Logic and Computation: Propositional Logic

Jonathan Buss

Spring 2018

Based in part on materials prepared by B. Bonakdarpour from Huth & Ryan’s text.

Additional thanks to D. Maftuleac, C. Roberts, R. Trefler, and P. Van Beek.
Broad Outline of Propositional Logic

- Introduction to Propositional Logic
- Syntax
- Semantics
- Proof Systems (Natural Deduction)
Logic is the systematic study of the principles of reasoning and inference.

We use logic throughout computer science,

- To model the computer hardware, software and embedded systems we create or encounter, in order to reason about those objects in a completely precise and rigorous manner.
- To understand how to develop systems that can themselves apply reason and make inferences ("artificial intelligence").

Historically, logic and computer science are closely linked.

- To define and build a “computer” required deep ideas from logic.
- Computer science gave the first real definition of “rigorous argument”: an argument that can be checked by a machine.
An Example Argument

Consider this example.

If the train arrives late and there are no taxis at the station, then John is late for his meeting.

John is not late for his meeting.

The train did arrive late.

*Therefore*, there were taxis at the station.

*Question*. Is this argument *valid*? Why, or why not?
Consider this example.

If the train arrives late and there are no taxis at the station, then John is late for his meeting.

John is not late for his meeting.

The train did arrive late.

*Therefore*, there were taxis at the station.

**Question.** Is this argument *valid*? Why, or why not?

**Question.** What is the structure of the argument?
Consider this example.

If the train arrives late and there are no taxis at the station, then John is late for his meeting.

John is not late for his meeting.

The train did arrive late.

*Therefore*, there were taxis at the station.

**Question.** Is this argument *valid*? Why, or why not?

**Question.** What is the structure of the argument?

We can represent the structure symbolically as

If $p$ and not $q$, then $r$. Not $r$. $p$. Therefore $q$. 
The argument in the previous example has the form

If $p$ and not $q$, then $r$. Not $r$. $p$. Therefore $q$.

where

$p$ stands for “the train arrives late.”
$q$ stands for “there are taxis at the station.”
$r$ stands for “John is late for his meeting.”
An Example Argument (2)

The argument in the previous example has the form

If \( p \) and not \( q \), then \( r \). Not \( r \). \( p \). Therefore \( q \).

where

\( p \) stands for “the train arrives late.”
\( q \) stands for “there are taxis at the station.”
\( r \) stands for “John is late for his meeting.”

What happens if we change our notion of \( p \), \( q \) and \( r \)? Perhaps

\( p \) stands for “It rains.”
\( q \) stands for “Jane takes her umbrella.”
\( r \) stands for “Jane gets very wet.”

Then the argument changes...
An Example Argument (3)

The essential argument: If \( p \) and not \( q \), then \( r \). Not \( r \). \( p \). Therefore \( q \).

New choices for \( p \), \( q \) and \( r \):

\( p \) stands for “It rains.”
\( q \) stands for “Jane takes her umbrella.”
\( r \) stands for “Jane gets very wet.”

The new argument is

If it rains, and Jane does not take her umbrella, then Jane gets very wet. Jane does not get very wet. It rains. Therefore, Jane takes her umbrella.

An equally valid argument!
What Is Logic?—Reprise

In the example argument,

- The factual content of the statements doesn’t matter.
- The relationships among the statements govern the argument.

Logic concerns careful reasoning about the process of reasoning.

As part of this care, we need to know

- What, exactly, constitutes a “statement”?
- What, precisely, do the logical relationships mean?

We shall start with a basic form of logic, called *propositional logic*. 
A proposition is a declarative sentence that is either true or false.

In other words, we make the following defining assumption:

For any particular proposition, in any particular situation (or “world”), either the proposition is true, or the proposition is false, and it is never the case that the proposition is both true and false.
Examples of Propositions

Each of the following is a proposition.

• The sum of 3 and 5 is 8.
• The sum of 3 and 5 is 35.
• $7 \geq 5$.
• $3 \geq 5$.
• Every even number greater than 2 is the sum of two prime numbers.

In some of the cases, we may not know whether the statement is true or false, but it’s one or the other—and not both.
### Connectives of Propositional Logic

<table>
<thead>
<tr>
<th>Connective</th>
<th>Meaning</th>
<th>Some alternative expressions</th>
</tr>
</thead>
<tbody>
<tr>
<td>((¬p))</td>
<td>not (p)</td>
<td>(p) does not hold; (p) is false; it is not the case that (p)</td>
</tr>
<tr>
<td>((p \land q))</td>
<td>(p) and (q)</td>
<td>(p) but (q); not only (p) but (q); (p) while (q); (p) despite (q); (p) yet (q); (p) although (q)</td>
</tr>
<tr>
<td>((p \lor q))</td>
<td>(p) or (q)</td>
<td>(p) or (q) or both; (p) and/or (q); (p) unless (q)</td>
</tr>
<tr>
<td>((p \rightarrow q))</td>
<td>if (p) then (q)</td>
<td>(p) implies (q); (q) if (p); (p) only if (q); (q) when (p); (p) is sufficient for (q); (q) is necessary for (p)</td>
</tr>
<tr>
<td>((p \leftrightarrow q))</td>
<td>(p) if and only if (q)</td>
<td>(p) is equivalent to (q); (p) exactly if (q); (p) is necessary and sufficient for (q)</td>
</tr>
</tbody>
</table>
“Translating” from English to Propositional Logic

Examples

1. She is clever and hard working.
2. He is clever but not hard working.
3. If it rains, he will be at home; otherwise he will go to the market or to school.
4. The sum of two numbers is even if and only if both numbers are even or both numbers are odd.

A non-example:

- Do you want bacon or sausage?
1. The sky is blue and the grass is red or Trudeau is prime minister.
English Can Be Ambiguous

1. The sky is blue and the grass is red or Trudeau is prime minister.
   
   (Does “the grass is red” belong with the “and” or the “or”??)
English Can Be Ambiguous

1. The sky is blue and the grass is red or Trudeau is prime minister.  
   
   (Does “the grass is red” belong with the “and” or the “or”??)

2. If it is sunny tomorrow, then I will play golf.
English Can Be Ambiguous

1. The sky is blue and the grass is red or Trudeau is prime minister.
   
   *(Does “the grass is red” belong with the “and” or the “or”?)*

2. If it is sunny tomorrow, then I will play golf.
   
   *(The next day it rains. I play golf anyway…)*
English Can Be Ambiguous

1. The sky is blue and the grass is red or Trudeau is prime minister.
   
   (Does “the grass is red” belong with the “and” or the “or”??)

2. If it is sunny tomorrow, then I will play golf.
   
   (The next day it rains. I play golf anyway....)

We can’t afford ambiguity in a precise logic.

Our definitions will exclude such possibilities.
Some sentences are not propositions. For example, a sentence might be

Interrogative: Where shall we go to eat?
Imperative: Please pass the salt.
Exhortative: Go, team!
Ambiguous: Time flies like an arrow.
Nonsense: Green ideas sleep furiously.
Paradoxical: This sentence is false.

In the field of “artificial intelligence,” one must deal with such sentences. For this course, however, we shall ignore them.
**Example:** Binary notation.

I have a natural number, less than 8, which I call $x$. Let’s define three propositions:

- $p_0$: The number $x$ is odd.
- $p_1$: If $x$ is divided by 4, it has a remainder of 2 or 3.
- $p_2$: If $x$ is divided by 8, it has a remainder of 4, 5, 6 or 7.

Suppose that $p_0$ is false, $p_1$ is true, and $p_2$ is true. What is $x$?

If we write 1 for “true” and 0 for “false”, and write the values in decreasing order, we get “110” — the binary representation of $x$. 
Example: Sudoku.

“Sudoku” is played on a square grid (we’ll use 3x3).

• Each cell gets a number: 1, 2, or 3.
• Every row must have one of each.
• Every column must have one of each.

Some cells have values given. A solution fills in the other cells.
Sudoku in propositional logic

To represent a Sudoku grid, we must choose binary-valued variables.

Many different choices work. I will use variables $p_{n,r,c}$ to indicate whether number $n$ appears in row $r$, column $c$.

Using these 27 variables, we can express

- a solution, or partial solution, or
- the requirements on a solution.

For example, setting the values $p_{1,1,2} = \text{true}$ and $p_{2,3,1} = \text{true}$ corresponds to the initial grid

```
1

2
```
Sudoku: Expressing requirements

**Requirement:** at most one number per cell

There are several ways to express this:

- **In words**
  - Not both $p_{i,r,c}$ and $p_{j,r,c}$ (for $j \neq i$)
  - Either not $p_{i,r,c}$ or not $p_{j,r,c}$
  - If $p_{i,r,c}$, then not $p_{j,r,c}$

- **Symbolically**
  - $\neg(p_{i,r,c} \land p_{j,r,c})$
  - $(\neg p_{i,r,c}) \lor (\neg p_{j,r,c})$
  - $(p_{i,r,c} \rightarrow (\neg p_{j,r,c}))$

**Exercise:** how many of these formulas are needed, in total?

**Exercise:** Find similar formulas for the other requirements.
Propositional Logic:
Syntax
In propositional logic, simple *atomic* propositions are the basic building blocks.

We connect atomic propositions into *compound* propositions, and then analyze sets of interrelated propositions.

Typical questions to consider:

- Does a given sequence of propositions form a valid argument?
- Can all propositions in a given set be true simultaneously?

First, however, we must answer the question,

What, exactly, is a proposition?
Symbols and expressions

Propositions are represented by *formulas*.

A formula consists of a sequence of *symbols*. There are three kinds of symbols.

**Propositional variables**: Usually lowercase Latin letters; e.g., $p$, $q$, $r$, etc., perhaps with subscripts ($p_1$, $p_2$, $q_{27}$, etc.).

**Connectives**: We shall use $\neg$, $\land$, $\lor$, $\rightarrow$ and $\leftrightarrow$. (Others are possible.)

**Punctuation**: ‘(’ and ‘)’.

Every formula is a sequence of symbols, but not every sequence of symbols is a formula.

An arbitrary finite sequence of symbols is an *expression* (or string).
An *expression* is a finite sequence (or “string”) of symbols.

The *length* of an expression is its number of symbols.

For example, \((\neg)(\lor)pq\rightarrow\) is an expression.

**Questions:**

- What is its length of this expression?
- Is it a formula?

We often use a letter that is not formally a symbol in order to name an expression. For example, we might denote the expression above by \(\alpha\).

This is an example of a “meta-symbol”. It is NOT a symbol!
Some terminology for expressions.

- Two expressions $\alpha$ and $\beta$ are **equal**, written as $\alpha = \beta$, iff they are of the same length, say $n$, and if $n > 0$ then for all $i \in [1..n]$ the $i$th symbol of $\alpha$ is the same as the $i$th symbol of $\beta$.

- We write $\alpha\beta$ to mean the **concatenation** of two expressions $\alpha$ and $\beta$. For example, if
  \[
  \alpha = (\neg)()
  \]
  and
  \[
  \beta = \lor pq \rightarrow
  \]
  then
  \[
  \alpha\beta = (\neg)()\lor pq \rightarrow .
  \]
Definition:

If $\alpha$ is an expression of length $i$ and $\beta$ is an expression of length $j$, then $\alpha \beta$ is an expression of length $i + j$. We have

The $k$th symbol of $\alpha \beta$ is

$$
\begin{cases}
\text{the } k\text{th symbol of } \alpha & \text{if } k \leq i \\
\text{the } (k - i)\text{th symbol of } \beta & \text{if } k > i
\end{cases}
$$
Definition of “well-formed formula”

Let $\mathcal{P}$ be a set of propositional variables. We define the set of well-formed formulas (or WFFs) over $\mathcal{P}$ inductively as follows.

1. An expression consisting of a single symbol of $\mathcal{P}$ is a well-formed formula.
2. If $\alpha$ is a well-formed formula, then $(\neg \alpha)$ is a well-formed formula.
3. If $\alpha$ is a well-formed formula and $\beta$ is a well-formed formula, then each of

$$\ (\alpha \land \beta) \ , \ (\alpha \lor \beta) \ , \ (\alpha \rightarrow \beta) \ , \ \text{and} \ (\alpha \leftrightarrow \beta)$$

is a well-formed formula.
4. Nothing else is a well-formed formula.

(Note the use of the meta-symbols ‘$\alpha$’ and ‘$\beta$’ to refer to formulas.)
Examples: well-formed formulas

Example: The following are well-formed formulas.

1. $p, q, r, s$  (rule 1)
2. $(\neg p)$  (rule 2, from #1)
3. $(r \land q)$  (rule 3, from #1)
4. $((\neg p) \rightarrow s)$  (rule 3, from #2 and #1)
5. $((r \land q) \lor ((\neg p) \rightarrow s))$  (rule 3, from #3 and #4)
6. $(\neg (r \land q))$  (rule 2, from #3)
We shall define the semantics (i.e. the meaning) of a formula from its syntax (the way its symbols are assembled).

Is this well-defined? Or can a formula get two different meanings?

**Theorem.** Every well-formed Propositional formula has a unique derivation as a well-formed formula. That is, each well-formed formula has exactly one of the following forms:

1. an atom, 2. $\neg \alpha$, 3. $\alpha \land \beta$, 4. $\alpha \lor \beta$, 5. $\alpha \to \beta$, or 6. $\alpha \leftrightarrow \beta$.

In each case, it is of that form in exactly one way.
As an example, consider \(((p \land q) \rightarrow r)\). It can be formed from the two formulas \((p \land q)\) and \(r\) using the connective \(\rightarrow\).

If we tried to form it using \(\land\), the two parts would need to be “\((p\) and \(q) \rightarrow r\)”. But neither of those is a formula!

The statement holds for this example.

How can we be sure the theorem holds for every formula?
To prove the theorem, we will use mathematical induction.

Before doing the proof, we will review mathematical induction.

This starts with the natural numbers.
Natural Numbers

The “natural numbers” are the numbers we use to count things.

Before we start, we count zero; as we find things we count one, two, etc.

The natural numbers form an unbounded sequence

\[0, 1, 2, 3, 4, \ldots\]

Suppose \( P \) names a property. We write “\( P(2) \)” to mean “2 has property \( P \)”\), or “\( P \) holds for 2”.

A statement “every natural number has property \( P \)” corresponds to a sequence of statements

\[P(0), P(1), P(2), P(3), P(4), \ldots\]
Mathematical Induction

Principle of mathematical induction:

Suppose we establish two things: that

- 0 has property $P$, and that
- whenever any number has property $P$, then the next number also has property $P$.

Then we may conclude that every natural number has property $P$.

Example: Show that $\sum_{x=0}^{n} x = \frac{n(n+1)}{2}$ for every natural number $n$.

Let $P$ be the property; that is, let $P(n)$ be “$\sum_{x=0}^{n} x = \frac{n(n+1)}{2}$.”
Proof for the example

Step 1 (basis): The property $P(0)$ is

$$
\sum_{x=0}^{0} x = \frac{0(0+1)}{2}.
$$

The left side of the equation is just 0. Also the right side evaluates to 0. Thus 0 has property $P$.

Step 2 (inductive step): hypothesize that some number has property $P$; in other words, that

$$
\sum_{x=0}^{\text{some number}} x = \frac{\text{some number}\,(\text{some number}+1)}{2}.
$$
Proof for the example

**Step 1** (basis): The property $P(0)$ is

$$\sum_{x=0}^{0} x = \frac{0(0+1)}{2}.$$  

The left side of the equation is just 0. Also the right side evaluates to 0. Thus 0 has property $P$.

**Step 2** (inductive step): hypothesize that some number has property $P$; in other words, that

$$\sum_{x=0}^{\text{some number}} x = \frac{\text{some number} (\text{some number}+1)}{2}.$$  

For simplicity, we give a name to the “some number”. I choose $k$.

Thus the hypothesis becomes

$$\sum_{x=0}^{k} x = \frac{k(k+1)}{2}.$$
**Step 2** (inductive step), continued:

We hypothesize that $k$ has property $P$; that is, 
\[ \sum_{x=0}^{k} x = \frac{k(k + 1)}{2}. \]

We need to demonstrate that $k + 1$ has property $P$; that is, 
\[ \sum_{x=0}^{k+1} x = \frac{(k + 1)((k + 1) + 1)}{2} = \frac{(k + 1)(k + 2)}{2}. \]

We calculate:

\[ \sum_{x=0}^{k+1} x = \left( \sum_{x=0}^{k} x \right) + (k + 1) \]
\[ = \frac{k(k+1)}{2} + (k + 1) \]
\[ = \left( \frac{k}{2} + 1 \right)(k + 1) \]
\[ = \frac{(k + 1)(k + 2)}{2} \]

DONE!
Step 2 (inductive step), continued:

We hypothesize that $k$ has property $P$; that is,\[\sum_{x=0}^{k} x = \frac{k(k+1)}{2} .\]

We need to demonstrate that $k + 1$ has property $P$; that is,
\[\sum_{x=0}^{k+1} x = \frac{(k + 1)((k + 1) + 1)}{2} = \frac{(k + 1)(k + 2)}{2} .\]

We calculate:
\[\sum_{x=0}^{k+1} x = \left(\sum_{x=0}^{k} x\right) + (k + 1) \quad \text{definition of } \sum\]
\[= \frac{k(k+1)}{2} + (k + 1) \quad \text{hypothesis}\]
\[= \left(\frac{k}{2} + 1\right)(k + 1) \quad \text{"algebra"}\]
\[= \frac{(k + 1)(k + 2)}{2} \quad \text{DONE!}\]
Observations/Techniques

To talk about something, give it a name.
  E.g., property $P$, number $k$, etc.

A formula is a textual object.
  In this text, we can substitute one symbol or expression for another.
  For example, we put “$k + 1$” in place of “$k$”.

The induction principle gives a “template” for a proof:

• The proof has two parts: the “basis” and the “inductive step”.
• In the inductive step, hypothesize $P(k)$ and prove $P(k + 1)$ from it.

But the induction principle does not say how to actually do either step.
We must invent the method ourselves.
**“Simple” Induction vs. “Strong” Induction**

<table>
<thead>
<tr>
<th></th>
<th><strong>Simple Induction</strong></th>
<th><strong>Strong Induction</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Basis:</td>
<td>Show $P(0)$</td>
<td>Show $P(0)$</td>
</tr>
<tr>
<td>Ind. Hypothesis:</td>
<td>$P(k)$ holds</td>
<td>$P(m)$ holds, <em>for every</em> $m \leq k$</td>
</tr>
<tr>
<td>Ind. Step:</td>
<td>Show $P(k+1)$ holds</td>
<td>Show $P(k+1)$ holds</td>
</tr>
<tr>
<td>Conclusion:</td>
<td>$P(k)$ holds for every $k$</td>
<td>$P(k)$ holds for every $k$</td>
</tr>
</tbody>
</table>

What is the difference?
**“Simple” Induction vs. “Strong” Induction**

| Simple Induction | Strong Induction  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Basis:</td>
<td>Show $P(0)$</td>
</tr>
<tr>
<td>Ind. Hypothesis:</td>
<td>$P(k)$ holds</td>
</tr>
<tr>
<td>Ind. Step:</td>
<td>Show $P(k+1)$ holds</td>
</tr>
<tr>
<td>Conclusion:</td>
<td>$P(k)$ holds for every $k$</td>
</tr>
<tr>
<td></td>
<td>Show $P(0)$</td>
</tr>
<tr>
<td></td>
<td>$P(m)$ holds, <em>for every</em> $m \leq k$</td>
</tr>
<tr>
<td></td>
<td>Show $P(k+1)$ holds</td>
</tr>
<tr>
<td></td>
<td>$P(k)$ holds for every $k$</td>
</tr>
</tbody>
</table>

What is the difference?

Define $Q(k)$ as the property “$P(m)$ holds, for every $m \leq k$”.

---

**Syntax**  
**Mathematical induction**
"Simple” Induction vs. “Strong” Induction

<table>
<thead>
<tr>
<th>Simple Induction</th>
<th>Strong Induction or Course of Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basis:</td>
<td>Show $P(0)$</td>
</tr>
<tr>
<td>Ind. Hypothesis: $P(k)$</td>
<td>$P(m)$ holds, \textit{for every} $m \leq k$</td>
</tr>
<tr>
<td>Ind. Step:</td>
<td>Show $P(k+1)$ holds</td>
</tr>
<tr>
<td>Conclusion:</td>
<td>$P(k)$ holds for every $k$</td>
</tr>
</tbody>
</table>

What is the difference?

Define $Q(k)$ as the property “$P(m)$ holds, for every $m \leq k$.”

- $Q(0)$ is equivalent to $P(0)$. 
“Simple” Induction vs. “Strong” Induction

| Simple Induction | Strong Induction  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Basis:</strong></td>
<td>Show $Q(0)$</td>
</tr>
<tr>
<td><strong>Ind. Hypothesis:</strong></td>
<td>$P(k)$ holds</td>
</tr>
<tr>
<td><strong>Ind. Step:</strong></td>
<td>Show $P(k+1)$ holds</td>
</tr>
<tr>
<td><strong>Conclusion:</strong></td>
<td>$P(k)$ holds for every $k$</td>
</tr>
</tbody>
</table>

What is the difference?

Define $Q(k)$ as the property “$P(m)$ holds, for every $m \leq k$”.

- $Q(0)$ is equivalent to $P(0)$.
- $Q(k+1)$ is equivalent to “$Q(k)$ and $P(k+1)$”.

To prove $Q(k+1)$ from $Q(k)$, need only to prove $P(k+1)$. 


“Simple” Induction vs. “Strong” Induction

<table>
<thead>
<tr>
<th>Simple Induction</th>
<th>Strong Induction or Course of Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basis:</td>
<td>Show $Q(0)$</td>
</tr>
<tr>
<td>Ind. Hypothesis:</td>
<td>$Q(k)$ holds</td>
</tr>
<tr>
<td>Ind. Step:</td>
<td>Show $Q(k + 1)$ holds</td>
</tr>
<tr>
<td>Conclusion:</td>
<td>$P(k)$ holds for every $k$</td>
</tr>
</tbody>
</table>

| Basis:           | Show $P(0)$                         |
| Ind. Hypothesis: | $P(m)$ holds, for every $m \leq k$ |
| Ind. Step:       | Show $P(k + 1)$ holds               |
| Conclusion:      | $P(k)$ holds for every $k$          |

What is the difference?

Define $Q(k)$ as the property “$P(m)$ holds, for every $m \leq k$”.

- $Q(0)$ is equivalent to $P(0)$.
- $Q(k + 1)$ is equivalent to “$Q(k)$ and $P(k + 1)$”.
  To prove $Q(k + 1)$ from $Q(k)$, need only to prove $P(k + 1)$.
- “$Q(k)$ for every $k$” is equivalent to “$P(k)$ for every $k$”.
“Simple” Induction vs. “Strong” Induction

<table>
<thead>
<tr>
<th>Simple Induction</th>
<th>Strong Induction or Course of Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Basis:</strong></td>
<td>Show ( Q(0) )</td>
</tr>
<tr>
<td><strong>Ind. Hypothesis:</strong></td>
<td>( Q(k) ) holds</td>
</tr>
<tr>
<td><strong>Ind. Step:</strong></td>
<td>Show ( Q(k+1) ) holds</td>
</tr>
<tr>
<td><strong>Conclusion:</strong></td>
<td>( Q(k) ) holds for every ( k )</td>
</tr>
<tr>
<td></td>
<td>Show ( P(0) )</td>
</tr>
<tr>
<td></td>
<td>( P(m) ) holds, <em>for every</em> ( m \leq k )</td>
</tr>
<tr>
<td></td>
<td>Show ( P(k+1) ) holds</td>
</tr>
<tr>
<td></td>
<td>( P(k) ) holds for every ( k )</td>
</tr>
</tbody>
</table>

What is the difference? No difference!

Define \( Q(k) \) as the property “\( P(m) \) holds, for every \( m \leq k \)’’.

- \( Q(0) \) is equivalent to \( P(0) \).
- \( Q(k + 1) \) is equivalent to “\( Q(k) \) and \( P(k + 1) \)”.
  To prove \( Q(k + 1) \) from \( Q(k) \), need only to prove \( P(k + 1) \).
- “\( Q(k) \) for every \( k \)” is equivalent to “\( P(k) \) for every \( k \)”.

Syntax Mathematical induction
Structural Induction

To prove: every formula has property \( P \).

How to prove such a statement? Can we use induction?

A formula is not a natural number....
To prove: every formula has property $P$.

How to prove such a statement? Can we use induction?

A formula is not a natural number, but it suffices to prove any one of the following.

- For every natural number $n$, every formula with $n$ or fewer symbols has property $P$.
- For every natural number $n$, every formula with $n$ or fewer connectives has property $P$. OR
- For every natural number $n$, every formula with $n$ or fewer connectives has property $P$. OR
Structural Induction (2)

*To prove:* every formula has property $P$.

For every natural number $n$, every formula whose parse tree has height $n$ or less has property $P$.

OR

For every natural number $n$, every formula producible with $n$ or fewer uses of the formation rules has property $P$.

OR

In each of these formulations, the induction step requires showing that

If $P(\alpha)$ and $P(\beta)$, then $P(\neg \alpha)$ and $P(\alpha \star \beta)$.

Formulas $\alpha$ and $\beta$ have smaller $n$ values than $(\neg \alpha)$ and $(\alpha \star \beta)$ do.
The Principle of Structural Induction

**Theorem.** Let $R$ be a property. Suppose that

1. for each atomic formula $p$, we have $R(p)$; and
2. for each formula $\alpha$, if $R(\alpha)$ then $R(\lnot \alpha)$; and
3. for each pair of formulas $\alpha$ and $\beta$, and each binary connective $\star$, if $R(\alpha)$ and $R(\beta)$ then $R((\alpha \star \beta))$.

Then $R(\alpha)$ for every formula $\alpha$.

Use of this principle is called *structural induction*.

Structural induction is a special case of mathematical induction.
As a first example of structural induction, we shall prove the following.

**Lemma.** Every well-formed formula has an equal number of left and right parentheses.

**Proof.** We use structural induction. The property to prove is

\[ R(\alpha) \text{: } \alpha \text{ has an equal number of left and right parentheses} \]

for every formula \( \alpha \).

**Base case:** \( \alpha \) is an atom.

\( \alpha \) has no parentheses—only a propositional variable. Thus \( R(\alpha) \) holds.

This completes the proof of the base case.
**Inductive step:**

**Hypothesis:** formulas $\alpha$ and $\beta$ both have property $R$.

**To prove:** each of the formulas $(\neg \alpha)$, $(\alpha \land \beta)$, $(\alpha \lor \beta)$, $(\alpha \rightarrow \beta)$, and $(\alpha \leftrightarrow \beta)$ has property $R$. 

**Inductive step:**

**Hypothesis:** formulas $\alpha$ and $\beta$ both have property $R$.

**To prove:** each of the formulas $\neg \alpha$, $(\alpha \land \beta)$, $(\alpha \lor \beta)$, $(\alpha \rightarrow \beta)$, and $(\alpha \leftrightarrow \beta)$ has property $R$.

W.l.o.g., we consider $(\alpha \land \beta)$.

Notation: For any formula $\gamma$, let $op(\gamma)$ denote the number of '(' in $\gamma$, and let $cl(\gamma)$ denote the number of ')' in $\gamma$.

We calculate $op((\alpha \land \beta))$:

$$op((\alpha \land \beta)) = 1 + op(\alpha) + op(\beta)$$

inspection
**Inductive step:**

**Hypothesis:** formulas $\alpha$ and $\beta$ both have property $R$.

**To prove:** each of the formulas $(\neg \alpha)$, $(\alpha \land \beta)$, $(\alpha \lor \beta)$, $(\alpha \rightarrow \beta)$, and $(\alpha \leftrightarrow \beta)$ has property $R$.

W.l.o.g., we consider $(\alpha \land \beta)$.

**Notation:** For any formula $\gamma$, let $op(\gamma)$ denote the number of ‘(’ in $\gamma$, and let $cl(\gamma)$ denote the number of ’)’ in $\gamma$.

We calculate $op((\alpha \land \beta))$:

$$op((\alpha \land \beta)) = 1 + op(\alpha) + op(\beta) \quad \text{inspection}$$
$$= 1 + cl(\alpha) + cl(\beta) \quad R(\alpha) \text{ and } R(\beta)$$
Example, cont’d — the Inductive Step

**Inductive step:**

**Hypothesis:** formulas $\alpha$ and $\beta$ both have property $R$.

**To prove:** each of the formulas $(\neg \alpha)$, $(\alpha \land \beta)$, $(\alpha \lor \beta)$, $(\alpha \rightarrow \beta)$, and $(\alpha \leftrightarrow \beta)$ has property $R$.

W.l.o.g., we consider $(\alpha \land \beta)$.

Notation: For any formula $\gamma$, let $\text{op}(\gamma)$ denote the number of ’(’ in $\gamma$, and let $\text{cl}(\gamma)$ denote the number of ’)’ in $\gamma$.

We calculate $\text{op}((\alpha \land \beta))$:

\[\text{op}((\alpha \land \beta)) = 1 + \text{op}(\alpha) + \text{op}(\beta)\]
\[= 1 + \text{cl}(\alpha) + \text{cl}(\beta)\]
\[= \text{cl}((\alpha \land \beta))\]
**Theorem.** Every well-formed Propositional formula is exactly one of an atom, $(\neg \alpha)$, $(\alpha \land \beta)$, $(\alpha \lor \beta)$, $(\alpha \rightarrow \beta)$ or $(\alpha \leftrightarrow \beta)$; and in each case it is of that form in exactly one way.

We want to prove this using structural induction. How will it go?
Back to the Unique Readability Theorem

**Theorem.** Every well-formed Propositional formula is exactly one of an atom, \((\neg \alpha)\), \((\alpha \land \beta)\), \((\alpha \lor \beta)\), \((\alpha \to \beta)\) or \((\alpha \leftrightarrow \beta)\); and in each case it is of that form in exactly one way.

We want to prove this using structural induction. How will it go?

The proof will consider formulas of the form \((\alpha \to \beta)\). One such is our previous example \(((p \land q) \to r)\), which has \((p \land q)\) for \(\alpha\) and \(r\) for \(\beta\).
Back to the Unique Readability Theorem

**Theorem.** Every well-formed Propositional formula is exactly one of an atom, (¬\(\alpha\)), (\(\alpha \land \beta\)), (\(\alpha \lor \beta\)), (\(\alpha \rightarrow \beta\)) or (\(\alpha \leftrightarrow \beta\)); and in each case it is of that form in exactly one way.

We want to prove this using structural induction. How will it go?

The proof will consider formulas of the form (\(\alpha \rightarrow \beta\)). One such is our previous example ((\(p \land q\) → \(r\)), which has (\(p \land q\)) for \(\alpha\) and \(r\) for \(\beta\).

Is this the only way to write the formula (\(p \land q\) → \(r\))?
Theorem. Every well-formed Propositional formula is exactly one of an atom, \((\neg \alpha)\), \((\alpha \land \beta)\), \((\alpha \lor \beta)\), \((\alpha \rightarrow \beta)\) or \((\alpha \leftrightarrow \beta)\); and in each case it is of that form in exactly one way.

We want to prove this using structural induction. How will it go?

The proof will consider formulas of the form \((\alpha \rightarrow \beta)\). One such is our previous example \(((p \land q) \rightarrow r)\), which has \((p \land q)\) for \(\alpha\) and \(r\) for \(\beta\).

Is this the only way to write the formula \(((p \land q) \rightarrow r)\)? What about

\[
((p \land q) \rightarrow r) = (\alpha' \land \beta'),
\]

where \(\alpha'\) is the expression "(p" and \(\beta'\) is the expression "q)→r"?
**Theorem.** Every well-formed Propositional formula is exactly one of an atom, \((\neg \alpha)\), \((\alpha \land \beta)\), \((\alpha \lor \beta)\), \((\alpha \rightarrow \beta)\) or \((\alpha \leftrightarrow \beta)\); and in each case it is of that form in exactly one way.

We want to prove this using structural induction. How will it go?

The proof will consider formulas of the form \((\alpha \rightarrow \beta)\). One such is our previous example \(((p \land q) \rightarrow r)\), which has \((p \land q)\) for \(\alpha\) and \(r\) for \(\beta\).

Is this the only way to write the formula \(((p \land q) \rightarrow r)\)? What about

\[ ((p \land q) \rightarrow r) = (\alpha' \land \beta') , \]

where \(\alpha'\) is the expression “\(p\)” and \(\beta'\) is the expression “\(q) \rightarrow r””? 

Fortunately, neither \(\alpha'\) nor \(\beta'\) is a formula. (Why?)
Does It Always Work?

The theorem worked out for one example.

How can we make sure the inductive step works for every formula? That is, if \((\alpha' \land \beta')\) is the same expression as \((\alpha \rightarrow \beta)\), how can we argue that neither \(\alpha'\) nor \(\beta'\) can be a formula?

Can \(\alpha'\) (or \(\beta'\)) have an equal number of left and right parentheses? If not, why not?
Does It Always Work?

The theorem worked out for one example.

How can we make sure the inductive step works for every formula? That is, if \((\alpha' \land \beta')\) is the same expression as \((\alpha \rightarrow \beta)\), how can we argue that neither \(\alpha'\) nor \(\beta'\) can be a formula?

Can \(\alpha'\) (or \(\beta'\)) have an equal number of left and right parentheses? If not, why not?

To do the proof, we actually need to know more about formulas.

This illustrates a common feature of inductive proofs: they often prove more than just the statement given in the theorem.
Proving Unique Readability

Property $P(\varphi)$:

A formula $\varphi$ has property $P$ iff it satisfies all three of the following.

A: The first symbol of $\varphi$ is either ‘(’ or a variable.

B: $\varphi$ has an equal number of ‘(’ and ‘)’, and each proper prefix of $\varphi$ has more ‘(’ than ‘)’.

C: $\varphi$ has a unique construction as a formula.

(Proper prefix of $\varphi$ is a non-empty expression $x$ such that $\varphi$ is $xy$ for some non-empty expression $y$.)

We prove property $P(\varphi)$ for all formulas $\varphi$, by Structural Induction.

The basis (an atom) is trivial.
Starting the Inductive Step

**Inductive hypothesis:** Formulas \( \beta \) and \( \gamma \) each has property \( P \).

**Inductive step:** Let \( \alpha \) be \( (\beta \star \gamma) \), for \( \star \) a binary connective.

Clearly, \( \alpha \) has property A; we must show it also has B and C.

For property B, we consider all proper prefixes of \( \alpha \). They are

- The expressions \( "(\)", \( "(\beta)\)", \( "(\beta \star)\)", and \( "(\beta \star \gamma)\)".
- Any expression \( "(x)\)", where \( x \) is a proper prefix of \( \beta \).
- Any expression \( "(\beta \star x)\)", where \( x \) is a proper prefix of \( \gamma \).

The formulas \( \beta \) and \( \gamma \), by B of the inductive hypothesis, each have the same number of open and close parentheses. Also by B, an expression \( x \) in the second and third cases has more ‘(’ than ‘)’. Thus, the prefix of \( \alpha \) has more open than close parentheses in each case; \( \alpha \) satisfies B.
Proving Property C

We have assumed that $\alpha$ is $(\beta \star \gamma)$, where both $\beta$ and $\gamma$ have property $P$.

For property C, we must show

If $\alpha$ is $(\beta' \star' \gamma')$ for formulas $\beta'$ and $\gamma'$, then $\beta = \beta'$, $\star = \star'$ and $\gamma = \gamma'$.

If $|\beta'| = |\beta|$, then $\beta' = \beta$ (both start at the second symbol of $\alpha$).
Thus also $\star = \star'$ and $\gamma = \gamma'$, as required.

If $0 < |\beta'| < |\beta|$, then $\beta'$ is a proper prefix of $\beta$.
Thus, by hypothesis $[\beta$ has B$]$, $\beta'$ is not a formula; we have nothing to prove.

If $|\beta'| > |\beta|$, then $\beta' = \beta \star y$, where $y$ is a proper prefix of $\gamma$ (or is empty).
By hypothesis, $\beta$ has equally many ‘(’ and ‘)’ $[P(\beta)]$,
while $y$ has more ‘(’ than ‘)’ $[P(\gamma)]$. Thus $\beta'$ has more ‘(’ than ‘)’;
it is not a formula, and we have nothing to prove.

Therefore $\alpha$ has a unique derivation; it has property C, as required.
Concluding the Proof

By the principle of structural induction, every Propositional formula has properties A, B and C.

This shows that Unique Readability (property C) holds for every Propositional Formula.

This is what we set out to prove.
Commentary

The “goal” of the proof is property C—unique formation.

However, properties A and B are required in order to prove C.

There are actually two equally good options for a proof:

- Prove A, B and C simultaneously, as a single “compound” property. (As done here.)
- Prove them separately: first A, then B, and finally C. (The text uses this method.)

Two fundamental techniques:

- If a proof doesn’t work, go back and fix it—as often as necessary.
- Start from the end and work backwards.
Two consequences of unique formation

We shall define the semantics (meaning) of a formula from its syntax. Thus unique formation ensures unambiguous formulas.

Given a formula, determine its sub-formulas by counting parentheses.

\[
1 \quad 2 \quad \ldots \quad m \quad m + 1 \quad m + 2 \quad \ldots \quad n - 1 \quad n
\]

\[
( \underbrace{\langle \text{rest of subformula} \rangle} \quad \ast \quad \langle \text{subformula 2} \rangle )
\]

*To determine* \( m \): count excess of ‘(’ over ‘)’

When the count returns to zero, the subformula has ended.

*(For efficient parsing of more-complicated formulas/programs, see CS 241.)*
Propositional Logic:
Semantics
The **semantics** of a logic describes how to interpret the well-formed formulas of the logic.

The semantics of propositional logic is “compositional”; in other words, the meaning of a whole formula derives from the meanings of its parts.

In propositional logic, we need to give meaning to atoms, connectives, and formulas.

For example, the interpretation of formula \((p \land q)\) depends on three things: the meaning of \(p\), the meaning of \(q\), and the meaning of \(\land\).
**Definition:**

A *truth valuation* is a function with the set of all proposition symbols as domain and \{F, T\} as range.

(Symbolically, a function \(t : \mathcal{P} \mapsto \{F, T\}\).)

In other words, a truth valuation assigns a value to every propositional variable.

- If \(t(p) = T\), then we say/write, “\(t\) makes \(p\) true”.
- If \(t(p) = F\), then we say/write, “\(t\) makes \(p\) false”.

A propositional variable has no intrinsic meaning; it gets a meaning only via a valuation.
Let $\alpha$ and $\beta$ be two formulas that express propositions $A$ and $B$. Intuitively, we give the following meanings to combinations.

\[
\begin{align*}
(\neg \alpha) & \quad \text{Not } A \\
(\alpha \land \beta) & \quad A \text{ and } B \\
(\alpha \lor \beta) & \quad A \text{ or } B \\
(\alpha \rightarrow \beta) & \quad \text{If } A, \text{ then } B \\
(\alpha \leftrightarrow \beta) & \quad A \text{ if and only if } B
\end{align*}
\]

The English, however, can be ambiguous. We want precise meanings for formulas.
Semantics of Connectives

Formally, a connective represents a function from truth values to truth values.

The connective $\neg$ is unary; it maps one value to one value. We can show its function in a picture, known as a truth table:

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$\neg\alpha$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

The other connectives are binary; they map two values to one value. Thus their truth tables require four lines to cover the possibilities.
Truth tables for connectives

The binary connectives:

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$(\alpha \land \beta)$</th>
<th>$(\alpha \lor \beta)$</th>
<th>$(\alpha \rightarrow \beta)$</th>
<th>$(\alpha \leftrightarrow \beta)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

$\land$ is as expected: $(\alpha \land \beta)$ is true if and only if both $\alpha$ and $\beta$ are true.

The column for $\lor$ shows that it means “one or the other or both”. (This is called “inclusive or”.)

The column for $\rightarrow$ may look a bit strange.
Some people find the meaning of → rather unintuitive. You may want to think of → as meaning “\textit{truth is preserved}”.

- The meaning of \(T \rightarrow T\) is \(T\) because truth is preserved.
- The meaning of \(T \rightarrow F\) is \(F\) because truth is not preserved.
- The meaning of \(F \rightarrow T\) and \(F \rightarrow F\) are both \(T\), because there is no truth to preserve.

For example, the following sentence comes out true:

\[
\text{If everyone is a child, then the moon is made of green cheese.}
\]

Some people prefer to call that sentence non-sensical, rather than true. But propositional logic gives every formula a meaning.
Summary: value of a formula

Fix a truth valuation \( t \). Every formula \( \alpha \) has a value under \( t \), denoted \( \alpha^t \), determined as follows.

1. \( p^t = t(p) \).

2. \( (\neg \alpha)^t = \begin{cases} T & \text{if } \alpha^t = F \\ F & \text{if } \alpha^t = T \end{cases} \)

3. \( (\alpha \land \beta)^t = \begin{cases} T & \text{if } \alpha^t = \beta^t = T \\ F & \text{otherwise} \end{cases} \)

4. \( (\alpha \lor \beta)^t = \begin{cases} T & \text{if } \alpha^t = T \text{ or } \beta^t = T \\ F & \text{otherwise} \end{cases} \)

5. \( (\alpha \rightarrow \beta)^t = \begin{cases} T & \text{if } \alpha^t = F \text{ or } \beta^t = T \\ F & \text{otherwise} \end{cases} \)

6. \( (\alpha \leftrightarrow \beta)^t = \begin{cases} T & \text{if } \alpha^t = \beta^t \\ F & \text{otherwise} \end{cases} \)

The value of a formula comes from the values of its variables, combined as given by its connectives.

The valuation \( t \) is necessary. Without a valuation, a formula has no value.
Evaluating Formulas

Recall that propositional logic is *compositional*. The value of two subformulas, determines the value of their composition using a propositional connective. Given a valuation \( t \):

\[
p^t = t(p)
\]

\[
(\neg \alpha)^t = \begin{cases} 
T & \text{if } \alpha^t = F \\
F & \text{if } \alpha^t = T 
\end{cases}
\]

\[
(\alpha \land \beta)^t = \begin{cases} 
T & \text{if } \alpha^t = \beta^t = T \\
F & \text{otherwise} 
\end{cases}
\]

\[
(\alpha \lor \beta)^t = \begin{cases} 
T & \text{if } \alpha^t = T \text{ or } \beta^t = T \\
F & \text{otherwise} 
\end{cases}
\]

\[
(\alpha \rightarrow \beta)^t = \begin{cases} 
T & \text{if } \alpha^t = F \text{ or } \beta^t = T \\
F & \text{otherwise} 
\end{cases}
\]

\[
(\alpha \leftrightarrow \beta)^t = \begin{cases} 
T & \text{if } \alpha^t = \beta^t \\
F & \text{otherwise} 
\end{cases}
\]

Using these rules, we can build a *truth table* considering all combinations.

For a formula with \( n \) variables, the full truth table has \( 2^n \) lines.
Example: evaluating a formula

**Example.** The truth table of \((p \lor q) \rightarrow (q \land r)\).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>(p \lor q)</th>
<th>(q \land r)</th>
<th>((p \lor q) \rightarrow (q \land r))</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

Note: Each possible truth valuation (on 3 variables) has a line in the truth table.
Exercise

Build the truth table of \(((p \rightarrow (\neg q)) \rightarrow (q \lor (\neg p)))\).
Another Example of Structural Induction

To illustrate Structural Induction, we present a very simple example.

**Claim:** Fix a truth valuation $t$. Every formula $\alpha$ has a value $\alpha^t$ in \{F, T\}.

**Proof:** The property for $R(\alpha)$ is “$\alpha$ has a value $\alpha^t$ in \{F, T\}”.

1. If $\alpha$ is atomic (i.e. a Propositional variable), then $t$ assigns it a value (by the definition of a truth valuation).
2. If $\alpha$ has a value in \{F, T\}, then $(\neg \alpha)$ also does, as given on Slide 56.
3. If $\alpha$ and $\beta$ each has a value in \{F, T\}, then $(\alpha \star \beta)$ also does for every binary connective $\star$, as given on Slide 56.

Thus, by the principle of Structural Induction, every formula has a value.

Now by the unique readability of formulas, we have proved that a formula has **only one** value under any truth valuation, $t$. 
Tautology, Satisfaction, Contradiction

Definition.

A formula $\alpha$ is a **tautology** if and only if for every truth valuation $t$, $\alpha^t = T$.

A formula $\alpha$ is a **contradiction** if and only if for every truth valuation $t$, $\alpha^t = F$.

A formula $\alpha$ is **satisfiable** if and only if there is some truth valuation $t$ such that $\alpha^t = T$.

Note: a formula is satisfiable if and only if it is not a contradiction.

Example. The formula $(p \land (\neg p))$ is a contradiction.

Example. $\left( ((p \land q) \rightarrow r) \land (p \rightarrow q) \right) \rightarrow (p \rightarrow r)$ is satisfiable. (Consider setting each variable to $T$.)
Example. Is \(((p \land q) \rightarrow (\neg r)) \land (p \rightarrow q)) \rightarrow (p \rightarrow (\neg r))\) a tautology?

One method: Fill out a truth table.
Every line will end up with \(T\) in the final column.

For larger formulas, with more variables, this can take a long time.

Can we do better? Is there some other method?
“Short-Cutting” a Truth Table

Rather than fill out an entire truth table, we can analyze what would happen if we did.

Let $A$ be a truth value; that is, $A \in \{F, T\}$. We can combine it with other truth values as follows.

\[
\begin{array}{ccccccc}
\neg T & F & A \land T & A & A \lor T & T & A \rightarrow T & T \\
\neg F & T & A \land F & F & A \lor F & A & A \rightarrow F & \neg A \\
T \land A & A & T \lor A & T & T \rightarrow A & A \\
F \land A & F & F \lor A & A & F \rightarrow A & T \\
A \land A & A & A \lor A & A & A \rightarrow A & T \\
\end{array}
\]

We can use these rules to evaluate a formula, by using a *valuation tree*. A valuation tree may sometimes be much smaller than the corresponding truth table.
Example: Valuation trees

**Example.** Show that \( (((p ∧ q) → (¬r)) ∧ (p → q)) → (p → (¬r))) \) is a tautology.
Example: Valuation trees

Example. Show that \( (((p \land q) \rightarrow (\neg r)) \land (p \rightarrow q)) \rightarrow (p \rightarrow (\neg r)) \) is a tautology.

In valuations with \( t(p) = T \), we put \( T \) in for \( p \):

\[
\left( \left( \left( (T \land q) \rightarrow (\neg r) \right) \land (T \rightarrow q) \right) \rightarrow (T \rightarrow (\neg r)) \right).
\]

From the previous table, this becomes \( \left( ((q \rightarrow (\neg r)) \land q) \rightarrow (\neg r) \right) \).
Example. Show that \( \left( \left( \left( p \land q \right) \rightarrow \left( \neg r \right) \right) \land \left( p \rightarrow q \right) \right) \rightarrow \left( p \rightarrow \left( \neg r \right) \right) \) is a tautology.

In valuations with \( t(p) = T \), we put \( T \) in for \( p \):

\[
\left( \left( \left( \left( T \land q \right) \rightarrow \left( \neg r \right) \right) \land \left( T \rightarrow q \right) \right) \rightarrow \left( T \rightarrow \left( \neg r \right) \right) \right).
\]

From the previous table, this becomes \( \left( \left( \left( q \rightarrow \left( \neg r \right) \right) \land q \right) \rightarrow \left( \neg r \right) \right) \).

If \( t(q) = T \), this yields \( \left( \neg r \rightarrow \left( \neg r \right) \right) \) and then \( T \). (Check!).

If \( t(q) = F \), it yields \( \left( F \rightarrow \left( \neg r \right) \right) \) and then \( T \). (Check!).
Example: Valuation trees

**Example.** Show that \( \left( \left( (p \land q) \rightarrow (\neg r) \right) \land (p \rightarrow q) \right) \rightarrow (p \rightarrow (\neg r)) \) is a tautology.

In valuations with \( t(p) = T \), we put \( T \) in for \( p \):

\[
\left( \left( (T \land q) \rightarrow (\neg r) \right) \land (T \rightarrow q) \right) \rightarrow (T \rightarrow (\neg r))
\]

From the previous table, this becomes \( \left( ((q \rightarrow (\neg r)) \land q) \rightarrow (\neg r) \right) \).

If \( t(q) = T \), this yields \( ((\neg r) \rightarrow (\neg r)) \) and then \( T \). (Check!).

If \( t(q) = F \), it yields \( (F \rightarrow (\neg r)) \) and then \( T \). (Check!).

On the other hand, in valuations with \( t(p) = F \), we get

\[
\left( \left( (F \land q) \rightarrow (\neg r) \right) \land (F \rightarrow q) \right) \rightarrow (F \rightarrow (\neg r))
\]

Simplification yields \( \left( ((F \rightarrow (\neg r)) \land T) \rightarrow T \right) \) and eventually \( T \).

Thus every valuation makes the formula true, as required.
Suppose that a formula \((\alpha \leftrightarrow \beta)\) is a tautology.

Then \(\alpha\) and \(\beta\) must have the same final column in their truth tables—they have the same value under any valuation.

In symbols: \(\alpha^t = \beta^t\), for every valuation \(t\).

Such formulas are called *equivalent* formulas. The notation

\[ \alpha \equiv \beta \]

means that \(\alpha\) and \(\beta\) are equivalent.

Note: \(\alpha \equiv \beta\) does not mean \(\alpha = \beta\) !!
Equivalent formulas are equivalent in any context.

**Lemma.** Suppose that $\alpha \equiv \beta$. Then for any formula $\gamma$, and any connective $\star$, the formulas $(\alpha \star \gamma)$ and $(\beta \star \gamma)$ are equivalent:

$$(\alpha \star \gamma) \equiv (\beta \star \gamma) .$$

Proof idea: a value $(\alpha \star \gamma)^t$ depends only on the values $\alpha^t$ and $\gamma^t$, and the identity of $\star$.

Example: Since $((\neg p) \rightarrow p) \equiv p$ [check this!], we get that $(((\neg p) \rightarrow p) \land q) \equiv (p \land q)$. 
Algebra of Formulas

Many equivalences of formulas look much like rules of ordinary arithmetic and/or algebra.

Commutativity

\[(\alpha \land \beta) \equiv (\beta \land \alpha)\]
\[(\alpha \lor \beta) \equiv (\beta \lor \alpha)\]

Associativity

\[(\alpha \land (\beta \land \gamma)) \equiv ((\alpha \land \beta) \land \gamma)\]
\[(\alpha \lor (\beta \lor \gamma)) \equiv ((\alpha \lor \beta) \lor \gamma)\]

Distributivity

\[(\alpha \lor (\beta \land \gamma)) \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma))\]
\[(\alpha \land (\beta \lor \gamma)) \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma))\]

Idempotence

\[(\alpha \lor \alpha) \equiv \alpha\]
\[(\alpha \land \alpha) \equiv \alpha\]

Double Negation

\[\neg(\neg\alpha) \equiv \alpha\]

De Morgan’s Laws

\[\neg(\alpha \land \beta) \equiv ((\neg\alpha) \lor (\neg\beta))\]
\[\neg(\alpha \lor \beta) \equiv ((\neg\alpha) \land (\neg\beta))\]
Simplification I (Absorbtion)

\[(\alpha \land T) \equiv \alpha\]
\[(\alpha \lor T) \equiv T\]
\[(\alpha \land F) \equiv F\]
\[(\alpha \lor F) \equiv \alpha\]

Simplification II

\[(\alpha \lor (\alpha \land \beta)) \equiv \alpha\]
\[(\alpha \land (\alpha \lor \beta)) \equiv \alpha\]

Implication

\[(\alpha \rightarrow \beta) \equiv ((\neg \alpha) \lor \beta)\]

Contrapositive

\[(\alpha \rightarrow \beta) \equiv ((\neg \beta) \rightarrow (\neg \alpha))\]

Equivalence

\[(\alpha \leftrightarrow \beta) \equiv ((\alpha \rightarrow \beta) \land (\beta \rightarrow \alpha))\]

Excluded Middle

\[(\alpha \lor (\neg \alpha)) \equiv T\]

Contradiction

\[(\alpha \land (\neg \alpha)) \equiv F\]

(“T” and “F” aren’t really formulas, but we’ll pretend.)
Examples of Using Identities

Prove or disprove each of the following.

1. \((p \land q) \lor (q \land r)\) \equiv (q \land (p \lor r))
2. \(((p \land q) \lor (p \land s)) \lor ((r \land q) \lor (r \land s)))\equiv ((p \lor r) \land (q \lor s))
3. (without using Simplification II) \((p \lor (p \land q))\equiv p\)
4. \(\neg((\neg p) \lor (\neg(r \lor s)))\equiv ((p \land r) \lor (p \land s))\)
5. \(\neg((\neg(p \land q)) \lor p)\equiv F\)
6. \((p \land (q \to p))\equiv p\)
7. \((p \land (\neg((\neg q) \land \neg p)) \lor p)\equiv p\)
8. \((p \land ((\neg((\neg q) \land \neg p)) \lor p)\equiv q\)

Note: Apply only one rule per line of your proof (but you may apply the rule multiple times).
We are given a deck of cards, in which each card has a letter on one side and a natural number on the other side.

**Claim:** In these four cards, each card that has a vowel on one side, has an even number on the other side.

Is the claim true?

How many cards must you turn over, in order to determine whether or not the claim is true? Which ones?
A Code Example

if ( (input > 0) OR NOT output ) {
    if ( NOT (output AND (queuelength < 100) ) ) {
        $P_1$
    } else if ( output AND NOT (queuelength < 100) ) {
        $P_2$
    } else {
        $P_3$
    }
} else {
    $P_4$
}

When does each piece of code get executed?

Let $i$: input > 0,
$u$: output,
$q$: queuelength < 100.
A Code Example, cont’d

```c
if ( i || !u ) {
    if ( !(u && q) ){
        \(P_1\)
    }
    else if (u && !q){
        \(P_2\)
    }
    else {
        \(P_3\)
    }
} else {
    \(P_4\)
}
```

<table>
<thead>
<tr>
<th>(i) (u) (q)</th>
<th>((i \lor (\neg u)))</th>
<th>((\neg (u \land q)))</th>
<th>((u \land (\neg q)))</th>
</tr>
</thead>
<tbody>
<tr>
<td>T   T   T</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T   T   F</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T   F   T</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T   F   F</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F   T   T</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F   T   F</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F   F   T</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F   F   F</td>
<td>T</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\(P_2\) is never executed.
A Code Example, cont’d

```java
if ( i || !u ) {
    if ( !(u && q) ) {
        \( P_1 \)
    } else if (u && !q) {
        \( P_2 \)
    } else {
        \( P_3 \)
    }
} else {
    \( P_4 \)
}
```

<table>
<thead>
<tr>
<th>i</th>
<th>u</th>
<th>q</th>
<th>((i \lor (\neg u)))</th>
<th>((\neg (u \land q)))</th>
<th>((u \land (\neg q)))</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

\(P_1\) is never executed.
A Code Example, cont’d

```c
if ( i || !u ) {
    if ( !(u && q) ){
        \( P_1 \)
    }
    else if (u && !q) {
        \( P_2 \)
    }
    else {
        \( P_3 \)
    }
} else {
    \( P_4 \)
}
```

<table>
<thead>
<tr>
<th>( i )</th>
<th>( u )</th>
<th>( q )</th>
<th>((i \lor (\neg u)))</th>
<th>((\neg (u \land q)))</th>
<th>((u \land (\neg q)))</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>( P_1 )</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>( P_1 )</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>( P_1 )</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>( P_4 )</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>( P_4 )</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>( P_1 )</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>( P_1 )</td>
</tr>
</tbody>
</table>

\( P_2 \) is never executed.
Prove that $P_2$ is dead code.

The conditions leading to $P_2$ can never be true.

\[
\left( \left( \left( i \lor (\neg u) \right) \land \neg \left( \neg \left( u \land q \right) \right) \right) \right) \land \left( u \land (\neg q) \right)
\]
\[
\equiv \left( \left( i \lor (\neg u) \right) \land (u \land q) \right) \land \left( u \land (\neg q) \right)
\]
\[
\equiv (i \lor (\neg u)) \land ((u \land q) \land (u \land (\neg q)))
\]
\[
\equiv (i \lor (\neg u)) \land (u \land (q \land (\neg q)))
\]
\[
\equiv (i \lor (\neg u)) \land (u \land F)
\]
\[
\equiv (i \lor (\neg u)) \land F
\]
\[
\equiv F
\]
Prove that $P_3$ is live code.

The conditions leading to $P_3$ can be true.

$P_3$ is executed when the formula

$$
\left( (i \lor (\neg u)) \land \left( (\neg (\neg (u \land q))) \land (\neg (u \land (\neg q))) \right) \right)
$$

is true.

Find a satisfying truth valuation for this formula.

For example: $t(i) = T, t(u) = T, t(q) = T$. 
Consider these two fragments of code. Are they equivalent?

**Fragment 1:**

```c
if ( i || !u ) {
    if ( !(u && q) ) {
        P1
    } else if ( u && !q ) {
        P2
    } else {
        P3
    }
} else {
    P4
}
```

**Fragment 2:**

```c
if ( i && u && q ) {
    P3
} else if ( !i && u ) {
    P4
} else {
    P1
}
```
Simplifying Code

To prove that the two fragments are equivalent, show that each block of code $P_1$, $P_2$, $P_3$, and $P_4$ is executed under equivalent conditions.

<table>
<thead>
<tr>
<th>Block</th>
<th>Fragment 1</th>
<th>Fragment 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>$(i \lor (\neg u)) \land (\neg (u \land q))$</td>
<td>$(\neg (i \land u \land q)) \land (\neg ((\neg i) \land u))$</td>
</tr>
<tr>
<td>$P_2$</td>
<td>$(i \lor (\neg u)) \land (\neg (\neg (u \land q)))$</td>
<td>$F$</td>
</tr>
<tr>
<td></td>
<td>$\land (u \land (\neg q))$</td>
<td></td>
</tr>
<tr>
<td>$P_3$</td>
<td>$(i \lor (\neg u)) \land (\neg (\neg (u \land q)))$</td>
<td>$(i \land u \land q)$</td>
</tr>
<tr>
<td></td>
<td>$\land (\neg (u \land (\neg q)))$</td>
<td></td>
</tr>
<tr>
<td>$P_4$</td>
<td>$(\neg (i \lor (\neg u)))$</td>
<td>$(\neg (i \land u \land q)) \land ((\neg i) \land u)$</td>
</tr>
</tbody>
</table>
• An electronic computer is made up of a number of **circuits**.

• The basic elements of circuits are called **logic gates**.

• A **logic gate** is an electronic device that operates on a collection of binary inputs and produces a binary output.
Logical Gates

- AND gate: □
- OR gate: △
- NOT gate: ︵
- XOR gate: ︵ △
Your instructors, Alice, Carmen, and Collin, are choosing questions to include on the midterm. For each candidate problem, each instructor votes either yes or not. A question is included if and only if it receives two or more yes votes. Design a circuit, which outputs $T$ whenever a question is included.
### Draw the truth table

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
Draw the truth table

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
Design the circuit

1. Create a Propositional formula which has the required truth table.

For convenience, we will use the symbol $\oplus$ to represent an exclusive OR connective, having this truth table.

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>$(x \oplus y)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>

Using $\oplus$ is a temporary convenience only. You are not allowed to use this connective unless it is explicitly specified in the problem that you can.

2. Then, create a circuit which mirrors the formula.
Solution 1

1. Convert each row of the truth table with output T into a conjunction.
   
   - \((x \land y) \land z\)
   - \((x \land y) \land \lnot z\)
   - \((x \land \lnot y) \land z\)
   - \((\lnot x \land y) \land z\)

2. Connect all conjunctions to form a disjunction.

   \[((x \land y) \land z) \lor ((x \land y) \land \lnot z) \lor ((x \land \lnot y) \land z) \lor ((\lnot x \land y) \land z)\]
Draw the circuit.
Solution 2

1. Converts rows 1-3 to a propositional formula.
   \((x \land (y \lor z))\)

2. Convert row 5 to a propositional formula.
   \(((\neg x) \land y) \land z\)

3. Connect all formulas into a disjunction.
   \((x \land (y \lor z)) \lor (((\neg x) \land y) \land z)\)
Draw the circuit.
Solution 3

1. Convert rows 1 and 5 into a propositional formula.
   \((y \land z)\)

2. Convert rows 2 and 3 into a propositional formula.
   \((x \land (y \oplus z))\)

3. Connect all formulas into a disjunction.
   \((y \land z) \lor (x \land (y \oplus z))\)
Draw the circuit.

\[ z \quad y \quad x \]
Restricted Use of Connectives:
Adequate sets and normal forms
Formulas \((\alpha \rightarrow \beta)\) and \(((\neg \alpha) \lor \beta)\) are equivalent.

Thus \(\rightarrow\) is said to be *definable* in terms of \(\neg\) and \(\lor\).

We never need to use \(\rightarrow\); we can always write an equivalent formula without it.

There are actually sixteen possible binary connectives. (Why?)

Of these, two are essentially nullary (they ignore the values they connect).

Four others are essentially unary (they ignore one value but not the other).
A set of connectives is said to be *adequate* iff any \( n \)-ary \((n \geq 1)\) connective can be defined in terms of the ones in the set.

**Lemma.** \( \{\land, \lor, \neg\} \) is an adequate set of connectives.

Proof: see the equivalence rules “Implication” and “Equivalence”.

**Lemma.** Each of the sets \( \{\land, \neg\} \), \( \{\lor, \neg\} \), and \( \{\rightarrow, \neg\} \) is adequate.

Proof: For the first two, use De Morgan’s laws. For the third, …?

**Theorem.** The set \( \{\land, \lor\} \) is *not* an adequate set of connectives.
**Question:** Is there a connective \( * \) such that the singleton set \( \{ * \} \) is adequate?

**Question:** Are there connectives \( c_1, c_2 \) and \( c_3 \) such that \( \{ c_1, c_2, c_3 \} \) is adequate, but none of \( \{ c_1, c_2 \}, \{ c_1, c_3 \}, \) or \( \{ c_2, c_3 \} \) is adequate? (Such a set is called a *minimal* adequate set.)

**Question:** Find all minimal adequate sets containing only binary, unary and nullary connectives.
Semantic Entailment
The notion of satisfiability extends to sets of formulas.

Let $\Sigma$ denote a set of formulas and $t$ a valuation. Define

$$\Sigma^t = \begin{cases} 
T & \text{if for each } \beta \in \Sigma, \beta^t = T \\
F & \text{otherwise}
\end{cases}$$

When $\Sigma^t = T$, we say that $t$ satisfies $\Sigma$.

A set $\Sigma$ is satisfiable iff there is some valuation $t$ such that $\Sigma^t = T$.

*Example.* The set $\{(p \rightarrow q) \lor r), (p \lor (q \lor s))\}$ is satisfiable.
Logical Consequence, a.k.a. Entailment

Let $\Sigma$ be a set of formulas, and let $\alpha$ be a formula. We say that

- $\alpha$ is a **logical consequence** of $\Sigma$, or
- $\Sigma$ **(semantically) entails** $\alpha$, or
- in symbols, $\Sigma \models \alpha$,

if and only if for any truth valuation $t$,

$$\text{if } \Sigma^t = T \text{ then also } \alpha^t = T.$$ 

We write $\Sigma \not\models \alpha$ for “not $\Sigma \models \alpha$”. That is, there exists a truth valuation $t$ such that $\Sigma^t = T$ and $\alpha^t = F$. 
Example.

\[ \{ (p \rightarrow q), (q \rightarrow r) \} \models (p \rightarrow r) . \]
Examples: Entailment

Example.

\[\{(p \rightarrow q), (q \rightarrow r)\} \models (p \rightarrow r) .\]

Example.

\[\{((p \rightarrow (\neg q)) \lor r), (q \land (\neg r)), (p \leftrightarrow r)\} \not\models (p \land (q \rightarrow r)) .\]
Examples: Entailment

**Example.**

\[\{ (p \to q), (q \to r) \} \models (p \to r) .\]

**Example.**

\[\{ ((p \to (\neg q)) \lor r), (q \land (\neg r)), (p \leftrightarrow r) \} \not\models (p \land (q \to r)) .\]

**Example.** \(\emptyset \models \alpha\) means that \(\alpha\) is a tautology. Why?
Examples: Entailment

Example.

\[ \{ (p \rightarrow q), (q \rightarrow r) \} \models (p \rightarrow r) \, . \]

Example.

\[ \{ ((p \rightarrow (\neg q)) \lor r), (q \land (\neg r)), (p \leftrightarrow r) \} \not\models (p \land (q \rightarrow r)) \, . \]

Example. \( \emptyset \models \alpha \) means that \( \alpha \) is a tautology. Why?

Example. \( \{ \alpha, (\neg \alpha) \} \models \beta \) is always true, whatever \( \alpha \) and \( \beta \) are. Why?
Equivalence and Entailment

Equivalence can be expressed using the notion of entailment.

Lemma. \( \alpha \equiv \beta \) if and only if both \( \{\alpha\} \models \beta \) and \( \{\beta\} \models \alpha \).
Proof Systems in Propositional Logic
What Is a “Proof”? 

A *proof* is a formal demonstration that a statement is true.

- It must be mechanically checkable. A reader need not apply any intuition or insight to verify that it is correct.
- In fact, a computer could verify its correctness.

A proof is generally syntactic, rather than semantic.

- Syntactic rules permit mechanical checking.
- The rules are chosen for semantic reasons, but their use remains purely syntactic.
What Makes a Proof?

Generically, a proof consists of a sequence of formulas.

- The premises, if any, appear first.
- Each subsequent formula must be a valid *inference* from preceding formulas.
  That is, there is an *inference rule* (defined by the proof system) that justifies the formula, based on the previous ones.
- The final formula is the conclusion.

The key here is the set of inference rules. A set of inference rules defines a *proof system*.

We notate “there is a proof with premises $\Sigma$ and conclusion $\varphi$” by

$$\Sigma \vdash \varphi.$$
Inference Rules

In general, an inference rule is written as

\[
\frac{\alpha_1 \ \alpha_2 \ \ldots \ \alpha_i}{\beta}.
\]

This means,

Suppose that each of the formulas \( \alpha_1, \alpha_2, \ldots, \alpha_i \) already appears in the proof (either assumed or previously inferred).

Then one may infer the formula \( \beta \) – i.e., write it as the next formula of the proof.

Examples of possible rules:

\[
\frac{\alpha \ \beta}{\alpha \land \beta} \ 	ext{A kind of definition of } \land.
\]

\[
\frac{\alpha \land \beta}{\alpha \lor \beta} \ 	ext{Rules need not be equivalences.}
\]
Questions About Proofs

Given a sequence of formulas, is it a proof?

Determined by examining the sequence, formula by formula. If the sequence always follows the rules, it is a proof; if it ever does not, then it is not a proof.

Why might we want a proof?

For some (carefully constructed) proof systems, the existence of a proof implies that the conclusion is a logical consequence of the premises. Such a system is called sound. If $S$ is a sound proof system,

$$
\Sigma \vdash_S \varphi \text{ implies } \Sigma \models \varphi.
$$
Proofs in Propositional Logic: Natural Deduction
We will consider a proof system called Natural Deduction.

- It closely follows how people (mathematicians, at least) normally make formal arguments.
- It extends easily to more-powerful forms of logic.
Overview of Natural Deduction

As with any proof system, a proof in Natural Deduction consists of a sequence of formulas, in some order, each with a justification.

- Natural Deduction does direct proof.
- Assumptions (formulas without a justification) play a crucial role.
- Using an assumption creates a “sub-proof”. Formulas inside a sub-proof may not be used outside it.
  An inference rule may refer to a completed sub-proof.

We use the same notation as before for existence of a proof. If there is a proof of a formula $\varphi$ from a set $\Sigma$ of assumptions, we write

$$\Sigma \vdash_{ND} \varphi \quad \text{or simply} \quad \Sigma \vdash \varphi.$$
The Basic Rules of Natural Deduction

The simplest rule is, if you have a formula in the proof already, you may write it down again. This is called \textit{reflexivity}.

We will write rules like this:

<table>
<thead>
<tr>
<th>Name</th>
<th>$\vdash$-notation</th>
<th>inference notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reflexivity, \ or Premise</td>
<td>$\Sigma, \alpha \vdash \alpha$</td>
<td>$\frac{\alpha}{\alpha}$</td>
</tr>
</tbody>
</table>

The notation on the right is as we had before: if we have the formula above the line available, we may write the formula below the line in the proof.

The version in the center reminds us of the role of assumptions in Natural Deduction. Other rules will make more use of it.
A First Example

Here is a proof of \( \{p, q\} \vdash p \).

1. \( p \) \hspace{1em} \text{Premise}
2. \( q \) \hspace{1em} \text{Premise}
3. \( p \) \hspace{1em} \text{Reflexivity: 1}

Alternatively, we could simply write

1. \( p \) \hspace{1em} \text{Premise}

and be done.

(Note: “extra” formulas never hurt anything.)
Rules for Conjunction: $\land i$

Each connective symbol has an “introduction rule” to conclude formulas that contain it, and an “elimination rule” to conclude a formula that removes it from an earlier formula.

We start with the introduction rule for $\land$.

<table>
<thead>
<tr>
<th>Name</th>
<th>$\vdash$-notation</th>
<th>inference notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\land$-introduction ($\land i$)</td>
<td>If $\Sigma \vdash \alpha$ and $\Sigma \vdash \beta$, then $\Sigma \vdash (\alpha \land \beta)$</td>
<td>$\alpha \beta \vdash (\alpha \land \beta)$</td>
</tr>
</tbody>
</table>

Rule $\land i$ means

If each of the formulas $\alpha$ and $\beta$ already appears in the proof, then we may write the formula $\alpha \land \beta$ as the next formula of the proof.
The elimination rule for $\land$ basically “undoes” the introduction.

<table>
<thead>
<tr>
<th>Name</th>
<th>$\vdash$-notation</th>
<th>Inference notation</th>
</tr>
</thead>
</table>
| $\land$-elimination ($\land e$) | If $\Sigma \vdash (\alpha \land \beta)$, then $\Sigma \vdash \alpha$ and $\Sigma \vdash \beta$ | $(\alpha \land \beta)$
|                    |                                       | $\alpha$ $\vdash$ $\beta$ |

Rule $\land e$ means

If the formula $(\alpha \land \beta)$ already appears in the proof, then we may write either $\alpha$ or $\beta$ as the next formula of the proof.
Example: Conjunction Rules

*Example.* Show that \( \{ (p \land q) \} \vdash (q \land p) \).

1. \( (p \land q) \) \hspace{1cm} Premise
2. \( q \) \hspace{1cm} \&e: 1
3. \( p \) \hspace{1cm} \&e: 1
4. \( (q \land p) \) \hspace{1cm} \&i: 2, 3
Example. Show that \( \{(p \land q), r\} \vdash (q \land r) \).

1. \( (p \land q) \)  Premise
2. \( r \)  Premise
3. \( q \)  \( \land e: 1 \)
4. \( (q \land r) \)  \( \land i: 3, 2 \)
The rule →-elimination requires two formulas earlier in the proof.

<table>
<thead>
<tr>
<th>Name</th>
<th>⊢-notation</th>
<th>inference notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>→-elimination</td>
<td>If $\Sigma \vdash (\alpha \rightarrow \beta)$ and $\Sigma \vdash \alpha$, then $\Sigma \vdash \beta$</td>
<td>$(\alpha \rightarrow \beta) \quad \alpha \quad \beta$</td>
</tr>
</tbody>
</table>

In words:

if you have that $\alpha$ implies $\beta$, and also that $\alpha$, than you may conclude $\beta$.

This rule is often referred to by its Latin name, *modus ponens*.

(Rumours that “modus ponens” is the Latin equivalent of “D’uh!” are untrue, however well justified.)
The $\rightarrow$-introduction rule is our first to employ a sub-proof.

<table>
<thead>
<tr>
<th>Name</th>
<th>$\vdash$-notation</th>
<th>inference notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\rightarrow$-introduction (→i)</td>
<td>If $\Sigma, \alpha \vdash \beta$, then $\Sigma \vdash (\alpha \rightarrow \beta)$</td>
<td>$\begin{array}{c} \alpha \ \vdots \ \beta \end{array}$</td>
</tr>
</tbody>
</table>

The rule uses the formula $\alpha$ as a *hypothesis*, or *assumption*. The assumption functions as a premise in the sub-proof, but it is not a premise of the main proof.

The “box” around the sub-proof of $\Sigma, \alpha \vdash \beta$ reminds us that nothing inside the sub-proof may come out. Outside of the sub-proof, we may use only the whole sub-proof, in a rule (like $\rightarrow$-introduction) that specifies a sub-proof.
Sub-Proof Rules

To use rule $\rightarrow i$, we must have a completed sub-proof.

Assumption Rule:

A sub-proof may be opened at any point.
Its first line, labelled “assumption”, may be any formula.

Sub-proof closure rules:

The most-recently opened sub-proof may be closed at any time.

No formula inside a closed sub-proof may be referenced.
Only the entire sub-proof may be used, once it is closed.

Finally: every sub-proof must be closed before the last line of the proof.
Example. Give a proof of \( \{(p \rightarrow q), (q \rightarrow r)\} \vdash (p \rightarrow r) \).

To start, we write down the premises at the beginning, and the conclusion at the end.

1. \( (p \rightarrow q) \)  Premise
2. \( (q \rightarrow r) \)  Premise

\( (p \rightarrow r) \)  ???

What next?

The goal \( (p \rightarrow r) \) contains \( \rightarrow \). Let's try rule \( \rightarrow i \) …

Inside the sub-proof, we can use rule \( \rightarrow e \). Done!
Example: Rule $\rightarrow i$ and sub-proofs

**Example.** Give a proof of $\{(p \to q), (q \to r)\} \vdash (p \to r)$.

To start, we write down the premises at the beginning, and the conclusion at the end.

1. $(p \to q)$  Premise
2. $(q \to r)$  Premise
3. $p$  Assumption
4. 
5. 
6. $(p \to r)$  $\rightarrow i$: ??

What next?

The goal “$(p \to r)$” contains $\to$. Let’s try rule $\rightarrow i$....
Example: Rule $\rightarrow i$ and sub-proofs

**Example.** Give a proof of $\{(p \rightarrow q), (q \rightarrow r)\} \vdash (p \rightarrow r)$. 

To start, we write down the premises at the beginning, and the conclusion at the end.

1. $(p \rightarrow q)$ Premise
2. $(q \rightarrow r)$ Premise
3. $p$ Assumption
4. $q \rightarrow e: 1, 3$
5. $r \rightarrow e: 2, 4$
6. $(p \rightarrow r) \rightarrow i: ??$

What next?

The goal “$(p \rightarrow r)$” contains $\rightarrow$.

Let’s try rule $\rightarrow i$.

Inside the sub-proof, we can use rule $\rightarrow e$. 

Natural Deduction Implication Rules
Example: Rule $\rightarrow i$ and sub-proofs

**Example.** Give a proof of $\{(p \rightarrow q), (q \rightarrow r)\} \vdash (p \rightarrow r)$.

To start, we write down the premises at the beginning, and the conclusion at the end.

1. $(p \rightarrow q)$  Premise
2. $(q \rightarrow r)$  Premise
3. $p$  Assumption
4. $q$  $\rightarrow e$: 1, 3
5. $r$  $\rightarrow e$: 2, 4
6. $(p \rightarrow r)$  $\rightarrow i$: 3–5

**What next?**

The goal “$(p \rightarrow r)$” contains $\rightarrow$. Let’s try rule $\rightarrow i$...

Inside the sub-proof, we can use rule $\rightarrow e$.

**Done!**
Rules of Disjunction: \( \lor i \) and \( \lor e \)

Rule \( \lor i \) is much like rule \( \land i \).

Rule \( \lor e \), however, is more complicated.

<table>
<thead>
<tr>
<th>Name</th>
<th>( \vdash )-notation</th>
<th>inference notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lor )-introduction ( (\lor i) )</td>
<td>If ( \Sigma \vdash \alpha ), then ( \Sigma \vdash \alpha \land \beta ) and ( \Sigma \vdash \beta \land \alpha )</td>
<td>( \frac{\alpha}{\alpha \lor \beta} )</td>
</tr>
<tr>
<td>( \lor )-elimination ( (\lor e) )</td>
<td>If ( \Sigma, \alpha_1 \vdash \beta ) and ( \Sigma, \alpha_2 \vdash \beta ), then ( \Sigma, \alpha_1 \lor \alpha_2 \vdash \beta )</td>
<td>( \frac{\alpha_1 \lor \alpha_2}{\beta} )</td>
</tr>
</tbody>
</table>

Rule \( \lor e \) is also known as “proof by cases”.

Example: Or-Introduction and \(-\)-Elimination

**Example:** Show that \(\{p \lor q\} \vdash (p \rightarrow q) \lor (q \rightarrow p)\).

1. \(p \lor q\) \hspace{1cm} \text{Premise}
2. \(p\) \hspace{1cm} \text{Assumption}
3. \(q\) \hspace{1cm} \text{Assumption}
4. \(p\) \hspace{1cm} \text{Reflexivity: 2}
5. \(q \rightarrow p\) \hspace{1cm} \rightarrow \text{i: 3–4}
6. \((p \rightarrow q) \lor (q \rightarrow p)\) \hspace{1cm} \lor \text{i: 5}

\[
\begin{align*}
\text{7. } q & \hspace{1cm} \text{Assumption} \\
\text{8. } p & \hspace{1cm} \text{Assumption} \\
\text{9. } q & \hspace{1cm} \text{Reflexivity: 7} \\
\text{10. } p \rightarrow q & \hspace{1cm} \rightarrow \text{i: 8–9} \\
\text{11. } (p \rightarrow q) \lor (q \rightarrow p) & \hspace{1cm} \lor \text{i: 10} \\
\text{12. } (p \rightarrow q) \lor (q \rightarrow p) & \hspace{1cm} \lor \text{e: 1, 2–6, 7–11}
\end{align*}
\]
Negation

We shall treat negation by considering contradictions.

We shall use the notation $\bot$ to represent any contradiction. It may appear in proofs as if it were a formula.

The elimination rule for negation:

<table>
<thead>
<tr>
<th>Name</th>
<th>$\vdash$-notation</th>
<th>inference notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\bot$-introduction, or $\neg$-elimination ($\neg$e)</td>
<td>$\Sigma, \alpha, (\neg\alpha) \vdash \bot$</td>
<td>$\alpha \quad (\neg\alpha)$</td>
</tr>
</tbody>
</table>

If we have both $\alpha$ and $(\neg\alpha)$, then we have a contradiction.
Negation Introduction ($\neg i$)

If an assumption $\alpha$ leads to a contradiction, then derive ($\neg \alpha$).

<table>
<thead>
<tr>
<th>Name</th>
<th>$\vdash$-notation</th>
<th>inference notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\neg$-introduction ($\neg i$)</td>
<td>If $\Sigma, \alpha \vdash \bot$, then $\Sigma \vdash (\neg \alpha)$</td>
<td>$\begin{array}{c} \alpha \ \vdots \ \bot \end{array}$ $(\neg \alpha)$</td>
</tr>
</tbody>
</table>
Example. Show that \( \{\alpha \rightarrow (\neg \alpha)\} \vdash (\neg \alpha) \).
**Example.** Show that \( \{\alpha \rightarrow (\neg \alpha)\} \vdash (\neg \alpha) \).

1. \( \alpha \rightarrow (\neg \alpha) \) \quad \text{Premise}

\[ (\neg \alpha) \quad ?? \]
Example. Show that $\{\alpha \rightarrow (\neg \alpha)\} \vdash (\neg \alpha)$.

1. $\alpha \rightarrow (\neg \alpha)$  Premise
2. $\alpha$  Assumption
3.  
4. $\bot$  ??
5. $(\neg \alpha)$  $\neg i$: 2–?
Example. Show that \( \{ \alpha \rightarrow (\neg \alpha) \} \vdash (\neg \alpha) \).

1. \( \alpha \rightarrow (\neg \alpha) \)  \hspace{1cm} \text{Premise}

2. \( \alpha \)  \hspace{1cm} \text{Assumption}

3. \( (\neg \alpha) \)  \hspace{1cm} \rightarrow e: 1, 2

4. \( \bot \)  \hspace{1cm} ??

5. \( (\neg \alpha) \)  \hspace{1cm} \neg i: 2–?
Example. Show that \( \{\alpha \rightarrow (\neg \alpha)\} \vdash (\neg \alpha) \).

1. \( \alpha \rightarrow (\neg \alpha) \quad \text{Premise} \)
2. \( \alpha \quad \text{Assumption} \)
3. \( (\neg \alpha) \quad \rightarrow e: 1, 2 \)
4. \( \bot \quad \neg e: 2, 3 \)
5. \( (\neg \alpha) \quad \neg i: 2–4 \)
The Last Two Basic Rules

Double-Negation Elimination:

<table>
<thead>
<tr>
<th>Name</th>
<th>⊢-notation</th>
<th>Inference notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>¬¬-elimination</td>
<td>If ( \Sigma \vdash (\neg(\neg\alpha)) ), then ( \Sigma \vdash \alpha )</td>
<td>( (\neg(\neg\alpha)) ) ( \alpha )</td>
</tr>
</tbody>
</table>

Contradiction Elimination:

<table>
<thead>
<tr>
<th>Name</th>
<th>⊢-notation</th>
<th>Inference notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>⊥-elimination</td>
<td>If ( \Sigma \vdash \bot ), then ( \Sigma \vdash \alpha )</td>
<td>( \bot ) ( \alpha )</td>
</tr>
</tbody>
</table>
A Redundant Rule

The rule of $\bot$-elimination is not actually needed.

Suppose a proof has

27. $\bot$ \textit{(some rule)}
28. $\alpha$ $\bot$e: 27.

We can replace these by

27. $\bot$ \textit{(some rule)}
28. $(\neg \alpha)$ Assumption
29. $\bot$ Reflexivity: 27
30. $(\neg (\neg \alpha))$ $\neg$i: 28–29
31. $\alpha$ $\neg \neg$e: 30.

Thus any proof that uses $\bot$e can be modified into a proof that does not.
Example: “Modus tollens”

The principle of *modus tollens*: \( \{ p \rightarrow q, (\neg q) \} \vdash (\neg p) \).
Example: “Modus tollens”

The principle of *modus tollens*: \( \{ p \rightarrow q, (\neg q) \} \vdash (\neg p) \).

1. \( p \rightarrow q \)  Premise
2. \( (\neg q) \)  Premise

\( (\neg p) \)  ??
Example: “Modus tollens”

The principle of *modus tollens*: \( \{p \rightarrow q, (\neg q)\} \vdash (\neg p) \).

1. \( p \rightarrow q \) Premise
2. \( (\neg q) \) Premise
3. \( p \) Assumption
4. 
5. \( \bot \) ??
6. \( (\neg p) \) \( \neg i: \) ??
Example: “Modus tollens”

The principle of *modus tollens*: \( \{p \rightarrow q, (\neg q)\} \vdash (\neg p) \).

1. \( p \rightarrow q \) Premise
2. \( (\neg q) \) Premise
3. \( p \) Assumption
4. \( q \rightarrow e: 3, 1 \)
5. \( \bot \) ??
6. \( (\neg p) \neg i: ?? \)
Example: “Modus tollens”

The principle of *modus tollens*: \( \{p \rightarrow q, (\neg q)\} \vdash (\neg p) \).

1. \( p \rightarrow q \)  
   Premise

2. \( (\neg q) \)  
   Premise

3. \( p \)  
   Assumption

4. \( q \)  
   \( \rightarrow e: 3, 1 \)

5. \( \bot \)  
   \( \neg e: 2, 4 \)

6. \( (\neg p) \)  
   \( \neg i: 3–5 \)

*Modus tollens* is sometimes taken as a “derived rule”:

\[
\begin{align*}
\alpha & \rightarrow \beta \\
(\neg \beta) & \quad \text{MT} \\
(\neg \alpha) & \quad \text{MT}
\end{align*}
\]
Derived Rules

Whenever we have a proof of the form $\Gamma \vdash \alpha$, we can consider it as a derived rule:

$$
\frac{}{\Gamma \vdash \alpha}
$$

If we use this in a proof, it can be replaced by the original proof of $\Gamma \vdash \alpha$. The result is a proof using only the basic rules.

Using derived rules does not expand the things that can be proved. But they can make it easier to find a proof.
Some Useful Heuristics

Ideas to construct a proof:

1. Start with the premises at the top and the conclusion at the bottom.
2. If you can apply an elimination rule to premises, do so. (In the case of $\lor$-elimination, open two sub-proofs.)
3. Next, work backwards from the end. If your target formula has a connective, try its introduction rule. This will yield a new target. Repeat steps 2 and 3 with the new target, until you reach premises and/or available assumptions.
4. Treat a subproof as if it were a full proof (with a new premise).

Sometimes these ideas will lead you to a proof; sometimes they will not. If not, try something else instead of an introduction rule (idea 3).

Sometime nothing works. Take a break, and perhaps try again later.
Further Examples of Natural Deduction

**Example.** Show that \( \{p \rightarrow q\} \vdash (r \lor p) \rightarrow (r \lor q) \).

Write down premises and conclusion (step 1).
No elimination applies (step 2). Thus try \( \rightarrow i \) (step 3).

1. \( p \rightarrow q \)  
   Premise

   \((r \lor p) \rightarrow (r \lor q)\)  
   ??
Further Examples of Natural Deduction

Example. Show that \( \{p \rightarrow q\} \vdash (r \lor p) \rightarrow (r \lor q) \).

In the sub-proof, try \( \lor \)-elimination on the assumption (step 2).

1. \( p \rightarrow q \)  
   \( \)  
   Premise

2. \( r \lor p \)  
   \( \)  
   Assumption

3. \( r \)  
   \( \)  
   Assumption

4. \( r \lor q \)  
   ??

5. \( p \)  
   \( \)  
   Assumption

6. \( q \)  
   ??

7. \( r \lor q \)  
   ??

8. \( r \lor q \)  
   ??

9. \( (r \lor p) \rightarrow (r \lor q) \)  
   ??
Further Examples of Natural Deduction

**Example.** Show that \( \{p \rightarrow q\} \vdash (r \lor p) \rightarrow (r \lor q) \).

To justify lines 4 and 7:
No elimination applies from the assumptions (step 2).
What about \(\lor\)-introduction for the conclusion (step 3)?

<table>
<thead>
<tr>
<th>No.</th>
<th>Line</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( p \rightarrow q )</td>
<td>Premise</td>
</tr>
<tr>
<td>2</td>
<td>( r \lor p )</td>
<td>Assumption</td>
</tr>
<tr>
<td>3</td>
<td>( r )</td>
<td>Assumption</td>
</tr>
<tr>
<td>4</td>
<td>( r \lor q )</td>
<td>??</td>
</tr>
<tr>
<td>5</td>
<td>( p )</td>
<td>Assumption</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>( r \lor q )</td>
<td>??</td>
</tr>
<tr>
<td>8</td>
<td>( r \lor q )</td>
<td>(\lor)e: ??</td>
</tr>
<tr>
<td>9</td>
<td>( (r \lor p) \rightarrow (r \lor q) )</td>
<td>(\rightarrow)i: 2–8</td>
</tr>
</tbody>
</table>
**Example.** Show that \( \{p \rightarrow q\} \vdash (r \lor p) \rightarrow (r \lor q) \).

It works!

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>( p \rightarrow q )</td>
<td>Premise</td>
</tr>
<tr>
<td>2.</td>
<td>( r \lor p )</td>
<td>Assumption</td>
</tr>
<tr>
<td>3.</td>
<td>( r )</td>
<td>Assumption</td>
</tr>
<tr>
<td>4.</td>
<td>( r \lor q )</td>
<td>( \lor i: 3 )</td>
</tr>
<tr>
<td>5.</td>
<td>( p )</td>
<td>Assumption</td>
</tr>
<tr>
<td>6.</td>
<td>( q )</td>
<td>( \rightarrow e: 5, 1 )</td>
</tr>
<tr>
<td>7.</td>
<td>( r \lor q )</td>
<td>( \lor i: 6 )</td>
</tr>
<tr>
<td>8.</td>
<td>( r \lor q )</td>
<td>( \lor e: 2, 3–4, 5–7 )</td>
</tr>
<tr>
<td>9.</td>
<td>( (r \lor p) \rightarrow (r \lor q) )</td>
<td>( \rightarrow i: 2–8 )</td>
</tr>
</tbody>
</table>
Example. Show that \( \vdash ((p \to q) \to p) \to p \).

1. \( ((p \to q) \to p) \to p \quad \text{Try } \to i\ldots \)
Example. Show that \[ \vdash (p \to q) \to p. \]

1. \[ (p \to q) \to p \] Assumption

5. \[ p \]

6. \[ ((p \to q) \to p) \to p \] Try \( \to \)
Example. Show that \( \vdash ((p \to q) \to p) \to p \).

1. \( (p \to q) \to p \) \hspace{1cm} \text{Assumption}
2. \hspace{1cm} \text{No elimination applies.}
3. 
4. ????
5. \( p \) \hspace{1cm} \text{No connective.}
6. \( ((p \to q) \to p) \to p \) \hspace{1cm} \text{Try } \to i...
Example. Show that $\vdash ((p \rightarrow q) \rightarrow p) \rightarrow p$.

1. $(p \rightarrow q) \rightarrow p$  Assumption
2.  
3.  
4.  
5. $p$  No connective.
6. $((p \rightarrow q) \rightarrow p) \rightarrow p$  Try $\rightarrow i$...

Time to try something ingenious....
Some Common Derived Rules

Proof by contradiction (*reductio ad absurdum*):

\[
\text{if } \Sigma, (\neg \alpha) \vdash \bot, \text{ then } \Sigma \vdash \alpha.
\]

The “Law of Excluded Middle” (*tertium non datur*): \( \vdash \alpha \lor (\neg \alpha) \).

Double-Negation Introduction: if \( \Sigma \vdash \alpha \) then \( \Sigma \vdash (\neg (\neg \alpha)) \).

You can try to prove these yourself, as exercises. (Hint: in the first two, the last step uses rule \( \neg\neg \text{e}: (\neg (\neg \alpha)) \vdash \alpha \).)

Or see pages 24–26 of Huth and Ryan.
Soundness and Completeness of Natural Deduction for Propositional Logic
Soundness and Completeness of Natural Deduction

We want to prove that Natural Deduction is both sound and complete.

*Soundness* of Natural Deduction means that the conclusion of a proof is always a logical consequence of the premises. That is,

\[
\text{If } \Sigma \vdash_{\text{ND}} \alpha, \text{ then } \Sigma \vDash \alpha .
\]

*Completeness* of Natural Deduction means that all logical consequences in propositional logic are provable in Natural Deduction. That is,

\[
\text{If } \Sigma \vDash \alpha, \text{ then } \Sigma \vdash_{\text{ND}} \alpha .
\]
To prove soundness, we use induction on the *length of the proof*:

For all deductions \( \Sigma \vdash \alpha \) which have a proof of length \( n \) or less, it is the case that \( \Sigma \models \alpha \).

That property, however, is not quite good enough to carry out the induction. We actually use the following property of a natural number \( n \).

Suppose that a formula \( \alpha \) appears at line \( n \) of a partial deduction, which may have one or more open sub-proofs. Let \( \Sigma \) be the set of premises used and \( \Gamma \) be the set of assumptions of open sub-proofs. Then \( \Sigma \cup \Gamma \models \alpha \).
Basis of the Induction

**Base case.** The shortest deductions have length 1, and thus are either

1. \( \alpha \)  Premise.

or

1. \( \alpha \)  Assumption.

We have either \( \alpha \in \Sigma \) (in the first case), or \( \alpha \in \Gamma \) (in the second case).

Thus \( \Sigma \cup \Gamma \models \alpha \), as required.
Proof of Soundness: Inductive Step

**Inductive step.** Hypothesis: the property holds for each $n < k$; that is,

If some formula $\alpha$ appears at line $k$ or earlier of some partial deduction, with premises $\Sigma$ and un-closed assumptions $\Gamma$, then $\Sigma \cup \Gamma \vdash \alpha$.

To prove: if $\alpha'$ appears at line $k + 1$, then $\Sigma \cup \Gamma' \vdash \alpha'$
(where $\Gamma' = \Gamma \cup \alpha'$ when $\alpha'$ is an assumption, and $\Gamma' = \Gamma$ otherwise).

The case that $\alpha'$ is an assumption is trivial.

Otherwise, formula $\alpha'$ must have a justification by some rule. We shall consider each possible rule.
Case I: \( \alpha' \) was justified by \( \land i \).

We must have \( \alpha' = \alpha_1 \land \alpha_2 \), where each of \( \alpha_1 \) and \( \alpha_2 \) appear earlier in the proof, at steps \( m_1 \) and \( m_2 \), respectively. Also, any sub-proof open at step \( m_1 \) or \( m_2 \) is still open at step \( k + 1 \).

Thus the induction hypothesis applies to both; that is, \( \Sigma \cup \Gamma \models \alpha_1 \) and \( \Sigma \cup \Gamma \models \alpha_2 \).

By the definition of \( \models \), this yields \( \Sigma \cup \Gamma \models \alpha' \), as required.
Case II: $\alpha'$ was justified by $\rightarrow i$.

Rule $\rightarrow i$ requires that $\alpha' = \alpha_1 \rightarrow \alpha_2$ and there is a closed sub-proof with assumption $\alpha_1$ and conclusion $\alpha_2$, ending by step $k$. Also, any sub-proof open before the assumption of $\alpha_1$ is still open at step $k + 1$.

The induction hypothesis thus implies $\Sigma \cup (\Gamma \cup \{\alpha_1\}) \models \alpha_2$.

Hence $\Sigma \cup \Gamma \models \alpha_1 \rightarrow \alpha_2$, as required.
Inductive Step, Cases III ff.

Case III: $\alpha'$ was justified by $\neg e$.

This requires that $\alpha'$ be the pseudo-formula $\perp$, and that the proof contain formulas $\alpha$ and $(\neg \alpha)$ for some $\alpha$, each using at most $k$ steps.

By the induction hypothesis, both $\Sigma \models \alpha$ and $\Sigma \models (\neg \alpha)$.

Thus $\Sigma$ is contradictory, and $\Sigma \models \alpha'$ for any $\alpha'$.

Cases IV–XIII:

The other cases follow by similar reasoning.

This completes the inductive step, and the proof of soundness.
We now turn to completeness.

Recall that \textit{completeness} means the following.

Let $\Sigma$ be a set of formulas and $\varphi$ be a formula.

$$\text{If } \Sigma \models \varphi, \text{ then } \Sigma \vdash \varphi.$$ 

That is, every consequence has a proof.

How can we prove this?
Proof of Completeness: Getting started

We shall assume that the set \( \Sigma \) of hypotheses is finite. The theorem is also true for infinite sets of hypotheses, but that requires a completely different proof.

Suppose that \( \Sigma \vdash \varphi \), where \( \Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_m\} \).
Thus the formula \( (\sigma_1 \land \sigma_2 \land \ldots \land \sigma_m) \rightarrow \varphi \) is a tautology.

**Lemma.** Every tautology is provable in Natural Deduction.

Once we prove the Lemma, the result follows. Given a proof of \( (\sigma_1 \land \sigma_2 \land \ldots \land \sigma_m) \rightarrow \varphi \), one can use \( \land_i \) and \( \rightarrow_e \) to complete a proof of \( \Sigma \vdash \varphi \).
Tautologies Have Proofs

For a tautology, every line of its truth table ends with T. We can mimic the construction of a truth table using inferences in Natural Deduction.

**Claim.** Let $\varphi$ have $k$ variables $p_1, \ldots, p_k$. Let $v$ be a valuation, and define $\ell_1, \ell_2, \ldots, \ell_k$ as

$$
\ell_i = \begin{cases} 
  p_i & \text{if } v(p_i) = T \\
  \neg p_i & \text{if } v(p_i) = F.
\end{cases}
$$

If $\varphi^v = T$, then $\{\ell_1, \ldots, \ell_k\} \vdash \varphi$, and if $\varphi^v = F$, then $\{\ell_1, \ldots, \ell_k\} \vdash (\neg \varphi)$.

To prove the claim, use structural induction on formulas (which is induction on the column number of the truth table).

Once the claim is proven, we can prove a tautology as follows....
Outline of the Proof of a Tautology

1. \( p_1 \lor (\neg p_1) \)  
   \[ \text{L.E.M.} \]
2. \( p_2 \lor (\neg p_2) \)  
   \[ \text{L.E.M.} \]
\[ \vdots \]
k. \( p_k \lor (\neg p_k) \)  
   \[ \text{L.E.M.} \]

\( k + 1. \)
\[
\begin{align*}
p_1 & \quad \text{assumption} \\
p_2 & \quad \text{assumption} \\
\vdots & \\
\phi & \\
(\neg p_2) & \quad \text{assumption}
\end{align*}
\]

\( m. \)
\[
\begin{align*}
\phi & \\
\text{\lor e: 2, ...}
\end{align*}
\]

\( m + 1. \)
\[
\begin{align*}
(\neg p_1) & \quad \text{assumption} \\
\vdots & \\
\phi & \\
\text{\lor e: m + 1, ...}
\end{align*}
\]

\( n. \)
\[
\begin{align*}
\phi & \\
\text{\lor e: 1, m - (k + 1),} \\
\text{n - (m + 1)}
\end{align*}
\]

Once each variable is assumed true or false, the previous claim provides a proof.
Proving the Claim

Hypothesis: the following hold for formulas $\alpha$ and $\beta$:

- If $\{l_1, \ldots, l_k\} \models \alpha$, then $\{l_1, \ldots, l_k\} \vdash \alpha$;
- If $\{l_1, \ldots, l_k\} \not\models \alpha$, then $\{l_1, \ldots, l_k\} \vdash (\neg \alpha)$;
- If $\{l_1, \ldots, l_k\} \models \beta$, then $\{l_1, \ldots, l_k\} \vdash \beta$; and
- If $\{l_1, \ldots, l_k\} \not\models \beta$, then $\{l_1, \ldots, l_k\} \vdash (\neg \beta)$.

If $\{l_1, \ldots, l_k\} \models (\alpha \land \beta)$, put the two proofs of $\alpha$ and $\beta$ together, and then infer $(\alpha \land \beta)$, by $\land i$.

If $\{l_1, \ldots, l_k\} \not\models (\alpha \rightarrow \beta)$ (i.e., $\{l_1, \ldots, l_k\} \models \alpha$ and $\{l_1, \ldots, l_k\} \not\models \beta$),

- Prove $\alpha$ and $(\neg \beta)$.
- Assume $(\alpha \rightarrow \beta)$; from it, conclude $\beta$ ($\rightarrow e$) and then $\bot$ ($\neg e$).
- From the sub-proof, conclude $(\neg (\alpha \rightarrow \beta))$, by $\neg i$.

The other cases are similar.