Logic and Proof
Mathematical Structures for CS (CSCI 3030)

Prof. Harley Eades (heades@gru.edu).

Read chapter 1.1 - 1.5,1.7,1.8

Logic is everywhere. It underlies computer science, mathematics, science, law, and even the humanities. Formally, a logic is a framework for reasoning about truth. This framework has operators for constructing logical statements we call **formulas**. A logical formula can be **either true or false, but not both**. Here are a few examples of formulas:

- It is raining outside and my shoes are wet.
- I was born in 1981 or 1991.
- If the glove fits, then the defendant is guilty.
- A program is either terminating or diverging.
- If $n >= 1$ and $LD(n) = n$, then n is prime.
- For every GRU student s, such that, s is taking CSCI:3030, the student s is not learning Haskell.
- There exists a bug in Gears of War 2 that allows one to escape the map.
- There are no natural numbers, a, b, and c, such that, for any natural number $n > 2$, $a^n + b^n = c^n$.

Every statement in the previous list is either true or false. Consider the statements in the following list:

- $f(x) = x^2 + 2$
- ```
  last (a:[]) = a
  last (x:xs) = last xs
  ```
- Look over there!

The previous statements are neither true nor false, and so these do not constitue formulas.

1 Propositional Logic

1.1 Introduction

Propositional logic is a logical framework that allows reasoning about a special class of formulas called propositional formulas.

Definition 1. *A **propositional formula** is one that is constructed from the following list of logical operators:*

- ***True**: True*
- ***False**: False*
  ```

- ***Conjunction**: I am a professor and I teach CS.*
- ***Disjunction**: We are either alone or aliens exist.*
- ***Implication**: If I dance, then we are happy.*
- ***Negation**: The negation of "I am happy" is "I am sad."*

Now we define each of the above concepts in detail. The proposition $True$ is the proposition that is always true, and its opposite $False$ is the proposition that is always false. A **propositional variable** is like a variable found in algebra, but it ranges over $True$ and $False$ instead of numbers.

Definition 2. *Suppose p and q are propositional variables. Then the **conjunction** of p and q is denoted $p \wedge q$. We call p and q the conjuncts of the conjunction. A conjunction is true only when both conjunctions are true.*

Consider the statement:

The Cardinal's won yesterday and I was there.

We can see that this statement is indeed a conjunction by setting $p =$ "The Cardinal's won yesterday" and $q =$ "I was there", and then the previous statement is equivalent to $p \wedge q$.

Exercise: Translate the following into propositional formulas:

- $0 \leq 5 \leq 200$
- The function, $f(x)$, threw an exception and the computer blew up.

The truthfulness of a proposition can be defined by a table known as a **truth table**. Given a proposition, P, a truth table has a column for each propositional variable in P, and then one for P itself. Then there are rows for each possible setting of $True$ and $False$ to each propositional variable. Finally, the final column of each row is the outcome of substituting the values of the propositional variable in that row in P. This implies that the table must contain every possible choice between $True$ and $False$ for each propositional variable. As we will learn later in the course there are 2^n possible choices where n is the number of propositional variables in P. Thus, there must be 2^n rows in a truth table. As an example we define the truth table for conjunction as follows:

p	q	$p \wedge q$
$True$	$True$	$True$
$True$	$False$	$False$
$False$	$True$	$False$
$False$	$False$	$False$

Definition 3. *Suppose p and q are propositional variables. Then the **disjunction** of p and q is denoted $p \vee q$. We call p and q the disjuncts of the disjunction. A disjunction is true when either of its disjuncts are true. The following truth table summarizes this:*

p	q	$p \vee q$
$True$	$True$	$True$
$True$	$False$	$True$
$False$	$True$	$True$
$False$	$False$	$False$

Exercise: Find an example like the example for conjunction.

Definition 4. *Suppose p is a proposition. Then the negation of p, denoted $\neg p$, is defined as follows:*

p	$\neg p$
$True$	$False$
$False$	$True$

The proposition $\neg p$ is read "not p."

Exercise: What is the negation of "I am hungry and the moon is far."?

Definition 5. *Suppose p and q are propositional variables. Then the **implication** of p and q is denoted $p \Rightarrow q$. The latter is read "p implies q" or "if p then q". Implication is defined as follows:*

p	q	$p \Rightarrow q$
$True$	$True$	$True$
$True$	$False$	$False$
$False$	$True$	$True$
$False$	$False$	$True$

Consider the following example:

$$\text{If } n = (n-1) + (n-2), \text{ then } n \text{ is a Fibonacci number.}$$

Now set $p =$ "$n = (n-1) + (n-2)$" and $q =$ "n is a Fibonacci number". Then we can see that the previous statement has the form $p \Rightarrow q$.

Explain the truth table and how p is the gatekeeper.

Given an implication, $P = p \Rightarrow q$, we can construct additional propositions. If we flip the implication resulting in $q \Rightarrow p$, then we obtain the **converse** proposition of P. If we negate p and q resulting in $\neg p \Rightarrow \neg q$, then we obtain the **inverse** of P. Finally, If we take the converse of P followed by the inverse resulting in $\neg q \Rightarrow \neg p$, then we obtain the **contrapositive** of P.

Exercise: Compute the truth tables for the converse, inverse, and contrapositive of $p \Rightarrow q$.

Implication is ubiquitous in both mathematics and computer science, and so, there are a lot of ways to express one. The following table are all equivalent:

if p, then q
q unless not p
p implies q
p only if q
q whenever p
q follows from p
p is sufficient for q
q is necessary for p
a necessary condition for p is q
a sufficient condition for q is p

1.2 Compound Propositions

So far we have only seen very basic propositions using only a single propositional connective at a time. In this section we introduce compound propositions which are propositions that are built from more than one connective. Consider the example:

$$(\text{if } x > 2, \text{ then } f(x-1) > 3) \text{ or } (\text{if } x \leq 2, \text{ then } f(x+1) == 2)$$

This is a compound statement made up of implication, disjunction, and negation. We can see this by setting $p = x > 2$, $q = f(x-1) > 3$, and $r = f(x+1) == 2$. Notice that $\neg p = x \leq 2$. Thus, the previous statement is equivalent to $(p \Rightarrow q) \vee (\neg p \Rightarrow r)$.

Now consider the following example:

$$p \wedge q \Rightarrow r$$

Should this be read as "(p and q) implies r" or "p and (q implies r)"? To overcome these problems we will use the following **convention**:

> Negation binds more tightly than disjunction or conjunction, and the latter two bind more tightly than implication. Finally, implication is right-associative; which means that an implication like $p \Rightarrow q \Rightarrow r$ is understood to mean $p \Rightarrow (q \Rightarrow r)$, and $p \Rightarrow q \Rightarrow r \Rightarrow s$ is understood to mean $p \Rightarrow (q \Rightarrow (r \Rightarrow s))$.

At this point we consider several compound statements:

- If the sun shines today, then it won't shine tomorrow.
- Today it will rain or shine, but not both.
- No shoes, no shirt, no service.
- My sister wants a black and white cat.

There is one very important compound proposition that stands for equivalence.

Definition 6. *Suppose p and q are propositional variables. Then the **biconditional** of p and q is denoted $p \Leftrightarrow q$. If a biconditional is true, then p and q are equivalent propositions. The truth table for the biconditional is as follows:*

p	q	$p \Leftrightarrow q$
$True$	$True$	$True$
$True$	$False$	$False$
$False$	$True$	$False$
$False$	$False$	$True$

Now the biconditional can be defined interms of implication and conjunction. The biconditional $p \Leftrightarrow q$ can be defined as $(p \Rightarrow q) \wedge (q \Rightarrow p)$. To check that this is true we can construct a truth table for the previous compound proposition:

p	q	$p \Rightarrow q$	$q \Rightarrow p$	$(p \Rightarrow q) \wedge (q \Rightarrow p)$
$True$	$True$	$True$	$True$	$True$
$True$	$False$	$False$	$True$	$False$
$False$	$True$	$True$	$False$	$False$
$False$	$False$	$True$	$True$	$True$

This example shows that truth tables can be used to understand when a compound proposition is true.

Another important compound proposition is called the **law-of-excluded-middle (LEM)**, and it is defined as $\neg p \vee p$. Consider the truth table of LEM:

p	$\neg p$	$\neg p \vee p$
$True$	$False$	$True$
$False$	$True$	$True$

The above truth table reveals that LEM is always true. A proposition that is always true is called a **tautology**. The intuitive meaning of LEM is that a proposition, p, is either true or false, and so, it makes sense that this is always true.

Disjunction states a proposition p is true or a proposition q is true, or both the propositions p and q are true. We can use negation, disjunction, and conjunction to define when one of two propositions p and q are true, but not both.

Definition 7. *Suppose p and q are propositional variables. Then the **exclusive-or** of p and q is denoted $p \oplus q$. The truth table for exclusive-or is as follows:*

p	q	$p \oplus q$
$True$	$True$	$False$
$True$	$False$	$True$
$False$	$True$	$True$
$False$	$False$	$False$

The exclusive-or is heavily used in circuit design, because it can be used to simply circuits, and thus use less materials when constructing the physical circuit. Finally, the definition of exclusive-or in terms of conjunction, disjunction, and negation is $(p \wedge \neg q) \vee (\neg p \wedge q)$. We can verify this by constructing the truth table:

p	q	$\neg p$	$\neg q$	$p \wedge \neg q$	$\neg p \wedge q$	$(p \wedge \neg q) \vee \neg p \wedge q)$
$True$	$True$	$False$	$False$	$False$	$False$	$False$
$True$	$False$	$False$	$True$	$True$	$False$	$True$
$False$	$True$	$True$	$False$	$False$	$True$	$True$
$False$	$False$	$True$	$True$	$False$	$False$	$False$

1.3 Reasoning Using Equivalences

So far to prove P is equivalent to a second proposition Q we must construct the truth tables for both P and Q and then compare them. This can be tedious and long, because when P and Q are long and complex the truth tables grow. In this section we will define a more suitable theory for reasoning about propositions using equivalences in much of the same manner as one does in algebra. Throughout this section and the remainder of the course we will denote arbitrary compound propositions by capital alphabet symbols like P, Q, R, S, etc.

First, the basic definition of logical equivalence.

Definition 8. *Propositions P and Q are **logically equivalent**, denoted by $P \equiv Q$, if and only if P and Q have the same truth table.*

Notice that logical equivalence itself is a proposition, because it can only be true or false. Thus, we can prove properties about logical equivalence. In fact, to aid in our understanding we will add one additional notion of equality.

Definition 9. *Suppose E_1 and E_2 are logical equivalences, then they are considered equal, denoted by $E_1 = E_2$, if E_1 and E_2 are both true.*

This new equality can be used to prove properties between logical equivalences. For example, consider the property $(p \wedge q) \equiv p = (p \wedge q) \equiv q$.

1.4 The Four-Queens Problem

We have completed the introduction to propositional formulas and truth tables. However, we have spent a lot of time working in the abstract, and it might not be apparent what applications these mathematical devices in computer science. It is true that logic plays at least a basic role when writing software[1], and a more fundamental role when reasoning about the correctness of that software. However, these two applications are obvious connections, but can logic be used solve more computational problems?

It is well-known that every propositional formula can be written in a special form called Conjunctive Normal Form (CNF).

Definition 10. *A formula ϕ is in **Conjunctive Normal Form (CNF)** if and only if ϕ is a conjunction of clauses where*

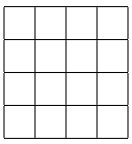
- *a **clause** is a formula of the form $x_1 \vee \cdots \vee x_i$ where each x_j for $1 \leq j \leq i$ are literals, and*
- *a **literal** is a formula of the form p or $\neg p$ where p is a propositional variable.*

We will not prove this, but any formula in propositional logic can be translated into CNF form. As an example consider the formula $p \Rightarrow q$, this is equivalent to the clause $\neg p \vee q$ which is actually a conjunction, because this formula is equivalent to the conjunction $(\neg p \vee q) \wedge True$. How about this formula $p \Rightarrow (q \wedge r)$? This formula is equivalent to $\neg p \vee (q \wedge r)$ which is equivalent to $(\neg p \vee q) \wedge (\neg p \vee r)$ and this formula is in CNF. In fact, using De Morgan's laws, and the distributivity laws for conjunction and disjunction any formula can be converted into CNF.

The reason CNF is a better form for propositional formulas is that they are easy to check automatically if they are valid. That is we can get a computer to do the natural deduction proof for us. In all honesty, the automatic proof of validity of a formula in CNF is not a proof at all, and is driven by the truth table construction.

Given a formula in CNF we know that it must be in the form of a conjunction of disjunctions of literals, where literals consist of a propositional variable or the negation of a propositional variable. Thus, each propositional variable is a premise. To check to see if a formula in CNF is valid it is enough to find a replacement of $True$ and $False$ for each propositional variable that makes the CNF formula true. This is called checking to see if the formula is **satisfiable**. If a formula is **satisfiable**, then we know there must exist a proof in natural deduction of its validity. There are various tools called **sat solvers** that can automatically check to see if any formula in CNF is satisfiable. Sat solvers have been used in a wide range of applications. We will show that a sat solver can be used to solve the four-queens problem.

Suppose we are given a four-by-four chessboard. It would look at little something like this:

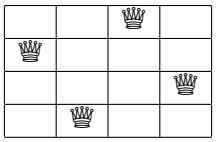


The four-queens problem is stated as follows:

> Is it possible to place four queens on the four-by-four chessboard in such away that no two queens are attacking?

Here is an example solution:

[1] This is an understatement.



We will solve this problem by first describing a set of constraints that must be true given a valid solution, and false otherwise. A **constraint** is a propositional formula that a solution to a particular problem must satisfy. For example, suppose `l = x1:x2:⋯:xn:[]` is a list of integers. Then `l` is sorted if $\texttt{x1} \leq \texttt{x2} \leq \cdots \leq \texttt{xn}$. The if-condition is the constraint. Constraints tell us when a proposed solution to a problem really is a solution.

What are the constraints each solution to the four-queens problem must satisfy? First, we know that each row must have exactly only queen in it, or else two queens will be attacking. Similarly, for each column. Lastly, there can be at most one queen per diagonal. No other constraints are necessary. If we can choose four cells to assign queens to, such that, the previous constraints holds, then we know we have found a valid solution. The next question we must solve is how do formally define the constraints of the four-queens problem? This is where logic really comes in.

Given a chessboard assign to each cell a propositional variable:

q_{11}	q_{12}	q_{13}	q_{14}
q_{21}	q_{22}	q_{23}	q_{24}
q_{31}	q_{32}	q_{33}	q_{34}
q_{41}	q_{42}	q_{43}	q_{44}

A queen is said to be assigned to the cell (i, j) if and only if q_{ij} is true. Now we need to define each constraint in CNF using the propositional variables labeling the board. Before trying to do this for the entire board itself first we try to define similar constraints for just a few propositional variables to get an idea for how it is done.

To make the task of constructing the constraints in CNF we will first define a well-known translation of arbitrary propositional formulas into equivalent propositional formulas in CNF form. The translation requires using the following logical equivalences:

$$
\begin{array}{lrcl}
1. & \phi \Rightarrow \psi & \equiv & \neg\phi \vee \psi \\
2. & \neg(\phi \wedge \psi) & \equiv & \neg\phi \vee \neg\psi \\
3. & \neg(\phi \vee \psi) & \equiv & \neg\phi \wedge \neg\psi \\
4. & \phi \wedge (\psi_1 \vee \psi_2) & \equiv & (\phi \wedge \psi_1) \vee (\phi \wedge \psi_2) \\
5. & \phi \vee (\psi_1 \wedge \psi_2) & \equiv & (\phi \vee \psi_1) \wedge (\phi \vee \psi_2)
\end{array}
$$

The translation, $CNF(\phi)$, from propositional logic into CNF form is defined as follows:

i. (Remove Implications) Remove every implication from ϕ using the first logical equivalence above;

ii. (Push Negations) Push every negation to literals using the second and third logical equivalences above;

iii. (Distributions) Apply the logical equivalences four and five above to put the formula in the CNF.

As an example suppose we wanted a propositional formula in CNF that is true only when exactly one of p and q are true. A propositional formula that captures this is exclusive or, and is defined as $(p \wedge \neg q) \vee (\neg p \wedge q)$, but this is not in CNF. We can obtain a CNF formula by applying the CNF translation above:

i. Step one is easy, because there are no implications.

ii. All negations are on literals, and so this step is also easy.

iii. The following steps are needed:

$$
\begin{aligned}
(p \wedge \neg q) \vee (\neg p \wedge q) &\equiv ((p \wedge \neg q) \vee \neg p) \wedge ((p \wedge \neg q) \vee q) \\
&\equiv (\neg p \vee (p \wedge \neg q)) \wedge (q \vee (p \wedge \neg q)) \\
&\equiv ((\neg p \vee p) \wedge (\neg p \vee \neg q)) \wedge (q \vee (p \wedge \neg q)) \\
&\equiv ((\neg p \vee p) \wedge (\neg p \vee \neg q)) \wedge ((q \vee p) \wedge (q \vee \neg q)) \\
&\equiv (\neg p \vee p) \wedge (\neg p \vee \neg q) \wedge (q \vee p) \wedge (q \vee \neg q) \\
&\equiv (\neg p \vee \neg q) \wedge (q \vee p) \wedge (q \vee \neg q) \\
&\equiv (\neg p \vee \neg q) \wedge (q \vee p) \\
&\equiv (q \vee p) \wedge (\neg p \vee \neg q)
\end{aligned}
$$

At this point consider the question of defining a propositional formula that is true when at most one of p, q, or r is true. The formula $(\neg p \vee \neg q) \wedge (\neg p \vee \neg r) \wedge (\neg q \vee \neg r)$, but this is already in CNF. Using these insights we can now define the constraints to model the problem.

The following define the constraints of the four-queens problem:

- Exactly one queen in each row:

$$
\begin{array}{l}
(q_{i1} \vee q_{i2} \vee q_{i3} \vee q_{i4}) \wedge \\
(\neg q_{i1} \vee \neg q_{i2}) \wedge \\
(\neg q_{i1} \vee \neg q_{i3}) \wedge \\
(\neg q_{i1} \vee \neg q_{i4}) \wedge \\
(\neg q_{i2} \vee \neg q_{i3}) \wedge \\
(\neg q_{i2} \vee \neg q_{i4}) \wedge \\
(\neg q_{i3} \vee \neg q_{i4}), \text{ where } 1 \leq i \leq 4
\end{array}
$$

- Exactly one queen in each column:

$$
\begin{array}{l}
(q_{1j} \vee q_{2j} \vee q_{3j} \vee q_{4j}) \wedge \\
(\neg q_{1j} \vee \neg q_{2j}) \wedge \\
(\neg q_{1j} \vee \neg q_{3j}) \wedge \\
(\neg q_{1j} \vee \neg q_{4j}) \wedge \\
(\neg q_{2j} \vee \neg q_{3j}) \wedge \\
(\neg q_{2j} \vee \neg q_{4j}) \wedge \\
(\neg q_{3j} \vee \neg q_{4j}), \text{ where } 1 \leq j \leq 4
\end{array}
$$

- At most one queen per diagonal:

$$
\begin{array}{l}
(\neg q_{11} \vee \neg q_{22}) \wedge \\
(\neg q_{11} \vee \neg q_{33}) \wedge \\
(\neg q_{11} \vee \neg q_{44}) \wedge \\
(\neg q_{22} \vee \neg q_{33}) \wedge \\
(\neg q_{22} \vee \neg q_{44}) \wedge \\
(\neg q_{33} \vee \neg q_{44})
\end{array}
\qquad
\begin{array}{l}
(\neg q_{12} \vee \neg q_{23}) \wedge \\
(\neg q_{12} \vee \neg q_{34}) \wedge \\
(\neg q_{23} \vee \neg q_{34}) \wedge
\end{array}
\qquad
(\neg q_{13} \vee \neg q_{24})
$$

$$
\begin{array}{l}
(\neg q_{21} \vee \neg q_{32}) \wedge \\
(\neg q_{21} \vee \neg q_{43}) \wedge \\
(\neg q_{32} \vee \neg q_{43}) \wedge
\end{array}
\qquad
(\neg q_{31} \vee \neg q_{42})
$$

$$
\begin{array}{l}
(\neg q_{14} \vee \neg q_{23}) \wedge \\
(\neg q_{14} \vee \neg q_{32}) \wedge \\
(\neg q_{14} \vee \neg q_{41}) \wedge \\
(\neg q_{23} \vee \neg q_{32}) \wedge \\
(\neg q_{23} \vee \neg q_{41}) \wedge \\
(\neg q_{32} \vee \neg q_{41})
\end{array}
\qquad
\begin{array}{l}
(\neg q_{24} \vee \neg q_{33}) \wedge \\
(\neg q_{24} \vee \neg q_{42}) \wedge \\
(\neg q_{33} \vee \neg q_{42}) \wedge
\end{array}
\qquad
(\neg q_{34} \vee \neg q_{43})
$$

$$
\begin{array}{l}
(\neg q_{13} \vee \neg q_{22}) \wedge \\
(\neg q_{13} \vee \neg q_{31}) \wedge \\
(\neg q_{22} \vee \neg q_{31}) \wedge
\end{array}
\qquad
(\neg q_{12} \vee \neg q_{21})
$$

1.5 Formal Proofs in Propositional Natural Deduction

So far we have introduced the notion of a propositional formula which is one that can only be true or false, and that is constructed using the propositional connectives: negation, conjunction, disjunction, and implication. However, we have not introduced a means of reasoning about arguments made up of propositional formulas, because, after all, we did define the notion of a logic to be one that supports reasoning about truth. Suppose we are given a list of propositional formulas to assume, and a conclusion. Then how do we reason using these assumptions that the conclusion is a tautology? Truth tables might be a possible solution, but suppose there are n propositional variables in our set of propositional formulas, then the truth table has 2^n rows, and so can be very large. We need a better way to do our reasoning.

In this subsection we introduce a set of rules called **natural deduction** for propositional logic[2]. It consists of several rules for constructing formulas using the propositional formulas, and rules for deconstructing formulas with a propositional connective. These rules can then be used to write structure arguments that begin with a list of assumptions and reach a desired conclusion in a consistent way. An argument of this type is called a **proof**, and the only time in these proofs one can conclude a formula to be true, is if it is either an assumption, or it is the result of applying one of the natural deduction rules.

1.5.1 Sequents

Every argument has a conclusion, and it is usually the first thing one states before giving an argument for its truth. For example, in mathematics the conclusions we state are usually called theorems, lemmas, or corollaries.

In propositional logic we need a way to state our desired result, and this comes in the form of a **sequent**. A logical sequent has the form:

$$P_1, \ldots, P_i \vdash Q$$

where the $P_1, \ldots, P_i$ are a list of arbitrary propositional formulas called the assumptions, hypotheses, or premises and Q is our desired conclusion. We will call these assumptions, premises from here on. An example of a logical sequent is $p, q \vdash p \wedge q$ which states that assuming that both p and q are true, then their conjunction is true.

One way of reading a logical sequent is as an implication, and so we often read a logical sequent as the hypotheses imply the conclusion. We call a sequent **valid** if and only if one has given a proof of the sequent conclusion starting with the sequents hypotheses.

1.5.2 The Rules of Natural Deduction

To construct formal arguments of a given logical sequent in a consistent way we need several guiding principles called the rules of natural deduction. A rule will be in the form of an **inference rule**.

An inference rule has the following form:

$$\frac{P_1 \qquad \cdots \qquad P_i}{Q}\ \textsc{Name}$$

They are read from the top down as an if-then statemant. That is, we read the above rule as "if $P_1, \cdots, P_i$ are true, then Q is true." Each rule is given a unique name so that is can be referenced within a proof. We often say that $P_1, \cdots, P_i$ infer (or deduce) Q.

[2]This chapter is based on the book "Logic in Computer Science: Modeling and Reasoning about Systems."

Note that from here on we will consider the propositional variables like p and q as variables that can only be replaced with $True$ or $False$, and nothing else. Then we will use P and Q to stand for arbitrary propositional formulas.

The rules for natural deduction come in two flavors called **introduction** and **elimination** rules. The former allow one to introduce a logical connective while the latter allow one to eliminate or deconstruct a logical connective into its constitute parts.

Conjunction Introduction. The first rule of natural deduction we give is the introduction rule conjunction. It is defined as follows:

$$\frac{P \qquad Q}{P \wedge Q} \wedge_i$$

This rule has two premises P and Q, and when we suppose these are true we are allowed to conclude $P \wedge Q$. Consider our previous example sequent $p, q \vdash p \wedge q$. We can prove the validity of this sequent by using the above rule:

1.	p	premise
2.	q	premise
3.	$p \wedge q$	$\wedge_i, 1, 2$

We use the above rule by setting $P = p$ and $Q = q$, and then the rule states that we know $P \wedge Q = p \wedge q$ is true.

The proofs we will be constructing are called **formal proofs** and they are all of the form of the previous example. Each line must begin with a number, then to the right of the number is the formula we are claiming is true, and to the right of that is the reason we are allowed to conclude it is true. If we conclude a formula in a proof is true using one of the rules of natural deduction, then we put the name of the rule and the line numbers the formulas occur earlier in the proof that are being used as the premises of the rule as the reason. Finally, the last line of the proof must be the conclusion stated in the sequent we are proving. Therefore, a formal proof of the sequent $P_1, \cdots, P_i \vdash Q$ has the form:

1.	P_1	premise
$\vdots$	$\vdots$	$\vdots$
$i.$	P_i	premise
$\vdots$	$\vdots$	$\vdots$
$n.$	Q	reason

where in between the assumptions and the conclusion there is some logical reasoning using the rules of natural deduction.

Conjunction Elimination. The previous rule reveals how to take two formulas that we have previously argued are true and introduce their conjunction. However, suppose we already have conjunction of two formulas, then by the truth table of the conjunction we know its conjuncts are true, and so we should be able to conclude them. This is the role of the elimination rules for conjuncts.

The elimination rules for conjunction are defined as follows:

$$\frac{P \wedge Q}{P} \wedge_{e_1} \qquad\qquad \frac{P \wedge Q}{Q} \wedge_{e_2}$$

We can use the introduction and elimination rules to prove the validity of the sequent $p \wedge (q \wedge r) \vdash (p \wedge q) \wedge r$:

1.	$p \wedge (q \wedge r)$	premise
2.	p	$\wedge_{e_1}, 1$
3.	$q \wedge r$	$\wedge_{e_2}, 1$
4.	q	$\wedge_{e_1}, 3$
5.	r	$\wedge_{e_2}, 3$
6.	$p \wedge q$	$\wedge_i, 2, 4$
7.	$(p \wedge q) \wedge r$	$\wedge_i, 6, 5$

Disjunction Introduction. As we saw by inspecting the truth table for disjunction if one of the disjuncts are true, then disjunction is true. We use this fact to define the introduction rules for disjunction:

$$\frac{P}{P \vee Q} \vee_{i_1} \qquad\qquad \frac{Q}{P \vee Q} \vee_{i_2}$$

As one can see these rules capture the idea that we may introduce a disjunction if we know at least one of the conjuncts are true. Consider the sequent $p \wedge q \vdash p \vee q$ we can prove the validity of it using the previous introduction rules:

1.	$p \wedge q$	premise
2.	p	$\wedge_{e_1}, 1$
4.	$p \vee q$	$\vee_{i_1}, 2$

There is a second proof using the second introduction rule for disjunction:

1.	$p \wedge q$	premise
2.	q	$\wedge_{e_2}, 1$
4.	$p \vee q$	$\vee_{i_2}, 2$

Disjunction Elimination. So far the introduction and elimination rules we have seen have been very straightforward and are easily seen to be consequences of their respective truth tables. Disjunction elimination, however, does not easily fit this mold, and is the first of the rules that take sometime to understand.

The disjunction elimination rules is as follows:

$$\frac{P \vee Q \qquad \boxed{\begin{array}{c} P \\ \vdots \\ Z \end{array}} \qquad \boxed{\begin{array}{c} Q \\ \vdots \\ Z \end{array}}}{Z} \vee_e$$

The previous rule states that if we know $P \vee Q$, then to eliminate the disjunction, we must first take P as an assumption, and then prove that Z holds, where Z can be any formula we can prove using what we already know, finally, we must also take Q as an assumption, and prove Z again. Note that these two proofs are independent of each other. That is, we are not allowed to use P in Q's proof, and vice versa, unless we already know them to be true before trying eliminate the disjunction. Also, note that P and Q need not be premises, and so we will call these assumptions to highlight the difference.

As an example using the previous rule consider the proof of the validity of the sequent $p \vee q \vdash q \vee p$:

1.	$p \vee q$	premise
2.	p	assumption
3.	$q \vee p$	$\vee_{i_2}, 2$
4.	q	assumption
5.	$q \vee p$	$\vee_{i_1}, 4$
6.	$q \vee p$	$\vee_e, 1, 2\text{–}3, 4\text{–}5$

The elimination rule for disjunction have three premises: i. the disjunction we are eliminating $P \vee Q$, ii. a proof of Z assuming P, and iii. a proof of Z assuming Q. In a formal proof we will use boxes to group the

two proofs of Z. These boxes are called **scope boxes** and the first line of any scope box is a new assumption, and then can consist of any number of lines after the first. In the case of disjunction elimination the last line of a scope box must be Z. When we introduce a new assumption we say that we are **opening** a new scope box, and then when we reach our desired conclusion we say that we are **closing** the scope box. When we close a scope box we say that we are **discharging** the assumption that opened the scope box.

If a proof eliminates a disjunction, then the proof must contain two scope boxes where the first line of the first scope box is the assumption of the first disjunct, and the first line of the second scope box is the second disjunct. Finally, the last line of both scope boxes must be the same formula. The role of a scope box is to keep track of where we are using our new assumption. This is a lot like the scope of a variable in programming. When we define a method in C# the variables we declare inside that method are local to the method, and cannot be accessed from, say, some other method. Thus, we say that these variables are locally scoped. The situation is similar in logic, when we introduce a new assumption we must keep track of its scope. Then once an assumption is not longer in scope, then it can no longer be used. Thus, the formulas inside a closed scope box are not useable outside of the box. For example, the following is an incorrect proof:

1.	$p \vee q$	premise
2.	p	assumption
3.	$p \vee (q \wedge r)$	$\vee_{i_1}, 2$
4.	q	assumption
5.	$p \vee (q \wedge r)$	$\vee_{i_1}, 2$
6.	$p \vee (q \wedge r)$	$\vee_{i_1}, 2$

Notice that the 5 line of the above proof uses the assumption inside the first scope box after it has been closed.

Remark. The line just after the second scope box must be again Z, and the reason on this line must be of the form $\vee_e, l, n\text{–}m, j\text{––}k$ where l is the line number of the disjunction being eliminated, n is the line number of the line that opens the first scope box, m is the line number of the line that closes the first scope box, j is the line number of the line that opens the second scope box, and k is the line number of the line that closes the second scope box.

A second example is the proof of the validity of the sequent $(p \vee q) \vee r \vdash p \vee (q \vee r)$:

1.	$(p \vee q) \vee r$	premise
2.	$p \vee q$	assumption
3.	p	assumption
4.	$p \vee (q \vee r)$	$\vee_{i_1}$
5.	q	assumption
6.	$q \vee r$	$\vee_{i_1}, 5$
7.	$p \vee (q \vee r)$	$\vee_{i_2}, 6$
8.	$p \vee (q \vee r)$	$\vee_e, 2, 3\text{–}4, 5\text{–}7$
9.	r	assumption
10.	$q \vee r$	$\vee_{i_2}, 9$
11.	$p \vee (q \vee r)$	$\vee_{i_2}, 10$
12.	$p \vee (q \vee r)$	$\vee_e, 1, 2\text{–}7, 9\text{–}11$

Implication Introduction. Introducing an implication is similar to eliminating a disjunction. We start with an assumption, P, and if we can obtain some conclusion,Q, then we can introduce the implication $P \Rightarrow Q$. We capture this intuition in the following rule:

$$\frac{\boxed{\begin{array}{c} P \\ \vdots \\ Q \end{array}}}{P \Rightarrow Q} \Rightarrow_i$$

This rule looks a lot like the second and third premises of the rule for disjunction elimination. In fact, in our proofs we will also use scope boxes to keep track of where the hypothesis P is inplay. Thus, when we introduce our assumption P we must open a new scope box, and then once we conclude Q we close the scope box, and then on the next line we must have $P \Rightarrow Q$ discharging P.

Remark. The reason on the line after closing the scope box that contains $P \Rightarrow Q$ must be $\Rightarrow_i, n$–m where n is the line number that opened the scope box, and m is the line number that closes the scope box.

The following is a proof of the validity of the sequent $\cdot \vdash (p \wedge q) \Rightarrow q$:

1.	$p \wedge q$	assumption
2.	q	$\wedge_{e_2}, 1$
3.	$(p \wedge q) \Rightarrow q$	$\Rightarrow_i, 1$–2

Compare the previous proof with that of the following. It shows the validity of the sequent $p \wedge q \vdash q$:

1.	$p \wedge q$	premise
2.	q	$\wedge_{e_2}, 1$
3.	$(p \wedge q) \Rightarrow q$	$\Rightarrow_i, 1$–2

The only difference is that the scope box has been removed. This hits on an important note. The only time we need to open a scope box is when we add an assumption. However, we are allowed to also use a premise as the assumption to create an implication. In this case, a scope box is not needed. In fact, we can could actually prove the following result:

Theorem 11 (Deduction Theorem)**.** *The sequent $P_1, \ldots, P_n \vdash Q$ holds if and only if $\cdot \vdash P_1 \Rightarrow (P_2 \Rightarrow (\cdots (P_{n-1} \Rightarrow (P_n \Rightarrow Q)) \cdots))$ holds.*

This is an important result, but its proof is outside the scope of this course. This theorem implies that the only time we need to open a new scope box, when using the introduction rule for implication, is if the assumption opening the box is new, and does not occur already in our proof, otherwise, no scope box is needed.

Implication Elimination. The following is the rule for implication elimination:

$$\frac{P \Rightarrow Q \qquad P}{Q} \Rightarrow_e$$

This rule is also known as modus ponens. It simply states that if we have already concluded that $P \Rightarrow Q$ holds, and we also know P holds, then Q also holds.

The following is a proof of the validity of the sequent $(p \wedge q) \Rightarrow r \vdash p \Rightarrow (q \Rightarrow r)$:

1.	$(p \wedge q) \Rightarrow r$	premise
2.	p	assumption
3.	q	assumption
4.	$p \wedge q$	$\wedge_i, 3, 4$
5.	r	$\Rightarrow_e, 1, 4$
6.	$q \Rightarrow r$	$\Rightarrow_i, 3$–5
7.	$p \Rightarrow (q \Rightarrow r)$	$\Rightarrow_i, 2$–6

We can actually prove the validity of the converse of the previous sequent:

1.	$p \Rightarrow (q \Rightarrow r)$	premise
2.	$p \wedge q$	assumption
3.	p	$\wedge_{e_1}, 2$
4.	$q \Rightarrow r$	$\Rightarrow_e, 1, 3$
5.	q	$\wedge_{e_2}, 2$
6.	r	$\Rightarrow_e, 4, 5$
7.	$(p \wedge q) \Rightarrow r$	$\Rightarrow_i, 2$–6

Absurdity Elimination. There are three rules for dealing with negation. The first says that if you can prove $False$, then you can prove anything. The intuition behind this is that if one has proven $False$, then one has reached the ultimate contradiction, and should be allowed to conclude anything we wish. This intuition is captured in the following natural deduction:

$$\frac{False}{P}\ False_e$$

The previous rule is called the elimination rule for absurdity or $False$, note that there is not introduction rule for $False$.

Negation Introduction. To introduce a negation we must start with an assumption, and reach a contradiction by proving $False$. If one can accomplish this, then the original assumption must have been false to begin with. We capture this intuition by the following rule:

$$\frac{\begin{array}{|c|}\hline P \\ \vdots \\ False \\ \hline\end{array}}{\neg P}\ \neg_i$$

Remark. This rule also requires the opening of a new scope box starting with P, and this box is closed when $False$ has been proven.

Negation Elimination. To eliminate a negation one must prove $\neg P$ as well as P, and if this can be done, then one may conclude $False$, because we have reached a contradiction. Thus, we have arrived at the following rule:

$$\frac{\neg P \qquad P}{False}\ \neg_e$$

The following example shows the validity of the sequent $\neg p \vee q \vdash p \Rightarrow q$:

1.	$\neg p \vee q$	premise
2.	$\neg p$	assumption
3.	p	assumption
4.	$False$	$\neg_e, 2, 3$
5.	q	$False_e, 4$
6.	$p \Rightarrow q$	$\Rightarrow_i, 3$–5
7.	q	assumption
8.	p	assumption
9.	q	copy 7
10.	$p \Rightarrow q$	$\Rightarrow_i, 8$–9
11.	$p \Rightarrow q$	$\vee_e, 1, 2$–$6, 7$–10

We show that the sequent $p \Rightarrow q, p \Rightarrow \neg q \vdash \neg p$ is valid:

1.	$p \Rightarrow q$	premise
2.	$p \Rightarrow \neg q$	premise
3.	p	assumption
4.	q	$\Rightarrow_e, 1, 3$
5.	$\neg q$	$\Rightarrow_e, 2, 3$
6.	$False$	$\neg_e, 4, 5$
7.	$\neg p$	$\neg_i, 3$–6

Now we show that the sequent $p \Rightarrow (q \Rightarrow r), p, \neg r \vdash \neg q$ is valid.

1.	$p \Rightarrow (q \Rightarrow r)$	premise
2.	p	premise
3.	$\neg r$	premise
4.	$q \Rightarrow r$	$\Rightarrow_e, 1, 2$
5.	q	assumption
6.	r	$\Rightarrow_e, 4, 5$
7.	$False$	$\neg_e, 3, 6$
8.	$\neg q$	$\neg_i, 5 - 7$

Finally, we show that the sequent $(p \wedge \neg q) \Rightarrow r, \neg r, p \vdash \neg\neg q$ is valid.

1.	$(p \wedge \neg q) \Rightarrow r$	premise
2.	$\neg r$	premise
3.	p	premise
4.	$\neg q$	assumption
5.	$p \wedge \neg q$	$\wedge_i, 3, 4$
6.	r	$\Rightarrow_e, 5, 1$
7.	$False$	$\neg_e, 2, 6$
8.	$\neg\neg q$	$\neg_i, 4 - 7$

Law of Double Negation. The law of double negation says that any propositional formula P is equivalent to its double negation $\neg\neg P$. We will express this law by two rules, LDN_i and LDN_e, the former can actually be proven in terms of the basic rules we have been introducing – see below – and so we only introduce the latter here:

$$\frac{\neg\neg P}{P}\ \mathsf{LDN}_e$$

This is an extremely powerful rule. In fact, we can use it to prove the law-of-excluded-middle (LEM) $\cdot \vdash P \vee \neg P$:

1.	$\neg(p \vee \neg p)$	assumption
2.	p	assumption
3.	$p \vee \neg p$	$\vee_{i_1}, 2$
4.	$False$	$\neg_e, 1, 3$
5.	$\neg p$	$\neg_i, 2$–4
6.	$p \vee \neg p$	$\vee_{i_2}, 5$
7.	$False$	$\neg_e, 1, 6$
8.	$\neg\neg(p \vee \neg p)$	$\neg_i, 1$–7
9.	$p \vee \neg p$	$\mathsf{LND}_e, 8$

Propositional Equivalences. At this point we have fully introduced propositional logic and natural deduction. In this subsection we introduce the concept of propositional equivalences.

Definition 12. *Two propositional formulas, P and Q, are **propositional equivalent** if and only if one can prove that $P \vdash Q$ and $Q \vdash P$. We denote a propositional equivalence by $P \equiv Q$.*

Note that P and Q are arbitrary formulas, and not just p and q. In the rest of this subsection we give several convenient and important propositional equivalencies.

Derived Rules. First, we make a remark about what are called derived rules. These are inference rules that are added to our list of useable rules within formal natural deduction proofs. However, a derived rule must be proven to hold before it can be used. This is similar to a theorem or lemma of mathematics. Once we prove a theorem or lemma they can be used in other proofs.

If a formula P is a tautology, then we can add the rule:

$$\frac{}{P}\ \text{AX}$$

To our list of usable rules. An example is the law-of-excluded-middle, $P \vee \neg P$. We showed using truth tables that this is a tautology, and thus, we obtain the rule:

$$\frac{}{P \vee \neg P}\ \text{LEM}$$

We can now use this rule whenever we want in a proof. We will use this rule below in the proof of the equivalence called the law of double negation.

If a the sequent $P_1, \cdots, P_n \vdash Q$ is valid, then we may add the rule:

$$\frac{P_1 \ \cdots \ P_n}{Q}\ \text{newRule}$$

Thus, propositional equivalences yield two different rules.

Law of Double Negation Introduction: We can prove the converse of LDN, $P \vdash \neg\neg P$, as follows:

1.	P	premise
2.	$\neg P$	assumption
3.	$False$	$\neg_e, 1, 2$
4.	$\neg\neg P$	$\neg_i, 2$–3

Thus, we now have the following derived rule:

$$\frac{P}{\neg\neg P}\ \mathsf{LDN}_i$$

Proof by Contradiction. This is perhaps one of the most used proof techniques found throughout both mathematics and computer science. One obtains a *contradiction* if one is able to prove both P and $\neg P$ for some formula P, and hence, using the rule $\neg_e$ we may prove False. Proof by contradiction capitalizes on this idea. Consider the following rule:

$$\frac{\boxed{\begin{array}{c}\neg P \\ \vdots \\ \mathsf{False}\end{array}}}{P}\ \mathsf{PBC}$$

First, compare this to the rule $\neg_i$, and note the difference. This rule says, to prove P we can first assume P is false, $\neg P$, and then obtain a contradiction, by proving False. We will make heavy use of this rule throughout the semester.

We derive this rule by proving the validity of the sequent $(\neg P \Rightarrow \mathsf{False}) \vdash P$:

1.	$\neg P \Rightarrow \mathsf{False}$	premise
2.	$P \vee \neg P$	LEM
3.	P	assumption
4.	$\neg P$	assumption
5.	False	$\Rightarrow_e, 1, 4$
6.	P	$\mathsf{False}_e, 5$
7.	P	$\vee_e, 2, 3\text{–}3, 4\text{–}6$

De Morgan's Laws. Augusta De Morgan was a 19th century British mathematician who proved that the conjunction is dual to disjunction and vice versa. Two propositional connectives are called dual if given any formula P interms of the first connective, the formula $\neg P$ can be translated into a propositional equivalent formula interms of the second and negation, and viceversa. The following lists De Morgan's laws or De Morgan's dualities:

$$\begin{array}{lrl} \text{(D1)} & \neg(P \wedge Q) & \equiv \quad \neg P \vee \neg Q \\ \text{(D2)} & \neg(P \vee Q) & \equiv \quad \neg P \wedge \neg Q \end{array}$$

We now turn to their proofs. Recall to prove $D1$ we must first prove that $\neg(P \wedge Q) \vdash \neg P \vee \neg Q$:

1.	$\neg(P \wedge Q)$	premise
2.	$\neg(\neg P \vee \neg Q)$	assumption
3.	$\neg P$	assumption
4.	$\neg P \vee \neg Q$	$\vee_i, 3$
5.	$False$	$\neg_e, 2, 4$
6.	P	PBC, 3–5
7.	$\neg Q$	assumption
8.	$\neg P \vee \neg Q$	$\vee_i, 3$
9.	$False$	$\neg_e, 2, 4$
10.	Q	PBC, 7–9
11.	$P \wedge Q$	$\wedge_i, 6, 10$
12.	$Fasle$	$\neg_e, 1, 11$
13.	$\neg P \vee \neg Q$	PBC, 2–12

Now we prove the inverse of D1 is valid $\neg P \vee \neg Q \vdash \neg(P \wedge Q)$:

1.	$\neg P \vee \neg Q$	premise
2.	$\neg\neg(P \wedge Q)$	assumption
3.	$P \wedge Q$	LDN, 2
4.	P	$\wedge_{e_1}, 3$
5.	Q	$\wedge_{e_2}, 3$
6.	$\neg P$	assumption
7.	$\neg Q$	assumption
8.	$False$	$\neg_e, 5, 7$
9.	$\neg P$	$False_e, 8$
10.	$\neg P$	$\vee_e, 1, 6, 7\text{–}9$
11.	$False$	$\neg_e, 4, 10$
12.	$\neg(P \wedge Q)$	PBC, 2–13

Moving onto D2 we show that $\neg(P \vee Q) \vdash \neg P \wedge \neg Q$ is valid:

1.		$\neg(P \vee Q)$	premise
2.	$\boxed{}$	P	assumption
3.	$\boxed{}$	$P \vee Q$	$\vee_{i_1}, 2$
4.	$\boxed{}$	$False$	$\neg_e, 1, 3$
5.		$\neg P$	$\neg_i, 2$–4
6.	$\boxed{}$	Q	assumption
7.	$\boxed{}$	$P \vee Q$	$\vee_{i_2}, 6$
8.	$\boxed{}$	$False$	$\neg_e, 1, 3$
9.		$\neg Q$	$\neg_i, 6$–8
10.		$\neg P \wedge \neg Q$	$\wedge_i, 5, 9$

Finally, we show that $\neg P \wedge \neg Q \vdash \neg(P \vee Q)$ is valid:

1.		$\neg P \wedge \neg Q$	premise
2.		$\neg P$	$\wedge_{e_1}, 1$
3.		$\neg Q$	$\wedge_{e_2}, 1$
4.	$\boxed{}$	$\neg\neg(P \vee Q)$	assumption
5.	$\boxed{}$	$P \vee Q$	LDN, 4
6.	$\boxed{}\boxed{}$	P	assumption
7.	$\boxed{}\boxed{}$	Q	assumption
8.	$\boxed{}\boxed{}$	$False$	$\neg_e, 3, 7$
9.	$\boxed{}\boxed{}$	P	$False_e, 8$
10.	$\boxed{}$	P	$\vee_e, 5, 6, 7$–9
11.	$\boxed{}$	$False$	$\neg_e, 2, 10$
12.		$\neg(P \vee Q)$	PBC, 4–11

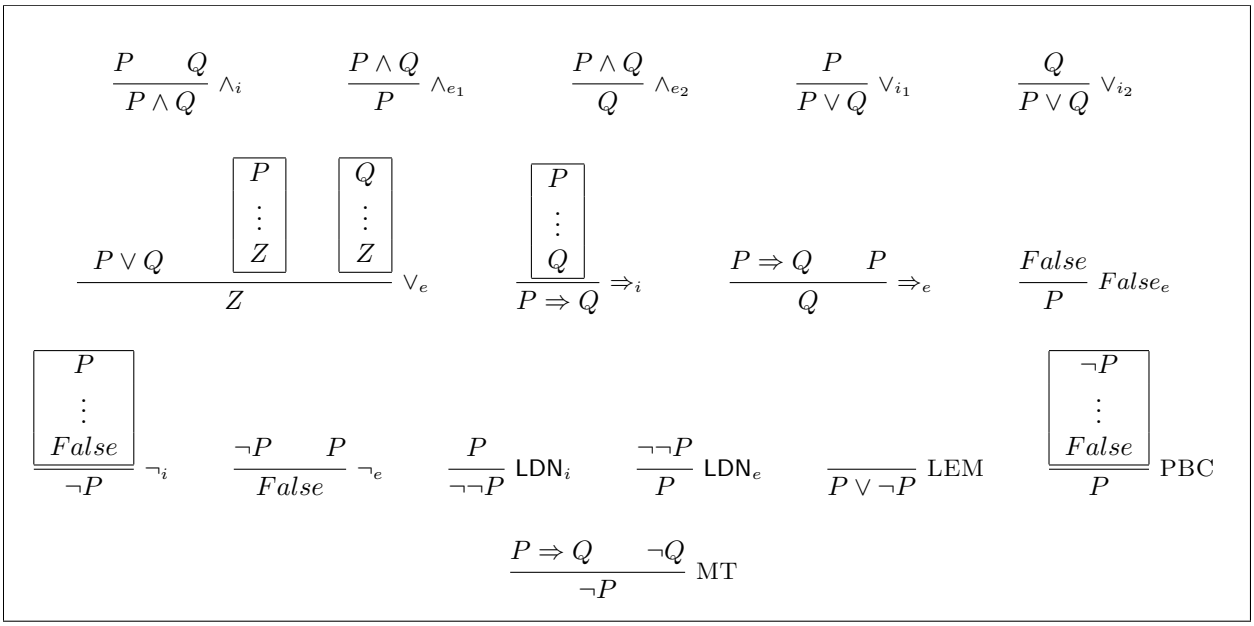
As yet another example, let's try and prove the following logical equivalence:

$$P \Rightarrow Q \equiv \neg P \vee Q$$

We have already proven above that $\neg P \vee Q \vdash P \Rightarrow Q$ for propositional variables, but the proof does not change for arbitrary P and Q, and hence, we only need to show $P \Rightarrow Q \vdash \neg P \vee Q$:

1.		$P \Rightarrow Q$	premise
2.		$P \vee \neg P$	LEM
3.	$\boxed{}$	P	assumption
4.	$\boxed{}$	Q	$\Rightarrow_e, 1, 3$
5.	$\boxed{}$	$\neg P \vee Q$	$\vee_{i_2}, 4$
6.	$\boxed{}$	$\neg P$	assumption
7.	$\boxed{}$	$\neg P \vee Q$	$\vee_{i_1}, 6$
8.		$\neg P \vee Q$	$\vee_e, 2, 3$–$5, 6$–7

The Basic and Derived Rules of Natural Deduction:



2 Logic with (First-Order) Quantifiers

Note: Read chapter 1.4 - 1.5.

Up until now we have been studying proportional logic and formal proof. However, propositional logic is not expressive enough to prove the validity of many of the statements we encounter in mathematics, natural language, or computer science. Consider the following statement:

> "Every communication channel between computers using 512 bit encryption is insecure."

Suppose that we know of a communication channel between two computers on the AU network. Then there are no rules of propositional logic for proving that the channel between them is insecure using the truth of the above statement.

Similarly, propositional logic cannot be used to express statements about the existence of objects. For example, suppose we know the following:

> "A hacker is conducting a person-in-the-middle attack on the communication channel between severs CS-Gödel and CS-Logic."

Then we cannot use this to prove the validity of the statement:

> "There exists an insecure channel between two servers on the AU network."

Furthermore, statements that express properties of a range of objects cannot be expressed in propositional logic. For example, consider the usual statement from mathematics:

$$x > 1 \Rightarrow x^2 > 1$$

Here the variable x is assumed to range over all real numbers. This is called a mathematical variable. Then if we have an actual number, like 2, we can use the previous statement to conclude that $2^2 = 4 > 1$. However, this is impossible in propositional logic.

We are now going to extend the language of propositional logic with the necessary tools to be able to prove the validity of the sort of statements we introduced above.

2.1 Predicates

A **predicate** is a formula containing one or more *data variables*. Data variables range over different types of data, for example, real numbers, integers, people, cars, houses, etc. They differ from propositional variables in that the latter range over only true or false. We will denote data variables by x, y, and z, and sometimes we simply call them variables, but it should be discernible from context which type of variable is which. Now if we simply say, x is a data variable, then it ranges over all data, but we can use formulas to cut this range down.

Again, predicates are formulas that contain one or more data variables, for example, a predicate in English is "The computer N is functional properly", and we can see that N corresponds to a data variable. One very important property of predicates is that when we replace all of the data variables by actual pieces of data, then we obtain a formula. Give the example predicate above if we replace N with it's IP address, 252.252.1.1, then we see that "The computer 252.252.1.1 is functional properly" is either true or false, and hence, is a formula, but when N is used, then we are not able to determine if the statement is true or false, because N could be any computer at all.

In general predicates are defined using definitions of the following form:

$$P(x_1, \ldots, x_i) := D$$

where P is the name of the predicate, and $x_1, \ldots, x_i$ are its input data variables, and D is some formula using the input variables. Think of $P(x_1, \ldots, x_i)$ as a method in C#, and think of the variables of a predicate as the subjects of the predicate they belong to. The name can be anything we wish, but we will often stick to using P, Q, or R.

It might be helpful to consider some example predicates:

- The statement: If x is a real number, then $1 \leq x < 10$, corresponds to the predicate $Q(x)$ below:

  $$\begin{aligned} &P(y) := y \text{ is a real number} \\ &L(z) := 1 \leq z < 10 \\ &Q(x) := P(x) \Rightarrow L(x) \end{aligned}$$

  Notice that we used the predicates $P(y)$ and $L(z)$ to define $Q(x)$.

- The statement: The computer N is functioning properly, corresponds to the predicate $Q(x)$ below:

  $$\begin{aligned} &C(N) := N \text{ is a computer} \\ &F(y) := y \text{ is functioning properly} \\ &Q(x) := C(x) \wedge F(x) \end{aligned}$$

- The statement: If x and y are natural numbers, then $x + y = y + x$, corresponds to the predicate $Q(x, y)$ below:

  $$\begin{aligned} &\mathsf{Nat}(n) := n \text{ is a natural number} \\ &P(x, y) := (x + y = y + x) \\ &Q(x, y) := (\mathsf{Nat}(x) \wedge \mathsf{Nat}(y)) \Rightarrow P(x, y) \end{aligned}$$

- The statement: $(x > 22) \Rightarrow (y < 11)$ where x and y are complex numbers, corresponds to the predicate $Q(x, y)$ below:

  $$\begin{aligned} &\mathsf{GT}(x, y) := x > y \\ &\mathsf{LT}(x, y) := x < y \\ &\mathsf{Complex}(n) := n \text{ is a complex number} \\ &Q(x, y) := (\mathsf{Complex}(x) \wedge \mathsf{Complex}(y)) \Rightarrow (\mathsf{GT}(x, 22) \Rightarrow \mathsf{LT}(y, 11)) \end{aligned}$$

  Notice that in the one above we actually plug in 22 and 11 into GT and LT respectively, and this really shows that x and y are data variables, and we can fill them with actual data.

- The statement: $\neg(x = 0) \Rightarrow y =$ ``Ok'' where x is an integer and y is a string, corresponds to $Q(x, y)$ below:

  $$\begin{aligned} &\mathsf{Eq}(x, y) := (x = y) \\ &\mathsf{Str}(s) := s \text{ is a string} \\ &\mathsf{Int}(i) := i \text{ is an integer} \\ &Q(x, y) := (\mathsf{Int}(x) \wedge \mathsf{Str}(y)) \Rightarrow (\neg \mathsf{Eq}(x, 0) \Rightarrow \mathsf{Eq}(y, \texttt{''Ok''})) \end{aligned}$$

Domains. Consider the predicate $P(x) := x \leq 10$, what is x? That is, what can replace x? Up front, we do not know this, and we have to add a predicate to indicate what x is. So a better predicate would be $P(x) := (x \text{ is an integer}) \Rightarrow x \leq 10$. Think of the predicate "x is an integer" as specifying the "type" of x, and we call this type the *domain* of x. Notice that all of the examples above specify the domain of each variable introduced, and this must always be the case.

2.2 Quantifiers

Moving from proposition to predicates opens up the possibility of adding two more types of formulas to our logic.

Universal Quantification. The first allows us to speak about the truth of a predicate across all possible values. That, is suppose we have a predicate $P(x)$ with one variable called x whose domain is the sequence of natural numbers, then if we can prove that for any natural number n, $P(n)$ (replacing x with the number n) is true, then we say that $P(x)$ is universally true. We have the following definition:

Definition 13. *The **universal quantification** of $P(x)$ is the statement "$P(x)$ for all values of x". We denote this statement by $\forall x.P(x)$, and it is read as "for all x,$P(x)$". The $\forall$ is called the universal quantifier. Finding an element y such that $P(y)$ is false is called finding a **counterexample** of $\forall x.P(x)$ (the element y is the counter example).*

Domains. We call a variables, x, universally quantified, if it is attached to the $\forall$ quantifier. For example, x in $\forall x.P(x)$ is universally quantified. Notice that in $\forall x.P(x)$ the predicate $P(x)$ must specify the domain of x, but when x is universally quantified $P(x)$ must have the following form:

$$P(x) := D(x) \Rightarrow R(x)$$

where $D(x)$ specifies the domain of x. As an example, consider the following:

$$\begin{aligned} D(x) &:= x \text{ is an integer} \\ R(x) &:= x \leq (x+1) \\ P(x) &:= D(x) \Rightarrow R(x) \end{aligned}$$

The formula $\forall x.P(x)$ now correctly specifies the domain of x and declared that it is an integer. Notice that when we are specifying the domain of a universally quantified variable, then we use an implication, but what is wrong with the following:

$$\begin{aligned} D(x) &:= x \text{ is an integer} \\ R(x) &:= x \leq (x+1) \\ P(x) &:= D(x) \wedge R(x) \end{aligned}$$

Now consider $\forall x.P(x)$ is this formula true or false? Well we are allowed to choose any x at all, and so, choose $x = \frac{1}{2}$, then we can ask if $P(\frac{1}{2}) = D(\frac{1}{2}) \wedge R(\frac{1}{2})$ is true, but we can see that it is false, because $\frac{1}{2}$ is not an integer. This is bad, because we really only want to consider integers, and filter out everything else, and so really, we want $P(\frac{1}{2})$ to be true, and not false. Thus, to specify the domain of a universally quantified variable we must use an implication.

Remark. Let $U(x) :=$ "x is a unicorn", $C(y) :=$ "y is a child", and $R(y, x) := y$ will ride x. Is $\forall x.U(x) \Rightarrow \forall y.C(y) \Rightarrow R(y, x)$ true? Since it is impossible for a unicorn to exist we know that there is no x such that $U(x)$ is true, and hence, the formula in question is true vacuously. Remember, that if the domain of the variable x in $\forall x.P(x)$ is **empty**, then it is true, because $P(x)$ must be of the form $D(x) \Rightarrow Q(x)$ for some predicates D and Q where D establishes the domain of x.

Recall from the above definition of universal quantification a formula $\forall x.P(x)$ is false if we can find at least one value v such that $P(v)$ is false. We only need to find one to show that $\forall x.P(x)$ is false.

Another way of viewing an universally quantified formula like $\forall x.P(x)$ is as the conjunction:

$$P(x_1) \wedge P(x_2) \wedge P(x_3) \wedge P(x_4) \wedge \cdots$$

where each x_i is some value.

The truth value of an universally quantified formula depends quite strongly on the domain of the variable being quantified over.

Example 14. *Let $P(x)$ be $x^2 \geq x$. First, what is the truth value of $P(x)$ when x ranges over the sequence of real numbers? It is false, because $((\frac{1}{2})^2 \not\geq \frac{1}{2})$. Note that $x^2 \geq x$ if and only if $x^2 - x \geq 0$ if and only if $x(x-1) \geq 0$. This then implies that $x^2 \geq x$ if and only if $x \leq 0 \vee x \geq 1$. Now to make $\forall x.P(x)$ false, then we must make $\neg(\forall x.P(x))$ true, and thus, if we negate our reasoning above, then we must make $\neg(x^2 \geq x)$ if and only if $\neg(x \leq 0 \vee x \geq 1)$ true. This is equivalent to $\neg(x^2 \geq x)$ if and only if $0 < x \wedge x < 1$ if and only if $0 < x < 1$. Therefore, $\forall x.P(x)$ is false exactly when we choose a real number between zero and one.*

Now consider when x ranges over the integers. Then $\forall x.P(x)$ is true, because of the fact that there are no integers between zero and one.

This example shows that the domain of the variable being universally quantified over matters significantly.

Existential Quantification. Dual to universal quantification is existential quantification. It allows one to assert the existence of a particular object with a certain property. For example, "there exists a logic such that LEM is false." The previous statement asserts that there is some logic in existence that refutes LEM. To prove such a statement is true we must find the an actual logic that does refute LEM, and we can, it is called intuitionistic logic[3].

It is very important to remember that the **only** way to prove an existential quantified statement is true like the one above is to find the actual element in the domain that makes the predicate true. You must go and point it out!

Definition 15. *The **existential quantification** of $P(x)$ is the statement "$P(x)$ for some value of x". We denote this statement by $\exists x.P(x)$, and it is read as "there exists a x,$P(x)$". The $\exists$ is called the existential quantifier. To prove an existential quantified formula, $\exists x.P(x)$, one must find a particular y such that $P(y)$ is true, but to disprove an existentially quantified formula, $\exists x.P(x)$, one must show that $P(y)$ is false for every y. So a negated existentially quantified formula turns into a universally quantified formula.*

Domains. The domain of the variable x in $\exists x.P(x)$ must **always** be specified or the formula means nothing, because we must find a particular object that makes the formula true. Suppose $\exists x.P(x)$, then $P(x)$ must have the following form:

$$P(x) := D(x) \wedge R(x)$$

Here $D(x)$ specifies the domain of x. The domain of an existentially quantified variable is always a conjunction, because we are speaking about a single element of the domain of x, and not doing any filtering. An example might be the following:

$$\begin{aligned} D(x) &:= x \text{ is a student} \\ F(y) &:= y \text{ is a professor} \\ R(x,y) &:= x \text{ is smarter than } y \\ P(x,y) &:= D(x) \wedge F(y) \wedge R(x,y) \end{aligned}$$

The formula $\exists x.\exists y.P(x,y)$ says that there is a student and a professor, such that, the student is smarter than the professor. The predicates P and F are domain predicates, and the conjunction in P forces us to find objects of those domains.

Similarly to universal quantification we can view an existentially quantified formula like $\exists x.P(x)$ is as the disjunction:

$$P(x_1) \vee P(x_2) \vee P(x_3) \vee P(x_4) \vee \cdots$$

2.3 First-Order Logic Example Translations

Translate the following to a first-order formula:

[3] `http://plato.stanford.edu/entries/logic-intuitionistic/#RejTerNonDat`

i. Not all birds can fly. First, list all of the predicates that we need:

$$B(x) := x \text{ is a bid}$$
$$F(x) := x \text{ can fly}$$

Then we can translate the sentence as $\neg(\forall x.B(x) \Rightarrow F(x))$.

ii. It's not the case that all birds can fly can be translated as $\exists x.B(x) \wedge \neg F(x)$.

iii. Every student is younger than some instructor. First, rewrite the formula so that we know what variables are needed:

Every student x is younger than some instructor y

List all of the predicates that we need:

$$S(x) := x \text{ is a student}$$
$$I(x) := x \text{ is an instructor}$$
$$Y(x, y) := x \text{ is younger than } y$$

Then we can translate the sentence as $\forall x.S(x) \Rightarrow (\exists y.I(y) \wedge Y(x, y))$.

iv. No books are gaseous. Dictionaries are books. Therefore, no dictionary is gaseous. The predicates are:

$$B(x) := x \text{ is a book}$$
$$G(x) := x \text{ is gaseous}$$
$$D(x) := x \text{ is a dictionary}$$

The translation is then $(\neg(\exists x.B(x) \wedge G(x)) \wedge (\forall x.D(x) \Rightarrow B(x))) \Rightarrow \neg(\exists x.(D(x) \wedge G(x))$.

v. Every child is younger than their parents. The predicates are:

$$C(x) := x \text{ is a child}$$
$$P(x, y) := x \text{ is the parent of } y$$
$$Y(x, y) := x \text{ is younger than } y$$

The translation is then $\forall x.\forall y.((C(x) \wedge P(y, x)) \Rightarrow Y(x, y))$.

2.4 Formal Proofs in First-Order Logic

Now that we understand the basics of first-order formulas we now move on to formula proofs in first-order logic. But, first we introduce a general schema representing every first-order formula.

2.4.1 Formulas

The schema for formulas in first-order logic is defined as follows:

- (Truth) $True$,
- (Falsity) $False$,
- (Predicates) $P(x_1, \ldots, x_n)$,
- (Conjunction) $P \wedge Q$, where P and Q are formulas,
- (Disjunction) $P \vee Q$, where P and Q are formulas,
- (Implication) $P \Rightarrow Q$, where P and Q are formulas,

- (Negation) $\neg P$, where P is a formula,
- (Universal Quantification) $\forall x.P(x)$, where P is a formula, and
- (Existential Quantification) $\forall x.P(x)$, where P is a formula.

No other expressions are first-order formulas, that is, all first-order formulas are generated using the previous rules. **One simplifying assumption about formulas that we adopt in this course is the assumption that all variables in a formula or proof are uniquely named.**

2.4.2 Rules of Predicate Logic

Next we extend natural deduction with rules for both universal and existential quantification. These new rules will make use of substitution for introducing and eliminating each type of quantifier.

Universal Quantification Introduction. The inference rule for introducing an universal quantification is as follows:

$$\frac{\boxed{\begin{array}{l}\text{d new}\\ \\ \vdots\\ \text{P(d)}\end{array}}}{\forall x.P}\ \forall_i$$

The statement, d new, means that d is a data variable that is not used anywhere else in the proof. The previous rule can be read "assuming we have an arbitrary new piece of data d, if we can prove that $P(d)$ is true, then we are allowed to conclude that $\forall x.P(x)$ holds for any x. This makes sense, because we were able to prove $P(d)$ for any d at all, and hence, our reasoning is completely arbitrary.

Slogan: Proving an universally quantified formula requires one to **reason in the arbitrary.** We are not allowed to know what specific d we have.

Let's see this in action and prove the sequent $\cdot \vdash \forall x.((P)(x) \Rightarrow P(x))$:

1.	y new	MS: $P(y) \Rightarrow P(y)$	
2.		$P(y)$	assumption
2.		$P(y)$	copy, 1
3.		$P(y) \Rightarrow P(y)$	$\Rightarrow_i$, 1-2
4.		$\forall x.(P(x) \Rightarrow P(x))$	$\forall_i$, 1-4

Universal Quantification Elimination. The inference rule for eliminating an universal quantification is as follows:

$$\frac{\forall x.P \qquad d \text{ is some data}}{P(d)}\ \forall_e$$

If we know that $\forall x.P$ is true, then it is true for any piece of data we replace x with. Thus, since we proved above that $\forall x.(P(x) \Rightarrow P(x))$ is true, then we know by the above rule that $(P(x) \Rightarrow P(x))(3) = (P(3) \Rightarrow P(3))$ is true regardless of what P is. In fact, we can substitute any data, t, at all for x and $P(t)$ will be true, because that's exactly what we proved.

Let's consider an example using this rule by proving $P(t), \forall x.(P(x) \Rightarrow \neg Q(x)) \vdash \neg Q(t)$:

1.	$P(t)$	Premise
2.	$\forall x.(P(x) \Rightarrow \neg Q(x))$	Premise
3.	$P(t) \Rightarrow \neg Q(t)$	$\forall_e$, 2
4.	$\neg Q(t)$	$\Rightarrow_e$, 3, 1

At this point we do several examples:

- $\forall x.(P(x) \Rightarrow Q(x)), \forall x.P(x) \vdash \forall x.Q(x)$

1.		$\forall x.(P(x) \Rightarrow Q(x))$	Premise
2.		$\forall x.P(x)$	Premise
3.	y new	MS: $Q(y)$	
4.		$P(y) \Rightarrow Q(y)$	$\forall_e, 1$
5.		$P(y)$	$\forall_e, 2$
6.		$Q(y)$	$\Rightarrow, 4, 5$
7.		$\forall x.Q(x)$	$\forall_i$, 3-6

- $\forall x.\forall y.P(x, y) \vdash \forall u.\forall v.P(u, v)$
- $\forall x.(\neg P(x) \wedge Q(x)) \vdash \forall x.(P(x) \Rightarrow Q(x))$
- $\forall x.(P(x) \wedge Q(x)) \vdash (\forall x.P(x)) \wedge (\forall x.Q(x))$
- $(\forall .P(x)) \vee (\forall x.Q(x)) \vdash \forall x.(P(x) \vee Q(x))$

Existential Introduction. The introduction rule for existential quantification is similar to the elimination rule for universal quantification:

$$\frac{P(d) \qquad d \text{ is some data}}{\exists x.P}\ \exists_i$$

This rule is better understood by reading it bottom up. It says that if we wish to prove $\exists x.P$ is true, then we must go out and find a piece of data, d, such that $P(d)$ is true. For example, to prove that $\exists x.(\mathsf{Real}(x) \wedge x^2 < x)$, we must find an actual real number, r, that makes, $(\mathsf{Real}(x) \wedge x^2 < x)(r) = (\mathsf{Real}(r) \wedge r^2 < r)$ true. This is easily done, by choosing $r = \frac{1}{2}$, and we can easily see that $(\mathsf{Real}(\frac{1}{2}) \wedge (\frac{1}{2})^2 < \frac{1}{2})$ is true.

Existential Elimination. The elimination rule is similar to the elimination rule for disjunction elimination:

$$\frac{\exists x.P \qquad \boxed{\begin{array}{ll} d \text{ new} & P(d) \\ & \vdots \\ & Q \end{array}}}{Q}\ \exists_e$$

This rule is easier to understand reading top to bottom. It says, if we know $\exists x.P$ is true, then we know there is some piece of data, d, making $P(d)$ true. So to eliminate $\exists x.P$ we open a box, and assume $P(d)$ for some data d, and prove Q, and if we can do that, then we can close the box, and get Q.

Data Rule. The data rule is defined as follows:

$$\frac{\boxed{\begin{array}{l} \text{d new} \\ \vdots \\ \text{Q} \end{array}} \qquad \text{Q does not use } d}{Q}\ data$$

Note: We are not allowed to assume anything about d, meaning, d is completely arbitrary and has no domain.

This rule is used when there are no premises nor assumptions that will allow us to obtain data needed to plug into some formula in our proof. We can use the data rule to open a box and give ourselves some data, but the only time we can close this box is if we prove some formula that does not depend on this new piece of data.

Here is an example proof of $\forall x.P(x) \vdash \exists x.P(x)$:

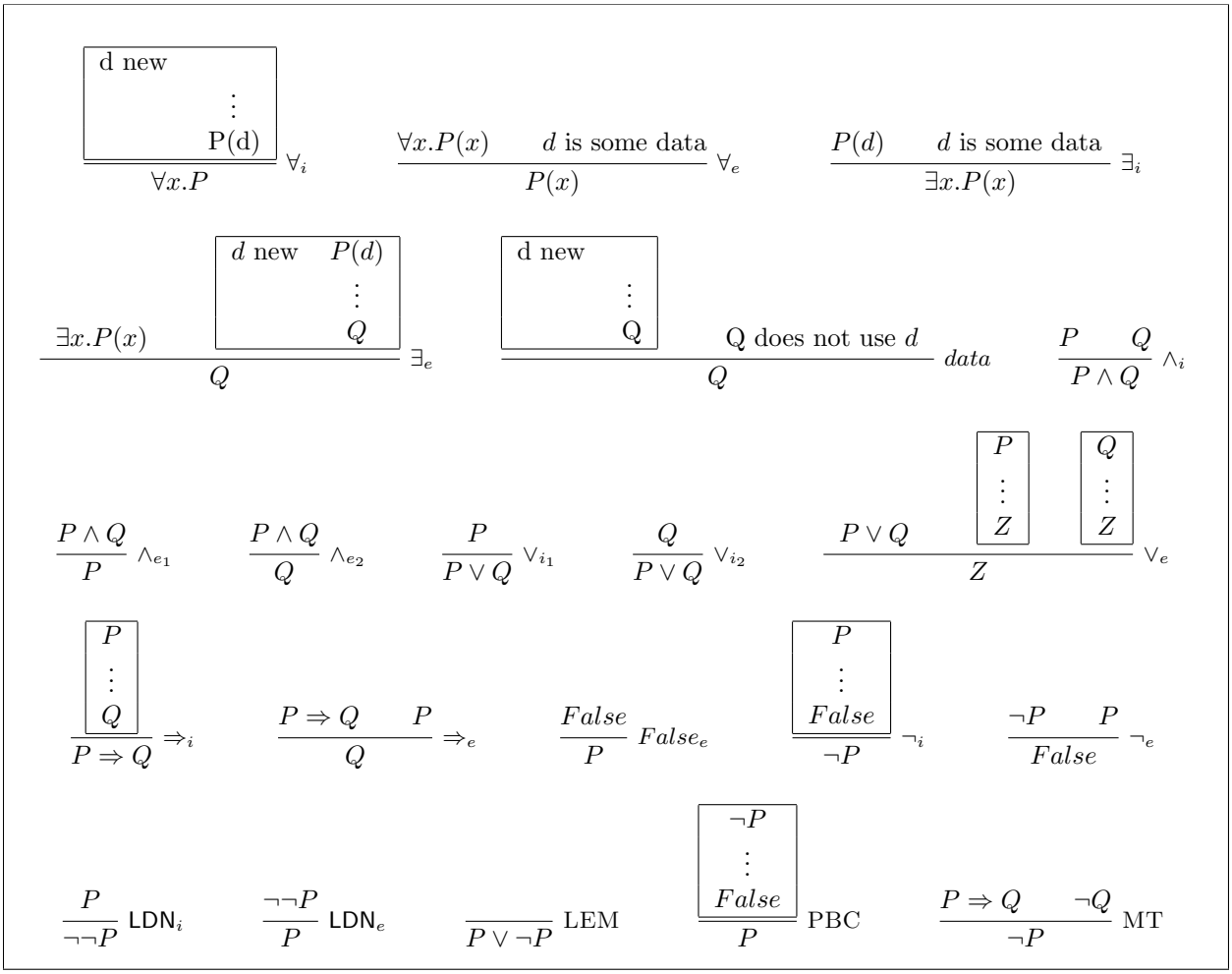
1.	$\forall x.P(x)$		Premise
3.	d new		
4.		$P(d)$	$\forall_e, 1$
5.		$\exists x.P(x)$	$\exists_i, 4$
6.	$\exists x.P(x)$		$data$, 2-5

Remark 16. *Before using this rule one should exhaust all possibility of being able to get data from a premise or assumption, because this rule can, if used incorrectly, lead one down paths that will not work.*

We now give several examples:

- $\forall x.(P(x) \Rightarrow Q(x)), \exists x.P(x) \vdash \exists x.Q(x)$
- $\forall x.(Q(x) \Rightarrow R(x)), \exists x.(P(x) \wedge Q(x)) \vdash \exists x.(P(x) \wedge Q(x))$
- $\exists x.\forall y.P(x,y) \vdash \forall y.\exists x.P(x,y)$
- $\exists x.(S \Rightarrow Q(x)) \vdash S \Rightarrow \exists x.Q(x)$

The Rules of Natural Deduction for First-Order Logic:



3 Theorems, Lemmas, Corollaries and Proofs

We have finished our stint into basic logic in both propositional and fist order forms. At this point we use what we have learned about basic logic to begin studying informal mathematical proof. In this section we will learn how to state a mathematical result, and then how to prove the stated result. In addition, we will cover several forms of mathematical proofs and when one form is better to use than another.

3.1 Theorems, Lemmas, and Corollaries

Throughout science mathematics is used to unequivocally convince someone of a scientific discovery, but how is it used? A mathematical result is stated in the form of either a theorem (proposition or result), lemma, or corollary. For example, the following is a theorem:

Theorem 17 (Fermat's Last Theorem)**.** *For any natural numbers, a, b, c, and $n > 2$, it is not the case that $a^n + b^n = c^n$.*

Thus, a theorem is nothing more than a first-order formula with a name. A lemma is similar. For example, the following is a lemma:

Lemma 18 (Pumping Lemma)**.** *Suppose L is a regular language. Then there exists a $p \in \mathbb{N}$ where if $w \in L$ and $|w| \geq p$, then w can be factored into three pieces, $w = xyz$, satisfying the following properties:*

1. *for any $i \in \mathbb{N}, xy^iz \in A$,*
2. *$|y| > 0$, and*
3. *$|xy| \leq p$.*

When describing mathematical results – even in CS – it is important to organize them into theorems and lemmas, because this makes the result under consideration apparent, it will also put a mathematical result together with its proof, but more importantly it makes it easier to reference the results in the proofs of later results. Science builds on previous results, and theorems and lemmas give us this ability.

So when do we use a lemma instead of a theorem? They both look very similar. A theorem is a significant main result, and lemmas are smaller results to which theorems depend on. A corollary is now slightly different from theorems or lemmas. They too are simply the statement of a first-order formula with a name, but they should be seen as direct consequences of theorems or lemmas. For example, the following is a corollary of Fermat's Last theorem:

Corollary 19. *For any natural numbers, a, b, and c, it is not the case that $a^3 + b^3 = c^3$.*

Corollaries usually are not accompanied by proofs, because they should be straightforward consequences of the theorem they are deduced from.

Keep in mind that sometimes universal quantifiers are left implicit in mathematics. For example, the following lemma as an implicit universal quantifier:

Lemma 20. *If $x > y$, where x and y are positive real numbers, then $x^2 > y^2$.*

The same result can be written as follows with an explicit universal quantifier:

Lemma 21. *For any positive real numbers x and y, if $x > y$, then $x^2 > y^2$.*

However, existential quantifiers are always explicit.

Open Problems. The driving force of mathematics is the notion of proof, but it is not a simple task to prove a mathematical result. Some theorems have been stated, but without proof for many years. For example, Fermat's last theorem had no proof for 357 years. A scientific problem without a solution is called

an **open problem**, and mathematical results without proofs are prefect examples. Thus, we can say that Fermat's last theorem was an open problem for 357 years. The first to give a successful proof was Andrew Wiles, and the proof weighs in at 150 pages long.

3.2 Informal Proofs

Mathematical results are always accompanied by a proof, and as of right now we will live our lives based on the motto that nothing capable of being proved should be accepted without proof (Dedekind 1872). A proof of a mathematical result is a rigorous argument of the validity of the mathematical statement under consideration. In this class a mathematical proof will be written primarily in English prose, but must be mathematically consistent. Proofs must be constructed with care, because a wrong proof can hinder the progress of science, because every future result is based on previous results. If a proof is done wrong then scientists may think that a result holds, but which in reality does not, and thus can waste a lot of time and money.

First, we consider an example proof:
Lemma 22. *For any natural numbers a, b, and c, if $a < b$ and $b < c$, then $a < c$.*

Proof. This is an example of a direct proof. XXXX □

Remark. Proofs can be extremely difficult to get right, and one should not be to hard on their selves for not finding a proof quickly. Mathematics is not a sprint, and should not be treated as one.

Consider our example proof from above:

Lemma 23. *For any natural numbers a, b, and c, if $a < b$ and $b < c$, then $a < c$.*

Proof. Suppose a, b, and c are natural numbers, and that $a < b$ and $b < c$. It suffices to conclude $a < c$.

We know by assumption that $a < b$, and thus, there must be some natural number $k > 0$ such that $b = a + k$. Similarly, since we know by assumption that $b < c$ there must exist a second natural number $k' > 0$ such that $c = b + k'$. Now substituting $a + k$ for b in the latter we obtain that $c = (a + k) + k'$. It is easy to see that $a < (a + k) + k'$, because k and k' are at least one, but $c = (a + k) + k'$, therefore, $a < c$. □

The statement of the lemma can be translated into first-order logic as $\forall a.\forall b.\forall c.((a < c) \wedge (b < c)) \Rightarrow (a < c)$. Now we match this against the conclusions of the introduction rules we have defined above and it will give us an algorithm for the proof. This statement corresponds to the conclusion of the implication introduction rule ($\Rightarrow_i$). This rule then says that to prove the implication we must first assume the assumptions are true, and then prove the conclusion. Note that this is exactly what the previous proof does. It assumes $a < b$ and $b < c$ are true, and then argues using these assumptions that $a < c$ using facts we know are true about natural numbers.

Consider a second example:

Lemma 24. *If a divides b, where a and b are integers, then*

i. $a \bmod b = 0$

ii. $a \neq 0$

The previous lemma is a common way of writing a conjunction. In fact, it can be translated into first-order logic as follows:

$$\forall a.\forall b.(a \text{ divides } b) \Rightarrow ((a \bmod b = 0) \wedge (a \neq 0))$$

The inference rules given above tell us that any proof of this lemma must first assume a divides b, and then prove the conjunction $(a \bmod b = 0) \wedge (a \neq 0)$. The rule for conjunction says that to prove a conjunction we must first prove the first conjunct and then prove the second conjunction. So the rest of the proof can be broken down into two steps, the first is a proof of the first conjunction, and the second step is the proof of the second.

3.2.1 Tips on Dealing with Quantifiers in Proofs

Suppose we had to prove the following implications:

$$\begin{array}{ll} i. & (\forall x.P(x)) \Rightarrow (\forall y.Q(y)) \\ ii. & (\forall x.P(x)) \Rightarrow (\exists y.Q(y)) \\ iii. & (\exists x.P(x)) \Rightarrow (\forall y.Q(y)) \\ iv. & (\exists x.P(x)) \Rightarrow (\exists y.Q(y)) \end{array}$$

Using the assumptions given above requires some care. Consider formula i and suppose we wanted to prove $Q(y)$. Then we are allowed to assume $\forall x.P(X)$ holds and that we have an arbitrary y. Then we would need to prove $Q(y)$. Lastly, assume that we need to use our first assumption. In order to do this we would need some x from our domain before we can conclude $P(x)$. Formula ii is similar.

Now consider formula iii. A proof of this formula is allowed to first assume $\exists x.P(x)$ holds, and that there is some arbitrary y. The assumption states that there is some x such that $P(x)$ holds, but we do not know which x it is, and thus, we must keep it arbitrary. Thus, we are allowed to use x throughout our proof, but we do not know anything about x except that $P(x)$ holds. Formula is is similar, but we must find a y such that $Q(y)$.

Now we give an example:

Lemma 25. *Show that each of the following can be used to express the fact that there exists an unique x, such that $P(x)$:*

i. There exists a x, such that for any y, $P(y)$ holds iff $x = y$.

ii. There exists a w, such that $P(w)$ holds and for any z, $P(z)$ implies $w = z$.

Proof. $(i \Rightarrow ii)$ Suppose there exists an x, such that for any y, $P(y)$ holds iff $x = y$. This is equivalent to the statement that there exists an x, such that for any y, if $P(y)$, then $x = y$ and if $x = y$, then $P(y)$. Thus, we know by assumption that for any y, if $P(y)$, then $x = y$.

Furthermore, taking $y = x$ in the our initial assumption we know that $P(x)$ holds iff $x = x$, and hence, this implies that $P(x)$ holds. Therefore, we know $P(x)$ and for any y, if $P(y)$, then $x = y$. Now take $w = x$ and $z = y$ and thus, we have there exists an w, such that $P(w)$ holds and for any z, $P(z)$ implies $w = z$.

$(ii \Rightarrow i)$ Suppose there exists a w, such that $P(w)$ holds and for any z, $P(z)$ implies $w = z$. Choose $x = w$. Now we must prove that for any z, $P(z)$ iff $w = z$.

$(\Rightarrow)$ Suppose $P(y)$ holds for any arbitrary y. We know by assumption that for any z, $P(z)$ implies $w = z$. So choosing $w = x$ and $z = y$ we obtain that $x = y$.

$(\Leftarrow)$ Suppose $x = y$, for any arbitrary y. Then we must show that $P(y)$ holds which is equivalent to showing that $P(x)$ holds, and hence is also equivalent to showing that $P(w)$ holds, but this holds by assumption.

Therefore, we obtain our result. □

3.2.2 Proof Strategies

Direct Proof. Many strategies can be seen as a trick on the statement or the proof we are conducting. The first strategy is called the **direct proof** because it uses no tricks, and can be seen as a direct argument of the given statement.

Consider an example. First, we need the following definition:

Definition 26. *An integer, n, is even when $n = 2k$ for some integer k, and odd when $n = 2k + 1$ for some integer k. Note that every integer is either even or odd, and not both.*

Now we use the previous definition to give the following example.

Example 27. *Give a direct proof of the following lemma:*

If n is an odd integer, then n^2 is odd.

Proof. Notice that this result translates into the first-order formula $\forall a.(n \text{ is odd}) \Rightarrow (n^2 \text{ is odd})$. Thus, following the rule $\Rightarrow_i$ we first must assume n is odd, and then prove that its square is also odd.

Assume n is odd. Then by the definition of an odd number there must exist an integer k such that $n = 2k+1$. Using this we know that $n^2 = (2k + 1)^2 = (2k)^2 + 2k + 1 = 2((2k^2) + k) + 1$. Let $r = ((2k^2) + k)$, then $n^2 = 2r + 1$, and thus, by definition n^2 is odd. □

Proof by Contraposition. This is the first non-direct strategy. We showed earlier in the course that if $\phi \Rightarrow \psi$ is true, then $\neg\psi \Rightarrow \neg\phi$ is true. Sometimes proving the latter is easier than proving the original statement. Proving the contraposition instead of the original implication is called **proof by contraposition**.

Example 28. *Prove that if n is an integer and $3n + 2$ is even, then n is even.*

Proof. Suppose n is an integer. We prove this using a proof by contraposition, and thus, we prove the following statement:

If n is not even, then $3n + 2$ is not even.

Suppose n is not even, then it must be odd. Thus, there exists an integer k such that $n = 2k+1$. At this point we must show that $3n+2$ is odd, but $3n+2 = 3(2k+1)+2 = (3\times2)k+1+2 = (3\times2)k+2+1 = 2(3k+1)+1$. Let $r = 3k + 1$, then $3n + 2 = 2r + 1$, and hence, is odd.

Therefore, by contraposition if $3n + 2$ is even, then n is even. □

Proof by Contradiction. A very powerful strategy is one in which we first assume the negation of the conclusion of the statement we are trying to prove, and then obtaining a contradiction. A contradiction is defined as the formula $\phi \wedge \neg\phi$, that is, we obtain a contradiction when we know a formula ϕ is true, and we also know that its negation is true. This is a logical impossibility.

Example 29. *Prove that for any natural number n, n is even if and only if $7n + 4$ is even.*

Proof. This example contains a couple new features. First, the statement has the form $\forall n.(n \text{ is even}) \Leftrightarrow (7n+4 \text{ is even})$. We know that a biconditional is defined as a conjunction $\forall n.((n \text{ is even}) \Rightarrow (7n+4 \text{ is even})) \wedge ((7n + 4 \text{ is even}) \Rightarrow (n \text{ is even}))$. So to prove this result we must first prove that if n is even, then $7n + 4$ is even. Then we must prove that if $7n + 4$ is even, then n is even.

($\Rightarrow$) We first prove that if n is even, then $7n + 4$ is even. We use proof by contradiction. Suppose n is even, and $7n + 4$ is odd (not even). Then $n = 2k$ for some natural number k. This implies that $7n+4 = 7(2k)+4 = 2(7k+2)$ is odd, because we assumed $7n+4$ is odd, but letting $r = 7k+2$ implies that $7n + 4 = 2r$ which is odd. This is a contradiction. Thus, it must be the case that $7n + 4$ is even.

($\Leftarrow$) Now we prove that if $7n + 4$ is even, then n is even. We also do this proof by contradiction. Suppose $7n + 4$ is even, and n is odd. Then $n = 2k + 1$. This implies that $7n + 4 = 7(2k + 1) + 4 = 2(2k) + 7 + 4 = 2(2k) + 4 + 7 = 2(2k) + 4 + 6 + 1 = 2(k + 5) + 1$. Letting $r = k + 5$ implies that $7n + 4 = 2r + 1$, but this is odd, and we assumed $7n+4$ is even, and thus, a contradiction. Therefore, it must be the case that n is even. Note that in this case we obtain a contradiction using the premise, and not the negation of the conclusion unlike the previous case. This is okay, because we just need a contradiction. □

Note: Have the students read about more proof strategies in Chapter 1.7 - 1.8.

Another example is the following.

Example 30. *Prove the **triangle inequality**: if x and y are real numbers, then $|x| + |y| \geq |x + y|$. Here $|x|$ represents the absolute value of x which equals x when $x \geq 0$, and equals $-x$ otherwise.*

Proof. Suppose x and y are real numbers. Prove by case splitting on the output of $|x|$ with inner case-splitting over the output of $|y|$.

Case. Suppose $x \geq 0$, then $|x| = x$. We must further case split on y.

- Case. Suppose $y \geq 0$, then $|y| = y$. Then we know it is the case that $|x| = x$ and $|y| = y$. Thus, $|x + y| = x + y = |x| + |y|$ which implies that $|x| + |y| \geq |x + y|$.
- Case. Suppose $y < 0$. Then it must be the case that there is a positive real number k such that $y = -k$. Then by definition we know that $|y| = -y = -(-k) = k$. It suffices to show that:

$$|x| + |y| = x + -y = x + k \geq |x - k| = |x + y|$$

  We have two further cases to consider:

  - Suppose $x \geq k$. Then $x - k \geq 0$ and $|x - k| = x - k$. Thus, $x + k \geq x - k$.
  - Suppose $x < k$. Then $|x - k| < 0$ and $|x - k| = -(x - k) = k - x$. Now it suffices to show that $x + k \geq k - x$, but this is certainly the case.

Case. Suppose $x < 0$. Then $|x| = -x$. We must further case split on y.

- Case. Suppose $y \geq 0$, then $|y| = y$. Similar to the second case above by interchanging y and x.
- Case. Suppose $y < 0$. Then $|y| = -y$. Similar to the first case above, but we must factor out the negative.

□