Contents

Chapter 1

Logic

1.1 Propositional Logic: The Algebra of Logic

1.1.1 Overview

In this section, we give an introduction to propositional logic, which is the mathematical formalization of logical relationships. We also discuss the disjunctive and conjunctive normal forms, how to convert formulas to each form, and conclude with a fundamental problem in computer science known as the satisfiability problem.

1.1.2 Propositions and Predicates

We must first understand what propositions and predicates are. Here we lay out definitions and examples of both. Note that this is all in preparation for our study of proofs and proof structure, but before we begin in that area we must have a foundation on which to draw.

Propositions

The goal of every mathematical proof is to demonstrate the truth value of a statement known as a proposition, so understanding the notion of propositions is essential to understanding proofs.

Definition 1. *A* proposition *is a statement that is either true or false.*

For example, the statement "$1 + 2 = 3$" is a true proposition, while the statement "$2 + 2 = 5$" is a false proposition. The statement "It will rain tomorrow" is *not* a proposition, because its truth value depends on unknown circumstances. We are often interested in propositions whose truth value is fixed but difficult to determine; we now give some examples.

Example 1: Consider the following proposition: for every positive integer n, the number $n^2 + n + 1$ is prime. How can we determine its truth value? One way is to simply test the statement for different values of n. For $n = 1, 2, 3$, this statement is easily verified, but for $n = 4$, we have $n^2 + n + 1 = 4^2 + 4 + 1 = 21$, which is not prime. Since 4 is a positive integer yet 21 is not prime, we can conclude that the proposition is false.

In the above example, $n = 4$ is a *counterexample* to the proposition. In general, providing a counterexample is one way to prove that any proposition is false. However, this method does not

allow us to show that a true statement is indeed true. Furthermore, finding counterexamples can often be difficult, as we will see.

Example 2: Consider the following proposition: for every positive integer n, the number $n^2 + n + 41$ is prime. Again, we can try to determine its truth value by testing the statement for different values of n. In this case, the statement is true for $n = 1, 2, 3, \ldots, 39$. However, if we examine the case $n = 40$, we notice the following:

$$n^2 + n + 41 = 40^2 + 40 + 41 = 40 \cdot (40 + 1) + 41 = 40 \cdot 41 + 41 = (40 + 1) \cdot 41,$$

which is clearly divisible by 41, and hence, not prime.

Example 3: If x is an even, prime integer, then $x = 2$.

Proof. The following proof technique is termed a *proof by contradiction*. Assume our claim is false. Then there must exist some integer x which is even, prime and strictly greater than two. However, since x is even, $x = 2 * a$ for some integer a. Since $x > 2$, $a \neq 1$. This means that x has factors 2 and a. This contradicts our definition of x as a prime number. Therefore, the assumption we started with, which states that our proposition is incorrect, must be false, and so the proposition is true. □

Example 4: The following proposition is known as *Euler's conjecture*, proposed by the great mathematician Leonhard Euler in 1769: For all integers k and n greater than 1, if there exist $n + 1$ positive integers $a_1, a_2, \ldots, a_n, b$ such that $a_1^k + a_2^k + \cdots + a_n^k = b^k$, then $k \leq n$.

This statement is a little difficult to parse, so let us first consider the case where $k = n = 2$. We can indeed find positive integers a_1, a_2, b such that $a_1^2 + a_2^2 = b^2$; any Pythagorean triple will do. Now if $k = 1$ and $n = 2$, then it is very trivial to find positive integers a_1, a_2, b such that $a_1^1 + a_2^1 = b^1$. The statement is saying that if $k > n$, then we will no longer be able to find $n + 1$ positive integers such that the inequality holds (because if we did, that would mean $k \leq n$).

Euler's conjecture was disproved in 1986 by Noam Elkies, who gave the following counterexample: $n = 3, k = 4, a_1 = 95800, a_2, = 217519, a_3 = 414560, b = 422481$. It is by no means obvious, but it is indeed the case that

$$95800^4 + 217519^4 + 414560^4 = 411481^4,$$

yet here we have $k > n$, so the conjecture is false. This is another example that shows how difficult it can be to find even one counterexample to a proposition; this one took mathematicians over 200 years to find!

Example 5: Euler's conjecture, given in Example 1.1.2, is a generalization of the following proposition proposed by Pierre de Fermat in 1637: for any integer $k \geq 1$, if there exist positive integers a_1, a_2, b such that $a_1^k + a_2^k = b^k$, then $k \geq 2$. This proposition is one of the most notable statements in all of mathematics, and is known as *Fermat's Last Theorem*. Unlike Euler's conjecture, Fermat's Last Theorem is true. The first complete proof was given by Andrew Wiles in 1995, over 300 years after the proposition was proposed, and involved many advanced techniques in mathematics. This example shows that proofs, like counterexamples, can also be very elusive.

Predicates

Predicates are special kinds of propositions that we often consider when writing mathematical proofs. In this section, we will look at examples of predicates that are always true, sometimes true, and never true.

Definition 2. *A* predicate *is a proposition whose truth value depends on the value of certain parameters (also known as arguments or variables).*

Example 1: Suppose $P(n) = n^2 + n + 1$. The following is a predicate: "$P(n)$ is prime." As we can see, the truth value of this predicate depends on the value of n: some values of n (e.g., 1, 2, 3) make the predicate true, while others (e.g., 4) make the predicate false.

Example 2: Suppose $P(n) = n^2 + 2n + 1$ and consider the following predicate: "$P(n)$ is a square number." This predicate is always true, regardless of the value of n, because $n^2 + 2n + 1 = (n + 1)^2$, so $P(n)$ can be written as $(n + 1)^2$, which is clearly a square number.

We now take the opportunity to introduce some notation. Again, suppose $P(n) = n^2 + 2n + 1$ and consider the predicate "$P(n)$ is a square number." Since this statement is true for all values of n, we can write the following proposition:

$$\forall n : P(n) \text{ is a square number.} \tag{1.1}$$

Here, the "$\forall$" symbol represents "for all", and the ":" represents "such that". Statement (1.1) is a proposition, not a predicate, because its truth value is fixed: it is either the case that $n^2 + 2n + 1$ is a square number for every value of n, or it is not the case.

Example 3: Suppose $P(n) = n^2 + n + 1$ and consider the predicate "$P(n)$ is a square number." Unlike the previous two example predicates, this predicate is always false. We can see this by noticing that in order for $P(n)$ to be a perfect square, it must be equal to m^2 for some integer m. Clearly, setting $m = n$ cannot work because $n^2 < n^2 + n + 1$, but setting $m = n + 1$ does not work either because $n^2 + n + 1 < (n + 1)^2$. Since there are no integers bigger than n *and* smaller than $n + 1$, we can conclude that the predicate is always false.

We conclude with some more notation. Suppose $P(n) = n^2 + 7n + 1$ and consider the predicate "$P(n)$ is a square number." We see that for $n = 1$ the predicate is true, but for $n = 2$, the predicate is false. Since the predicate is not true for all values of n, the proposition shown in (1.1) is false. However, since the predicate is true for at least one value of n, the following proposition is true:

$$\exists n : P(n) \text{ is a square number.}$$

Here, the "$\exists$" symbol represents "there exists".

1.1.3 Propositional Logic

We now establish the fundamentals of writing formal proofs by reasoning about logical statements.

Throughout this section, we will maintain a running analogy of propositional logic with the system of arithmetic with which we are already familiar. In general, capital letters (e.g., A, B, C) represent propositional variables, while lowercase letters (e.g., x, y, z) represent arithmetic variables.

But what is a propositional variable? A propositional variable is the basic unit of propositional logic. Recall that in arithmetic, the variable x is a placeholder for any real number, such as 7, 0.5, or -3. In propositional logic, the variable A is a placeholder for any truth value. While x has (infinitely) many possible values, there are only two possible values of A: True (denoted **T**) and False (denoted **F**). Since propositional variables can only take one of two possible values, they are also known as *Boolean* variables, named after their inventor George Boole.

By letting x denote any real number, we can make claims like "$x^2 - 1 = (x+1)(x-1)$" regardless of the numerical value of x. Similarly, as we shall see, letting A, B, C, etc. denote Boolean variables allows us to make claims regarding these variables regardless of their truth values.

Combining Propositions

In arithmetic, the basic operators are addition ($+$) and multiplication ($\times$). Using these operators, we can define new variables such as "$z = x^2 + y$." Similarly, in propositional logic, the basic operators are AND (denoted by $\wedge$), OR ($\vee$), and NOT ($\neg$). This allows us to combine propositions to create new, more complicated propositions.

Furthermore, arithmetic contains shorthand that enables us to write longer formulas more concisely. For example, x^4 is shorthand for $x \times x \times x \times x$. Similarly, in propositional logic, we have notational shorthand, e.g.:

- $A \to B$ is shorthand for $\neg A \vee B$,
- $A \leftrightarrow B$ is shorthand for $(A \to B) \wedge (B \to A)$.

(Note that the "$\neg$" in "$\neg A \vee B$" modifies "A", not "$A \vee B$". In the latter case, we would write "$\neg(A \vee B)$". We will elaborate on this order of operations in the following section.)

In our example above, if we define z as $x^2 + y$, then if $x = 2$ and $y = 1$, we must have $z = 5$ in order to obey the rules of addition and multiplication. Similarly, there are rules that must be obeyed when creating propositions using $\wedge$, $\vee$, and $\neg$. These are given below, in the form of truth tables:

A	B	$A \wedge B$	$A \vee B$
T	**T**	**T**	**T**
T	**F**	**F**	**T**
F	**T**	**F**	**T**
F	**F**	**F**	**F**

A	$\neg A$
T	**F**
F	**T**

Note that the truth tables given above match our understanding of the corresponding words in English: $A \wedge B$ is true if both A and B are true, $A \vee B$ is true if either A or B (or both) is true, and $\neg A$ is simply the opposite of A.

Note: Although the mathematical definitions align with the English meaning in this case, in general, it is not a good idea to rely on intuition when reasoning about logical statements. In fact, the whole point of propositional logic is to remove our dependence on English, which is full of logical ambiguities. For example, what do "Do you want cake or ice cream?" and "We will go golfing unless it rains" mean, exactly?

Now we fill out the truth table for $A \to B$, also known as "A implies B," "If A, then B," and "A is sufficient for B." Recall this expression is defined as $\neg A \vee B$, so if A is **F** then $A \to B$ is **T**, and if

A is **T** then $A \to B$ is the value of B. This yields the following truth table:

A	B	$A \to B$
T	**T**	**T**
T	**F**	**F**
F	**T**	**T**
F	**F**	**T**

Similarly, we can reason about the truth table for $A \leftrightarrow B$, defined as $(A \to B) \wedge (B \to A)$. The expression "$A \leftrightarrow B$" is also known as "$A$ if and only if B", "A iff B," "A is equivalent to B,", and "$A = B$." It is not too difficult to see that if A and B have the same truth value, then $A \leftrightarrow B$ is **T**, and otherwise, $A \leftrightarrow B$ is **F**. Again, this corresponds to the intuitive idea of "equivalence" in English. The truth table is below:

A	B	$A \leftrightarrow B$
T	**T**	**T**
T	**F**	**F**
F	**T**	**F**
F	**F**	**T**

If the above truth table is not apparent, it is a good idea to fill in the details by first writing the truth table for $C = (A \to B)$ and $D = (B \to A)$, and then considering $C \wedge D$. The following section contains more opportunities to practice filling out truth tables.

Axiomatic Rules

In this section, we present rules that allow us to manipulate logical expressions while maintaining logical equivalence. Most, but not all of the rules have analogous counterparts in arithmetic, and we will point these out as they appear. Most rules also have reasonable interpretations in English, but as noted above, we should not rely on semantics in the English language when reasoning about mathematics.

Note: Just like in arithmetic (e.g., $3 + 2 \times 5$ is 13, not 25), logical symbols follow a standard order of operations. In decreasing order of precedence, this is: $\neg, \wedge, \vee, \to, \leftrightarrow$. For clarity, the statement of the rules below contain parentheses, some of which are unnecessary. Also, as noted above, we often use $A = B$ and $A \leftrightarrow B$ interchangeably.

1. Commutativity: The $\wedge$ and $\vee$ operators are commutative, which means

$$A \wedge B = B \wedge A \quad \text{and} \quad A \vee B = B \vee A.$$

   (In arithmetic, we know $2 + 3 = 3 + 2$ and $2 \times 3 = 3 \times 2$.)

2. Associativity: The $\wedge$ and $\vee$ operators are associative, which means

$$\Big((A \wedge B) \wedge C\Big) = \Big(A \wedge (B \wedge C)\Big) \quad \text{and} \quad \Big((A \vee B) \vee C\Big) = \Big(A \vee (B \vee C)\Big).$$

   (In arithmetic, we know $(3 + 4) + 2 = 3 + (4 + 2)$ and $(3 \times 4) \times 2 = 3 \times (4 \times 2)$.) Together, commutativity and associativity essentially mean "order doesn't matter."

3. Idempotence: The $\wedge$ and $\vee$ operators are idempotent, which means
$$A \wedge A = A \text{ and } A \vee A = A.$$
(This rule has no analogy in arithmetic: $3 + 3$ is not 3, and neither is 3×3.)

4. Identity: The truth value **T** is the identity for $\wedge$ and **F** is the identity for $\vee$. In other words,
$$(A \wedge \mathbf{T}) = A, \quad \text{and} (A \vee \mathbf{F}) = A.$$
Moreover, **F** is the "zero" for $\wedge$ and **T** makes $\vee$ valid:
$$(A \wedge \mathbf{F}) = \mathbf{F}, \quad (A \vee \mathbf{T}) = \mathbf{T}.$$
(We know $3 + 0 = 3$, $3 \times 1 = 3$, and $3 \times 0 = 0$. There is no counterpart of the last rule.)

5. Distributivity: The operators $\wedge$ and $\vee$ distribute over each other, which means
$$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C) \quad \text{and} \quad A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C).$$
(In arithmetic, the analogy only applies in one case: $3 \times (2 + 5) = 3 \times 2 + 3 \times 5$, but the statement is not true if we swap the $\times$ and $+$ symbols.)

6. Negations: The $\neg$ operator cancels itself, which means
$$\neg(\neg A) = A.$$
(In arithmetic, we know $-(-5) = 5$.)

7. Tautologies and Contradictions: A tautology is a logical expression that always has truth value **T**, such as $A \vee \neg A$. A contradiction is a logical expression that always has truth value **F**, such as $A \wedge \neg A$. (There is no illustrative analogy in arithmetic.)

8. De Morgan's Laws: Named after Augustus De Morgan, these rules allow us to distribute the $\neg$ operator over $\wedge$ and $\vee$. More specifically, we have
$$\neg(A \vee B) = \neg A \wedge \neg B \quad \text{and} \quad \neg(A \wedge B) = \neg A \vee \neg B.$$
In other words, the $\neg$ gets distributed and the corresponding operator "flips." (There is no illustrative example in arithmetic.)

Again, every one of these equivalences can be formally verified by completing the corresponding truth tables for two equivalent expressions. Each entry of these tables can be derived from the basic principles given in Section 1.1.3.

Example 1: We briefly give an example of simplification of a proposition
$$\begin{aligned}
&(\neg(A \vee B)) \wedge ((\neg C) \vee (\neg(B \wedge A))) \\
&\quad = (\neg A \wedge \neg B) \wedge (\neg C \vee (\neg B \vee \neg A)) \\
&\quad = (\neg A \wedge \neg B) \wedge (\neg C \vee \neg B \vee \neg A) \\
&\quad = \neg A \wedge \neg B \wedge (\neg A \vee \neg B \vee \neg C) \\
&\quad = (\neg A \wedge \neg B \wedge \neg A) \vee (\neg A \wedge \neg B \wedge \neg B) \vee (\neg A \wedge \neg B \wedge \neg C) \\
&\quad = (\neg A \wedge \neg B) \vee (\neg A \wedge \neg B) \vee (\neg A \wedge \neg B \wedge \neg C) \\
&\quad = (\neg A \wedge \neg B) \vee (\neg A \wedge \neg B \wedge \neg C) \\
&\quad \stackrel{?}{=} \neg A \wedge \neg B
\end{aligned}$$

The last statement can be intuitively reasoned. It is left as an exercise to derive the last result using axiomatic rules.

1.1.4 Disjunctive and Conjunctive Normal Forms

The rules given in the previous section (such as De Morgan's) allow us to transform propositional formulas (expressions) to equivalent formulas. In this lecture, we will see two "standard forms" that can represent every propositional formula.

Definition 3. *A propositional formula is in* disjunctive normal form (DNF) *if it is the disjunction of conjunctions of literals.*

On its own, this definition may seem somewhat cryptic, so we explain each term below:

- A *literal* is a Boolean variable, or the negation of a Boolean variable (e.g., P, $\neg P$).
- A *conjunction* is a logical formula that is the AND ($\wedge$) of (smaller) formulas (e.g., $P \wedge \neg Q \wedge R$).
- A *disjunction* is a logical formula that is the OR ($\vee$) of (smaller) formulas (e.g., $P \vee \neg Q \vee R$).

With this in mind, the meaning of DNF should be clearer: a DNF formula is an OR of AND's. For example,

$$X = (A \wedge \neg B \wedge \neg D) \vee (B \wedge C) \vee (C \wedge \neg D \wedge E)$$

is a DNF formula: the literals of X are $A, \neg B, \neg D, B, C, \neg D$, and E; the conjunctions are $A \wedge \neg B \wedge \neg D$, $B \wedge C$, and $C \wedge \neg D \wedge E$; and X is the disjunction (OR) of these three conjunctions. On the other hand,

$$Y = (A \vee \neg B) \wedge (B \vee C \vee \neg D)$$

is not a DNF formula, because the structure of the operators is inverted. But this formula *is* in the other "standard" form (CNF) for propositional formulas.

Definition 4. *A propositional formula is in* conjunctive normal form (CNF) *if it is the conjunction of disjunctions of literals.*

As noted above, Y is a CNF formula because it is an AND of OR's. The literals of Y are $A, \neg B, B, C$, and $\neg D$; the disjunctions are $(A \vee \neg B)$ and $(B \vee C \vee \neg D)$; and Y is the conjunction (AND) of the two disjunctions.

Converting to DNF

We now describe a procedure that converts any propositional formula X into disjunctive normal form. Note that the resulting DNF is not necessarily the shortest expression equivalent to X (or even the shortest DNF equivalent to X), but the result is guaranteed to be a DNF formula.

We illustrate the procedure on the following example:

$$X = \neg\Big((A \vee B) \wedge (A \vee C)\Big).$$

The first step is to write the truth table of X:

A	B	C	X
T	**T**	**T**	**F**
T	**T**	**F**	**F**
T	**F**	**T**	**F**
T	**F**	**F**	**F**
F	**T**	**T**	**F**
F	**T**	**F**	**T**
F	**F**	**T**	**T**
F	**F**	**F**	**T**

(Note that if $A = \mathbf{T}$, then the $\wedge$ "succeeds", so $X = \mathbf{F}$. Otherwise, the $\wedge$ "succeeds" only if $B = C = \mathbf{T}$, and so $X = \mathbf{F}$ in this case and $X = \mathbf{T}$ in all remaining rows.)

The second step is to identify the rows that contain $X = \mathbf{T}$; in our case, these are rows 6, 7, and 8. Finally, we encode each row as a conjunction of the corresponding literals, and the final result is the disjunction of these conjunctions. Intuitively, we can think of the DNF we're constructing as the following expression:

$$(\text{row } 6) \vee (\text{row } 7) \vee (\text{row } 8).$$

In our example, we have the following encoding:

$$\begin{aligned}
&\text{row } 6 : (\neg A \wedge B \wedge \neg C) \\
&\text{row } 7 : (\neg A \wedge \neg B \wedge C) \\
&\text{row } 8 : (\neg A \wedge \neg B \wedge \neg C).
\end{aligned}$$

Taking the disjunction of these conjunctions yields the following DNF formula for X:

$$(\neg A \wedge B \wedge \neg C) \vee (\neg A \wedge \neg B \wedge C) \vee (\neg A \wedge \neg B \wedge \neg C).$$

Converting to CNF

Similarly, we can convert any propositional formula into conjunctive normal form. We will illustrate the process on the same propositional formula X from the previous subsection. In this case, the intuition is the following: $X = \mathbf{T}$ if the truth assignment does not belong to any of the rows that evaluate to **F**.

More specifically, we first write the truth table, as before. We then identify the rows that contain $X = \mathbf{F}$; in our case, these are rows 1, 2, 3, 4, and 5. Finally, we "negate" each row and return the conjunction over these "negated" rows. Intuitively, we can think of the CNF we're constructing as the following expression:

$$(\text{NOT row } 1) \wedge (\text{NOT row } 2) \wedge (\text{NOT row } 3) \wedge (\text{NOT row } 4) \wedge (\text{NOT row } 5).$$

Again, the intuition is the following: if the truth assignment for A, B, C does not correspond to row

1, nor 2, nor 3, nor 4, nor 5, then $X = \mathbf{T}$. Our encoding of the rows is given below:

$$\begin{aligned}
&\text{NOT row 1} : \neg(A \wedge B \wedge C) = \neg A \vee \neg B \vee \neg C \\
&\text{NOT row 2} : \neg(A \wedge B \wedge \neg C) = \neg A \vee \neg B \vee C \\
&\text{NOT row 3} : \neg(A \wedge \neg B \wedge C) = \neg A \vee B \vee \neg C \\
&\text{NOT row 4} : \neg(A \wedge \neg B \wedge \neg C) = \neg A \vee B \vee C \\
&\text{NOT row 5} : \neg(\neg A \wedge B \wedge C) = A \vee \neg B \vee \neg C.
\end{aligned}$$

Notice that we've applied De Morgan's Law and the double negation rule to produce a disjunction. The final CNF is the conjunction of these disjunctions:

$$(\neg A \vee \neg B \vee \neg C) \wedge (\neg A \vee \neg B \vee C) \wedge (\neg A \vee B \vee \neg C) \wedge (\neg A \vee B \vee C) \wedge (A \vee \neg B \vee \neg C).$$

Proving Equivalence of Formulas

A typical problem is the following: given two propositional formulas X and Y, are X and Y equivalent? That is, what is the truth value of the proposition $X \leftrightarrow Y$ (denoted by $X = Y$)?

Roughly speaking, there are three ways to prove that X and Y are equivalent:

1. Use a truth table to check if the columns corresponding to X and Y are identical.
2. Apply a sequence of rules to X until Y appears. (We can also "work backwards" from Y and meet X at some formula "in the middle.")
3. Convert X and Y to DNF (or CNF) using the procedure described above, and check if the resulting formulas are identical. This is essentially a special case of Method 2.

In general, Method 1 is the most straightforward, but also the most tedious. Method 2 often yields a more succinct proof that $X = Y$, but it is sometimes unclear which rules to apply. In this section, we explore Method 3; in particular, we will convert one propositional formula to DNF, and then to CNF. However, instead of writing the truth table and applying the procedures described above, we will apply the rules we learned in the previous lecture.

The propositional formula that we will consider is the following:

$$X = \neg\Big((A \vee \neg B) \wedge (\neg A \vee C)\Big).$$

As promised, we first convert X to DNF:

$$\begin{aligned}
X &= \neg\Big((A \vee \neg B) \wedge (\neg A \vee C)\Big) \\
&= \neg(A \vee \neg B) \vee \neg(\neg A \wedge C) && \text{(De Morgan's for } \wedge) \\
&= (\neg A \wedge \neg\neg B) \vee (\neg\neg A \vee \neg C) && \text{(De Morgan's for } \vee \text{ and } \wedge) \\
&= (\neg A \wedge B) \vee (A \wedge \neg C). && \text{(double negation)}
\end{aligned}$$

Note that this final formula is the OR of AND's, so it is indeed in DNF. Instead of writing the entire truth table of X, we simply applied De Morgan's Laws and the double negation rule a few times.

Now we continue where we left off, and convert X into CNF:

$$\begin{aligned} X &= (\neg A \wedge B) \vee (A \wedge \neg C) \\ &= \Big(\neg A \vee (A \wedge \neg C)\Big) \wedge \Big(B \vee (A \wedge \neg C)\Big) && \text{(distribution of } \vee \text{ over } \wedge) \\ &= \Big((\neg A \vee A) \wedge (\neg A \vee \neg C)\Big) \wedge \Big((B \vee A) \wedge (B \vee \neg C)\Big) && \text{(distribution of } \vee \text{ over } \wedge) \end{aligned}$$

Technically, this final expression is in CNF: it is the AND of four OR's, each of which has two literals. But recall that

$$(\neg P \vee P) = \mathbf{T} \quad \text{and} \quad \mathbf{T} \wedge Q = Q.$$

These two facts allow us to simplify our CNF for X to a shorter CNF:

$$X = (A \vee B) \wedge (\neg A \vee \neg C) \wedge (B \vee \neg C).$$

Note that for stylistic purposes, we have alphabetized the order of variable appearances, which we are allowed to do because $\wedge$ and $\vee$ are both commutative and associative.

The Satisfiability Problem

We now present one of the most well-known problems in computer science: the satisfiability problem, which we denote by SAT. In SAT, we are given a propositional formula X in CNF and must decide whether or not there exists a truth assignment to the variables of X that makes X evaluate to TRUE. If there is such an assignment, we return "Yes," and if there is no satisfying assignment, then we return "No."

For example, let's consider the propositional formula from the previous section:

$$X = (A \vee B) \wedge (\neg A \vee \neg C) \wedge (B \vee \neg C).$$

If we set $A = \mathbf{T}, B = \mathbf{T}, C = \mathbf{F}$, then $X = \mathbf{T}$, which means X is satisfiable, so we return "Yes." Since X only has a total of 6 literals, finding this assignment shouldn't take too long. But in general, how can we solve this problem?

The simplest solution is the following: given X, write the truth table for X, and if at any point we see a $\mathbf{T}$ in the column belonging to X, return "Yes." If every row has $X = \mathbf{F}$, then return "No." This algorithm clearly works, but if X has n variables, then its truth table has 2^n rows. In the worst case, we would check every row, so informally, the worst-case running time is roughly 2^n "steps."

But 2^n can be quite large: if $n = 100$, then 2^n has 31 digits! Computer scientists consider a solution to be *efficient* if its worst-case running time has a polynomial (e.g., $2n, n^2, 5n^3$) number of steps (or less). Since 2^n is not a polynomial, the "truth table" solution is not efficient.

Solving SAT efficiently is one formulation of the "P versus NP" problem, which is perhaps the most well-known unsolved problem in computer science. The first person to give an efficient algorithm for SAT (or prove that no such algorithm exists) will receive $1,000,000 from the Clay Mathematics Institute. An efficient algorithm for SAT would likely have significant consequences: online financial transactions would no longer be secure, and many modern-day security systems would crumble!

Before we conclude this section, let's consider the SAT problem with a slight modification: instead of the input being in CNF, assume we are given X in DNF. Then there's an efficient solution:

for each conjunction of X, assign the variables of the conjunction with their appropriate truth values. If any conjunction evaluates to TRUE, then return "Yes," and otherwise, return "No."

For example, suppose we are given the same propositional formula X in DNF:

$$X = (\neg A \wedge B) \vee (A \wedge \neg C).$$

Looking at the first conjunction, if we simply set $A = \mathbf{F}$ and $B = \mathbf{T}$, then that conjunction is satisfied, so $X = \mathbf{T}$ and we can return "Yes." (If that assignment didn't work, we would try the next conjunction.) Since this algorithm essentially "scans" the input just once, it is efficient, so this version of SAT is easy. However, the original version of SAT (given a CNF) is still difficult, because the conversion process from a CNF to a DNF can require roughly 2^n steps.

$\mathcal{P}$ vs $\mathcal{NP}$

The previous problem, SAT is an example of a "non-deterministic polynomial" problem. The precise definitions for $\mathcal{P}$ versus $\mathcal{NP}$ is beyond the scope of this course. That said, in very loose terms $\mathcal{P}$ is the set of problems which can be explicitly solved in polynomial time. For example, solving the modification of the SAT problem in which we are given a DNF is a problem in P, since we have the described linear time algorithm to explicitly answer whether the DNF is satisfiable. In contrast, the SAT problem on CNFs is known to be in NP. So, while SAT on CNFs cannot be explicitly solved in polynomial time, if an algorithm were given a solution and a certificate (such as a particular configuration of true and false literals), it could determine if the solution is valid in polynomial time. This leads into a discussion on complexity theory that is beyond the scope of this class. In short, and as mentioned in the previous section, if you find an efficient solution to SAT, this would imply an efficient solution to many problems (such as those that form the basis of online security).

1.1.5 Summary

In this section, we studied propositional logic, discussed the basic axiomatic rules for propositional logic, and learned how to verify these rules using truth tables. We also learned how to convert every propositional formula to DNF and CNF, and took a look at the SAT problem.

1.2 First Order Logic: Quantifiers and Predicates, Godel's Incompleteness Theorem

1.2.1 Overview

In this section, we study predicate logic, which builds on propositional logic. We end with a brief discussion of Gödel's incompleteness theorems and their implications.

1.2.2 Predicate Logic and quantifiers

The general idea of predicate logic is similar to propositional logic—there are variables and operators—but now we have predicates.

Definition 5. *A* PREDICATE *is a proposition whose truth value depends on certain parameters.*

For example,

$$E(x) = \text{“}x \text{ is even.”} \quad \text{and} \quad O(x) = \text{“}x \text{ is odd.”}$$

are two examples of predicates. We use quantifiers to interact with predicates. There are two quantifiers that we should know: the universal quantifier (denoted by $\forall$, pronounced “for all”) and the existential quantifier ($\exists$, “there exists”). By using quantifiers with predicates (and other logical symbols), we can create predicate formulas such as the following:

$$P = (\forall x \in \mathbb{Z}^+. E(x) \vee O(x)).$$

(Recall that $\mathbb{Z}^+$ denotes the set of positive integers.) In words, proposition P states that every positive integer is even or odd (or possibly both, although we know that's never the case).

Negating Quantifiers

Since P (defined above) is not only a predicate formula but also a proposition, P can be negated. To do this, we use a version of De Morgan's Laws for quantifiers. These laws are the following:

$$\neg(\forall x. Q(x)) = \exists x. \neg Q(x)$$
$$\neg(\exists x. Q(x)) = \forall x. \neg Q(x).$$

In other words, the $\neg$ is distributed and the quantifier “flips” (similarly to the regular De Morgan's Laws). Now why are these laws true, and why are they a version of De Morgan's Laws?

Imagine that the variable x can only take one of three possible values: a, b, or c. Then notice that

$$\forall x. Q(x) = (Q(a) \wedge Q(b) \wedge Q(c)),$$

which means

$$\begin{aligned} \neg(\forall x. Q(x)) &= \neg(Q(a) \wedge Q(b) \wedge Q(c)) \\ &= \neg Q(a) \vee \neg Q(b) \vee \neg Q(c). && \text{(De Morgan's for } \wedge) \end{aligned}$$

This last expression states that $Q(x)$ is false for at least one of the three possible values of x; in other words, this formula is equivalent to $\exists x, \neg Q(x)$. The other direction is similar:

$$\begin{aligned} \neg(\exists x. Q(x)) &= \neg(Q(a) \vee Q(b) \vee Q(c)) \\ &= \neg Q(a) \wedge \neg Q(b) \wedge \neg Q(c) && \text{(De Morgan's for } \vee) \\ &= \forall x. \neg Q(x). \end{aligned}$$

So the negation of our proposition P is the following:

$$\begin{aligned} \neg P &= \neg(\forall x \in \mathbb{Z}^+. E(x) \vee O(x)) \\ &= (\exists x \in \mathbb{Z}^+. \neg(E(x) \vee O(x))) && \text{(De Morgan's for quantifiers)} \\ &= (\exists x \in \mathbb{Z}^+. \neg E(x) \wedge \neg O(x)). && \text{(De Morgan's for } \vee) \end{aligned}$$

In English, $\neg P$ states that there is a positive integer that is neither even nor odd, which of course is false. This confirms the fact that the proposition P is always TRUE.

Combining Quantifiers

In propositional logic, we can combine multiple $\wedge$'s, $\vee$'s, and $\neg$'s to create complicated propositional formulas. Similarly, we can combine $\forall$'s and $\exists$'s to create complicated predicate formulas. Consider the following statement:

Every even number greater than 2 is the sum of two primes.

This proposition is known as *Goldbach's conjecture*, and despite its simplicity, it is one of the oldest and most well-known unsolved problems in mathematics. Let's reformulate the statement into predicate logic: let E denote the set of even numbers greater than 2, and let P denote the set of prime numbers. Also, define R as the following predicate:

$$R(a, b, c) \leftrightarrow (a = b + c).$$

Then Goldbach's conjecture states the following:

$$\forall x \in E \, \exists p \in P \, \exists q \in P. \, R(x, p, q).$$

Notice that this predicate formula has three quantifiers: one $\forall$ and two $\exists$'s. For notational convenience, we often collect variables under the same quantifier whenever possible, so this predicate formula is equivalent to the following:

$$\forall x \in E \, \exists p, q \in P. \, R(x, p, q).$$

Order of Quantifiers

In Goldbach's conjecture, and in most predicate formulas, the order of quantifiers matters. For example, consider the following alteration of Goldbach's conjecture:

$$\exists p, q \in P \, \forall x \in E. \, R(x, p, q).$$

In English, this proposition says that there are two prime numbers whose sum is equal to every even number greater than two. Of course, this is false because the sum of two numbers can only be a single number.

So swapping the order of quantifiers does not create an equivalent predicate formula, but that's not all. Consider the following predicate formulas:

$$A = (\forall x \, \exists y. \, R(x, y))$$
$$B = (\exists y \, \forall x. \, R(x, y)).$$

Goldbach's conjecture has the same form as A, and we just saw that the corresponding B is false. So even if Goldbach's conjecture is true, it is not the case that A implies B. However, it is the case that B implies A. (In other words, the proposition "$B \to A$" is true.)

To illustrate why this statement is true, let us again imagine that the variable x can only be one of three values, which we denote by x_1, x_2, and x_3. Then B implies there is some value of y, which we denote by y^*, such that

$$R(x_1, y^*) \wedge R(x_2, y^*) \wedge R(x_3, y^*).$$

To prove that A is true, we must show that for any possible value of x, there exists some value of y such that $R(x, y)$ is true. But B implies the existence of y^*, which we can exhibit as the value of y regardless of the value of x we are given. In other words, A is equivalent to

$$\exists y_1, y_2, y_3.\, R(x_1, y_1) \wedge R(x_2, y_2) \wedge R(x_3, y_3),$$

and we prove A by setting $y_1 = y_2 = y_3 = y^*$. On the other hand, if A is true, then the appropriate values y_1, y_2, y_3 exist but may not be the same. Thus, B is not necessarily true, because B demands a single value of y that works for all possible values of x. The takeaway is the following: the order of quantifiers in predicate formulas is important to consider when writing proofs.

1.2.3 Gödel's Incompleteness Theorems

In this section, we informally state Gödel's incompleteness theorems and provide some brief commentary. Although a formal proof is outside the scope of this class, these theorems relate to the foundations of mathematics, so we'd like to have a rough idea of what they mean.

Recall from Lecture 1 that ZFC is a set of axioms that form the basis of modern-day mathematics. But ZFC is only one system; there are other axiomatic systems that mathematicians can use as well. Any reasonable system, however, should satisfy the following properties:

- Soundness: The system should not be able to prove false statements.
- Completeness: The system should be able to prove every true statement.

Notice that satisfying exactly one of the properties is easy: for example, a system that does not allow us to prove anything would not be able to prove false statements.

In the early 20th century, mathematicians sought a system that is both sound and complete. However, in 1931, Kurt Gödel showed that this is impossible. More precisely (but still informally), Gödel proved the following:

If X is a sound logical system, then X is not complete.

To show that X is not complete, Gödel had to state a proposition that any sound system X would not be able to prove. That proposition is the following:

The system X is complete.

In other words, Gödel showed that any sound logical system cannot prove its own completeness, and therefore, cannot be complete. This statement is known as Gödel's *second* incompleteness theorem. Gödel *first* incompleteness theorem came to a similar conclusion but was conditioned on the logical system being related to the arithmetic of numbers. It (informally) states the following:

If a logical system X can perform arithmetic on the positive integers and is sound, then X is incomplete.

1.2.4 Summary

In this section, we extended our study of logic to quantifiers and predicate formulas and saw a brief commentary on Gödel's incompleteness theorems.

Chapter 2

Proofs

2.1 Implications and Equivalences, Basic Proof Techniques

2.1.1 Overview

In this section, we will study mathematical proofs. We will see some basic proof techniques and how to use the techniques to prove statements.

2.1.2 Rules of inference

Modus Ponens: If P is true and $P \to Q$ is true, then Q is true.

Proof.

P	Q	$\neg P \vee Q$
T	**T**	**T**
T	**F**	**F**
F	**T**	**T**
F	**F**	**T**

By the truth table we can see when P and $\neg P \vee Q$ are both true, Q is true as well. □

Chain rule: If $P \to Q$, and $Q \to R$, then $P \to R$.

Proof.

P	Q	R	$(\neg P \vee Q)$	$(\neg Q \vee R)$	$(\neg P \vee R)$
T	**T**	**T**	**T**	**T**	**T**
T	**T**	**F**	**T**	**F**	**F**
T	**F**	**T**	**F**	**T**	**T**
T	**F**	**F**	**F**	**T**	**F**
F	**T**	**T**	**T**	**T**	**T**
F	**T**	**F**	**T**	**F**	**T**
F	**F**	**T**	**T**	**T**	**T**
F	**F**	**F**	**T**	**T**	**T**

We focus on the rows where $(\neg P \vee Q)$ and $(\neg Q \vee R)$ are true, which are the 1st, 5th, 7th, and 8th rows. Clearly in these rows, $(\neg P \vee R)$ are all true. □

Example 1: Prove the following statement: if $0 \leq x \leq 2$, then $-x^3 + 4x + 1 > 0$. Before we write a formal proof of the theorem, let's understand why this statement is true. Notice the following:

$$-x^3 + 4x + 1 = x(4 - x^2) + 1 = x(2 + x)(2 - x) + 1$$

Since $x \geq 0$, we know $x \geq 0$ and $2 + x \geq 0$. Furthermore, since $x \leq 2$, we have $2 - x \geq 0$. Thus, we have $x(2 + x)(2 - x) \geq 0$, so adding one yields a positive number as desired. We now provide a formal argument, using the rules of inference stated above.

1. The expression $0 \leq x \leq 2$ is shorthand for $x \geq 0$ and $x \leq 2$.
2. Thus, x, $2 + x$, and $2 - x$ are all nonnegative numbers.
3. The product of nonnegative numbers is nonnegative. Together with Step 2, we can apply Modus Ponens to conclude that $x(2 + x)(2 - x)$ is nonnegative, i.e., $x(2 + x)(2 - x) \geq 0$.
4. The sum of a nonnegative number and a positive number is positive. Since $x(2 + x)(2 - x)$ is nonnegative and 1 is positive, we can apply Modus Ponens to conclude that $x(2 + x)(2 - x) + 1$ is positive.

Modus Tollens (Contrapositive): If $\neg P \to \neg Q$ is true, then so is $Q \to P$.

Proof. We prove this claim by applying the definition of $\to$. Notice the following:

$$\neg P \to \neg Q = \neg(\neg P) \vee \neg Q = P \vee \neg Q = \neg Q \vee P = Q \to P. \qquad \square$$

Note that $\neg P \to \neg Q$ does *not* imply $P \to Q$.

Example 2: Recall the definition of rational number: a number x is *rational* if $x = \frac{p}{q}$ for some integers p, q. If x is not rational, then x is *irrational*. Prove the following statement: if r is irrational, then $\sqrt{r}$ is also irrational.

Proof. Instead of proving the statement directly, we shall use Modus Tollens. In other words, it suffices to prove the following: if $\sqrt{r}$ is rational, than r is rational. Notice that if $\sqrt{r}$ is rational, then $\sqrt{r} = p/q$ for some integers p, q, which implies $r = p^2/q^2$. Since p^2 and q^2 are also integers, this implies r is rational, as desired.

$\square$

2.1.3 Proof Techniques

Proof by contradiction

Suppose we want to show that a proposition P is true, and we've managed to show $\neg P \to \neg Q$ for some proposition Q that we know is true. By Modus Tollens, this means we've shown that $Q \to P$ is true. So finally, by Modus Ponens, we can conclude that P is true, as desired.

The general framework is the following: assume P is false, show that this implies Q is false for some Q we know to be true, and conclude that P must be true.

Example 3: Let us show that $\sqrt{2}$ is irrational by giving a proof by contradiction.

Proof. Assume the statement is false, that is, $\sqrt{2} = p/q$ for some integers p, q. By simplifying this fraction, we can further assume that p and q have no common divisors (other than 1). By squaring both sides and rearranging, we have $p^2 = 2q^2$, which means p is even. This means $p = 2s$ for some integer s, and substituting back yields $(2s)^2 = 2q^2$, which means $q^2 = 2s^2$. By the same reasoning, this means q is even as well.

Thus, p and q are both even, which mean they share 2 as a divisor. Yet this cannot occur, because we can always write a fraction in lowest terms. Therefore, our assumption that $\sqrt{2}$ is rational is false, so we can conclude that $\sqrt{2}$ is irrational, as desired.

Example 4: Let a, b, n be positive integers such that $a \cdot b = n$. Prove that $\min(a, b) \leq \sqrt{n}$.

Proof. For contradiction, assume $\min(a, b) > \sqrt{n}$. This implies $a > \sqrt{n}$ and $b > \sqrt{n}$, which implies $a \cdot b > \sqrt{n} \cdot \sqrt{n} = n$. However, the conclusion $a \cdot b > n$ contradicts the fact that $a \cdot b = n$. Therefore, $\min(a, b) \leq \sqrt{n}$. □

Example 5: Prove that $\log_4 6$ is irrational.

Proof. For contradiction, assume that $\log_4 6$ is rational, i.e., $\log_4 6 = \frac{a}{b}$, for some integers a, b where $b \neq 0$. This implies $6 = 4^{a/b}$, which implies $6^b = 4^a$. Substituting $6 = 2 \cdot 3$ and $4 = 2^2$ yields $2^b \cdot 3^b = 2^{2a}$. Dividing both sides by 2^b yields $3^b = 2^{2a-b}$. Notice that the only way for this equality to hold is $b = 2a - b = 0$. However, $b = 0$ contradicts the condition that $b \neq 0$. Therefore, $\log_4 6$ must be irrational. □

Disproving propositions

Usually, our goal in this class is to prove that a proposition is true. However, we sometimes want to prove that a statement is false, and to do this, it suffices to show that its negation is true.

Example 6: Disprove the following: if a positive integer is even, then it is not prime. Let $E(x) =$ "x is even" and $P(x) =$ "x is prime." To disprove the original statement, we want to show the following proposition is true:

$$\neg \left(\forall x \in \mathbb{N}.\, E(n) \rightarrow \neg P(n) \right).$$

Now notice the following:

$$\begin{aligned}
\neg \left(\forall x \in \mathbb{N}.\, E(n) \rightarrow \neg P(n) \right) &= \exists n \in \mathbb{N}.\, \neg (E(n) \rightarrow \neg P(n)) && \text{(distribution of } \neg) \\
&= \exists n \in \mathbb{N}.\, \neg (\neg E(n) \vee \neg P(n)) && \text{(definition of } \rightarrow) \\
&= \exists n \in \mathbb{N}.\, (\neg\neg E(n) \wedge \neg\neg P(n)) && \text{(De Morgan's law)} \\
&= \exists n \in \mathbb{N}.\, E(n) \wedge P(n) && \text{(double negation)}
\end{aligned}$$

The last proposition states that there exists a positive integer that is even and prime. This is easily shown to be true: 2 is such a number. Therefore, the negation of this proposition, i.e., the original statement, is false.

2.1.4 Proving equivalences

Let A and B be two propositions. Recall that $A \leftrightarrow B$ is shorthand for $(A \to B) \wedge (B \to A)$. Thus, if the proposition $A \leftrightarrow B$ is true, then we say that A and B are *equivalent*. Another way of saying this is "A if and only if B"; and we sometimes use "iff" as an abbreviation for "if and only if."

One way to prove a statement of the form $A \leftrightarrow B$ is to prove both $A \to B$ and $B \to A$. Another way is to construct a sequence of logically equivalent statements, by proving $A \leftrightarrow C_1$, $C_1 \leftrightarrow C_2$, and so on, until you've shown $C_n \leftrightarrow B$.

Example 7: Let $x_1, x_2, \ldots, x_n$ be real numbers for some integer $n \geq 2$. We define the *mean* as $\mu = \frac{x_1+x_2+\ldots+x_n}{n}$, and the *standard deviation* as the following value:

$$\sigma = \sqrt{\frac{(x_1-\mu)^2 + (x_2-\mu)^2 + \cdots + (x_n-\mu)^2}{n}}$$

(Note that the square of the standard deviation is known as the *variance*.) Prove that $\sigma = 0$ if and only if $x_1 = x_2 = \cdots = x_n = \mu$.

Proof. We shall present two proofs. The first proof begins with the following claim: for any real numbers $a_1, \ldots, a_n$, we have

$$a_1^2 + \cdots + a_n^2 = 0 \longleftrightarrow a_1 = a_2 = \cdots = a_n = 0. \tag{2.1}$$

Proving the "reverse" direction of this claim is simple: for each $i \in \{1, \ldots, n\}$, we have $a_i = 0$, which implies $a_i^2 = 0$, so $\sum_{i=1}^n a_i^2 = 0$. For the opposite direction, notice that a_i^2 can never be negative, so we must have $a_i^2 = 0$ for every $i \in \{1, \ldots, n\}$. This implies $a_i = 0$ for every $i \in \{1, \ldots, n\}$, so we've proven the claim.

Now let $a_i = x_i - \mu$ for every $i \in \{1, \ldots, n\}$. Then the left-hand side of (2.1) is equivalent to $\sigma = 0$, and the right-hand side of (2.1) is equivalent to $x_1 = x_2 = \cdots = x_n = \mu$. Thus, by applying this claim, we have proven the original desired statement.

Another way to prove the statement is through the following sequence of statements, all of which are logically equivalent:

$$\begin{aligned}
&\sigma = \sqrt{\frac{(x_1-\mu)^2 + \cdots + (x_n-\mu)^2}{n}} = 0 \\
&\quad \leftrightarrow (x_1-\mu)^2 + \cdots + (x_n-\mu)^2 = 0 && \text{(by squaring and multiplying by } n\text{)} \\
&\quad \leftrightarrow (x_i-\mu)^2 = 0 \quad \forall i \in \{1, \ldots, n\} && \text{(since the terms are non-negative and sum to 0)} \\
&\quad \leftrightarrow x_i - \mu = 0 \quad \forall i \in \{1, \ldots, n\} \\
&\quad \leftrightarrow x_i = \mu \quad \forall i \in \{1, \ldots, n\}.
\end{aligned}$$

□

2.1.5 Proof by cases

Now we demonstrate how to prove a statement by analyzing the set of possible cases.

Example 8: In any group of 6 people, there are either 3 mutual friends (every pair knows each other) or 3 mutual strangers (no pair knows each other).

Proof. Let's model the situation as follows: the set of people is $\{A, B, C, D, E, F\}$, and for each pair of people, we add a blue edge if they are friends and a red edge if they are strangers. Our goal is to prove that no matter how we color the edges, there will always exist a red triangle or a blue triangle. See Fig. 2.1 for an example.

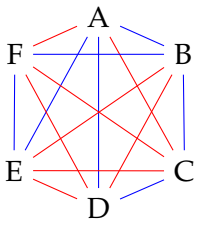


Figure 2.1: An example of the friendship graph for this problem. Notice that the set $\{B, D, E\}$ forms a blue triangle, that is, B, D, and E are mutual friends.

Now we begin the proof by considering the person A. There are two possible cases: A does not have at least three friends (in the set $\{B, C, D, E\}$), or A does have at least three friends.

1. A **does not have at least three friends:** This means A has is incident to at least three red edges; suppose these three strangers are B, C, and D (see Fig. 2.2). If we consider the set

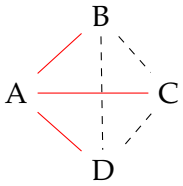
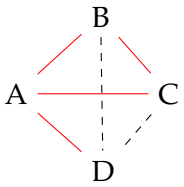
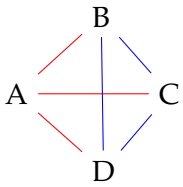


Figure 2.2: Case 1

$\{B, C, D\}$, there are two possible subcases: (a) these three people contain at least one red edge, or (b) they do not. In case (a), that red edge together with the corresponding red edges to A form a red triangle, so we are done. In the second case, $\{B, C, D\}$ forms a blue triangle, so we are also done. (See the correspnding figures below.)



(a) Case 1(a)



(b) Case 1(b)

2. A **has at least three friends:** The argument is essentially identical. If these three friends form the set $\{B, C, D\}$, then any friendship within these three would create a blue triangle, and if B, C, and D are mutual strangers, then we've found a red triangle.

In both cases, we have proven that a red or blue triangle exists, so we are done.

□

Example 9: Prove the following statement: For any two real numbers r and s, $|r+s| \leq |r| + |s|$.

Proof. Considering the positivity/negativity of r and s, we have four cases:

- Case 1: $r \geq 0, s \geq 0$. Then $|r| + |s| = r + s = |r+s|$
- Case 2: $r \geq 0, s \leq 0$. Then $|r| + |s| = r - s$. We do not know if $r - s$ is positive or not, so we have following subcases.

  Case 2(a): $r + s \geq 0$. Then $|r+s| = r + s < r - s = |r| + |s|$.
  Case 2(b): $r + s < 0$. Then $|r+s| = -(s+r) = -s - r < -s + r = |r| + |s|$.

- Case 3: $r \leq 0, s \geq 0$. Then $|r| + |s| = s - r$. Similarly to Case 2, we have following subcases.

  Case 3(a): $r + s \geq 0$. Then $|r+s| = r + s < s - r = |r| + |s|$.
  Case 3(b): $r + s < 0$. Then $|r+s| = -(s+r) = -s - r < s + r = |r| + |s|$.

- Case 4: $r \leq 0, s \leq 0$. Then $|r| + |s| = -r - s = -(r+s) = |r+s|$.

We can see that in all four cases $|r+s| \leq |r| + |s|$, and thus the statement is true. □

Example 10: Recall bubble sort: starting with the first element of an unsorted array, swap the current element with the next element if the former is greater than the latter. Continue to the next element until we've reached the end of the array, and we repeat all of these steps until no adjacent pair is out of order. Let's prove that this algorithm correctly returns a sorted list.

Proof. For contradiction, assume that in the final output array, there exists a pair of numbers a, b such that $a > b$ but a appears before b in the array. There are two cases:

1. a **and** b **are adjacent:** This case cannot happen because bubble sort repeatedly swaps adjacent pairs that are out of order.
2. a **and** b **are not adjacent:** By suitably adjusting the values of a and b, we can assume that among all pairs that are out of order, a and b are the closest to each other. Let c be the element immediately preceding b in the output. There are two possible cases:
   - $c > b$: Again, this case is not possible because bubble sort does not stop until all adjacent pairs are in order.
   - $c \leq b$: In this case, we have $c \leq b < a$, which implies that a and c are also out of order. However, c is one step closer to a than b is, so (a, b) is not the closest violating pair, contradicting our previous assumption.

We've shown that all of the cases are not possible, which proves the correctness of bubble sort. □

2.2 Induction

In this section, we study mathematical induction. We are going to see three types of induction: weak induction, strong induction, and structural induction. We will see their differences as well as examples.

2.2.1 Weak Induction

We begin by studying weak (also known as ordinary) induction. Suppose we want to prove that some predicate $P(n)$ is true for every $n \in \mathbb{Z}^+$, the set of nonnegative integers. We first state the driving principle behind weak induction, which is that the following implication is true:

$$[P(1) \wedge (\forall n.\, P(n) \to P(n+1))] \to [\forall n.\, P(n)],$$

where the universe of n is $\mathbb{Z}^+$. Notice that the consequence (the part after the second "$\to$") of the above implication is what we want to prove. So if we can prove the antecedent (the part before the second "$\to$"), then since the entire implication is true, we would be done. Proving the antecedent requires proving two things: $P(0)$ and, for every $n \in \mathbb{Z}_0^+$, $P(n) \to P(n+1)$. The proposition $P(1)$ is known as the "base case"; $P(n)$, what we assume is true, is known as the "inductive hypothesis"; and $P(n) \to P(n+1)$ is known as the "inductive step."

Now why is the implication true? If the antecedent is true, then $P(1)$ is true, and furthermore, $P(1) \to P(2)$ is true, so $P(2)$ is true. Continuing, since $P(2)$ and $P(2) \to P(3)$ are true, we can conclude $P(3)$ is also true. Thus, through this chain of implications, we can conclude $P(n)$ is true for every $n \in \mathbb{Z}_0^+$, as desired. We now illustrate weak induction with an example.

Example 1: For any $n \in \mathbb{Z}^+$, the following holds:

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}.$$

In this example, we can set $P(n) =$"$\sum_{i=0}^{n} i = n(n+1)/2$", so the equation states

$$\forall n \in \mathbb{Z}^+.\, P(n).$$

We shall now prove the statement using ordinary induction.

Proof. Recall that in ordinary induction, we must prove the following things:

$$P(1) \text{ and } \forall n \in \mathbb{Z}^+.\, P(n) \to P(n+1).$$

Base case: $P(1)$ is true because both sides of the equality in $P(1)$ are equal to 1:

$$\sum_{i=1}^{1} i = 1 = \frac{1(1+1)}{2}.$$

Inductive Step: We now show $P(n) \to P(n+1)$ for every $n \in \mathbb{Z}^+$. Let n be any positive integer, and assume $P(n)$ is true. This assumption is known as the **inductive hypothesis**. In our case, assuming $P(n)$ is true is equivalent to assuming

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

is true. We want to use this inductive hypothesis to prove $P(n+1)$, which is the following:

$$\sum_{i=1}^{n+1} i = \frac{(n+1)(n+2)}{2}.$$

The left side of $P(n+1)$ can be written as follows:

$$\begin{aligned}
\sum_{i=1}^{n+1} i &= 1 + 2 + \cdots + (n-1) + n + (n+1) \\
&= \sum_{i=1}^{n} i + (n+1) \\
&= \frac{n(n+1)}{2} + (n+1) && \text{(by the inductive hypothesis)} \\
&= (n+1)\left(\frac{n}{2} + 1\right) \\
&= \frac{(n+1)(n+2)}{2}.
\end{aligned}$$

This final expression is the right side of $P(n+1)$. Thus, by assuming $P(n)$ is true, we have shown that $P(n+1)$ is true. By the principle of induction, we can conclude that $P(n)$ is true for all $n \in \mathbb{Z}^+$. □

To summarize, a proof by weak induction that proves a predicate $P(n)$ for $n \in \mathbb{Z}^+$ has the following steps:

1. **Base Case:** Prove that $P(1)$ is true.
2. **Inductive Hypothesis:** Precisely state the hypothesis that $P(n)$ is true.
3. **Inductive Step:** Prove that $P(n+1)$ is true using the inductive hypothesis.

Example 2: Suppose $F(1) = 2$ and $F(n+1) = 2 \cdot F(n)$. Prove that $F(n) = 2^n$.

Proof. We need to prove two things for induction: $F(1) = 2^1$, and if $F(n) = 2^n$, then $F(n+1) = 2^{n+1}$. We formalize the proof as follows.

Base Case: When $n = 1$, $F(1) = 2^1 = 2$, which is true.

Inductive Hypothesis: Let n be a positive integer, and assume $F(n) = 2^n$.

Inductive Step: We want to show that $F(n+1) = 2^{n+1}$.

The left hand side can be written as:

$$\begin{aligned}
F(n+1) &= 2 \cdot F(n) \\
&= 2 \cdot 2^n && \text{(by the inductive hypothesis)} \\
&= 2^{n+1}.
\end{aligned}$$

Thus, by the principle of induction, we can conclude $P(n)$ is true for all $n \in \mathbb{Z}^+$. □

Example 3: The Fibonacci sequence is defined as follows: $F(0) = 1, F(1) = 1, F(n) = F(n-1) + F(n-2)$. We want to show $F(n) \geq (\sqrt{2})^{n-1}$.

Proof. We proceed with induction. Notice that in this example, our base case starts with both $n = 0$ and $n = 1$, because they are both necessary for the statement to hold when $n = 2$. Furthermore, our inductive hypothesis is slightly stronger than the usual hypothesis, but it is still valid.

> **Base Case:** First, $F(0) = 1 \geq (\sqrt{2})^{0-1} = 1/\sqrt{2}$. Second, $F(1) = 1 \geq (\sqrt{2})^{1-1} = 1$. Therefore the base case is true.
>
> **Inductive Hypothesis:** Let n be a positive integer, and assume $F(n-1) \geq (\sqrt{2})^{n-2}$ and $F(n) \geq (\sqrt{2})^{n-1}$.
>
> **Inductive Step:** We want to show that $F(n+1) \geq (\sqrt{2})^{n}$.

The left hand side can be written as:

$$\begin{aligned} F(n+1) &= F(n) + F(n-1) \\ &\geq (\sqrt{2})^{n-2} + (\sqrt{2})^{n-1} && \text{(by the inductive hypothesis)} \\ &= (\sqrt{2})^{n-2}(\sqrt{2}+1) \\ &\geq (\sqrt{2})^{n-2}(\sqrt{2})^{2} \\ &= (\sqrt{2})^{n} \end{aligned}$$

which is the right hand side. Now we can conclude that $F(n) \geq (\sqrt{2})^{n-1}$ for every $n \in \mathbb{Z}^+$. □

2.2.2 Strong Induction

Now we will introduce a more general version of induction known as *strong induction*. The driving principle behind strong induction is the following proposition which is quite similar to that behind weak induction:

$$\big[P(1) \wedge \big(\forall n.\, (P(1) \wedge P(2) \wedge \cdots \wedge P(n)) \rightarrow P(n+1)\big)\big] \rightarrow [\forall n.\, P(n)],$$

Again, the universe of n is $\mathbb{Z}^+$. Notice that this is similar to weak induction: the conclusion is the same, we still have $P(1)$ as a base case, and the second part of the antecedent is also an implication. This time, however, the antecedent of the implication is

$$P(1) \wedge P(2) \wedge \cdots \wedge P(n).$$

Recall that in weak induction, the antecedent was only $P(n)$. This means that strong induction allows us to assume n predicates are true, rather than just 1, when proving $P(n+1)$ is true.

For example, in ordinary induction, we must prove $P(4)$ is true assuming $P(3)$ is true. But in strong induction, we must prove $P(4)$ is true assuming $P(1)$ and $P(2)$ and $P(3)$ are *all* true. Note that any proof by weak induction is also a proof by strong induction—it just doesn't make use of the remaining $n-1$ assumptions. We now proceed with examples.

Example 1: Define a recursive function $f(n) = f(n-1) + f(n-2) + ... + f(1) + f(0)$ where $n \in \mathbb{Z}^+$ and define $f(0) = 1$. Let's prove $f(n) = 2^{n-1}$ for all integer $n > 0$.

Proof. We proceed with strong induction, because the recursive function has all terms from $f(0)$ to $f(n-1)$.

Base case: When $n = 1, f(1) = f(0) = 1 = 2^{1-1}$.

Inductive hypothesis: Let n be a positive integer, and assume $f(k) = 2^{k-1}$ for every $k \in \{1, 2, \ldots, n\}$. In other words, we assume $f(1) = 2^0, f(2) = 2^1, f(3) = 2^2, \ldots, f(n) = 2^{n-1}$.

Inductive step: We want to show that $f(n+1) = 2^n$.

The left hand side can be written as:

$$\begin{aligned} f(n+1) &= f(0) + f(1) + \cdots + f(n) \\ &= 1 + 2^0 + 2^1 + \cdots + 2^{n-1} && \text{(by the inductive hypothesis)} \\ &= 1 + 1 + 2 + \cdots + 2^{n-1} \\ &= 2^n \end{aligned}$$

which is the right hand side. Therefore we conclude that $f(n) = 2^{n-1}$ for all $n \in \mathbb{Z}^+$. □

Example 2: Define a recursive function $f(n) = f(n-1) + f(n-2) + ... + f(1) + f(0) + n$ for $n \in \mathbb{Z}^+$ and define $f(0) = 0$. Our task is to express this function as a closed-form expression (i.e., there should be no summations). Notice that $f(1) = 1$, $f(2) = 3$, and $f(3) = 7$. We conjecture that $f(n) = 2^n - 1$ for $n \in \mathbb{Z}_{\geq 0}$, and proceed with a proof by strong induction.

Base case: When $n = 0$, $f(0) = 2^0 - 1 = 0$, which is true by definition.

Inductive hypothesis: Let n be a positive integer and assume $f(k) = 2^k - 1$ for every $k \leq n$.

Inductive step: We want to show $f(n+1) = 2^{n+1} - 1$.

The left hand side can be written as:

$$\begin{aligned} f(n+1) &= f(0) + f(1) + \cdots + f(n) + (n+1) \\ &= 2^0 - 1 + 2^1 - 1 + \cdots + 2^n - 1 + n + 1 && \text{(by the inductive hypothesis)} \\ &= (2^0 + 2^1 + \cdots + 2^{n-1}) - 1 \cdot (n+1) + n + 1 \\ &= 2^n - 1 \end{aligned}$$

which is the right hand side. Therefore we conclude $f(n) = 2^n - 1$ for all $n \in \mathbb{Z}^+$.

Example 3: Recall the breadth-first search (BFS) applied to undirected graphs. The source vertex, which we denote by s, is labeled $d(s) = 0$. In the first iteration, we visit each neighbor v of s and set $d(v) = 1$. In general, in the i-th iteration, we label every un-visited neighbor v of vertices u such that $d(u) = i - 1$ with $d(v) = i$. The process stops until all vertices are labeled. We now prove that for every vertx v, $d(v)$ is the shortest path length from s.

More formally, we define $P(i)$ as the following property: after the i-th iteration, every vertex whose distance from s is at most i is correctly labeled. We now prove that the property holds for every $i \in \{0, 1, \ldots, n-1\}$. Note that the maximum length from s to any vertex is at most $n - 1$, so $P(n-1)$ implies that all labels are correct.

Base case: In the beginning of the algorithm, we set $d(s) = 0$. This is correct because the length of the shortest path from s to itself is indeed 0, so $P(0)$ holds.

Inductive Hypothesis: Assume that $P(i)$ holds for iterations $i \in \{0, 1, \ldots, k\}$ for some nonnegative integer k.

Inductive step: We now prove $P(k+1)$ holds. Consider a vertex v visited in iteration $k+1$ from its parent u, so $d(u) = k$. We must show that the shortest distance from s to v is indeed $k+1$.

For contradiction, assume there is a path of length j from s to v for some $j \leq k$, and let w denote the vertex preceding v on this path. Note that the distance from s to w is shorter than the distance from s to u. By our inductive hypothesis, this implies that $d(w)$ is correctly labeled. However, this means that the algorithm would have visited v from w in an iteration earlier than $k+1$, contradicting our selection of v as a vertex visited in iteration $k+1$.

Thus, the property holds for every iteration i, so BFS correctly labels every vertex.

2.2.3 Structural Induction

We now turn our attention to structural induction ,which is essentially induction applied to particular data structures. First, let us define some familiar data structures recursively.

1. Linked List

   Base case: An empty linked list is a linked list

   Constructive: Attaching a node to a linked list yields a linked list.

2. Binary Tree

   Base case: An empty tree is a binary tree.

   Constructive: A node pointing to two binary trees (a left child and a right child) is the root of a binary tree.

3. String

   Base case: The empty string, denoted by λ, is a string.

   Constructive: If x is a symbol and s is a string, then the concatenation of x and s, denoted by $x.s$, is also a string.

The general idea is the following: given a data structure, our inductive hypothesis allows us to assume that some property holds for some particular substructures. Using this, we can show that the property holds for the entire structure. We now proceed with a few examples.

Example 1: Let $\ell(s)$ denote the length of a string, where we define $\ell(\lambda) = 0$. Let us prove that for any two strings s and t, we have $\ell(s.t) = \ell(s) + \ell(t)$.

Proof. We induct on the length of the string s.

Base case: If $s = \lambda$, then $\ell(s.t) = \ell(\lambda.t) = \ell(t) = 0 + \ell(t) = \ell(\lambda) + \ell(t)$.

Inductive hypothesis: Any substring r of s (excluding s) satisfies $\ell(r.t) = \ell(r) + \ell(t)$.

Inductive step: Suppose $s = x.r$ for some symbol x and string r, and assume that $\ell(r.t) = \ell(r) + \ell(t)$. Notice the following:

$$\begin{aligned}\ell(s.t) &= \ell(x.r.t) \\ &= \ell(x.(r.t)) \\ &= 1 + \ell(r.t) \\ &= 1 + \ell(r) + \ell(t) \qquad \text{(by applying the inductive hypothesis to } r\text{)}\end{aligned}$$

Therefore the property that $\ell(s.t) = \ell(s) + \ell(t)$ holds. □

Example 2: The *depth* of a non-empty tree is defined as follows: the depth of a single node is 1. Let $d(T)$ denote the depth of a tree T and define $d(T) = 1 + \max(d(T_1), d(T_2))$, where T_1 and T_2 are the left and right child of T, respectively. Let us prove that a non-empty binary tree of depth d has at most $2^d - 1$ nodes.

Proof. We apply structural induction on the tree.

Base case: For a single node, we have $1 \leq 2^1 - 1$.

Inductive hypothesis: Any subtree of T with depth $d' < d$ has at most $2^{d'} - 1$ nodes.

Inductive step: Let T_1 and T_2 denote the left and right subtrees of T, with depths d_1 and d_2, respectively. We know $d = d(T) = \max(d_1, d_2) + 1$. By the inductive hypothesis, T_1 has at most $2^{d_1} - 1$ nodes and T_2 has at most $2^{d_2} - 1$ nodes. Therefore, the number of nodes in T is at most

$$1 + 2^{d_1} - 1 + 2^{d_2} - 1 \leq 2 \cdot 2^{d-1} - 1 = 2^d - 1.$$ □

Example 3: A *leaf* of a tree is a node with no children. Let us prove that a binary tree with depth d has at most 2^{d-1} leaves.

Proof. Again, we apply structural induction on the tree.

Base case: For a single node, we have $1 \leq 2^{1-1}$.

Inductive Step: Any subtree of T with depth d' has at most $2^{d'-1}$ leaves.

Inductive steps: Let T_1 and T_2 denote the left and right subtrees of T, with depths d_1 and d_2, respectively. We know $d = d(T) = \max(d_1, d_2) + 1$. By the inductive hypothesis, T_1 has at most 2^{d_1-1} leaves and T_2 has at most 2^{d_2-1} leaves. Therefore, the number of leaves in T is at most

$$2^{d_1-1} + 2^{d_2-1} \leq 2^{d-2} + 2^{d-2} = 2^{d-1}.$$ □

2.3 The Well-Ordering Principle

2.3.1 Overview

In this section, we will introduce an intuitive principle and illustrate how it can serve as a powerful technique in proving some non-obvious theorems.

2.3.2 Well-Ordering Principle

The well-ordering principle (*WOP*):

There is a smallest number in any set of positive integers.

While this may seem obvious, what if instead we had claimed there is a greatest number in any set of positive integers?

Example 1: Consider the set of even numbers. The smallest even number is 2, but there is no greatest even integer in this set.

Our sets may not have finite size. Here are a few more examples:

Example 2: The set of positive odd integers: The smallest element in this set is 1.
Example 3: The set of positive integers divisible by 3: The smallest element in this set is 3.

Example 4: The set of positive integers *not* divisible by 3: The smallest element in this set is 1.

We will consider a few theorems to demonstrate how we may use the *WOP* in a proof.

Theorem 1. *For any rational number q we can express it as a ratio of two integers, x and y such that $y \neq 0$ and x, y are coprime.*

Recall:

Definition 6. *Two integers x and y are coprime if $gcd(x, y) = 1$, i.e. they share no common factors other than 1.*

Theorem 1 seems apparent: any rational number can be represented as a ratio of two integers. Take one such ratio, and cancel common factors until you have a fraction in lowest terms. However, this intuition does not prove the statement. Let's prove the statement formally.

Proof of Theorem 1. Assume the theorem is false, that is assume there exist rational numbers that cannot be expressed as the ratio of two integers with *gcd* 1. Let q be such a rational number. Without loss of generality, assume $q > 0$. If $q < 0$, we can redefine q as $-q$. Consider all possible ratios q can be expressed as:

$$q = \frac{x_1}{y_1} = \frac{x_2}{y_2} = \dots$$

where $y_1 \neq 0, y_2 \neq 0, \dots$. Furthermore, $y_1 > 0, y_2 > 0, \dots$ and $x_1 > 0, x_2 > 0, \dots$ since $q > 0$. We define a set of positive integers by taking the numerators of the above fractions:

$$S = \{x_1, x_2, \dots\}$$

This set may be infinite, but it is a set of positive integers so we can apply the well-ordering principle. The *WOP* implies there exists a smallest integer in S. Call this integer x^*. Then, $q = \frac{x^*}{y^*}$ for some $y^* > 0$. By our original assumption, $gcd(x^*, y^*) \geq 2$ so x^* and y^* must share a factor greater than 1, call this factor p. We have:

$$q = \frac{x^*}{y^*} = \frac{x' \cdot p}{y' \cdot p} = \frac{x'}{y'}$$

$y' \neq 0$ because $y^* \neq 0$. Thus, x' is a numerator of a ratio of two integers equal to q, so by the definition of S, it must be that $x' \in S$. However, $x' < x^*$ because $p > 1$. This is a contradiction since we assumed x^* was the smallest element of S. Therefore, our original assumption was incorrect and the theorem holds. □

Consider another theorem you are familiar with:

Theorem 2. *Every positive integer can be written as the product of primes.*

Proof. We will proceed with a proof by contradiction. Assume there exists some positive integer that cannot be written as the product of primes. Let S be the set of positive integers that do not have a prime factorization. By the *WOP* on S, there is a smallest positive integer that cannot be written as the product of primes. Call this integer n^*. Consider the following cases:

- n^* is prime. In this case, the prime factorization of n^* is n^* itself. This is a contradiction, as $n^* \in S$ so by definition does not have a prime factorization.
- n^* is composite (not prime). Then $n^* = a \cdot b$ such that $a \neq n^*, a \neq 1$ and $b \neq n^*, b \neq 1$. This implies that $a < n^*$ and $b < n^*$. Since n^* is the smallest element of S, $a \notin S$ and $b \notin S$. Therefore, a, b have prime factorizations:

  $$a = p_1 \cdots p_k$$

  $$b = q_1 \cdots q_\ell$$

  for prime numbers $p_1, \ldots, p_k, q_1, \ldots, q_\ell$. But,

  $$n^* = a \cdot b = p_1 \cdot \cdots \cdot p_k \cdot q_1 \cdot \cdots \cdot q_\ell.$$

  This is a prime factorization of n^*. This is a contradiction of our original assumption.

In all cases, we reached a contradiction. Thus, our original assumption was incorrect and every positive integer can be written as the product of primes. □

2.3.3 Well-Ordered Sets

Definition 7. *A* well-ordered set *is a set S such that each non-empty subset of S has a minimum element.*

In other words, the well-ordering principle states that the set of all positive integers is well-ordered. Consider the following:

Claim 1. *The set of integers that are $\geq -n$ for any integer n is well-ordered.*

Proof. Let S be a subset of integers such that $\forall s \in S, s \geq -n$. Define a new set as follows:

$$S' = \{s + n + 1 \ : \ s \in S\}$$

This notation means that for every element s in S add element $(s + n + 1)$ to set S'. For each $s \in S, s \geq -n$ implies that $s + n \geq 0$, so $s + n + 1 \geq 1$. Thus, S' is a set of positive integers. By the well-ordering principle there is a smallest integer in S'. Let this integer be x^*. Then, the smallest integer in S is $s = x^* - (n + 1)$ and the claim is true. □

Theorem 3. *Any set of integers with a lower bound is well-ordered.*

Before proving this theorem, let's define a lower bound:

Definition 8. *A* lower bound *of a set S is any real number y such that $y \leq s \; \forall s \in S$.*

Observe that a lower bound is not unique. $0, 1$, and -100 are all lower bounds on the set of positive integers. For the set of all positive integers, 1 is the largest lower bound. We can similarly define an upper bound:

Definition 9. *A* upper bound *of a set S is any real number y such that $s \leq y \; \forall s \in S$.*

We can now prove the theorem:

Proof of Theorem 3. Let X be a set of integers with a lower bound, and let S be a non-empty subset of X. Let y be one lower bound of X. Define a new set as follows:

$$S' = \{s - y + 1 | s \in S\}$$

Since y is a lower bound, $s \geq y \; \forall s \in S$. This implies $s - y \geq 0$, and thus $s - y + 1 \geq 1$. Therefore, S' is a set of positive integers. By the well-ordering principle there is a smallest integer in S'. Let this integer be x^*. Then, the smallest integer in S is $s = x^* + y - 1$, and X is well-ordered. □

Theorem 4. *Any set of integers with an upper bound has a maximum element.*

Proof. Let S be a subset of integers with an upper bound. Let y be one such upper bound. Define a new set as follows:

$$S' = \{y + 1 - s | s \in S\}$$

Since y is an upper bound, $s \leq y \; \forall s \in S$. For all $s \in S$, this implies $y - s \geq 0$ and therefore $y + 1 - s \geq 1$. Thus, S' is a set of positive integers. By the well-ordering principle, S' has a smallest integer. Let this integer be x^*. Then, the largest integer in S is $s = y + 1 - x^*$. □

Why are these theorems not immediately obvious? Consider the following (false) claim:

Claim 2. *Any set of rational numbers with a lower bound is well-ordered.*

Counter Example: Define the following set of rational numbers:

$$S = \{1 + x \mid x \text{ is any rational number s.t. } 0 < x < 1\}$$

S has a lower bound of 1. For the claim to hold, we would need to show that every non-empty subset of S has a minimum element. In particular, this implies that S must have a minimum element. Let this be $x^* = 1 + \frac{p}{q}$ for some integers p, q where $p, q > 0$. Now define $x' = 1 + \frac{p}{q+1}$.

$0 < \frac{p}{q+1} < 1$ because $0 < \frac{p}{q} < 1$. Additionally $\frac{p}{q+1}$ is rational because $\frac{p}{q}$ is rational. Thus, $x' \in S$. However, $x' < x^*$, which contradicts our assumption that x^* was the smallest element of S. Thus, S is not well-ordered and the claim is false.

An intuitive idea about the well-ordered property is as follows: You start with some number in the set and you consider smaller and smaller elements. Eventually, there will not be a smaller element left to consider. For the set of positive integers you may start at 5 and begin counting down to 1, where you stop. This idea captures the fact that there can be no infinite length decreasing

sequence from a fixed starting point in a well-ordered set. Otherwise, there would always exist a smaller element in the set and there would be no minimum. Recall some examples of familiar infinite decreasing sequences:

Example 5: The negative integers: $-1, -2, -3, \ldots$

Example 6:

$$\frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \ldots \frac{1}{x}, \frac{1}{x+1}, \ldots$$

Although a well-ordered set does not have a decreasing sequence of infinite length, we will see in the next lecture that a well-ordered set *can* have decreasing sequences of arbitrarily large finite length.

2.3.4 Summary

In this section, we considered the well-ordering principle and discussed well-ordered sets. We used the well-ordering principle to formally prove familiar theorems. We also showed that this principle can be used to show that various sets are well-ordered.