

Contents

| | | |
|----------|---------------------------------------|-----------|
| 1 | Proof Techniques | 1 |
| 1.1 | Direct Proof | 1 |
| 1.2 | Proof by Cases | 1 |
| 1.3 | Counterexample | 1 |
| 1.4 | Contradiction | 1 |
| 1.5 | Proving by Contrapositive | 2 |
| 2 | Logic | 2 |
| 2.1 | Preliminary Definitions | 2 |
| 2.2 | Implication | 3 |
| 2.3 | Normal Forms | 3 |
| 2.4 | First-Order Logic | 4 |
| 3 | Sets and Notation | 7 |
| 3.1 | Membership vs. Subsets | 7 |
| 3.2 | Set Operations | 7 |
| 3.3 | Power Sets | 7 |
| 3.4 | Product | 7 |
| 3.5 | Proof by Set-Element Method | 7 |
| 3.6 | Set Algebra | 8 |
| 4 | Relations | 10 |
| 4.1 | Cartesian Products | 10 |
| 4.2 | Reflexivity | 10 |
| 4.3 | Symmetry and Transitivity | 10 |
| 4.4 | Equivalence Relation | 10 |
| 4.5 | Equivalence Classes | 11 |
| 5 | Functions | 12 |
| 5.1 | Formal Definition | 12 |
| 5.2 | Injectivity | 12 |
| 5.3 | Surjectivity | 12 |
| 5.4 | Bijectivity | 13 |

1 Proof Techniques

1.1 Direct Proof

We directly use our hypotheses to reason that our conclusion is correct.

1.2 Proof by Cases

Proof by exhaustion, also known as proof by cases, is a method of mathematical proof in which the statement to be proved is split into a finite number of cases and each case is solved to show that, for every possible “angle” in the domain of a claim, we can exhaustively show that the claim can be proved.

1.3 Counterexample

Counterexamples help us prove that a certain claim is not true. A counterexample is a tangible example, that fits appropriately within the domain of a problem, that disproves the claim being made. Note that not every negative statement can be shown by counterexample (e.g., statements of the form “there does not exist. . .”).

However, you **cannot** prove a claim by showing one example of it. Counterexamples are used to *disprove*. (Alternatively, used to prove an inequality, as of sets.)

For example, the claim “all CS22 students like dinosaurs” can be disproved by finding a student who does not like dinosaurs. Finding this counterexample, however, will not prove that no 22 students like dinosaurs.

1.4 Contradiction

To prove the *negation* of a statement $\neg p$, we show that it is impossible for p to hold. This is known as *proof by contradiction*. It proceeds as follows:

1. Assume p is true.
2. Given p is true, use a direct proof to obtain a contradiction.
3. Since p being true leads us to a contradiction, p must be false, i.e., $\neg p$ must be true.

Occasionally, we can also use a similar technique to prove a positive (i.e., not negated) statement. *Before using contradiction, see if a direct approach would suffice.*

Here is how we would prove a (positive) proposition p by contradiction:

1. Assume p is not true.

2. Given p is false, use a direct proof to obtain a contradiction.
3. Since p being false leads us to a contradiction, p must be true.

1.5 Proving by Contrapositive

Alternatively, to prove “if p then q ” we suppose that q is false and show that p must be false. This is called a proof by contrapositive. Given a proposition, “if p then q ”, the proposition “if $\neg q$ then $\neg p$ ” is called the **contrapositive** of the first proposition.

Samples of different proof types can be found in the resources section of the [22 website](#).

2 Logic

2.1 Preliminary Definitions

1. A **propositional formula** is a condensed representation of a truth table using logical operators and variables. We call a propositional formula a *proposition* for short.
2. The term **logical expression** is often used synonymously with the word proposition.
3. Two propositions are **logically equivalent** when they represent the same truth table. We can prove propositions are logically equivalent by either comparing their truth tables or using logical rewrite rules. A full list of the rules you can use is on our course website.
4. A **valid proposition** is one that evaluates to true on any choice of inputs; it is true no matter what. It is also sometimes called a tautology. The classic example of a valid proposition is $b \vee \neg b$ (thanks, Shakespeare).
5. A proposition is **satisfiable** if it evaluates to true on *some* choice of inputs; that is, that there is some assignment of the input variables to true and false that makes the proposition true.
6. A proposition is **unsatisfiable** if it is false on any choice of inputs; it is false no matter what. It is also sometimes called a contradiction. The classic example of an unsatisfiable proposition is $p \wedge \neg p$.

Let’s now review the interpretation of each of the following logical operators:

| P | Q | P | $P \wedge Q$ | $P \vee Q$ | $P \oplus Q$ | $P \rightarrow Q$ | $P \leftrightarrow Q$ |
|-----|-----|-----|--------------|------------|--------------|-------------------|-----------------------|
| T | T | F | T | T | F | T | T |
| T | F | F | F | T | T | F | F |
| F | T | T | F | T | T | T | F |
| F | F | T | F | F | F | T | T |

2.2 Implication

In the formula $P \rightarrow Q$, we call P the **hypothesis** and Q the **conclusion**. $P \rightarrow Q$ is logically equivalent to $\neg P \vee Q$. In words, this means that for $P \rightarrow Q$ to be true, Q must be true or P must be false.

This choice can seem a little strange at first. Why is $P \rightarrow Q$ true when P is false? Consider the following statement: "If it is raining, I will bring my umbrella." Here are the events that could possibly occur.

- It rains, and I bring my umbrella. That seems fine. The statement is consistent with the situation.
- It rains, and I don't bring my umbrella. The statement does not fit with the situation.
- It doesn't rain, and I bring my umbrella. This situation doesn't seem to directly conflict with the statement. After all, what if I brought my umbrella to block the sun instead? As a result, we say the statement is still consistent with the situation.
- It doesn't rain, and I don't bring my umbrella. The statement seems consistent with this situation, too.

The only scenario where the statement doesn't fit is the second, which is why $P \rightarrow Q$ is only false when P is true and Q is false.

- $\neg Q \rightarrow \neg P$ is called the **contrapositive** of $P \rightarrow Q$ and is logically equivalent. As a result, we have a useful proof technique: to prove the statement "if p, then q" we can instead prove "if not q, then not p."
- $Q \rightarrow P$ is called the **converse** of $P \rightarrow Q$. It is **not** logically equivalent to $P \rightarrow Q$. If both a statement and its converse are true, then the biconditional $P \leftrightarrow Q$ is true.

2.3 Normal Forms

We say a proposition is in **DNF (disjunctive normal form)** when it is the disjunction (clauses ORed together (\vee)) of conjunctions (literals ANDed together (\wedge)).

We say a proposition is in **CNF (conjunctive normal form)** when it is the conjunction (clauses ANDed together (\wedge)) of disjunctions (literals ORed together (\vee)).

Here's a truth table, and propositions in DNF and CNF that represent it:

| P | Q | R | $?$ |
|-----|-----|-----|-----|
| T | T | T | F |
| T | T | F | T |
| T | F | T | F |
| T | F | F | T |
| F | T | T | F |
| F | T | F | F |
| F | F | T | T |
| F | F | F | T |

DNF: $(P \wedge Q \wedge \neg R) \vee (P \wedge \neg Q \wedge \neg R) \vee (\neg P \wedge \neg Q \wedge R) \vee (\neg P \wedge \neg Q \wedge \neg R)$

CNF: $(\neg P \vee \neg Q \vee \neg R) \wedge (\neg P \vee Q \vee \neg R) \wedge (P \vee \neg Q \vee \neg R) \wedge (P \vee \neg Q \vee R)$

If we have an arbitrary truth table, here are two ways we can think about describing it:

- Listing the true rows.
- Listing the false rows.

Since every row must be either true or false, both of these ways will uniquely describe our truth table.

These two ways correspond to DNF and CNF, respectively. To write a proposition in DNF, we can think about it like this: we find all rows where our proposition should evaluate to true, and we say that we must be in one of those rows. On the other hand, to write a proposition in CNF, we find all rows where our proposition should evaluate to false, and say we are not in any of those rows.

For DNF, we \wedge the true variables and negations of the false variables (to be in the row, the inputs must exactly correspond to the row). For CNF, we \vee the false variables and the negations of the true variables (to not be in the row, we just need at least one variable to be different).

In this way, we can represent any truth table in DNF or CNF. We can also rewrite any logical expression to be in DNF or CNF.

2.4 First-Order Logic

In propositional logic, we only consider “atomic” propositions, represented by propositional variables like p and q . First-order logic is more expressive: it allows us to write propositions *about* particular data (like numbers).

In particular, first-order logic lets us write expressions that make assertions about particular entities. Such expressions are called **predicates**; you can think of these a bit like functions from the data in question to the values “true” and “false.” For instance, we might define a predicate $\text{Odd}(n)$ that holds of a natural number $n : \mathbb{N}$ if and only if n is odd. Predicates are syntactically represented by *predicate variables* (like Odd), with the value of which they are being asserted written in parentheses after the predicate variable.

A given predicate can only make assertions about a certain *kind* of object: for instance, it wouldn't really make sense to apply the predicate Odd above to an irrational number like $\sqrt{2}$. We therefore define for each predicate a **domain**, the collection of all the possible values of which the predicate can be asserted. (The domain of Odd would be \mathbb{N} .)

Given some predicate, we may wish to make claims about whether it holds of *any*, or of *all*, elements in its domain. **Quantifiers** allow us to express such claims in first-order logic. There are two quantifiers of note:

1. Universal quantifier: denoted by the \forall symbol, it represents that a predicate holds for *every* element in its domain.
2. Existential quantifier: denoted by the \exists symbol, it represents that a predicate holds for *some* element in its domain.

For instance, the formula $\forall n : \mathbb{N}, \text{Odd}(n)$ asserts that every natural number is odd (this is false!). On the other hand, $\exists n : \mathbb{N}, \text{Odd}(n)$ asserts that at least one odd natural number exists (this is true!).

Note that a universal quantification over an empty domain is always true, while an existential quantification over an empty domain is always false.

We can also chain quantifiers in sequence to represent a more complex proposition.

Example

Problem: Render *Goldbach's conjecture*, that every integer greater than 2 is the sum of two primes, in first-order logic.^a

We first can reformulate this in English in a way that better matches our first-order syntax: "For every even integer n greater than 2, there exist primes p and q such that $n = p + q$ ".
Note: the TAs find this to be especially helpful!

We can then define some predicates. Let G be the predicate on natural numbers defined by $G(n) := n \geq 2$; that is, $G(n)$ is true just in case $n \geq 2$. Let P be the predicate on natural numbers such that $P(n)$ holds just in case n is prime.

We can thus describe the conjecture in first-order logic as follows:

$$\forall n : \mathbb{N}, G(n) \rightarrow \exists p, q : \mathbb{N}, P(p) \wedge P(q) \wedge n = p + q$$

Note that the order of quantifiers is essential. If we switched the order of the quantifiers, we would essentially assert that there are two prime numbers whose sum is equal to every number greater than 2. (This is clearly false!)

Note also the different ways we "restrict" universal and existential quantifiers (so that we are only considering n satisfying G and so that the witnesses p and q must have property P). If we switched the \rightarrow and \wedge symbols in the above, our formula would be incorrect!

(Think about why.)

^a*Mathematics for Computer Science* Eric Lehman. 3.6.

3 Sets and Notation

A set is a collection of objects without order or repetition.

3.1 Membership vs. Subsets

If an object s is a member of a set S , we say $s \in S$. If a set T is a subset of a set S , we write $T \subseteq S$. This means that every member of T is also a member of S .

3.2 Set Operations

- The union $A \cup B$ of two sets A and B is the set of all elements that are in A or B .
- The intersection $A \cap B$ of two sets A and B is the set of all elements that are in A and B .
- The set difference $B \setminus A$ of two sets A and B is the set of all elements that are in B , but that are not in A .
- The complement \bar{A} of a set A is the set of all elements that are *not* in A (where “all elements” refers to all elements in some universal set U .)
- The cardinality $|A|$ of a set A is the number of elements of A . Remember that sets have no duplicates!

3.3 Power Sets

The *power set* of a set S , denoted $\mathcal{P}(S)$, is the set of all subsets of S . The power set of S has cardinality $2^{|S|}$. We proved this last result by noticing that there are the same number of subsets of a set of size n as there are binary strings of length n (see the sample [bijective proof](#) on the website).

3.4 Product

The product of two sets A and B , denoted $A \times B$, is the set of all ordered pairs (a, b) for $a \in A$, $b \in B$. The product of a single set, A , is the set of all ordered pairs (a, a) where $a \in A$.

3.5 Proof by Set-Element Method

How do you prove that some set A equals some set B ? First show that $A \subseteq B$ and then you show that $B \subseteq A$. If every element in A is also an element in B and every element in B is also an element of A , then A must equal B .

To show that $A \subseteq B$ you consider an arbitrary element in A and show it is also in B . In use, this looks like the following:

1. Let x be an element of set A .
2. Prove that x is also an element of B .
3. Conclude that $A \subseteq B$.

Example

Claim: $A \cap (A \cup B) = A$.

Proof. We show that both $A \cap (A \cup B) \subseteq A$ and $A \subseteq A \cap (A \cup B)$.

We first prove the subclaim that $A \cap (A \cup B) \subseteq A$.

Consider any $x \in A \cap (A \cup B)$. By definition of intersection, this means that $x \in A$ and $x \in A \cup B$. Because every arbitrary x in $x \in A \cap (A \cup B)$ is in A , we can conclude that $A \cap (A \cup B) \subseteq A$.

We then prove the subclaim that $A \subseteq A \cap (A \cup B)$.

Consider any $x \in A$. By definition of union, we can reason that $x \in A \cup B$. Because we know that $x \in A \wedge x \in (A \cup B)$, by definition of intersection, we conclude $x \in A \cap (A \cup B)$. It follows that because every arbitrary x in A is in $A \cap (A \cup B)$, $A \subseteq A \cap (A \cup B)$.

Therefore, by the set element method we have proved that $A \cap (A \cup B) = A$. □

3.6 Set Algebra

1. Conversion of one side of the equation to the other (or conversion of both sides to an identical expression) using stated laws of set algebra. (See list of [set identities](#) on course website!)
2. Conclusion based on the biconditionality of the steps taken.

Note: Do not assume equality before applying set identities! Either rewrite one side to look like the other or rewrite both sides separately to look like the same expression.

Example

Claim: $(A \cap B) \cup (A \setminus B) = A \cap (B \cup (A \setminus B))$.

Proof:

$$\begin{aligned} & (A \cap B) \cup (A \setminus B) \\ &= (A \cap B) \cup (A \cap \overline{B}) && \text{(Set Difference Law)} \\ &= A \cap (B \cup \overline{B}) && \text{(Distributive Law)} \\ &= A \cap U && \text{(Complement Law)} \\ &= A && \text{(Identity Law)} \\ &= A \cap (A \cup B) && \text{(Absorption)} \\ &= A \cap (B \cup A) && \text{(Commutativity)} \\ &= A \cap ((B \cup A) \cap U) && \text{(Identity Law)} \\ &= A \cap ((B \cup A) \cap (B \cup \overline{B})) && \text{(Complement Law)} \\ &= A \cap (B \cup (A \cap \overline{B})) && \text{(Distributive Law)} \\ &= A \cap (B \cup (A \setminus B)) && \text{(Set Difference Law)} \end{aligned}$$

□

4 Relations

4.1 Cartesian Products

A *binary* (or *two-place*) *relation* R consists of a set A , called the *domain*; a set B , called the *codomain*; and a subset of the Cartesian product $A \times B$ called the *graph*. If we say that a relation R is *on* a set A , we mean that both its domain and codomain are A .

Always remember to specify the set(s) on which the relation is defined!

We write aRb or $(a, b) \in R$ to mean that a is related to b by R .

4.2 Reflexivity

A relation R on A is *reflexive* if for all $a \in A$, $(a, a) \in R$. In other words, a relation is reflexive if *every element* in the set A is related to itself in R . This is why it's important to specify a set when talking about a relation: you can't tell if a relation is reflexive if you don't know which elements have to be related to themselves (and every element must be!).

4.3 Symmetry and Transitivity

A relation R is *symmetric* if for all a, b in its domain, the following holds: **if** $(a, b) \in R$, **then** $(b, a) \in R$.

A relation R is *transitive* if for all a, b, c in its domain, the following holds: **if** $(a, b) \in R$ and $(b, c) \in R$, **then** $(a, c) \in R$. Remember that a , b , and c do not need to be different elements.

It's important to note that the definitions of symmetry and transitivity are phrased as if-then statements. A relation is symmetric/transitive *unless* it violates the appropriate if-then condition. To violate the condition, you must simultaneously satisfy the if-clause, and violate the then-clause.

Consider the following example of a relation that is not transitive: the ordered pairs $(1, 2)$ and $(2, 1)$ are in the relation (this satisfies the if-clause of the transitivity definition) but there is no pair $(1, 1)$ in the relation (this violates the then-clause).

As another illustrative example: any empty relation is reflexive, symmetric, and transitive, as there are no ordered pairs in the empty relation to satisfy the if-clause of the definitions.

4.4 Equivalence Relation

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

4.5 Equivalence Classes

Let R be an equivalence relation on A . Then the *equivalence class* of $a \in A$ is defined as

$$[a]_R := \{x \mid x \in A, (x, a) \in R\}.$$

Note that a is not unique (unless it is the only element in its equivalence class.) Rather, any element in the same equivalent class can serve equally well as the representative for the class. An equivalence relation splits a set into equivalence classes. In other words, it forms partitions.

A *partition* of a set A is a collection of nonempty subsets B_1, \dots, B_k of A such that

1. $B_1 \cup \dots \cup B_k = A$, and
2. $B_i \cap B_j = \emptyset \quad \forall i, j$ where $i \neq j$.

5 Functions

5.1 Formal Definition

A *function* $f : A \rightarrow B$ is a relation on A and B with the following property: for every $a \in A$ there exists exactly one pair (a, b) in the relation, where $b \in B$.

We call A the domain and B the codomain.

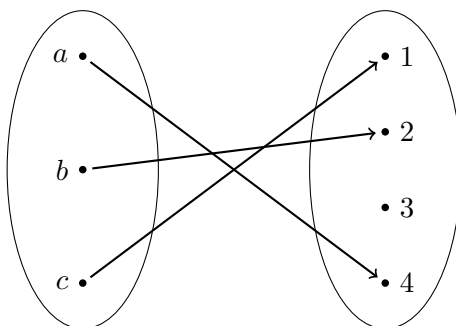
It's important to note that a function is characterized not only by the “rule” that maps inputs to outputs, but also by the domain and codomain.

Additionally, we call the set of all $b \in B$ such that there exists $a \in A$ where $f(a) = b$ the *image* of f . In other words, the image is the set of all elements mapped to by f .

5.2 Injectivity

A function is injective if for all $b \in B$, there exists at most one $a \in A$ such that $f(a) = b$. In other words, no two distinct elements map to the same thing!

If a function $f : A \rightarrow B$ is injective, we know that $|A| \leq |B|$. This is because every element in A needs some unmatched element in B , so B needs to have at least as many elements as A !



There are two ways to prove that a function is injective:

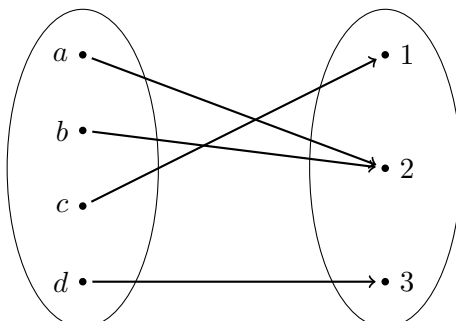
1. Consider two arbitrary elements a and b of the domain, and show that if $f(a) = f(b)$, then we must have $a = b$.
2. Consider two arbitrary distinct elements $a \neq b$ in the domain. Show that they must map to distinct outputs $f(a) \neq f(b)$.

5.3 Surjectivity

A function is surjective if for all $b \in B$, there exists at *least* one $a \in A$ such that $f(a) = b$. In other words, no element in the codomain gets left behind: there is always some element

that maps to it. Equivalently, a function is surjective if the image of the function is the entire codomain.

If a function $f : A \rightarrow B$ is surjective, we know that $|A| \geq |B|$. This is because every element in B needs some element in A to map to it, so A needs to have at least as many elements as B .



To prove that a function is surjective, consider an arbitrary element in the codomain, and construct the specific element in the domain that maps to it.

5.4 Bijection

A bijection is a function that is both injective and surjective. Thus, to prove that a function is a bijection, prove that it is injective and surjective.

If we combine our results from injectivity and surjectivity, we know that the cardinality of the domain must be less than or equal to that of the codomain (by injectivity), and that the cardinality of the domain must be greater than or equal to that of the codomain (by surjectivity.) Thus, the cardinalities of the two sets must be equal. This is a powerful result:

There exists a bijection between two sets if and only if they have equal cardinality.

Thus, to prove that the sizes of two sets are equal, it suffices to prove that there exists a bijection between them.

