Mathematical Reasoning
Foundations of Logic

*Mathematical Logic* is a tool for working with elaborate *compound* statements. It includes:

- A language for expressing them.
- A concise notation for writing them.
- A methodology for objectively reasoning about their truth or falsity.
- It is the foundation for expressing formal proofs in all branches of mathematics.
Foundations of Logic: Overview

1. Propositional logic
2. Predicate logic and Quantifiers
3. Quantifiers and Logical Operators
4. Logical Inference
5. Methods of Proof
Propositional Logic

Propositional Logic is the logic of compound statements built from simpler statements using so-called Boolean connectives.

Some applications in computer science:
• Design of digital electronic circuits.
• Expressing conditions in programs.
• Queries to databases & search engines.

George Boole (1815-1864)
Definition of a *Proposition*

**Assertion:** Statement

**Proposition:** A *proposition* is an assertion which is either true or false, but not both.

(However, you might not *know* the actual truth value, and it might be situation-dependent.)

[Later in *probability theory* we assign *degrees of certainty* to propositions. But for now: think True/False only!]
Examples of Propositions

• “It is raining.” (In a given situation.)
• “Beijing is the capital of China.” • “1 + 2 = 3”

But, the following are **NOT** propositions:
• “Who’s there?” (interrogative, question)
• “La la la la la la.” (meaningless interjection)
• “Just do it!” (imperative, command)
A Paradox

• “I am lying”: Is he speaking the truth or lying? True or False??
  – Neither True nor False.
  – If the statement is true, then he says he is lying, that is if he says the truth he is lying
  – If the statement is false, then his statement, “I am lying” is false, which means he is telling the truth
  – Thus, although it appears that the statement is a proposition, this is not. As this cannot be assigned a truth value.
Operators / Connectives

An operator or connective combines one or more operand expressions into a larger expression. (E.g., “+” in numeric exprs.)

Unary operators take 1 operand (e.g., −3); binary operators take 2 operands (eg 3 × 4).

Propositional or Boolean operators operate on propositions or truth values instead of on numbers.
Some Popular Boolean Operators

<table>
<thead>
<tr>
<th>Formal Name</th>
<th>Nickname</th>
<th>Arity</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negation operator</td>
<td>NOT</td>
<td>Unary</td>
<td>¬</td>
</tr>
<tr>
<td>Conjunction operator</td>
<td>AND</td>
<td>Binary</td>
<td>∧</td>
</tr>
<tr>
<td>Disjunction operator</td>
<td>OR</td>
<td>Binary</td>
<td>∨</td>
</tr>
<tr>
<td>Exclusive-OR operator</td>
<td>XOR</td>
<td>Binary</td>
<td>⊕</td>
</tr>
<tr>
<td>Implication operator</td>
<td>IMPLIES</td>
<td>Binary</td>
<td>⇒</td>
</tr>
<tr>
<td>Biconditional operator</td>
<td>IFF</td>
<td>Binary</td>
<td>⇔</td>
</tr>
</tbody>
</table>
The Negation Operator

The unary negation operator “¬” (NOT) transforms a prop. into its logical negation. 

E.g. If \( p = "I \text{ have brown hair.}" \) then \( \neg p = "I \text{ do not have brown hair.}" \)

Truth table for NOT:

<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>

T :\(\equiv\) True; F :\(\equiv\) False

“:\(\equiv\)” means “is defined as”
The Conjunction Operator

The binary conjunction operator “∧” (AND) combines two propositions to form their logical conjunction.

E.g. If $p=\text{“I will have salad for lunch.”}$ and $q=\text{“I will have chicken for dinner.”}$, then $p \land q=\text{“I will have salad for lunch and I will have chicken for dinner.”}$

Remember: “∧” points up like an “A”, and it means “AND”
Conjunction Truth Table

- Note that a conjunction $p_1 \land p_2 \land \ldots \land p_n$ of $n$ propositions will have $2^n$ rows in its truth table.

- Also: $\neg$ and $\land$ operations together are sufficient to express any Boolean truth table!
The Disjunction Operator

The binary disjunction operator “\( \lor \)” (OR) combines two propositions to form their logical disjunction.

\[ p = \text{“My car has a bad engine.”} \]
\[ q = \text{“My car has a bad carburetor.”} \]
\[ p \lor q = \text{“Either my car has a bad engine, or my car has a bad carburetor.”} \]

Meaning is like “and/or” in English.

After the downward-pointing “axe” of “\( \lor \)” splits the wood, you can take 1 piece OR the other, or both.
Disjunction Truth Table

• Note that \( p \lor q \) means that \( p \) is true, or \( q \) is true, or both are true!

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

• So, this operation is also called *inclusive or*, because it *includes* the possibility that both \( p \) and \( q \) are true.

• “\( \neg \)” and “\( \lor \)” together are also universal.
Nested Propositional Expressions

- Use parentheses to group sub-expressions:
  “I just saw my old friend, and either he’s grown or I’ve shrunk.” = \( f \land (g \lor s) \)
  - \((f \land g) \lor s\) would mean something different
  - \(f \land g \lor s\) would be ambiguous

- By convention, “¬” takes precedence over both “∧” and “∨”.
  - \(\neg s \land f\) means \((\neg s) \land f\), not \(\neg (s \land f)\)
A Simple Exercise

Let \( p = \text{“It rained last night”} \),
\( q = \text{“The sprinklers came on last night,”} \)
\( r = \text{“The lawn was wet this morning.”} \)

Translate each of the following into English:
\[
\neg p = \text{“It didn’t rain last night.”}
\]
\[
r \land \neg p = \text{“The lawn was wet this morning, and it didn’t rain last night.”}
\]
\[
\neg r \lor p \lor q = \text{“Either the lawn wasn’t wet this morning, or it rained last night, or the sprinklers came on last night.”}
\]
The Exclusive Or Operator

The binary exclusive-or operator “⊕” (XOR) combines two propositions to form their logical “exclusive or” (exjunction?).

\[ p = \text{“I will earn an A in this course,”} \]
\[ q = \text{“I will drop this course,”} \]
\[ p \oplus q = \text{“I will either earn an A for this course, or I will drop it (but not both!”} \]
Note that $p \oplus q$ means that $p$ is true, or $q$ is true, but not both!

This operation is called exclusive or, because it excludes the possibility that both $p$ and $q$ are true.

“¬” and “⊕” together are not universal.

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$p \oplus q$</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>

Note difference from OR.
Natural Language is Ambiguous

Note that English “or” can be ambiguous regarding the “both” case!

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

Need context to disambiguate the meaning!

For this class, assume “or” means inclusive.
The *Implication* Operator

The *implication* \( p \implies q \) states that \( p \) implies \( q \). **antecedent** **consequent**

I.e., If \( p \) is true, then \( q \) is true; but if \( p \) is not true, then \( q \) could be either true or false.

E.g., let \( p = \text{“You study hard.”} \)

\[ q = \text{“You will get a good grade.”} \]

\( p \implies q = \text{“If you study hard, then you will get a good grade.”} \) (else, it could go either way)
Implication Truth Table

- **p → q** is **false** only when p is true but q is **not** true.
- **p → q** does **not** say that p **causes** q!
- **p → q** does **not** require that p or q **are ever true**!
- **E.g.** “(1=0) → pigs can fly” is TRUE!

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>p→q</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

The only False case!

For simplicity, I shall denote the implication operator by the symbol → and the iff operator by ↔
Examples of Implications

• “If this lecture ends, then the sun will rise tomorrow.” True or False?
• “If Tuesday is a day of the week, then I am a bird.” True or False?
• “If 1+1=6, then Bush is president.” True or False?
• “If the moon is made of green cheese, then I am richer than Bill Gates.” True or False?
Why does this seem wrong?

• Consider a sentence like,
  – “If I wear a red shirt tomorrow, then Arnold Schwarzenegger will become governor of California.”

• In logic, we consider the sentence True so long as either I don’t wear a red shirt, or Arnold wins.

• But in normal English conversation, if I were to make this claim, you would think I was lying.
  – Why this discrepancy between logic & language?
Resolving the Discrepancy

• In English, a sentence “if $p$ then $q$” usually really *implicitly* means something like,
  – “In all possible situations, if $p$ then $q$.”
    • That is, “For $p$ to be true and $q$ false is impossible.”
    • Or, “I guarantee that no matter what, if $p$, then $q$.”
• This can be expressed in *predicate logic* as:
  – “For all situations $s$, if $p$ is true in situation $s$, then $q$ is also true in situation $s$”
  – Formally, we could write: $\forall s, P(s) \rightarrow Q(s)$
• *That* sentence is logically *False* in our example, because for me to wear a red shirt and for Arnold to lose is a *possible* (even if not actual) situation.
  – Natural language and logic then agree with each other.
English Phrases Meaning $p \rightarrow q$

- “$p$ implies $q$”
- “if $p$, then $q$”
- “if $p$, $q$”
- “when $p$, $q$”
- “whenever $p$, $q$”
- “$q$ if $p$”
- “$q$ when $p$”
- “$q$ whenever $p$”

- “$p$ only if $q$”
- “$p$ is sufficient for $q$”
- “$q$ is necessary for $p$”
- “$q$ follows from $p$”
- “$q$ is implied by $p$”

If $p$ is true, that is enough, $q$ has to be true for the implication to hold (sufficiency)

If $q$ is false, $p$ cannot be true; It is necessary that $q$ be true for $p$ to be true (necessity)
Converse, Inverse, Contrapositive

Some terminology, for an implication \( p \rightarrow q \):

- Its converse is: \( q \rightarrow p \).
- Its inverse is: \( \neg p \rightarrow \neg q \).
- Its contrapositive: \( \neg q \rightarrow \neg p \).
- One of these three has the same meaning (same truth table) as \( p \rightarrow q \). Can you figure out which?
How do we know for sure?

Proving the equivalence of $p \rightarrow q$ and its contrapositive using truth tables:

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$</td>
<td>$q$</td>
<td>$\neg q$</td>
<td>$\neg p$</td>
<td>$p \rightarrow q$</td>
<td>$\neg q \rightarrow \neg p$</td>
</tr>
<tr>
<td>F→F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F→T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T→F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T→T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
The *biconditional* operator

The *biconditional* $p \leftrightarrow q$ states that $p$ is true *if and only if* (IFF) $q$ is true.

$p = \text{“Bush wins the 2004 election.”}$

$q = \text{“Bush will be president for all of 2005.”}$

$p \leftrightarrow q = \text{“If, and only if, Bush wins the 2004 election, Bush will be president for all of 2005.”}$

I'm still here!
Biconditional Truth Table

• $p \leftrightarrow q$ means that $p$ and $q$ have the same truth value.

• Note this truth table is the exact opposite of $\oplus$’s!

• $p \leftrightarrow q$ does not imply that $p$ and $q$ are true, or cause each other.

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

- We have seen 1 unary operator (out of the 4 possible) and 5 binary operators (out of the 16 possible). Their truth tables are below.

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>¬p</th>
<th>p ∧ q</th>
<th>p ∨ q</th>
<th>p ⊕ q</th>
<th>p → q</th>
<th>p ↔ q</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</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>
<td>T</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>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

Topic #1.0 – Propositional Logic: Operators
## Some Alternative Notations

<table>
<thead>
<tr>
<th>Name</th>
<th>not</th>
<th>and</th>
<th>or</th>
<th>xor</th>
<th>implies</th>
<th>iff</th>
</tr>
</thead>
<tbody>
<tr>
<td>Propositional logic:</td>
<td>¬</td>
<td>∧</td>
<td>∨</td>
<td>⊕</td>
<td>→</td>
<td>↔</td>
</tr>
<tr>
<td>Boolean algebra:</td>
<td>(\overline{p})</td>
<td>(pq)</td>
<td>+</td>
<td>⊕</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C/C++/Java (wordwise):</td>
<td>!</td>
<td>&amp; &amp;</td>
<td></td>
<td></td>
<td></td>
<td>!=</td>
</tr>
<tr>
<td>C/C++/Java (bitwise):</td>
<td>¬</td>
<td>&amp;</td>
<td></td>
<td>^</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logic gates:</td>
<td><img src="image1" alt="Not Gate" /></td>
<td><img src="image2" alt="And Gate" /></td>
<td><img src="image3" alt="Or Gate" /></td>
<td><img src="image4" alt="Xor Gate" /></td>
<td><img src="image5" alt="Implies Gate" /></td>
<td><img src="image6" alt="Iff Gate" /></td>
</tr>
</tbody>
</table>
Bits and Bit Operations

- A *bit* is a binary (base 2) digit: 0 or 1.
- Bits may be used to represent truth values.
- By convention:
  - 0 represents “false”; 1 represents “true”.
- *Boolean algebra* is like ordinary algebra except that variables stand for bits, + means “or”, and multiplication means “and”.

John Tukey (1915-2000)
Bit Strings

- A *bit string* of *length* $n$ is an ordered series or sequence of $n \geq 0$ bits.
  - More on sequences in §3.2.
- By convention, bit strings are written left to right: e.g. the first bit of “1001101010” is 1.
- When a bit string represents a base-2 number, by convention the first bit is the *most significant* bit. Ex. $1101_2 = 8 + 4 + 1 = 13$. 
Counting in Binary

• Did you know that you can count to 1,023 just using two hands?
  – How? Count in binary!
    • Each finger (up/down) represents 1 bit.

• To increment: Flip the rightmost (low-order) bit.
  – If it changes 1→0, then also flip the next bit to the left,
    • If that bit changes 1→0, then flip the next one, etc.

• 0000000000, 0000000001, 0000000010, …
  …, 1111111101, 1111111110, 1111111111
Bitwise Operations

• Boolean operations can be extended to operate on bit strings as well as single bits.

• E.g.:
  01 1011 0110
  11 0001 1101

  11 0001 1101

  Bit-wise OR
  Bit-wise AND
  Bit-wise XOR
Summary

You have learned about:

- Propositions: What they are.
- Propositional logic operators’
  - Symbolic notations.
  - English equivalents.
  - Logical meaning.
  - Truth tables.

- Nested propositions.
- Alternative notations.
- Bits and bit-strings.
- Next section:
  - Propositional equivalences.
  - How to prove them.
Propositional Equivalence

Two syntactically (i.e., textually) different compound propositions may be the semantically identical (i.e., have the same meaning). We call them equivalent. Learn:

- Various equivalence rules or laws.
- How to prove equivalences using symbolic derivations.
Tautologies and Contradictions

A *tautology* is a compound proposition that is **true no matter what** the truth values of its atomic propositions are!

*Ex. $p \lor \neg p$ [What is its truth table?]*

A *contradiction* is a compound proposition that is **false no matter what!** *Ex. $p \land \neg p$ [Truth table?]*

Other compound props. are *contingencies* (which is neither a tautology nor a contradiction)
Logical Equivalence

Compound proposition $p$ is \textit{logically equivalent} to compound proposition $q$, written $p \iff q$, \textbf{IFF} the compound proposition $p \leftrightarrow q$ is a tautology.

Compound propositions $p$ and $q$ are logically equivalent to each other \textbf{IFF} $p$ and $q$ contain the same truth values as each other in \textit{all} rows of their truth tables.
Ex. Prove that $p \lor q \iff \neg (\neg p \land \neg q)$.

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$p \lor q$</th>
<th>$\neg p$</th>
<th>$\neg q$</th>
<th>$\neg p \land \neg q$</th>
<th>$\neg (\neg p \land \neg q)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>
Constructing Truth table

*Construct a truth table for \( q \land \neg p \rightarrow p \).*

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

• These are similar to the arithmetic identities you may have learned in algebra, but for propositional equivalences instead.

• They provide a pattern or template that can be used to match all or part of a much more complicated proposition and to find an equivalence for it.
Equivalence Laws - Examples

• **Identity**: \( p \land T \iff p \quad p \lor F \iff p \)
• **Domination**: \( p \lor T \iff T \quad p \land F \iff F \)
• **Idempotent**: \( p \lor p \iff p \quad p \land p \iff p \)
• **Double negation**: \( \neg\neg p \iff p \)
• **Commutative**: \( p \lor q \iff q \lor p \quad p \land q \iff q \land p \)
• **Associative**: \( (p \lor q) \lor r \iff p \lor (q \lor r) \)
\( (p \land q) \land r \iff p \land (q \land r) \)
More Equivalence Laws

- **Distributive**: \[ p \lor (q \land r) \iff (p \lor q) \land (p \lor r) \]
  \[ p \land (q \lor r) \iff (p \land q) \lor (p \land r) \]

- **De Morgan’s**:\[ \neg (p \land q) \iff \neg p \lor \neg q \]
  \[ \neg (p \lor q) \iff \neg p \land \neg q \]

- **Trivial tautology/contradiction**:\[ p \lor \neg p \iff T \]
  \[ p \land \neg p \iff F \]
Defining Operators via Equivalences

Using equivalences, we can define operators in terms of other operators.

- **Exclusive or:** \( p \oplus q \iff (p \lor q) \land \neg (p \land q) \)
  \( p \oplus q \iff (p \land \neg q) \lor (q \land \neg p) \)

- **Implies:** \( p \to q \iff \neg p \lor q \)

- **Biconditional:** \( p \leftrightarrow q \iff (p \to q) \land (q \to p) \)
  \( p \leftrightarrow q \iff \neg (p \oplus q) \)
An Example Problem

• Check using a symbolic derivation whether

\[(p \land \neg q) \rightarrow (p \oplus r) \iff \neg p \lor q \lor \neg r.\]

\[(p \land \neg q) \rightarrow (p \oplus r) \iff \]

[Expand definition of \(\rightarrow\)] \(\neg(p \land \neg q) \lor (p \oplus r)\)

[Defn. of \(\oplus\)] \(\iff \neg(p \land \neg q) \lor ((p \lor r) \land \neg(p \land r))\)

[DeMorgan’s Law]

\(\iff (\neg p \lor q) \lor ((p \lor r) \land \neg(p \land r))\)

cont.
Example Continued...

\[(\neg p \lor q) \lor ((p \lor r) \land \neg(p \land r)) \iff [\lor \text{ commutes}]\]
\[\iff (q \lor \neg p) \lor ((p \lor r) \land \neg(p \land r)) \iff [\lor \text{ associative}]\]
\[\iff q \lor (\neg p \lor ((p \lor r) \land \neg(p \land r))) \iff [\text{distrib.} \lor \text{ over } \land]\]
\[\iff q \lor (((\neg p \lor (p \lor r)) \land (\neg p \lor \neg(p \land r))) \iff [\text{assoc.}]\]
\[\iff q \lor (((\neg p \lor p) \lor r) \land (\neg p \lor \neg(p \land r))) \iff [\text{trivial taut.}]\]
\[\iff q \lor ((\mathbf{I} \lor r) \land (\neg p \lor \neg(p \land r))) \iff [\text{domination}]\]
\[\iff q \lor (\mathbf{I} \land (\neg p \lor \neg(p \land r))) \iff [\text{identity}]\]
\[\iff q \lor (\neg p \lor \neg(p \land r)) \iff \text{cont.}\]
End of Long Example

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

[DeMorgan’s] \iff q \lor (\neg p \lor (\neg p \lor \neg r))

[Assoc.] \iff q \lor ((\neg p \lor \neg p) \lor \neg r)

[Idempotent] \iff q \lor (\neg p \lor \neg r)

[Assoc.] \iff (q \lor \neg p) \lor \neg r

[Commut.] \iff \neg p \lor q \lor \neg r

Q.E.D. (quod erat demonstrandum)

(Which was to be shown.)
Review: Propositional Logic

- Atomic propositions: \( p, q, r, \ldots \)
- Boolean operators: \( \neg \land \lor \oplus \to \leftrightarrow \)
- Compound propositions: \( s \equiv (p \land \neg q) \lor r \)
- Equivalences: \( p \land \neg q \leftrightarrow \neg (p \to q) \)
- Proving equivalences using:
  - Truth tables.
  - Symbolic derivations. \( p \leftrightarrow q \leftrightarrow r \ldots \)
Predicate Logic

• Language of propositions not sufficient to make all assertions needed in mathematics
  – $x=3$, $x+y=z$
  – They are not propositions (Why?)
  – However if values are assigned they do

• Consider the assertion:
  – He is tall and dark
  – These assertions are formed using variables, in a template. The template is called the predicate.
Contd…

• Assertion: \( x \) is tall and dark.
  – \( x \) is the variable
  – “is tall and dark” is the predicate
Applications of Predicate Logic

It is the formal notation for writing perfectly clear, concise, and unambiguous mathematical definitions, axioms, and theorems for any branch of mathematics.

Predicate logic with function symbols, the “=” operator, and a few proof-building rules is sufficient for defining any conceivable mathematical system, and for proving anything that can be proved within that system!
Other Applications

- Predicate logic is the foundation of the field of mathematical logic, which culminated in Gödel’s incompleteness theorem, which revealed the ultimate limits of mathematical thought:
  - Given any finitely describable, consistent proof procedure, there will still be some true statements that can never be proven by that procedure.
- I.e., we can’t discover all mathematical truths, unless we sometimes resort to making guesses.
Practical Applications

• Basis for clearly expressed formal specifications for any complex system.
• Basis for *automatic theorem provers* and many other Artificial Intelligence systems.
Subjects and Predicates

• In the sentence “The dog is sleeping”:
  – The phrase “the dog” denotes the subject - the object or entity that the sentence is about.
  – The phrase “is sleeping” denotes the predicate: a property that is true of the subject.

• In predicate logic, a predicate is modeled as a function $P(\cdot)$ from objects to propositions.
  – $P(x) =$ “x is sleeping” (where x is any object).
More About Predicates

• Convention: Lowercase variables $x, y, z...$ denote objects/entities; uppercase variables $P, Q, R...$ denote propositional functions (predicates).

• Keep in mind that the result of applying a predicate $P$ to an object $x$ is the proposition $P(x)$. But the predicate $P$ itself (e.g. $P$="is sleeping") is not a proposition (not a complete sentence).
  – E.g. if $P(x) = "x$ is a prime number",
    $P(3)$ is the proposition “3 is a prime number.”
Propositional Functions

• Predicate logic generalizes the grammatical notion of a predicate to also include propositional functions of any number of arguments, each of which may take any grammatical role that a noun can take.

  – *E.g.* let $P(x,y,z) = \text{“}x \text{ gave } y \text{ the grade } z\text{”},$ then if $x=\text{“}Mike\text{”}, \ y=\text{“}Mary\text{”}, \ z=\text{“}A\text{”},$ then $P(x,y,z) = \text{“}Mike \text{ gave } Mary \text{ the grade A}\text{”}\text{.}
Universes of Discourse (U.D.s)

• The power of distinguishing objects from predicates is that it lets you state things about *many* objects at once.

• E.g., let $P(x) = \text{“}x+1>x\text{”}. We can then say, “For *any* number $x$, $P(x)$ is true” instead of $\text{(0+1>0) \land (1+1>1) \land (2+1>2) \land ...}$

• The collection of values that a variable $x$ can take is called $x$’s *universe of discourse.*
Types of predicates

• Consider a predicate: $P(c_1, c_2, \ldots, c_n)$

• **Defn:**
  
  – **Valid:** Value of $P$ is true for all choices of the argument
  
  – **Satisfiable:** Value of $P$ is true for some value of the argument
  
  – **Unsatisfiable:** Value of $P$ is never true for the possible choices of the argument
Quantifier Expressions

- **Quantifiers** provide a notation that allows us to quantify (count) how many objects in the universe of discourse satisfy a given predicate.

- “∀” is the FOR ALL or universal quantifier. ∀x P(x) means *for all* x in the u.d., P holds.

- “∃” is the EXISTS or existential quantifier. ∃x P(x) means there exists an x in the u.d. (that is, 1 or more) such that P(x) is true.
The Universal Quantifier $\forall$

- Example:
  Let the u.d. of $x$ be parking spaces at IITM. Let $P(x)$ be the predicate “$x$ is full.” Then the universal quantification of $P(x)$, $\forall x P(x)$, is the proposition:
  - “All parking spaces at IITM are full.”
  - *i.e.*, “Every parking space at IITM is full.”
  - *i.e.*, “For each parking space at IITM, that space is full.”
The Existential Quantifier \( \exists \)

- Example:
  Let the u.d. of \( x \) be parking spaces at IITM. Let \( P(x) \) be the predicate “\( x \) is full.” Then the existential quantification of \( P(x) \), \( \exists x \ P(x) \), is the proposition:
  - “Some parking space at IITM is full.”
  - “There is a parking space at IITM that is full.”
  - “At least one parking space at IITM is full.”
Question

• What is a predicate with zero variables called?
Free and Bound Variables

• An expression like $P(x)$ is said to have a free variable $x$ (meaning, $x$ is undefined).

• A quantifier (either $\forall$ or $\exists$) operates on an expression having one or more free variables, and binds one or more of those variables, to produce an expression having one or more bound variables.

• Binding converts a predicate to a proposition
Example of Binding

• $P(x,y)$ has 2 free variables, $x$ and $y$.
• $\forall x P(x,y)$ has 1 free variable, and one bound variable. [Which is which?]
• “$P(x)$, where $x=3$” is another way to bind $x$.
• An expression with zero free variables is a bona-fide (actual) proposition
• An expression with one or more free variables is still only a predicate: $\forall x P(x,y)$
Nesting of Quantifiers

Example: Let the u.d. of $x$ & $y$ be people. Let $L(x,y) = \text{"} x \text{ likes } y \text{"} $ (a predicate w. 2 f.v.’s)

Then $\exists y \ L(x,y) = \text{"} There is someone whom } x \text{ likes.\"} $ (A predicate w. 1 free variable, $x$)

Then $\forall x (\exists y L(x,y)) =$

"Everyone has someone whom they like.\" (A __________ with ___ free variables.)

Proposition 1
Review: Propositional Logic

- Atomic propositions: \( p, q, r, \ldots \)
- Boolean operators: \( \neg \land \lor \oplus \rightarrow \leftrightarrow \)
- Compound propositions: \( s \equiv (p \land \neg q) \lor r \)
- Equivalences: \( p \land \neg q \iff \neg (p \rightarrow q) \)
- Proving equivalences using:
  - Truth tables.
  - Symbolic derivations. \( p \iff q \iff r \ldots \)
Review: Predicate Logic

• Objects $x, y, z, \ldots$
• Predicates $P, Q, R, \ldots$ are functions mapping objects $x$ to propositions $P(x)$.
• Multi-argument predicates $P(x, y)$.
• Quantifiers: $[\forall x P(x)] \equiv \text{“For all x’s, } P(x).”$ $[\exists x P(x)] \equiv \text{“There is an x such that } P(x).”$
• Universes of discourse, bound & free vars.
Quantifier Exercise

If $R(x,y) = "x relies upon y,” express the following in unambiguous English:

\[ \forall x (\exists y \ R(x,y)) = \]
\[ \exists y (\forall x \ R(x,y)) = \]
\[ \exists x (\forall y \ R(x,y)) = \]
\[ \forall y (\exists x \ R(x,y)) = \]
\[ \forall x (\forall y \ R(x,y)) = \]

- Everyone has someone to rely on.
- There’s a poor overburdened soul whom everyone relies upon (including himself)!
- There’s some needy person who relies upon everybody (including himself).
- Everyone has someone who relies upon them.
- Everyone relies upon everybody, (including themselves)!
Natural language is ambiguous!

• “Everybody likes somebody.”
  – For everybody, there is somebody they like,
    • $\forall x \exists y \text{Likes}(x,y)$ [Probably more likely.]
  – or, there is somebody (a popular person) whom everyone likes?
    • $\exists y \forall x \text{Likes}(x,y)$

• “Somebody likes everybody.”
  – Same problem: Depends on context, emphasis.
Game Theoretic Semantics

• Thinking in terms of a competitive game can help you tell whether a proposition with nested quantifiers is true.

• The game has two players, both with the same knowledge:
  – Verifier: Wants to demonstrate that the proposition is true.
  – Falsifier: Wants to demonstrate that the proposition is false.

• The Rules of the Game “Verify or Falsify”:
  – Read the quantifiers from left to right, picking values of variables.
  – When you see “∀”, the falsifier gets to select the value.
  – When you see “∃”, the verifier gets to select the value.

• If the verifier can always win, then the proposition is true.
• If the falsifier can always win, then it is false.
Let’s Play, “Verify or Falsify!”

Let $B(x,y) \equiv \text{“}x\text{’s month of birthday is the same as that of } y\text{”}$

Suppose I claim that among you:

$$\forall x \exists y \ B(x,y)$$

Your turn, as falsifier:
- You pick any $x \rightarrow (so\text{-}and\text{-}so)$
- $\exists y \ B(so\text{-}and\text{-}so,y)$

My turn, as verifier:
- I pick any $y \rightarrow (such\text{-}and\text{-}such)$
- $B(so\text{-}and\text{-}so,such\text{-}and\text{-}such)$

Let’s play it in class.
Who wins this game?
What if I switched the quantifiers, and I claimed that
$\exists y \ \forall x \ B(x,y)$?
Who wins in that case?
Still More Conventions

- Sometimes the universe of discourse is restricted within the quantification, e.g.,
  - $\forall x > 0 \ P(x)$ is shorthand for “For all $x$ that are greater than zero, $P(x)$.”
    
    $= \forall x \ (x > 0 \rightarrow P(x))$

  - $\exists x > 0 \ P(x)$ is shorthand for “There is an $x$ greater than zero such that $P(x)$.”
    
    $= \exists x \ (x > 0 \land P(x))$
More to Know About Binding

• $\forall x \exists x P(x)$ - $x$ is not a free variable in $\exists x P(x)$, therefore the $\forall x$ binding isn’t used.

• $(\forall x P(x)) \land Q(x)$ - The variable $x$ is outside of the scope of the $\forall x$ quantifier, and is therefore free. Not a complete proposition!

• $(\forall x P(x)) \land (\exists x Q(x))$ – This is legal, because there are 2 different $x$’s!
Commutativity of Quantifiers

- $\forall x \exists y \ P(x,y) \neq \exists y \forall x \ P(x,y)$
- $\forall x \ \forall y \ P(x,y) = \forall y \ \forall x \ P(x,y)$
- $\exists x \ \exists y \ P(x,y) = \exists y \ \exists x \ P(x,y)$

It is easy to disprove (give a counter-example)

Prove or disprove the above statements
Quantifier Equivalence Laws

• Definitions of quantifiers: If u.d.=a,b,c,…
  \( \forall x \ P(x) \iff P(a) \land P(b) \land P(c) \land \ldots \)
  \( \exists x \ P(x) \iff P(a) \lor P(b) \lor P(c) \lor \ldots \)

• From those, we can prove the laws:
  \( \forall x \ P(x) \iff \neg \exists x \ \neg P(x) \)
  \( \exists x \ P(x) \iff \neg \forall x \ \neg P(x) \)

• Which propositional equivalence laws can be used to prove this?
More Equivalence Laws

- \( \forall x \forall y P(x,y) \iff \forall y \forall x P(x,y) \)
  \( \exists x \exists y P(x,y) \iff \exists y \exists x P(x,y) \)

- \( \forall x (P(x) \land Q(x)) \iff (\forall x P(x)) \land (\forall x Q(x)) \)
  \( \exists x (P(x) \lor Q(x)) \iff (\exists x P(x)) \lor (\exists x Q(x)) \)

- Exercise:
  See if you can prove these yourself.
  - What propositional equivalences did you use?
Review: Predicate Logic

• Objects $x, y, z, \ldots$
• Predicates $P, Q, R, \ldots$ are functions mapping objects $x$ to propositions $P(x)$.
• Multi-argument predicates $P(x, y)$.
• Quantifiers: $(\forall x \ P(x))$ = “For all x’s, $P(x)$.”
  $(\exists x \ P(x))$ = “There is an $x$ such that $P(x)$.”
Defining New Quantifiers

As per their name, quantifiers can be used to express that a predicate is true of any given *quantity* (number) of objects.

Define $\exists! x \ P(x)$ to mean “$P(x)$ is true of exactly one $x$ in the universe of discourse.”

$\exists! x \ P(x) \iff \exists x \ (P(x) \land \neg \exists y \ (P(y) \land y \neq x))$

“There is an $x$ such that $P(x)$, where there is no $y$ such that $P(y)$ and $y$ is other than $x$.”
More about Quantifiers

• State True or False with reasons:
  – $\forall$ distributes over $\land$
  – $\forall$ distributes over $\lor$
  – $\exists$ distributes over $\land$
  – $\exists$ distributes over $\lor$
  – $\exists x[P(x) \land Q(x)] \rightarrow \exists xP(x) \land \exists xQ(x)$
  – $\forall x[P(x) \lor Q(x)] \rightarrow \forall xP(x) \lor \forall xQ(x)$
Prove or disprove:
\[ \exists x [P(x) \rightarrow Q(x)] \iff [\exists x P(x) \rightarrow \exists x Q(x)] \]
\[ \exists x [P(x) \rightarrow Q(x)] \iff \exists x [\neg P(x) \lor Q(x)] \]
\[ \iff \exists x [\neg P(x)] \lor \exists x Q(x) \iff \neg \forall x P(x) \lor \exists x Q(x) \]
\[ \iff \forall x P(x) \rightarrow \exists x Q(x) \]

Hence we are to check:
\[ [\forall x P(x) \rightarrow \exists x Q(x)] \iff [\exists x P(x) \rightarrow \exists x Q(x)] \]
# Truth Table

<table>
<thead>
<tr>
<th>$\forall x P(x)$</th>
<th>$\exists x P(x)$</th>
<th>$\exists x Q(x)$</th>
<th>$\forall x P(x) \rightarrow \exists x Q(x)$</th>
<th>$\exists x P(x) \rightarrow \exists x Q(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>n.a</td>
<td>n.a</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>n.a</td>
<td>n.a</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Building Counter-example

• Build the counter-example, so that we satisfy the line of the truth-table which makes the difference:
  – Here, $\forall x P(x) = 0$, $\exists x P(x) = 1$, $\exists x Q(x) = 0$
  – Example: $P(x)$ is satisfiable and $Q(x)$ is unsatisfiable
  – $P(x)$: $x = 0$, $Q(x)$: $x \neq x$. 
Some Number Theory Examples

• Let u.d. = the natural numbers 0, 1, 2, …
• “A number $x$ is even, $E(x)$, if and only if it is equal to 2 times some other number.”
  \[ \forall x \ (E(x) \leftrightarrow (\exists y \ x=2y)) \]
• “A number is prime, $P(x)$, iff it’s greater than 1 and it isn’t the product of two non-unity numbers.”
  \[ \forall x \ (P(x) \leftrightarrow (x>1 \land \neg \exists yz \ x=yz \land y\neq1 \land z\neq1)) \]
Goldbach’s Conjecture (unproven)

Using \( E(x) \) and \( P(x) \) from previous slide,
\[
\forall E(x>2): \exists P(p), P(q): p+q = x
\]
or, with more explicit notation:
\[
\forall x \ [x>2 \land E(x)] \rightarrow \\
\exists p \ \exists q \ P(p) \land P(q) \land p+q = x.
\]
“Every even number greater than 2 is the sum of two primes.”
Deduction Example

• Definitions:
  \[ s \equiv \text{Socrates (ancient Greek philosopher);} \]
  \[ H(x) \equiv \text{“}x\text{ is human”;} \]
  \[ M(x) \equiv \text{“}x\text{ is mortal”;} \]

• Premises:
  \[ H(s) \quad \text{Socrates is human.} \]
  \[ \forall x \ H(x) \rightarrow M(x) \quad \text{All humans are mortal.} \]

Prove, Socrates is mortal!!
Some valid conclusions you can draw:

\( H(s) \rightarrow M(s) \) \[\text{Instantiate universal.}\] If Socrates is human then he is mortal.

\( \neg H(s) \lor M(s) \)

Socrates is inhuman or mortal.

\( H(s) \land (\neg H(s) \lor M(s)) \)

Socrates is human, and also either inhuman or mortal.

\( (H(s) \land \neg H(s)) \lor (H(s) \land M(s)) \) \[\text{Apply distributive law.}\]

\( F \lor (H(s) \land M(s)) \) \[\text{Trivial contradiction.}\]

\( H(s) \land M(s) \) \[\text{Use identity law.}\]

\( M(s) \)

Socrates is mortal.
Another Example

• Definitions: $H(x) \equiv \text{“}x \text{ is human”};$
  $M(x) \equiv \text{“}x \text{ is mortal”};$  $G(x) \equiv \text{“}x \text{ is a god”}$

• Premises:
  – $\forall x \ H(x) \rightarrow M(x)$ (“Humans are mortal”) and
  – $\forall x \ G(x) \rightarrow \neg M(x)$ (“Gods are immortal”).

• Show that $\neg \exists x \ (H(x) \land G(x))$
  (“No human is a god.”)
Summary

• From these sections you should have learned:
  – Predicate logic notation & conventions
  – Conversions: predicate logic ↔ clear English
  – Meaning of quantifiers, equivalences
  – Simple reasoning with quantifiers

• Upcoming topics:
  – Introduction to proof-writing.
  – Then: Set theory –
    • a language for talking about collections of objects.