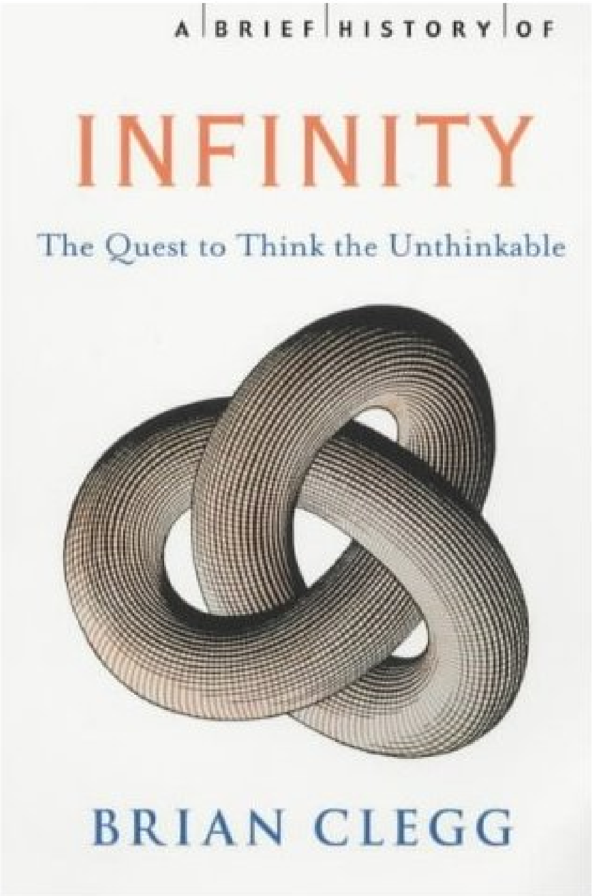
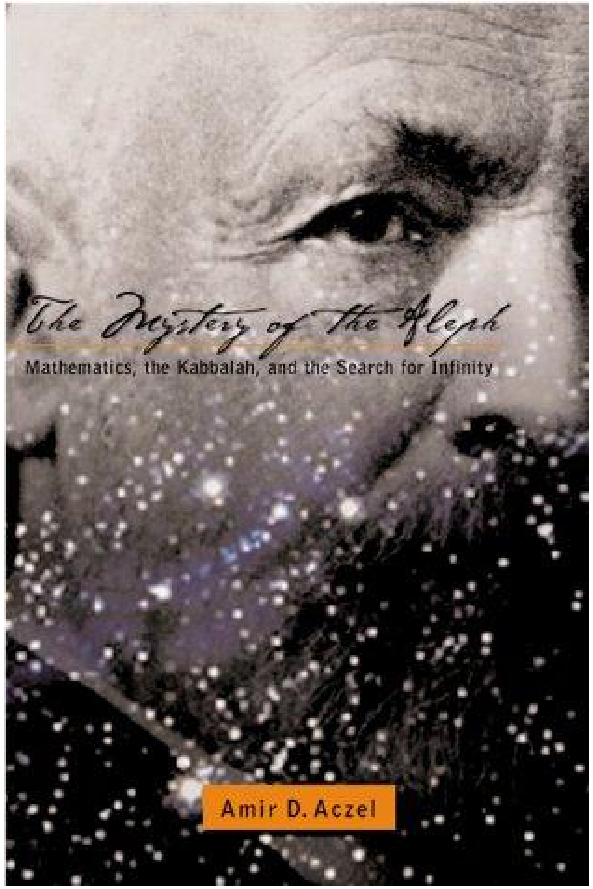
Direct Proofs

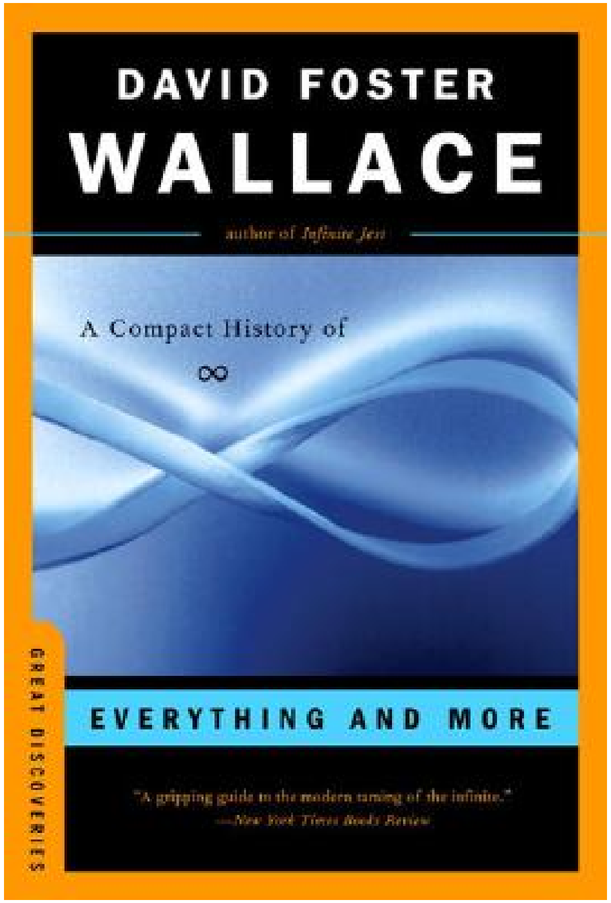
Recommended Reading



A Brief History of Infinity



The Mystery of the Aleph



Everything and More

Recommended Courses

Math 161: Set Theory

Outline for Today

- What is a Proof?
- Direct Proofs
- Universal and Existential Statements
- Extended Example: XOR

What is a Proof?

A ***proof*** is an argument that demonstrates why a conclusion is true.

A ***mathematical proof*** is an argument that demonstrates why a mathematical statement is true.

Structure of a Mathematical Proof

- Despite what you might think, mathematical proofs are not supposed to be jumbles of dense symbols.
- Good mathematical proofs read like argumentative essays that happen to use math to convey their arguments.
- Typically, proofs begin with a set of assumptions, combine those assumptions together in a series of steps, and ultimately arrive at the conclusion.
- They're best explored by example.

Two Quick Definitions

- An integer n is ***even*** if there is some integer k such that $n = 2k$.
  - This means that 0 is even.
- An integer n is ***odd*** if there is some integer k such that $n = 2k + 1$.
- We'll assume the following for now:
  - Every integer is either even or odd.
  - No integer is both even and odd.

Our First Direct Proof

Theorem: If n is an even integer, then n^2 is even.

Proof: Let n be an even integer.

Since n is even, there is some integer k such that $n = 2k$.

This means that $n^2 = (2k)^2 = 4k^2 = 2(2k^2)$.

From this, we see that there is an integer m (namely, $2k^2$) where $n^2 = 2m$.

Therefore, n^2 is even. ■

Our First Direct Proof

Theorem: If n is an even integer, then n^2 is even.

Proof: Let n be an even integer.

Since n is even, there is some integer k such that $n = 2k$.

This means that $n^2 = (2k)^2 = 4k^2 = 2(2k^2)$.

From this, we see that there is an integer m (namely, $2k^2$) where $n^2 = 2m$.

Therefore, n^2 is even. ■

This symbol means “end of proof”

Our First Direct Proof

Theorem: If n is an even integer, then n^2 is even.

Proof: Let n be an even integer.

Since n i

such tha

This mea

).

From thi

r

m (name

Therefor

To prove a statement of the form

“If *P*, then *Q*”

Assume that ***P*** is true, then show that ***Q*** must be true as well.

Our First Direct Proof

Theorem: If n is an even integer, then n^2 is even.
Proof: Let n be an even integer.

Since n is even, there is some integer k such that $n = 2k$.

This means th ^{2}).

From this, we er
m (namely, $2k$

Therefore, n^2

This is the definition of an even integer. When writing a mathematical proof, it's common to call back to the definitions.

Our First Direct Proof

Theorem: If n is an even integer, then n^2 is even.

Proof: Let n be an even integer.

Since n is even, there is some integer k such that $n = 2k$.

This means that $n^2 = (2k)^2 = 4k^2 = 2(2k^2)$.

Fr

m

Th

Notice how we use the value of k that we obtained above. Giving names to quantities, even if we aren't fully sure what they are, allows us to manipulate them. This is similar to variables in programs.

Our First Direct Proof

Theorem: If n i n.

Proof: Let n b

Since n

such th

This means that $n^2 = (2k)^2 = 4k^2 = 2(2k^2)$.

Our ultimate goal is to prove that **n^2** is even. This means that we need to find some **m** such that **$n^2 = 2m$**. Here, we're explicitly showing how we can do that.

From this, we see that there is an integer m (namely, $2k^2$) where $n^2 = 2m$.

Therefore, n^2 is even. ■

Our First Direct Proof

Theorem: If n is an even integer, then n^2 is even.

Proof: Let n be an even integer.

Since n is even, there is some integer k such that $n = 2k$.

This mea).

From thi r m (name

Hey, that's what we were trying to show! We're done now.

Therefore, n^2 is even. ■

That wasn't so bad! Let's do another one.

Some Helpful Set Theory

- Set equality is defined as follows:

If A and B are sets, then $A = B$ precisely when every element of A is an element of B and vice-versa.

- In practice, this definition is a bit tricky to work with.
- It's often easier to use the following result to show that two sets are equal:

For any sets A and B,
if $A \subseteq B$ and $B \subseteq A$, then $A = B$.

Theorem: For any sets A and B, if $A \subseteq B$ and $B \subseteq A$, then $A = B$.

How do we prove that this is true for *any* choice of sets?

Proving Something Always Holds

- Many statements have the form

For any *x*, `[some-property]` holds of *x*.

- Examples:

For all integers n, if n is even, n^2 is even.

For any sets A and B, if $A \subseteq B$ and $B \subseteq A$, then $A = B$.

For all sets S, $|S| < |\wp(\mathrm{S})|$.

Everything that drowns me makes me wanna fly.

- How do we prove these statements when there are (potentially) infinitely many cases to check?

Arbitrary Choices

- To prove that some property holds true for all possible *x*, show that no matter what choice of *x* you make, that property must be true.
- Start the proof by making an ***arbitrary choice*** of *x*:
  - “Let *x* be chosen arbitrarily.”
  - “Let *x* be an arbitrary even integer.”
  - “Let *x* be an arbitrary set containing 137.”
  - “Consider any *x*.”
- Demonstrate that the property holds true for this choice of *x*.

Theorem: For any sets A and B, if $A \subseteq B$ and $B \subseteq A$, then $A = B$.

Proof: Let A and B be arbitrary sets where $A \subseteq B$ and $B \subseteq A$.

We're showing here that regardless of what **A** and **B** you pick, the result will still be true.

Theorem: For any sets A and B, if $A \subseteq B$ and $B \subseteq A$, then $A = B$.

Proof: Let A and B be arbitrary sets where $A \subseteq B$ and $B \subseteq A$.

To prove a statement of the form

"If *P*, then *Q*"

Assume that ***P*** is true, then show that ***Q*** must be true as well.

Theorem: For any sets A and B, if $A \subseteq B$ and $B \subseteq A$, then $A = B$.

Proof: Let A and B be arbitrary sets where $A \subseteq B$ and $B \subseteq A$.

Because $A \subseteq B$, if we take an arbitrary $x \in A$, we know that $x \in B$. Similarly, since $B \subseteq A$, if we take an arbitrary $x \in B$, we'll see that $x \in A$ as well.

Therefore, every element of A is an element of B and every element of B is an element of A. Therefore, by definition of set equality, we see that $A = B$. ■

An Incorrect Proof

Theorem: If A and B are sets, then $A \subseteq A \cap B$.

Proof: Consider two arbitrary sets, say, $A = \emptyset$ and $B = \mathbb{N}$. Since $\emptyset$ is a subset of every set and $A = \emptyset$, we see that $A \subseteq A \cap B$. Since our choices of A and B were arbitrary, we conclude that if A and B are any sets, then $A \subseteq A \cap B$. ■

An Incorrect Proof

Theorem: If A and B are sets, then $A \subseteq A \cap B$.

Proof: Consider two arbitrary sets, say, $A = \emptyset$ and $B = \mathbb{N}$. Since Ø is a subset of every set and $A = \emptyset$, we see that $A \subseteq A \cap B$. Since our choices of A and B were arbitrary, we conclude that if A and B are any sets, then $A \subseteq A \cap B$. ■

ar·bi·trar·y
adjective /ˈärbiˌtrerē/

…not this one!

1. Based on random choice or personal whim, rather than any reason or system - *“his mealtimes were entirely arbitrary”*

2. *(of power or a ruling body)* Unrestrained and autocratic in the use of authority - *“arbitrary rule by King and bishops has been made impossible”*

3. *(of a constant or other quantity)* Of unspecified value

Use this definition…

Source: Google

To prove something is true for all *x*,
don't choose an *x* and base the proof
off of your choice.

Instead, leave *x* unspecified
and show that no matter what *x* is,
the specified property must hold.

Another Incorrect Proof

Theorem: If A and B are sets, then $A \subseteq A \cap B$.

Proof: We need to show that if $x \in A$, then $x \in A \cap B$ as well.

Consider any arbitrary $x \in A \cap B$. This means that $x \in A$ and $x \in B$.

In particular, we see that $x \in A$, as required. ■

Another Incorrect Proof

Theorem: If A and B are sets, then $A \subseteq A \cap B$.

Proof: We need to show that if $x \in A$, then $x \in A \cap B$ as well.

Consider any arbitrary $x \in A \cap B$. This means that $x \in A$ and $x \in B$.

In particular, we see that $x \in A$, as required. ■

If you want to prove that P implies Q,
assume P and prove Q.

Don't assume Q and then prove P!

An Entirely Different Proof

Theorem: There exists a natural number $n > 0$ such that the sum of all natural numbers less than n is equal to n.

This is a fundamentally different type of proof that what we've done before. Instead of showing that *every* object has some property, we want to show that *some* object has a given property.

Universal vs. Existential Statements

- A ***universal statement*** is a statement of the form

For all *x*, `[some-property]` holds for *x*.

- We've seen how to prove these statements.
- An ***existential statement*** is a statement of the form

There is some *x* where `[some-property]` holds for *x*.

- How do you prove an existential statement?

Proving an Existential Statement

- We will see several different ways to prove an existential statement.
- Simple approach: Just go and find some x where the property is true.
  - In our case, we need to find a positive natural number n such that that sum of all smaller natural numbers is equal to n.
  - Can we find one?

An Entirely Different Proof

Theorem: There exists a natural number $n > 0$ such that the sum of all natural numbers less than n is equal to n.

Proof: Take $n = 3$.

The three natural numbers smaller than three are 0, 1, and 2.

Notice that $0 + 1 + 2 = 3$.

Therefore, three is a natural number greater than zero equal to the sum of all smaller natural numbers. ■

Time-Out for Announcements!

Piazza

- We now have a Piazza site for CS103.
- Sign in to www.piazza.com and search for the course CS103 to sign in.
- Feel free to ask us questions!
- ***Use the site to find partners for the problem sets!***
- You can also email the staff list with questions: cs103-spr1415-staff@lists.stanford.edu .

DiversityBase
Coding for All

Valerie Taylor on Diversity in CS

Lecture by Professor Taylor
Thursday, April 2
Hewlett 201 @ 4pm

Reception and Group Discussion
Gates 211 and 219 @ 5pm

Valerie Taylor is the Senior Associate Dean for Academic Affairs and Royce E. Wisenbaker Professor at the Texas A&M School of Computer Science and Engineering. She has worked tirelessly to promote diversity in CS, and is frequently a speaker at the Richard Tapia and Grace Hopper Conferences.

Join DiversityBase and the CS Department to hear her take on what we can do to increase diversity in computing fields. Afterwards, at we will have a reception where we can discuss actions we can take at Stanford (over delicious food, of course!)

CS + Social Good Board

- Interested in making a difference at the intersection of technology and social good? Apply for the CS+SG Board!
- Application online at

  https://docs.google.com/a/stanford.edu/forms/d/1HrceGe5uE5AgeXiZqIiyZGgcVqvik7pjztrjCwYTPfE/viewform?c=0&w=1
- Applications due on April 10.

Back to CS103!

Extended Example: **XOR**

Logical Operators

- *A **bit** is a value that is either 0 or 1.*
- The set $\mathbb{B} = \{0, 1\}$ is the set of all bits.
- A ***logical operator*** is an operator that takes in some number of bits and produces a new bit as output.
- Example: the ***logical not*** operator, denoted $\neg x$, flips 0s to 1s and vice-versa:

$$\neg 0 = 1 \qquad\qquad \neg 1 = 0$$

Logical XOR

- The ***exclusive OR*** operator (called ***XOR*** for short) operates on two bits and produces 0 if the bits are the same and 1 if they are different.
    - Since XOR operates on two values, it is called a ***binary operator***.
- We denote the XOR of a and b by $\boldsymbol{a \oplus b}$.
- Formally, XOR is defined as follows:

$$0 \oplus 0 = 0 \qquad 0 \oplus 1 = 1$$

$$1 \oplus 0 = 1 \qquad 1 \oplus 1 = 0$$

Fun with XOR

- The XOR operator has numerous uses throughout computer science.
  - Applications in cryptography, data structures, error-correcting codes, networking, machine learning, etc.
- XOR is useful because of four key properties:
  - XOR has an ***identity element***.
  - XOR is ***self-inverting***.
  - XOR is ***associative***.
  - XOR is ***commutative***.

Identity Elements

- An ***identity element*** for a binary operator ★ is some value z such that for any a:

$$a \star z = z \star a = a$$

Identity Elements

An ***identity element*** for a binary operator ★ is some value z such that for any a:

$$a \star z = z \star a = a$$

In math-speak, the term "***for any a***" is synonymous with "for every a" or "***for every possibly choice of a.***" It does not mean "***for some specific choice of a.***"

Identity Elements

- An ***identity element*** for a binary operator $\star$ is some value z such that for any a:

$$a \star z = z \star a = a$$

- Example: 0 is an identity element for +:

$$a + 0 = 0 + a = a$$

- Example: 1 is an identity element for ×:

$$a \times 1 = 1 \times a = a$$

Theorem: 0 is an identity element for $\oplus$.

Proof: We will prove that for any $b \in \mathbb{B}$ that $b \oplus 0 = b$ and that $0 \oplus b = b$. To do this, consider an arbitrary $b \in \mathbb{B}$. We consider two cases:

Case 1: $b = 0$.

Case 2: $b = 1$.

This is called a proof by cases (alternatively, a proof by exhaustion) and works by showing that the theorem is true regardless of what specific outcome arises.

Theorem: 0 is an identity element for ⊕.

Proof: We will prove that for any $b \in \mathbb{B}$ that $b \oplus 0 = b$ and that $0 \oplus b = b$. To do this, consider an arbitrary $b \in \mathbb{B}$. We consider two cases:

Case 1: $b = 0$. Then we have

$b \oplus 0 = 0 \oplus 0$ $\qquad$ $0 \oplus b = 0 \oplus 0$

In a proof by cases, after demonstrating each case, you should summarize the cases afterwards to make your point clearer.

Case 2

$b \oplus$ $\qquad$ 1

$= b$ $\qquad$ $= b$

In both cases, we find $b \oplus 0 = 0 \oplus b = b$.

Theorem: 0 is an identity element for ⊕.

Proof: We will prove that for any $b \in \mathbb{B}$ that $b \oplus 0 = b$ and that $0 \oplus b = b$. To do this, consider an arbitrary $b \in \mathbb{B}$. We consider two cases:

Case 1: $b = 0$. Then we have

$$\begin{aligned} b \oplus 0 &= 0 \oplus 0 \\ &= 0 \\ &= b \end{aligned} \qquad\qquad \begin{aligned} 0 \oplus b &= 0 \oplus 0 \\ &= 0 \\ &= b \end{aligned}$$

Case 2: $b = 1$. Then we have

$$\begin{aligned} b \oplus 0 &= 1 \oplus 0 \\ &= 1 \\ &= b \end{aligned} \qquad\qquad \begin{aligned} 0 \oplus b &= 0 \oplus 1 \\ &= 1 \\ &= b \end{aligned}$$

In both cases, we find $b \oplus 0 = 0 \oplus b = b$. Thus 0 is an identity element for ⊕. ■

Self-Inverting Operators

- A binary operator ★ with identity element z is called ***self-inverting*** when for any a, we have

$$a \star a = z$$

- Is + self-inverting?
- Is – self-inverting?
  - Tricky tricky: minus doesn't have an identity element, so it can't be self-inverting.

XOR is Self-Inverting

Theorem: $\oplus$ is self-inverting.

Proof: Since $\oplus$ has identity element 0, we will prove for any $b \in \mathbb{B}$ that $b \oplus b = 0$. To do this, consider any $b \in \mathbb{B}$. We consider two cases:

Case 1: $b = 0$. Then $b \oplus b = 0 \oplus 0 = 0$.

Case 2: $b = 1$. Then $b \oplus b = 1 \oplus 1 = 0$.

In both cases we have $b \oplus b = 0$, so $\oplus$ is self-inverting. ■

Associative Operators

- A binary operator ★ is called ***associative*** when for any *a*, *b* and *c*, we have

$$a \bigstar (b \bigstar c) = (a \bigstar b) \bigstar c$$

- Is + associative?
- Is – associative?
- Is × associative?

Theorem: ⊕ is associative.

Proof: Consider any a, b, $c \in \mathbb{B}$. We will prove that $a \oplus (b \oplus c) = (a \oplus b) \oplus c$. To do this, we consider two cases:

Case 1: $c = 0$. Then we have that

$$\begin{aligned} a \oplus (b \oplus c) &= a \oplus (b \oplus 0) \\ &= a \oplus b && \textit{(since 0 is an identity)} \\ &= (a \oplus b) \oplus 0 && \textit{(since 0 is an identity)} \\ &= (a \oplus b) \oplus c \end{aligned}$$

Case 2: $c = 1$. Then we have that

$$\begin{aligned} a \oplus (b \oplus c) &= a \oplus (b \oplus 1) \\ &= \mathbf{?} \end{aligned}$$

When You Get Stuck

- When writing proofs, you are bound to get stuck at some point.
- When this happens, it can mean multiple things:
  - What you're proving is incorrect.
  - You are on the wrong track.
  - You're on the right track, but you need to prove an additional result to get to your goal.
- Unfortunately, there is no general way to determine which case you are in.
- You'll build this intuition through experience.

Where We're Stuck

- Right now, we have the expression

  $$a \oplus (b \oplus 1)$$

  and we don't know how to simplify it.
- Let's focus on the $(b \oplus 1)$ part and see what we find:
  - $\mathbf{0} \oplus 1 = \mathbf{1}$
  - $\mathbf{1} \oplus 1 = \mathbf{0}$
- It seems like $b \oplus 1 = \neg b$. Could we prove it?

Relations Between Proofs

- Proofs often build off of one another: large results are almost often accomplished by building off of previous work.
    - Like writing a large program – split the work into smaller methods, across different classes, etc. instead of putting the whole thing into **`main`**.
- A result that is proven specifically as a stepping stone toward a larger result is called a ***lemma***.
- Our result that $b \oplus 1 = \neg b$ serves as a lemma in our larger proof that $\oplus$ is associative.

Lemma 1: For any $b \in \mathbb{B}$, we have $b \oplus 1 = \neg b$.
Proof: Consider any $b \in \mathbb{B}$. We consider two cases:

Case 1: $b = 0$. Then

$$\begin{aligned} b \oplus 1 &= 0 \oplus 1 \\ &= 1 \\ &= \neg 0 \\ &= \neg b. \end{aligned}$$

Case 2: $b = 1$. Then

$$\begin{aligned} b \oplus 1 &= 1 \oplus 1 \\ &= 0 \\ &= \neg 1 \\ &= \neg b. \end{aligned}$$

In both cases, we find that $b \oplus 1 = \neg b$, which is what we needed to show. ■

Theorem: ⊕ is associative.

Proof: Consider any a, b, $c \in \mathbb{B}$. We will prove that $a \oplus (b \oplus c) = (a \oplus b) \oplus c$. To do this, we consider two cases:

Case 1: $c = 0$. Then we have that

$$\begin{aligned} a \oplus (b \oplus c) &= a \oplus (b \oplus 0) && \\ &= a \oplus b && \textit{(since 0 is an identity)} \\ &= (a \oplus b) \oplus 0 && \textit{(since 0 is an identity)} \\ &= (a \oplus b) \oplus c && \end{aligned}$$

Case 2: $c = 1$. Then we have that

$$\begin{aligned} a \oplus (b \oplus c) &= a \oplus (b \oplus 1) && \\ &= a \oplus \neg b && \textit{(by lemma 1)} \\ &= \textbf{??} && \end{aligned}$$

Lemma 2: For any $a, b \in \mathbb{B}$, we have $a \oplus \neg b = \neg(a \oplus b)$.
Proof: Consider any $a, b \in \mathbb{B}$. We consider two cases:

Case 1: $b = 0$. Then

$$\begin{aligned} a \oplus \neg b &= a \oplus \neg 0 \\ &= a \oplus 1 \\ &= \neg a && \textit{(using lemma 1)} \\ &= \neg(a \oplus 0) && \textit{(since 0 is an identity)} \\ &= \neg(a \oplus b) \end{aligned}$$

Case 2: $b = 1$. Then

$$\begin{aligned} a \oplus \neg b &= a \oplus \neg 1 \\ &= a \oplus 0 \\ &= a && \textit{(since 0 is an identity)} \\ &= \neg(\neg a) \\ &= \neg(a \oplus 1) && \textit{(using lemma 1)} \\ &= \neg(a \oplus b) \end{aligned}$$

In both cases, we find that $a \oplus \neg b = \neg(a \oplus b)$, as required. ■

Theorem: $\oplus$ is associative.

Proof: Consider any $a, b, c \in \mathbb{B}$. We will prove that $a \oplus (b \oplus c) = (a \oplus b) \oplus c$. We consider two cases:

Case 1: $c = 0$. Then we have that

$$\begin{aligned} a \oplus (b \oplus c) &= a \oplus (b \oplus 0) \\ &= a \oplus b && \textit{(since 0 is an identity)} \\ &= (a \oplus b) \oplus 0 && \textit{(since 0 is an identity)} \\ &= (a \oplus b) \oplus c \end{aligned}$$

Case 2: $c = 1$. Then we have that

$$\begin{aligned} a \oplus (b \oplus c) &= a \oplus (b \oplus 1) \\ &= a \oplus \neg b && \textit{(using lemma 1)} \\ &= \neg(a \oplus b) && \textit{(using lemma 2)} \\ &= (a \oplus b) \oplus 1 && \textit{(using lemma 1)} \\ &= (a \oplus b) \oplus c \end{aligned}$$

In both cases we have $a \oplus (b \oplus c) = (a \oplus b) \oplus c$, and therefore $\oplus$ is associative. ■

Commutative Operators

- A binary operator ★ is called ***commutative*** when the following is always true:

$$a \star b = b \star a$$

- Is + commutative?
- Is – commutative?

Theorem: $\oplus$ is commutative.

Proof: Consider any $a, b \in \mathbb{B}$. We will prove $a \oplus b = b \oplus a$. To do this, let $x = a \oplus b$. Then

$$\begin{aligned}
&x = a \oplus b \\
&x \oplus b = (a \oplus b) \oplus b \\
&x \oplus b = a \oplus (b \oplus b) && \textit{(since } \oplus \textit{ is associative)} \\
&x \oplus b = a \oplus 0 && \textit{(since } \oplus \textit{ is self-inverting)} \\
&x \oplus b = a && \textit{(since 0 is an identity of } \oplus\textit{)} \\
&x \oplus (x \oplus b) = x \oplus a \\
&(x \oplus x) \oplus b = x \oplus a && \textit{(since } \oplus \textit{ is associative)} \\
&0 \oplus b = x \oplus a && \textit{(since } \oplus \textit{ is self-inverting)} \\
&b = x \oplus a && \textit{(since 0 is an identity of } \oplus\textit{)} \\
&b \oplus a = (x \oplus a) \oplus a \\
&b \oplus a = x \oplus (a \oplus a) && \textit{(since } \oplus \textit{ is associative)} \\
&b \oplus a = x \oplus 0 && \textit{(since } \oplus \textit{ is self-inverting)} \\
&b \oplus a = x && \textit{(since 0 is an identity of } \oplus\textit{)}
\end{aligned}$$

This means that $a \oplus b = x = b \oplus a$. Therefore, $\oplus$ is commutative. ■

Theorem: ⊕ is commutative.

Proof: Consider any $a, b \in \mathbb{B}$. We will prove $a \oplus b = b \oplus a$. To do this, let $x = a \oplus b$. Then

$$
\begin{aligned}
x &= a \oplus b \\
x \oplus b &= (a \oplus b) \oplus b \\
x \oplus b &= a \oplus (b \oplus b) \\
x \oplus b &= a \oplus 0 \\
x \oplus b &= a \\
x \oplus (x \oplus b) &= x \oplus a \\
(x \oplus x) \oplus b &= x \oplus a \\
0 \oplus b &= x \oplus a \\
b &= x \oplus a \\
b \oplus a &= (x \oplus a) \oplus a \\
b \oplus a &= x \oplus (a \oplus a) \\
b \oplus a &= x \oplus 0 \\
b \oplus a &= x
\end{aligned}
$$

This means that $a \oplus b = x = b \oplus a$. Therefore, ⊕ is commutative. ■

The only properties of ⊕ that we used here are that it is associative, has an identity, and is self-inverting. This same proof works for any operator with these three properties!

Binary operators that have this property give rise to **boolean groups** (but you don't need to know that for this class).

Application: ***Encryption***

Bitstrings

- A ***bitstring*** is a finite sequence of zero or more 0s and 1s.
- Internally, computers represent all data as bitstrings.
  - For details on how, take CS107 or CS143.

Bitstrings and ⊕

- We can generalize the ⊕ operator from working on individual bits to working on bitstrings.
- If A and B are bitstrings of length n, then we'll define $A \oplus B$ to be the bitstring of length n formed by applying ⊕ to the corresponding bits of A and B.
- For example:

$$\begin{array}{r} \mathtt{110110} \\ \oplus\ \mathtt{011010} \\ \hline \mathtt{101100} \end{array}$$

Encryption

- Suppose that you want to send me a secret bitstring M of length n.
- You should be able to read the message, but anyone who intercepts the secret message should not be able to read it.
- How might we accomplish this?

$\oplus$ and Encryption

- In advance, you and I share a randomly-chosen bitstring K of length n (called the ***key***) and keep it secret.
- To send me message M secretly, you send me the string $C = M \oplus K$.
    - C is called the ***ciphertext***.
- To decrypt the ciphertext C, I compute the string $C \oplus K$. This is

$$\begin{aligned} C \oplus K &= (M \oplus K) \oplus K \\ &= M \oplus (K \oplus K) \\ &= M \end{aligned}$$

An Example

PUPPIES

M	01010000010101010101000001010000010010010100010101010011
K	11011100101110111100010011010101111001101111011111000010
C	10001100111011101001010010000101101011111011001010010001

Œî”…©² ‘

An Example

Œî”…©²‘

C	**10001100111011101001010010000101101011111011001010010001**
K	**11011100101110111100010011010101111001101111011111000010**
M	**01010000010101010101000001010000010010010100010101010011**

PUPPIES

An Example

Œî”…©²‘

C	**10001100111011101001010010000101101011111011001010010001**
K?	**11000000101000011101100011000011111011101111101111011101**
M?	**01001100010011110100110001000110010000010100100101001100**

LOLFAIL

Some Caveats

- This scheme is insecure if you encrypt multiple messages using the same key.
  - Good exercise: Figure out why this is!
- This scheme guarantees security if the key is random, but it isn't tamperproof.
  - Good exercise: Figure out why this is!
- General good advice: ***never implement your own cryptography!***
- Take CS255 for more details!

Next Time

- **Indirect Proofs**
  - Proof by contradiction.
  - Proof by contrapositive.