 \ 7 \ D \ 0$$

is true, while

$$8x \in \mathbb{R}: 5x^2 - 7 > 0$$

is not true.

There are several ways to express the notions of "always true" and "sometimes true" in English. The table below gives some general formats on the left and specific examples using those formats on the right. You can expect to see such phrases hundreds of times in mathematical writing!

Always True	
For all $x \in D$, $P(x)$ is true.	For all $x \in \mathbb{R}$, $x^2 \geq 0$. $P(x)$ is true for every x in the set, D. $x^2 \geq 0$ for every $x \in \mathbb{R}$.
Sometimes True	
There is an $x \in D$ such that $P(x)$ is true.	There is an $x \in \mathbb{R}$ such that $5x^2 - 7 > 0$. $P(x)$ is true for some x in the set, D. $5x^2 - 7 > 0$ for some $x \in \mathbb{R}$. $P(x)$ is true for at least one $x \in D$. $5x^2 - 7 > 0$ for at least one $x \in \mathbb{R}$.

All these sentences "quantify" how often the predicate is true. Specifically, an assertion that a predicate is always true is called a *universal quantification*, and an assertion that a predicate is sometimes true is an *existential quantification*. Some times the English sentences are unclear with respect to quantification:

> If you can solve any problem we come up with,
> then you get an *A* for the course. (3.16)

The phrase "you can solve any problem we can come up with" could reasonably be interpreted as either a universal or existential quantification:

> you can solve *every* problem we come up with, (3.17)

or maybe

> you can solve *at least one* problem we come up with. (3.18)

To be precise, let Probs be the set of problems we come up with, Solves.x/ be the predicate "You can solve problem x," and G be the proposition, "You get an *A* for the course." Then the two different interpretations of (3.16) can be written as follows:

$$\forall x \in \text{Probs}: \text{Solves}(x) \text{ IMPLIES } G; \text{ for (3.17)};$$
$$\exists x \in \text{Probs}: \text{Solves}(x) \text{ IMPLIES } G: \text{ for (3.18)}:$$

3.6.2 Mixing Quantifiers

Many mathematical statements involve several quantifiers. For example, we al ready described

> Goldbach's Conjecture 1.1.8: Every even integer greater than 2 is the sum of two primes.

Let's write this out in more detail to be precise about the quantification:

> For every even integer n greater than 2, there exist primes p and q such that $n = p + q$.

Let Evens be the set of even integers greater than 2, and let Primes be the set of primes. Then we can write Goldbach's Conjecture in logic notation as follows:

$$8\, n\, 2\ \text{Evens}\quad 9\, p\, 2\ \text{Primes}\ \ 9q\, 2\ \text{Primes}$$
$$\text{„}\quad \underset{\text{for every even}}{\underline{f,\ldots}}\ \text{„}\quad \underset{\text{there exist primes integer n>2}}{\underline{f,\ldots}}\ \underset{\text{p and q such that}}{}: n\ D\ p\ C\ q:$$

3.6.3 Order of Quantifiers

Swapping the order of different kinds of quantifiers (existential or universal) usually changes the meaning of a proposition. For example, let's return to one of our initial, confusing statements:

"Every American has a dream."

This sentence is ambiguous because the order of quantifiers is unclear. Let A be the set of Americans, let D be the set of dreams, and define the predicate H.a; d / to be "American a has dream d." Now the sentence could mean there is a single dream that every American shares—such as the dream of owning their own home:

$$9\ 2\ 8\ 2\ d\ D\ a\ A: H.a; d /$$

Or it could mean that every American has a personal dream:

$$8a\ 2\ A\ 9\ d\ 2\ D: H.a; d /$$

For example, some Americans may dream of a peaceful retirement, while others dream of continuing practicing their profession as long as they live, and still others may dream of being so rich they needn't think about work at all.

Swapping quantifiers in Goldbach's Conjecture creates a patently false statement that every even number 2 is the sum of *the same* two primes:

$$\underset{\text{„}}{9}\ p\ 2\ \text{Primes}\ 9\ q\ 2\ \text{Primes}: 8n\ 2\ \text{Evens}\ n\ D\ p\ C\ q:$$
$$\underset{\text{p and q such that}}{\underline{f,\ldots}}\ \text{„}\quad \underset{\text{there exist primes integer n>2}}{\underline{\ldots}}\ \underset{\text{for every even}}{}$$

3.6.4 Variables Over One Domain

When all the variables in a formula are understood to take values from the same nonempty set, D, it's conventional to omit mention of D. For example, instead of 8x 2 D 9y 2 D: Q.x; y/ we'd write 8x9y: Q.x; y/. The unnamed nonempty set that x and y range over is called the *domain of discourse*, or just plain *domain*, of the formula.

It's easy to arrange for all the variables to range over one domain. For exam ple, Goldbach's Conjecture could be expressed with all variables ranging over the domain N as

8n: n 2 Evens IMPLIES .9 p 9 q: p 2 Primes AND q 2 Primes AND n D p C q/:

3.6.5 Negating Quantifiers

There is a simple relationship between the two kinds of quantifiers. The following two sentences mean the same thing:

Not everyone likes ice cream.

There is someone who does not like ice cream.

The equivalence of these sentences is a instance of a general equivalence that holds between predicate formulas:

NOT.8x: P .x// is equivalent to 9x: NOT.P .x//: (3.19) Similarly,

these sentences mean the same thing:

There is no one who likes being mocked.

Everyone dislikes being mocked.

"mcs" — 2015/5/18 — 1:43 — page 60 — #68

The corresponding predicate formula equivalence is

NOT.9x: P .x// is equivalent to 8x: NOT.P .x//: (3.20)

The general principle is that *moving a* NOT *across a quantifier changes the kind of quantifier.* Note that (3.20) follows from negating both sides of (3.19).

3.6.6 Validity for Predicate Formulas

The idea of validity extends to predicate formulas, but to be valid, a formula now must evaluate to true no matter what the domain of discourse may be, no matter what values its variables may take over the domain, and no matter what interpreta tions its predicate variables may be given. For example, the equivalence (3.19) that gives the rule for negating a universal quantifier means that the following formula is valid:

NOT.8x: P .x// IFF 9x: NOT.P .x//: (3.21)

Another useful example of a valid assertion is

9x8y: P .x; y/ IMPLIES 8y9x: P .x; y/: (3.22)

Here's an explanation why this is valid:

Let D be the domain for the variables and P_0 be some binary predi cate[2] on D. We need to show that if

9x 2 D:8y 2 D: P_0.x; y/ (3.23)

holds under this interpretation, then so does

8y 2 D 9x 2 D: P_0.x; y/: (3.24)

So suppose (3.23) is true. Then by definition of 9, this means that some element d_0 2 D has the property that

8y 2 D: $P_0.d_0$; y/:

By definition of 8, this means that

$P_0.d_0$;d/

is true for all d 2 D. So given any d 2 D, there is an element in D,

namely, d_0, such that $P_0.d_0;d/$ is true. But that's exactly what (3.24) means, so we've proved that (3.24) holds under this interpretation, as required.

[2]That is, a predicate that depends on two variables.

We hope this is helpful as an explanation, but we don't really want to call it a "proof." The problem is that with something as basic as (3.22), it's hard to see what more elementary axioms are ok to use in proving it. What the explanation above did was translate the logical formula (3.22) into English and then appeal to the meaning, in English, of "for all" and "there exists" as justification.

In contrast to (3.22), the formula

$$8y9x: P .x; y/ \text{ IMPLIES } 9x8y: P .x; y/: \quad (3.25)$$

is *not* valid. We can prove this just by describing an interpretation where the hy pothesis, 8y9x: P .x; y/, is true but the conclusion, 9x8y: P .x; y/, is not true. For example, let the domain be the integers and P .x; y/ mean x>y. Then the hy pothesis would be true because, given a value, n, for y we could choose the value of x to be n C 1, for example. But under this interpretation the conclusion asserts that there is an integer that is bigger than all integers, which is certainly false. An interpretation like this that falsifies an assertion is called a *counter model* to that assertion.

3.7 References

[18]

Problems for Section 3.1

Practice Problems

Problem 3.1.
Some people are uncomfortable with the idea that from a false hypothesis you can prove everything, and instead of having P IMPLIES Q be true when P is false, they want P IMPLIES Q to be false when P is false. This would lead to IMPLIES having the same truth table as what propositional connective?

Problem 3.2.
Your class has a textbook and a final exam. Let P, Q, and R be the following propositions:

PWWD You get an A on the final exam.

QWWD You do every exercise in the book.

RWWD You get an A in the class.

Translate following assertions into propositional formulas using P, Q, R and the propositional connectives AND; NOT; IMPLIES.

(a) You get an A in the class, but you do not do every exercise in the book.

(b) You get an A on the final, you do every exercise in the book, and you get an A in the class.

(c) To get an A in the class, it is necessary for you to get an A on the final.

(d) You get an A on the final, but you don't do every exercise in this book; never theless, you get an A in this class.

Class Problems

Problem 3.3.
When the mathematician says to his student, "If a function is not continuous, then it is not differentiable," then letting D stand for "differentiable" and C for continuous, the only proper translation of the mathematician's statement would be

$$\text{NOT.C / IMPLIES NOT.D/;}$$

or equivalently,

$$\text{D IMPLIES C:}$$

But when a mother says to her son, "If you don't do your homework, then you can't watch TV," then letting T stand for "can watch TV" and H for "do your homework," a reasonable translation of the mother's statement would be

$$\text{NOT.H / IFF NOT.T /;}$$

or equivalently,

$$\text{H IFF T:}$$

Explain why it is reasonable to translate these two IF-THEN statements in dif ferent ways into propositional formulas.

Homework Problems

Problem 3.4.
Describe a simple procedure which, given a positive integer argument, n,

produces a width n array of truth-values whose rows would be all the possible truth-value assignments for n propositional variables. For example, for n D 2, the array would be:

T T
T F
F T
F F

Your description can be in English, or a simple program in some familiar lan guage such as Python or Java. If you do write a program, be sure to include some sample output.

Problems for Section 3.2

Class Problems

Problem 3.5.
Propositional logic comes up in digital circuit design using the convention that T corresponds to 1 and F to 0. A simple example is a 2-bit *half-adder* circuit. This circuit has 3 binary inputs, a_1; a_0 and b, and 3 binary outputs, c; s_1; s_0. The 2-bit word a_1a_0 gives the binary representation of an integer, k, between 0 and 3. The 3-bit word cs_1s_0 gives the binary representation of k C b. The third output bit, c, is called the final *carry bit*.

So if k and b were both 1, then the value of a_1a_0 would be 01 and the value of the output cs_1s_0 would 010, namely, the 3-bit binary representation of 1 C 1. In fact, the final carry bit equals 1 only when all three binary inputs are 1, that is, when k D 3 and b D 1. In that case, the value of cs_1s_0 is 100, namely, the binary representation of 3 C 1.

This 2-bit half-adder could be described by the following formulas:

$$c_0 \text{ D } b$$
$$s_0 \text{ D } a_0 \text{ XOR } c_0$$
$$c_1 \text{ D } a_0 \text{ AND } c_0 \text{ the carry into column 1}$$
$$s_1 \text{ D } a_1 \text{ XOR } c_1$$
$$c_2 \text{ D } a_1 \text{ AND } c_1 \text{ the carry into column 2}$$
$$c \text{ D } c_2:$$

(a) Generalize the above construction of a 2-bit half-adder to an n C 1 bit half adder with inputs a_n;:::;a_1; a_0 and b and outputs c; s_n;:::;s_1; s_0. That is, give simple formulas for s_i and c_i for 0 i n C 1, where c_i is the carry into column i C 1, and c D c_{nC1}.

(b) Write similar definitions for the digits and carries in the sum of two n C 1-bit binary numbers a_n :::a_1a_0 and b_n :::b_1b_0.

Visualized as digital circuits, the above adders consist of a sequence of single digit half-adders or adders strung together in series. These circuits mimic ordinary pencil-and-paper addition, where a carry into a column is calculated directly from the carry into the previous column, and the carries

have to ripple across all the columns before the carry into the final column is determined. Circuits with this design are called *ripple-carry* adders. Ripple-carry adders are easy to understand and remember and require a nearly minimal number of operations. But the higher order output bits and the final carry take time proportional to n to reach their final values.

(c) How many of each of the propositional operations does your adder from part (b) use to calculate the sum?

Homework Problems

Problem 3.6.

There are adder circuits that are *much* faster, and only slightly larger, than the ripple-carry circuits of Problem 3.5. They work by computing the values in later columns for both a carry of 0 and a carry of 1, *in parallel*. Then, when the carry from the earlier columns finally arrives, the pre-computed answer can be quickly selected. We'll illustrate this idea by working out the equations for an .n C 1/-bit parallel half-adder.

Parallel half-adders are built out of parallel *add1* modules. An .n C 1/-bit *add1* module takes as input the .n C 1/-bit binary representation, $a_n :::a_1a_0$, of an inte ger, s, and produces as output the binary representation, c $p_n :::p_1p_0$, of s C 1.

(a) A 1-bit *add1* module just has input a_0. Write propositional formulas for its outputs c and p_0.

(b) Explain how to build an .nC1/-bit parallel half-adder from an .nC1/-bit *add1* module by writing a propositional formula for the half-adder output, o_i, using only the variables a_i, p_i, and b.

We can build a double-size *add1* module with 2.n C 1/ inputs using two single size *add1* modules with nC1 inputs. Suppose the inputs of the double-size module are $a_{2nC1};:::;a_1$; a_0 and the outputs are $c;p_{2nC1};:::;p_1$; p_0. The setup is illus trated in Figure 3.1.

Namely, the first single size *add1* module handles the first n C 1 inputs. The in puts to this module are the low-order nC1 input bits $a_n;:::;a_1$; a_0, and its outputs will serve as the first n C 1 outputs $p_n;:::;p_1$; p_0 of the double-size module. Let $c_{.1/}$ be the remaining carry output from this module.

The inputs to the second single-size module are the higher-order n C 1 input bits $a_{2nC1};:::;a_{n\ 2\ C\ C}$; $a_{n\ 1}$. Call its first n C 1 outputs $r_n;:::;r_1$; r_0 and let $c_{.2/}$ be its carry.

(c) Write a formula for the carry, c, in terms of $c_{.1/}$ and $c_{.2/}$.

(d) Complete the specification of the double-size module by writing propositional formulas for the remaining outputs, p_i, for n C 1 i 2n C 1. The formula for p_i should only involve the variables a_i, $r_{i\ .n\ 1/\ C}$, and $c_{.1/}$.

(e) Parallel half-adders are exponentially faster than ripple-carry half-adders. Con firm this by determining the largest number of propositional operations required to compute any one output bit of an n-bit add module. (You may assume n is a power of 2.)

Exam Problems

Problem 3.7.

There are exactly two truth environments (assignments) for the variables M; N; P; Q; R; S that satisfy the following formula:

$$\left(Q \text{ AND } \left(\neg Q \text{ OR } \left(R \text{ AND } \left(\neg R \text{ OR } \left(S \text{ AND } \left(\neg S \text{ OR } \left(P \text{ AND } M \text{ AND } N\right)\right)\right)\right)\right)\right)\right) \cdot \neg P$$

clause (1) clause (2) clause (3) clause (4)

(a) This claim could be proved by truth-table. How many rows would the truth table have?

(b) Instead of a truth-table, prove this claim with an argument by cases according to the truth value of P.

$s_o\ s_2\ s_1$

d

3/oD30-bit add2 module

$b_o\ b_2\ b_1$ /oD20-bit add2

$b_{3oD2}\ b_{oD3}\ b_{oD2}$

$d_{/20}\ d_{/30}$ /oD20-bit add2

$q_{3oD2}\ q_{OD3}\ q_{oD2}\ q_o\ q_2\ q_1$ Figure 3.1 Structure of a

Double-size *add1* Module.

Problems for Section 3.3

Practice Problems

Problem 3.8.

Indicate whether each of the following propositional formulas is valid (V), satis fiable but not valid (S), or not satisfiable (N). For the satisfiable ones, indicate a satisfying truth assignment.

M IMPLIES Q

M IMPLIES .P OR Q/

M IMPLIES ŒM AND .P IMPLIES M /ç

.P OR Q/ IMPLIES Q

.P OR Q/ IMPLIES .P AND Q/

.P OR Q/ IMPLIES ŒM AND .P IMPLIES M /ç

.P XOR Q/ IMPLIES Q

.P XOR Q/ IMPLIES .P OR Q/

.P XOR Q/ IMPLIES ŒM AND .P IMPLIES M /ç

Problem 3.9.
Prove that the propositional formulas

P OR Q OR R

and

.P AND NOT.Q// OR.Q AND NOT.R// OR.R AND NOT.P // OR.P ANDQ ANDR/: are

equivalent.

Problem 3.10.
Prove by truth table that OR distributes over AND, namely,

P OR .Q AND R/ is equivalent to .P OR Q/ AND .P OR R/ (3.26)

Class Problems

Problem 3.11. (a) Verify by truth table that

.P IMPLIES Q/ OR .Q IMPLIES P /

is valid.

(b) Let P and Q be propositional formulas. Describe a single formula, R, using only AND's, OR's, NOT's, and copies of P and Q, such that R is valid iff P and Q are equivalent.

(c) A propositional formula is *satisfiable* iff there is an assignment of truth values to its variables—an *environment*—which makes it true. Explain why

P is valid iff NOT.P / is *not* satisfiable.

(d) A set of propositional formulas $P_1;:::;P_k$ is *consistent* iff there is an envi ronment in which they are all true. Write a formula, S, so that the set $P_1;:::;P_k$ is *not* consistent iff S is valid.

Problem 3.12.
This problem[3] examines whether the following specifications are

satisfiable: 1. If the file system is not locked, then

(a) new messages will be queued.

(b) new messages will be sent to the messages buffer.

(c) the system is functioning normally, and conversely, if the system is functioning normally, then the file system is not locked.

2. If new messages are not queued, then they will be sent to the messages buffer. 3. New messages will not be sent to the message buffer.

(a) Begin by translating the five specifications into propositional formulas using four propositional variables:

L WWD file system locked;

Q WWD new messages are queued;

B WWD new messages are sent to the message buffer;

N WWD system functioning normally:

[3]Revised from Rosen, 5th edition, Exercise 1.1.36

(b) Demonstrate that this set of specifications is satisfiable by describing a single truth assignment for the variables L; Q; B; N and verifying that under this assign ment, all the specifications are true.

(c) Argue that the assignment determined in part (b) is the only one that does the job.

Problems for Section 3.4

Practice Problems

Problem 3.13.
A half dozen different operators may appear in propositional formulas, but just AND, OR, and NOT are enough to do the job. That is because each of the operators is equivalent to a simple formula using only these three operators. For example, A IMPLIES B is equivalent to NOT.A/ OR B. So all occurences of IMPLIES in a formula can be replaced using just NOT and OR.

(a) Write formulas using only AND, OR, NOT that are equivalent to each of AIFFB and A XOR B. Conclude that every propositional formula is equivalent to an AND OR-NOT formula.

(b) Explain why you don't even need AND.

(c) Explain how to get by with the single operator NAND where A NAND B is equivalent by definition to NOT.A AND B/.

Class Problems

Problem 3.14.
The propositional connective NOR is defined by the rule

P NOR Q WWD .NOT.P / AND NOT.Q//:

Explain why every propositional formula—possibly involving any of the usual op erators such as IMPLIES, XOR, . . .—is equivalent to one whose only connective is NOR.

Problem 3.15.
Explain how to find a conjunctive form for a propositional formula directly from a disjunctive form for its complement.

Homework Problems

Problem 3.16.
Use the equivalence axioms of Section 3.4.2 to convert the following formula to disjunctive form:

A XOR B XOR C:

Problems for Section 3.5

Homework Problems

Problem 3.17.
A 3-conjunctive form (3CF) formula is a conjunctive form formula in which each OR-term is an OR of at most 3 variables or negations of variables. Although it may be hard to tell if a propositional formula, F , is satisfiable, it is always easy to construct a formula, *C*.F /, that is

- ✏ in 3-conjunctive form,
- ✏ has at most 24 times as many occurrences of variables as F , and
- ✏ is satisfiable iff F is satisfiable.

To construct *C*.F /, introduce a different new variables for each operator that occurs in F . For example, if F was

..P XOR Q/ XOR R/ OR .P AND S / (3.27)

we might use new variables X_1, X_2, O, and A corresponding to the operator oc currences as follows:

XOR Q/ „f,…
..P „f,… „f,….P X_1 X_2
XOR R/ OR O
A
AND S /: „f,…

Next we write a formula that constrains each new variable to have the same truth value as the subformula determined by its corresponding operator. For the example above, these constraining formulas would be

X_1 IFF .P XOR Q/;

X_2 IFF .X_1 XOR R/;

A IFF .P AND S /;

O IFF .X_2 OR A/

(a) Explain why the AND of the four constraining formulas above along with a fifth formula consisting of just the variable O will be satisfiable iff (3.27) is satisfi able.

(b) Explain why each constraining formula will be equivalent to a 3CF formula with at most 24 occurrences of variables.

(c) Using the ideas illustrated in the previous parts, explain how to construct *C*.F / for an arbitrary propositional formula, F .

Problem 3.18.
It doesn't matter whether we formulate the SAT problem (Section 3.5 in terms of propositional formulas or digital circuits. Here's why:

Let f be a Boolean function of k variables. That is, f W fT; Fg^k ! fT; Fg. When P is a propositional formula that has, among its variables, propositional variables labelled X_1;::::;X_k. For any truth values b_1;::::;b_k 2 fT; Fg, we let let P .b_1;::::;b_k/ be the result of substituting b_i for all occurrences of X_i in P, for 1 i k.

If P_f is a formula such that P_f .b_1;::::;b_k/ is satisfiable exactly when f .b_1;::::;b_k/ D T, we'll say that P_f *SAT-represents* f .

Suppose there is a digital circuit using two-input, one-output binary gates (like the circuits for binary addition in Problems 3.5 and 3.6) that has n wires and com putes the function f . Explain how to construct a formula P_f of size cn that SAT represents f for some small constant c. (Letting c D 6 will work).

Conclude that the SAT problem for digital circuits—that is, determining if there is some set of input values that will lead a circuit to give output 1—is no more difficult than the SAT problem for propositional formulas.

Hint: Introduce a new variable for each wire. The idea is similar to the one used in Problem 3.17 to show that satisfiablity of 3CNF propositional formmulas is just as hard as for arbitrary formulas.

Problems for Section 3.6

Practice Problems

Problem 3.19.
For each of the following propositions:

1. 8x 9y: 2x y D 0