The Foundations: Logic and Proofs

Part I
Chapter Summary

- Propositional Logic
  - The Language of Propositions
  - Applications
  - Logical Equivalences
- Predicate Logic
  - The Language of Quantifiers
  - Logical Equivalences
  - Nested Quantifiers
- Proofs
  - Rules of Inference
  - Proof Methods
  - Proof Strategy
Propositional Logic Summary

- The Language of Propositions
  - Connectives
  - Truth Values
  - Truth Tables

- Applications
  - Translating English Sentences
  - System Specifications
  - Logic Puzzles
  - Logic Circuits

- Logical Equivalences
  - Important Equivalences
  - Showing Equivalence
  - Satisfiability
Propositional Logic

Section 1.1
Section Summary

Propositions

Connectives
- Negation
- Conjunction
- Disjunction
- Implication; contraposition, inverse, converse
- Biconditional

Truth Tables
Propositions

A proposition is a declarative sentence that is either true or false.

- A declarative sentence (also known as a statement) makes a statement and ends with a period. It's named appropriately because it declares or states something.

Examples of propositions:

a) The Moon is made of green cheese.
b) Trenton is the capital of New Jersey.
c) Toronto is the capital of Canada.
d) $1 + 0 = 1$
e) $0 + 0 = 2$

Examples that are not propositions:

a) Sit down!
b) What time is it?
c) $x + 1 = 2$
d) $x + y = z$
Propositions

Propositional Variables: $p, q, r, s, ...$

- Each variable represents a proposition, just as in algebra, letters are often used to denote numberical variables.

The truth value of a proposition is true, denoted by $T$, if it is a true proposition.

The truth value of a proposition is false, denoted by $F$, if it is a false proposition.

*Propositional calculus* or *propositional logic* is the area of logic that deals with propositions.
Propositional Logic

- The proposition that is always true is denoted by $T$ and the proposition that is always false is denoted by $F$.

Constructing Propositions
- Start with propositional variables
- Construct compound propositions from propositions using logical operators
  - Negation $\neg$
  - Conjunction $\land$
  - Disjunction $\lor$
  - Implication $\rightarrow$
  - Biconditional $\leftrightarrow$
Compound Propositions: Negation

The *negation* of a proposition \( p \) is denoted by \( \neg p \) and has this truth table:

<table>
<thead>
<tr>
<th>( p )</th>
<th>( \neg p )</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

It is the statement “It is not the case that \( p \)”

- Also read “not \( p \)”
Compound Propositions: Negation

The *negation* of a proposition \( p \) is denoted by \( \neg p \) and has this truth table:

<table>
<thead>
<tr>
<th>( p )</th>
<th>( \neg p )</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

**Example:** If \( p \) denotes “The earth is round.”, then \( \neg p \) denotes “It is not the case that the earth is round,” or more simply “The earth is not round.”
Truth Table

- Has a row for each possible value of any propositional variable in a formula
  - Ex. If only variable is p, then a row corresponding to the case that p is true, and one corresponding to the case that p is false
  - Ex. If two variables, then four rows.
- A column for each logical expression of interest
Compound Propositions: Negation

The *negation* of a proposition $p$ is denoted by $\neg p$ and has this truth table:

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

**Example:** If $p$ denotes “The earth is round.”, then $\neg p$ denotes “It is not the case that the earth is round,” or more simply “The earth is not round.”
Conjunction

The *conjunction* of propositions \( p \) and \( q \) is denoted by \( p \land q \) and has this truth table:

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

Read “\( p \) and \( q \)”

\( p \land q \) is true if and only if both \( p \) and \( q \) are true.
Conjunction

The *conjunction* of propositions \( p \) and \( q \) is denoted by \( p \land q \) and has this truth table:

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

**Example:** If \( p \) denotes “I am at home.” and \( q \) denotes “It is raining.” then \( p \land q \) denotes “I am at home and it is raining.”
Disjunction

The *disjunction* of propositions $p$ and $q$ is denoted by $p \lor q$ and has this truth table:

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$p \lor q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

Read "p or q". $p \lor q$ is true if and only if $p$ is true or $q$ is true or both are true.
Disjunction

The *disjunction* of propositions $p$ and $q$ is denoted by $p \lor q$ and has this truth table:

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$p \lor q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

**Example:** If $p$ denotes “I am at home.” and $q$ denotes “It is raining.” then $p \lor q$ denotes “I am at home or it is raining.”
The “Connective Or” in English

- In English “or” has two distinct meanings.
  - “Inclusive Or” - p is true, or q is true, or both are true
  - “Exclusive Or” - p is true, or q is true, but not both are true
    - Ex. “Soup or salad comes with this entrée”
    - We don’t expect to be able to get both soup and salad.
Exclusive Or

The exclusive or of propositions $p$ and $q$ is denoted $p \oplus q$ and has this truth table

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$p \oplus q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

$p \oplus q$ is true if and only if $p$ is true, or $q$ is true, but not both are true.
If $p$ and $q$ are propositions, then $p \rightarrow q$ is a conditional statement or implication which is read as “if $p$, then $q$” and has this truth table:

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$p \rightarrow q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

**Example:** If $p$ denotes “I am at home.” and $q$ denotes “It is raining.” then $p \rightarrow q$ denotes “If I am at home then it is raining.”

In $p \rightarrow q$, $p$ is the hypothesis (antecedent or premise) and $q$ is the conclusion (or consequence).
Implication

The truth table for implication is often confusing:

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$p \rightarrow q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

If $p$ is true and $q$ is false, then the implication should be false, because a “true” statement should never imply a “false” statement.

Similarly, if $p$ is true and $q$ is true, then the implication should be true: a true statement has implied another true statement.

But what about those last two lines?
One way to think about it: $p \rightarrow q$ is an assertion that tells us something when we are in the situation that $p$ is true. Namely, that $q$ should be true. If it turns out that $q$ is NOT true in this situation, then the assertion $p \rightarrow q$ has lied to us.

BUT, if $p$ is not true, then the assertion has not promised us anything. It cannot have “lied” to us because it in fact did not promise us anything!
Implication

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$p \rightarrow q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

Henning Makholm: “One should note that these arguments are ultimately not the reason why implication has the truth table it has. The real reason is because that truth table is the \textit{definition} of $\rightarrow$. Expressing $p \rightarrow q$ as "If $p$, then $q"$ is not a definition of $\rightarrow$, but an explanation of how the words "if" and "then" are used by mathematicians, given that one already knows how $\rightarrow$ works. The intuitive explanations are supposed to convince you (or not) that it is reasonable to use those two English words to speak about logical implication, not that logical implication ought to work that way in the first place.”

And besides: propositional language is an artificial language – it is designed to make logical inference engines work correctly
- English language parallels are only used to make these easy to remember
Understanding Implication

In $p \rightarrow q$ there does not need to be any connection between the antecedent or the consequent. The “meaning” of $p \rightarrow q$ depends only on the truth values of $p$ and $q$.

These implications are perfectly fine, but would not be used in ordinary English.

- “If the moon is made of green cheese, then I have more money than Bill Gates.”
- “If the moon is made of green cheese then I’m on welfare.”
- “If $1 + 1 = 3$, then your grandma wears combat boots.”
One way to view the logical conditional is to think of an obligation or contract.

- “If I am elected, then I will lower taxes.”
- “If you get 100% on the final, then you will get an A.”

If the politician is elected and does not lower taxes, then the voters can say that he or she has broken the campaign pledge. Something similar holds for the professor. This corresponds to the case where $p$ is true and $q$ is false.
Different Ways of Expressing $p \rightarrow q$

- If $p$, then $q$
- If $p$, $q$
- $q$ unless $\neg p$
- $q$ if $p$
- $q$ whenever $p$
- $q$ follows from $p$

- $p$ implies $q$
- $p$ only if $q$
- $q$ when $p$
- $p$ is sufficient for $q$
- $q$ is necessary for $p$
Converse, Contrapositive, and Inverse

From $p \rightarrow q$ we can form new conditional statements.
- $q \rightarrow p$ is the **converse** of $p \rightarrow q$
- $\neg q \rightarrow \neg p$ is the **contrapositive** of $p \rightarrow q$
- $\neg p \rightarrow \neg q$ is the **inverse** of $p \rightarrow q$

**Example**: Find the converse, inverse, and contrapositive of “It raining is a sufficient condition for my not going to town.”

**Solution:**
- **converse**: If I do not go to town, then it is raining.
- **inverse**: If it is not raining, then I will go to town.
- **contrapositive**: If I go to town, then it is not not raining.
Biconditional

If $p$ and $q$ are propositions, then we can form the *biconditional* proposition $p \leftrightarrow q$, read as “$p$ if and only if $q$.” The biconditional $p \leftrightarrow q$ denotes the proposition with this truth table:

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$p \leftrightarrow q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

Think of it as $p \rightarrow q \land q \rightarrow p$
- Let’s consider truth table for this...
If $p$ and $q$ are propositions, then we can form the biconditional proposition $p \iff q$, read as “$p$ if and only if $q$.” The biconditional $p \iff q$ denotes the proposition with this truth table:

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$p \iff q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

If $p$ denotes “I am at home.” and $q$ denotes “It is raining.” then $p \iff q$ denotes “I am at home if and only if it is raining.”
Expressing the Biconditional

Some alternative ways “$p$ if and only if $q$” is expressed in English:

- $p$ is necessary and sufficient for $q$
- if $p$ then $q$, and conversely
- $p$ iff $q$
Truth Tables For Compound Propositions

Construction of a truth table:

Rows
- Need a row for every possible combination of values for the atomic propositions.

Columns
- Need a column for the compound proposition (usually at far right)
- Need a column for the truth value of each expression that occurs in the compound proposition as it is built up.
  - This includes the atomic propositions
Construct a truth table for $p \lor q \rightarrow \neg r$

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>r</th>
<th>$\neg r$</th>
<th>$p \lor q$</th>
<th>$p \lor q \rightarrow \neg r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</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>F</td>
<td>T</td>
<td>F</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>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>
We want a truth table for $p \lor q \rightarrow \neg r$

BUT,
- Is this $(p \lor q) \rightarrow \neg r$?
- Or is it $p \lor (q \rightarrow \neg r)$?

They are not necessarily the same!
**Precedence of Logical Operators**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>¬</td>
<td>1</td>
</tr>
<tr>
<td>∧</td>
<td>2</td>
</tr>
<tr>
<td>∨</td>
<td>3</td>
</tr>
<tr>
<td>→</td>
<td>4</td>
</tr>
<tr>
<td>↔</td>
<td>5</td>
</tr>
</tbody>
</table>

\[ p \lor q \rightarrow \neg r \] is equivalent to \[(p \lor q) \rightarrow \neg r\]

If the intended meaning is \( p \lor (q \rightarrow \neg r) \) then parentheses must be used.

BUT, you should always use parenthesis. Relying on someone knowing the precedence rules is dicey at best! (And confusing at worst!)
Example Truth Table

Construct a truth table for $p \lor q \rightarrow \neg r$

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>r</th>
<th>$\neg r$</th>
<th>$p \lor q$</th>
<th>$p \lor q \rightarrow \neg r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</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>F</td>
<td>T</td>
<td>F</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>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</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>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

Consider F to be 1 and T to be 0 and count in binary
Equivalent Propositions

Two propositions are *equivalent* if they always have the same truth value.

**Example:** Show using a truth table that the conditional is equivalent to the contrapositive.

**Solution:**

<table>
<thead>
<tr>
<th></th>
<th>q</th>
<th>¬p</th>
<th>¬q</th>
<th>p →q</th>
<th>¬q → ¬p</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</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>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
Using a Truth Table to Show Non-Equivalence

**Example:** Show using truth tables that neither the converse nor inverse of an implication are not equivalent to the implication.

**Solution:**

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$\neg p$</th>
<th>$\neg q$</th>
<th>$p \rightarrow q$</th>
<th>$\neg p \rightarrow \neg q$</th>
<th>$q \rightarrow p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</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>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</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>
<td>T</td>
</tr>
</tbody>
</table>

But note converse and inverse are equivalent! Why?
Problem

How many rows are there in a truth table with $n$ propositional variables?

**Solution:** $2^n$ We will see how to do this in Chapter 6.

Note that this means that with $n$ propositional variables, we can construct $2^n$ distinct (i.e., not equivalent) propositions.
Applications of Propositional Logic

Section 1.2
Applications of Propositional Logic: Summary

- Translating English to Propositional Logic
- System Specifications
- Boolean Searching
- Logic Puzzles
- Logic Circuits
- AI Diagnosis Method (Optional)
Translating English Sentences

Steps to convert an English sentence to a statement in propositional logic

- Identify atomic propositions and represent using propositional variables.
- Determine appropriate logical connectives

“If I go to Harry’s or to the country, I will not go shopping.”

- \(p\): I go to Harry’s
- \(q\): I go to the country.
- \(r\): I will go shopping.

\[ (p \lor q) \rightarrow \neg r \]
Example

**Problem:** Translate the following sentence into propositional logic:

“You can access the Internet from campus only if you are a computer science major or you are not a freshman.”

**One Solution:** Let $a$, $c$, and $f$ represent respectively “You can access the internet from campus,” “You are a computer science major,” and “You are a freshman.”

$$ a \rightarrow (c \lor \neg f) $$
System Specifications

System and Software engineers take requirements in English and express them in a precise specification language based on logic.

**Example:** Express in propositional logic: “The automated reply cannot be sent when the file system is full”

**Solution:** One possible solution: Let $p$ denote “The automated reply can be sent” and $q$ denote “The file system is full.”

$$q \rightarrow \neg p$$
Consistent System Specifications

**Definition**: A list of propositions is *consistent* if it is possible to assign truth values to the proposition variables so that each proposition is true.
Exercise: Are these specifications consistent?

- “The diagnostic message is stored in the buffer or it is retransmitted.”
- “The diagnostic message is not stored in the buffer.”
- “If the diagnostic message is stored in the buffer, then it is retransmitted.”

Solution: Let \( p \) denote “The diagnostic message is stored in the buffer.” Let \( q \) denote “The diagnostic message is retransmitted.” The specification can be written as: \( p \lor q, \neg p, p \rightarrow q \). When \( p \) is false and \( q \) is true all three statements are true. So the specification is consistent.

- What if “The diagnostic message is not retransmitted is added.”
**Consistent System Specifications**

**Exercise:** Are these specifications consistent?

- “The diagnostic message is stored in the buffer or it is retransmitted.”
- “The diagnostic message is not stored in the buffer.”
- “If the diagnostic message is stored in the buffer, then it is retransmitted.”

**Solution:** Let \( p \) denote “The diagnostic message is stored in the buffer.” Let \( q \) denote “The diagnostic message is retransmitted.”

The specification can be written as: \( p \lor q, \neg p, p \rightarrow q \). When \( p \) is false and \( q \) is true all three statements are true. So the specification is consistent.

- What if “The diagnostic message is not retransmitted is added.”

  **Solution:** Now we are adding \( \neg q \) and there is no satisfying assignment. So the specification is not consistent.
An island has two kinds of inhabitants, *knights*, who always tell the truth, and *knaves*, who always lie.

You go to the island and meet A and B.

- A says “B is a knight.”
- B says “The two of us are of opposite types.”

**Example:** What are the types of A and B?
An island has two kinds of inhabitants, *knights*, who always tell the truth, and *knaves*, who always lie.

You go to the island and meet A and B.

- A says “B is a knight.”
- B says “The two of us are of opposite types.”

**Example:** What are the types of A and B?

**Solution:** Let $p$ and $q$ be the statements that A is a knight and B is a knight, respectively. So, then $\neg p$ represents the proposition that A is a knave and $\neg q$ that B is a knave.

- If A is a knight, then $p$ is true. Since knights tell the truth, $q$ must also be true. Then $(p \land \neg q) \lor (\neg p \land q)$ would have to be true, but it is not. So, A is not a knight and therefore $\neg p$ must be true.
- If A is a knave, then B must not be a knight since knaves always lie. So, then both $\neg p$ and $\neg q$ hold since both are knaves.
Logic Circuits  
(Studied in depth in Chapter 12)

- Electronic circuits; each input/output signal can be viewed as a 0 or 1.
  - 0 represents False
  - 1 represents True

- Complicated circuits are constructed from three basic circuits called gates.

- The inverter (NOT gate) takes an input bit and produces the negation of that bit.
- The OR gate takes two input bits and produces the value equivalent to the disjunction of the two bits.
- The AND gate takes two input bits and produces the value equivalent to the conjunction of the two bits.

- More complicated digital circuits can be constructed by combining these basic circuits to produce the desired output given the input signals by building a circuit for each piece of the output expression and then combining them. For example:
A circuit with a single output is said to be *satisfiable* if there is an assignment of inputs that make the output T (or equivalently 1).

Simple right?
A circuit with a single output is said to be *satisfiable* if there is an assignment of inputs that make the output T (or equivalently 1)

Simple right? Wrong!

- For even moderately sized circuits, this general problem is computationally intractible!
- E.g., if circuit has 100 inputs, then there are $2^{100}$ possibilities to check. That’s approximately $1.26 \times 10^{30}$. If a computer can check one billion input tuples per second, it would take about 40 trillion years to check them all!

Note: This doesn’t mean that it takes this long to determine satisfiability for every circuit! (There might be tricks for some.)
Diagnosis of Faults in an Electrical System (*Optional*)

AI Example (from *Artificial Intelligence: Foundations of Computational Agents* by David Poole and Alan Mackworth, 2010)

Need to represent in propositional logic the features of a piece of machinery or circuitry that are required for the operation to produce observable features. This is called the **Knowledge Base (KB)**.

We also have observations representing the features that the system is exhibiting now.
Have lights (l1, l2), wires (w0, w1, w2, w3, w4), switches (s1, s2, s3), and circuit breakers (cb1).

The next page gives the knowledge base describing the circuit and the current observations.
Representing the Electrical System in Propositional Logic

- We need to represent our common-sense understanding of how the electrical system works in propositional logic.
- For example: “If l1 is a light and if l1 is receiving current, then l1 is lit.
  - \[ \text{light}_l1 \land \text{live}_l1 \land \text{ok}_l1 \rightarrow \text{lit}_l1 \]
- Also: “If w1 has current, and switch s2 is in the up position, and s2 is not broken, then w0 has current.”
  - \[ \text{live}_w1 \land \text{up}_s2 \land \text{ok}_s2 \rightarrow \text{live}_w0 \]
- This task of representing a piece of our common-sense world in logic is a common one in logic-based AI.
Knowledge Base (opt)

- live_outside
- light_l1
- light_l2
- live_w0 → live_l1
- live_w1 ∧ up_s2 ∧ ok_s2 → live_w0
- live_w2 ∧ down_s2 ∧ ok_s2 → live_w0
- live_w3 ∧ up_s1 ∧ ok_s1 → live_w1
- live_w3 ∧ down_s1 ∧ ok_s1 → live_w2
- live_w4 → live_l2
- live_w3 ∧ up_s3 ∧ ok_s3 → live_w4
- live_outside ∧ ok_cb1 → live_w3
- light_l1 ∧ live_l1 ∧ ok_l1 → lit_l1
- light_l2 ∧ live_l2 ∧ ok_l2 → lit_l2

We have outside power. Both l1 and l2 are lights.

If s2 is ok and s2 is in a down position and w2 has current, then w0 has current.
Observations (opt)

Observations need to be added to the KB

- Both Switches up
  - up_s1
  - up_s2

- Both lights are dark
  - ¬lit_l1
  - ¬lit_l2
Diagnosis (*opt*)

- We assume that the components are working ok, unless we are forced to assume otherwise. These atoms are called *assumables*.
- The assumables (ok_cb1, ok_s1, ok_s2, ok_s3, ok_l1, ok_l2) represent the assumption that we assume that the switches, lights, and circuit breakers are ok.
- If the system is working correctly (all assumables are true), the observations and the knowledge base are consistent (i.e., satisfiable).
- The augmented knowledge base is clearly not consistent if the assumables are all true. The switches are both up, but the lights are not lit. Some of the assumables must then be false. This is the basis for the method to diagnose possible faults in the system.
- A diagnosis is a minimal set of assumables which must be false to explain the observations of the system.
See *Artificial Intelligence: Foundations of Computational Agents* (by David Poole and Alan Mackworth, 2010) for details on this problem and how the method of consistency based diagnosis can determine possible diagnoses for the electrical system.

The approach yields 7 possible faults in the system. At least one of these must hold:

- Circuit Breaker 1 is not ok.
- Both Switch 1 and Switch 2 are not ok.
- Both Switch 1 and Light 2 are not ok.
- Both Switch 2 and Switch 3 are not ok.
- Both Switch 2 and Light 2 are not ok.
- Both Light 1 and Switch 3 are not ok.
- Both Light 1 and Light 2 are not ok.
Propositional Equivalences

Section 1.3
Section Summary

- Tautologies, Contradictions, and Contingencies.
- Logical Equivalence
  - Important Logical Equivalences
  - Showing Logical Equivalence
- Normal Forms *(optional, covered in exercises in text)*
  - Disjunctive Normal Form
  - Conjunctive Normal Form
- Propositional Satisfiability
  - Sudoku Example
Tautologies, Contradictions, and Contingencies

A **tautology** is a proposition which is always true.
- Example: $p \lor \neg p$

A **contradiction** is a proposition which is always false.
- Example: $p \land \neg p$

A **contingency** is a proposition which is neither a tautology nor a contradiction, such as $p$

<table>
<thead>
<tr>
<th>$p$</th>
<th>$\neg p$</th>
<th>$p \lor \neg p$</th>
<th>$p \land \neg p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>
Two compound propositions \( p \) and \( q \) are *logically equivalent* if \( p \leftrightarrow q \) is a tautology.

We write this as \( p \equiv q \) or as \( p \equiv q \) where \( p \) and \( q \) are compound propositions.

Two compound propositions \( p \) and \( q \) are equivalent if and only if the columns in a truth table giving their truth values agree.

This truth table shows that \( \neg p \lor q \) is equivalent to \( p \to q \).

<table>
<thead>
<tr>
<th>( p )</th>
<th>( q )</th>
<th>( \neg p )</th>
<th>( \neg p \lor q )</th>
<th>( p \to q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<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>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
De Morgan’s Laws

\[ \neg(p \land q) \equiv \neg p \lor \neg q \]
\[ \neg(p \lor q) \equiv \neg p \land \neg q \]

This truth table shows that De Morgan’s Second Law holds.

<table>
<thead>
<tr>
<th>(p)</th>
<th>(q)</th>
<th>(\neg p)</th>
<th>(\neg q)</th>
<th>(p \lor q)</th>
<th>(\neg(p \lor q))</th>
<th>(\neg p \land \neg q)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<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>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
Key Logical Equivalences

Identity Laws: \( p \land T \equiv p \quad p \lor F \equiv p \)

Domination Laws: \( p \lor T \equiv T \quad p \land F \equiv F \)

Idempotent laws: \( p \lor p \equiv p \quad p \land p \equiv p \)

Double Negation Law: \( \neg(\neg p) \equiv p \)

Negation Laws: \( p \lor \neg p \equiv T \quad p \land \neg p \equiv F \)
Key Logical Equivalences (cont)

**Commutative Laws:** \( p \lor q \equiv q \lor p \)  \( p \land q \equiv q \land p \)

**Associative Laws:**
\[
(p \land q) \land r \equiv p \land (q \land r) \\
(p \lor q) \lor r \equiv p \lor (q \lor r)
\]

**Distributive Laws:**
\[
(p \lor (q \land r)) \equiv (p \lor q) \land (p \lor r) \\
(p \land (q \lor r)) \equiv (p \land q) \lor (p \land r)
\]

**Absorption Laws:**
\[
p \land (p \lor q) \equiv p \\
p \lor (p \land q) \equiv p
\]
More Logical Equivalences

**TABLE 7** Logical Equivalences Involving Conditional Statements.

\[
\begin{align*}
    p \rightarrow q &\equiv \neg p \lor q \\
    p \rightarrow q &\equiv \neg q \rightarrow \neg p \\
    p \lor q &\equiv \neg p \rightarrow q \\
    p \land q &\equiv \neg (p \rightarrow \neg q) \\
    \neg (p \rightarrow q) &\equiv p \land \neg q \\
    (p \rightarrow q) \land (p \rightarrow r) &\equiv p \rightarrow (q \land r) \\
    (p \rightarrow r) \land (q \rightarrow r) &\equiv (p \lor q) \rightarrow r \\
    (p \rightarrow q) \lor (p \rightarrow r) &\equiv p \rightarrow (q \lor r) \\
    (p \rightarrow r) \lor (q \rightarrow r) &\equiv (p \land q) \rightarrow r
\end{align*}
\]

**TABLE 8** Logical Equivalences Involving Biconditional Statements.

\[
\begin{align*}
    p \leftrightarrow q &\equiv (p \rightarrow q) \land (q \rightarrow p) \\
    p \leftrightarrow q &\equiv \neg p \leftrightarrow \neg q \\
    p \leftrightarrow q &\equiv (p \land q) \lor (\neg p \land \neg q) \\
    \neg (p \leftrightarrow q) &\equiv p \leftrightarrow \neg q
\end{align*}
\]
Constructing New Logical Equivalences

We can show that two expressions are logically equivalent by developing a series of logically equivalent statements.

To prove that \( A \equiv B \) we produce a series of equivalences beginning with \( A \) and ending with \( B \).

\[
A \equiv A_1 \\
\vdots \\
A_n \equiv B
\]

Keep in mind that whenever a proposition (represented by a propositional variable) occurs in the equivalences listed earlier, it may be replaced by an arbitrarily complex compound proposition.
Equivalence Proofs

**Example:** Show that \( \neg(p \lor (\neg p \land q)) \) is logically equivalent to \( \neg p \land \neg q \)

**Solution:**

\[
\begin{align*}
\neg(p \lor (\neg p \land q)) & \equiv \neg p \land \neg(\neg p \land q) & \text{by the second De Morgan law} \\
& \equiv \neg p \land [\neg(\neg p) \lor \neg q] & \text{by the first De Morgan law} \\
& \equiv \neg p \land (p \lor \neg q) & \text{by the double negation law} \\
& \equiv (\neg p \land p) \lor (\neg p \land \neg q) & \text{by the second distributive law} \\
& \equiv F \lor (\neg p \land \neg q) & \text{because } \neg p \land p \equiv F \\
& \equiv (\neg p \land \neg q) \lor F & \text{by the commutative law for disjunction} \\
& \equiv (\neg p \land \neg q) & \text{by the identity law for } F
\end{align*}
\]
Equivalence Proofs

Example: Show that \((p \land q) \rightarrow (p \lor q)\) is a tautology.

Solution:

\[
(p \land q) \rightarrow (p \lor q) \equiv \neg (p \land q) \lor (p \lor q) \quad \text{by truth table for } \rightarrow
\]
\[
\equiv (\neg p \lor \neg q) \lor (p \lor q) \quad \text{by the first De Morgan law}
\]
\[
\equiv (\neg p \lor p) \lor (\neg q \lor q) \quad \text{by associative and commutative laws for disjunction}
\]
\[
\equiv T \lor T \quad \text{by truth tables}
\]
\[
\equiv T \quad \text{by the domination law}
\]
Disjunctive Normal Form

A propositional formula is in *disjunctive normal form* if it consists of a disjunction of \((1, \ldots, n)\) disjuncts where each disjunct consists of a conjunction of \((1, \ldots, m)\) atomic formulas or the negation of an atomic formula.

- **Yes**: \((p \land \neg q) \lor (\neg p \land q)\)
- **No**: \(p \land (p \lor q)\)

Disjunctive Normal Form is important for circuit design methods.
**Example:** Show that every compound proposition can be put in disjunctive normal form.

**Solution:** Construct the truth table for the proposition. Then an equivalent proposition is the disjunction with \( n \) disjuncts (where \( n \) is the number of rows for which the formula evaluates to \( T \)). Each disjunct has \( m \) conjuncts where \( m \) is the number of distinct propositional variables. Each conjunct includes the positive form of the propositional variable if the variable is assigned \( T \) in that row and the negated form if the variable is assigned \( F \) in that row. This proposition is in disjunctive normal form.
Example: Show that every compound proposition can be put in disjunctive normal form.

Solution: Construct the truth table for the proposition. Then an equivalent proposition is the disjunction with \( n \) disjuncts (where \( n \) is the number of rows for which the formula evaluates to \( T \)). Each disjunct has \( m \) conjuncts where \( m \) is the number of distinct propositional variables. Each conjunct includes the positive form of the propositional variable if the variable is assigned \( T \) in that row and the negated form if the variable is assigned \( F \) in that row. This proposition is in disjunctive normal form.

Say what?!
An Aside

- I want to design a circuit that does one bit binary addition
- This circuit has three (?) inputs and two (?) outputs
An Aside

I want to design a circuit that does one bit binary addition

This circuit has three (?) inputs and two (?) outputs

- a (first summand bit)
- b (second summand bit)
- CI - carry in!
- Sum
- CO – carry out!
Full One-bit Adder

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>CarryIn</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

*FIGURE C.5.3* Input and output specification for a 1-bit adder.
Full One-bit Adder

Q: How to create a circuit for this?
A: Write each output in disjunctive normal form! (But how?!)
Example: Show that every compound proposition can be put in disjunctive normal form.

Solution: Construct the truth table for the proposition. Then an equivalent proposition is the disjunction with $n$ disjuncts (where $n$ is the number of rows for which the formula evaluates to $T$). Each disjunct has $m$ conjuncts where $m$ is the number of distinct propositional variables. Each conjunct includes the positive form of the propositional variable if the variable is assigned $T$ in that row and the negated form if the variable is assigned $F$ in that row. This proposition is in disjunctive normal from.

Lots of words for something straightforward!
Disjunctive Normal Form

**Example:** Find the Disjunctive Normal Form (DNF) of

\[(p \lor q) \rightarrow \neg r\]

**Solution:** This proposition is true when \( r \) is false or when both \( p \) and \( q \) are false.

\[(\neg p \land \neg q) \lor \neg r\]
One More Aside

Now that we have our one-bit full adder, how do we use it to create a 32-bit binary adder?
One More Aside

Now that we have our one-bit full adder, how do we use it to create a 32-bit binary adder?
Conjunctive Normal Form

- A compound proposition is in *Conjunctive Normal Form* (CNF) if it is a conjunction of disjunctions.
- Every proposition can be put in an equivalent CNF.
- Conjunctive Normal Form (CNF) can be obtained by eliminating implications, moving negation inwards and using the distributive and associative laws.
- Important in resolution theorem proving used in artificial Intelligence (AI).
- A compound proposition can be put in conjunctive normal form through repeated application of the logical equivalences covered earlier.
Conjunctive Normal Form

**Example:** Put the following into CNF:
\[-(p \rightarrow q) \lor (r \rightarrow p)\]

**Solution:**

1. Eliminate implication signs:
   \[-(\neg p \lor q) \lor (\neg r \lor p)\]
2. Move negation inwards; eliminate double negation:
   \[(p \land \neg q) \lor (\neg r \lor p)\]
3. Convert to CNF using associative/distributive laws
   \[(p \lor \neg r \lor p) \land (\neg q \lor \neg r \lor p)\]
Propositional Satisfiability

A compound proposition is *satisfiable* if there is an assignment of truth values to its variables that make it true. When no such assignments exist, the compound proposition is *unsatisfiable*.

A compound proposition is unsatisfiable if and only if its negation is a tautology.

- Think about it: if no assignment makes it true, then every assignment makes it false!
Questions on Propositional Satisfiability

Example: Determine the satisfiability of the following compound propositions:

\[(p \lor \neg q) \land (q \lor \neg r) \land (r \lor \neg p)\]

Solution: Satisfiable. Assign T to p, q, and r.

\[(p \lor q \lor r) \land (\neg p \lor \neg q \lor \neg r)\]

Solution: Satisfiable. Assign T to p and F to q.

\[(p \lor \neg q) \land (q \lor \neg r) \land (r \lor \neg p) \land (p \lor q \lor r) \land (\neg p \lor \neg q \lor \neg r)\]

Solution: Not satisfiable. Check each possible assignment of truth values to the propositional variables and none will make the proposition true.
Questions on Propositional Satisfiability

**Example:** Determine the satisfiability of the following compound propositions:

\[(p \lor q \lor r) \land (\neg p \lor \neg q \lor \neg r)\]

**Solution:** Satisfiable. Assign T to \(p\) and F to \(q\).

**Note:** Given this assignment to \(p\) and \(q\), it doesn’t matter what \(r\) is assigned – proposition will be true either way. In digital logic design, we would say that \(r\) is a “don’t care” value. These help simplify chip design.
Questions on Propositional Satisfiability

**Example:** Determine the satisfiability of the following compound propositions:

\[
(p \lor q \lor r) \land (\neg p \lor \neg q \lor \neg r)
\]

**Solution:** Satisfiable. Assign T to \(p\) and F to \(q\).

**Note:** Given this assignment to \(p\) and \(q\), it doesn’t matter what \(r\) is assigned – proposition will be true either way. In digital logic design, we would say that \(r\) is a “don’t care” value. These help simplify chip design. And sometimes create security holes!
Notation

\[ \vee_{j=1}^{n} p_j \text{ is used for } p_1 \lor p_2 \lor \ldots \lor p_n \]

\[ \wedge_{j=1}^{n} p_j \text{ is used for } p_1 \land p_2 \land \ldots \land p_n \]

Think of these like summation (\(\Sigma\)) or product (\(\Pi\))
A Sudoku puzzle is represented by a $9 \times 9$ grid made up of nine $3 \times 3$ subgrids, known as blocks. Some of the 81 cells of the puzzle are assigned one of the numbers 1, 2, ..., 9.

The puzzle is solved by assigning numbers to each blank cell so that every row, column and block contains each of the nine possible numbers.

Example
Encoding as a Satisfiability Problem

Let \( p(i,j,n) \) denote the proposition that is true when the number \( n \) is in the cell in the \( i \)th row and the \( j \)th column.

There are \( 9 \times 9 \times 9 = 729 \) such propositions.

In the sample puzzle \( p(5,1,6) \) is true, but \( p(5,j,6) \) is false for \( j = 2,3,...9 \).
Encoding (cont)

- For each cell with a given value, assert $p(i,j,n)$, when the cell in row $i$ and column $j$ has the given value.
- Assert that every row contains every number.

\[ \bigwedge_{i=1}^{9} \bigwedge_{n=1}^{9} \bigvee_{j=1}^{9} p(i,j,n) \]

- Assert that every column contains every number.

\[ \bigwedge_{j=1}^{9} \bigwedge_{n=1}^{9} \bigvee_{i=1}^{9} p(i,j,n) \]
Encoding (cont)

Assert that each of the 3 x 3 blocks contain every number.

\[ \bigwedge_{r=0}^{2} \bigwedge_{s=0}^{2} \bigwedge_{n=1}^{9} \bigwedge_{i=1}^{3} \bigwedge_{j=1}^{3} p(3r + i, 3s + j, n) \]

(this is tricky - ideas from chapter 4 help)

Assert that no cell contains more than one number. Take the conjunction over all values of \( n, n', i, \) and \( j, \) where each variable ranges from 1 to 9 and \( n \neq n', \)

of \( p(i, j, n) \rightarrow \neg p(i, j, n') \)

Put another way, if a cell contains \( n, \) this implies that it cannot contain any \( n' \) not equal to \( n. \)
Solving Satisfiability Problems

To solve a Sudoku puzzle, we need to find an assignment of truth values to the 729 variables of the form $p(i,j,n)$ that makes the conjunction of all of these assertions true. Those variables that are assigned T yield a solution to the puzzle.

- Which you can then read off

A truth table can always be used to determine the satisfiability of a compound proposition. But this is too complex even for modern computers for large problems.

There has been much work on developing efficient methods for solving satisfiability problems as many practical problems can be translated into satisfiability problems.

- But then, you knew that from our discussion on the first day!