UC Davis
ECS20, Winter 2017

Discrete Mathematics for Computer Science

Prof. Raissa D’Souza
(slides adopted from Michael Frank and Haluk Bingöl)
Foundations of Logic: Overview

- Propositional logic (§1.1-1.2):
  - Basic definitions. (§1.1)
  - Equivalence rules & derivations. (§1.2)

- Predicate logic (§1.3-1.4)
  - Predicates.
  - Quantified predicate expressions.
  - Equivalences & derivations.
Topic #1 – Propositional Logic

**Definition of a *Proposition***

- **Definition:** A *proposition* (denoted *p*, *q*, *r*, ...) is simply:
  - A *declarative statement* with *some definite meaning*, (not vague or ambiguous)
  - *having a truth value* that is either *true* (**T**) or *false* (**F**)  
    - it is **never** both, neither, or somewhere “in between”
      - However, you might not *know* the actual truth value,
      - and, the truth value might *depend* on the situation or context.
  - Algebra, mathematics expressed in terms of unknown *variables* like *x* and *n* are **NOT** propositions: *predicate logic next topic*
    - *X + 10 = 30*
    - *1024 > 2^n*
Example – Are these statements propositions?

• $p = \text{“This statement is true”} \quad (\text{assert } p = T)$
  • Yes a proposition; consistent $T/F$ assignment
  • If $p = T$ then statement is true
  • If $p = F$ then $\neg p$ and the statement is not true

• $p = \text{“This statement is false”} \quad (\text{assert } p = F)$
  • No, not a proposition; cannot assign $T$ or $F$
  • If $p = F$ then, in fact, $p = T$
  • If $p = T$ then, in fact, $p = F$

(Recall, a proposition cannot be both $T$ and $F$, or partway)

The compound proposition $p \land \neg p$ is $F$
## 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>
Truth tables

- To evaluate the T or F status of a compound proposition
- Enumerate over all T and F combinations of all the propositions
<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>

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

<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>
Nested Propositional Expressions

• Use parentheses to **group sub-expressions**: “I just saw my old friend, and either he’s grown or I’ve shrunk.”

• First break it down into propositions:
  • \( f = “I \text{ just saw my old friend}” \)
  • \( g = \text{“he’s grown”} \)
  • \( s = \text{“I’ve shrunk”} \)

• \( = f \land (g \lor s) \)
  • \((f \land g) \lor s\) would mean something different
  • \(f \land g \lor s\) would be ambiguous

• By convention, “\( \neg \)” takes precedence over both “\( \land \)” and “\( \lor \)”
  • \( \neg s \land f\) means \((\neg s) \land f\), **not** \(\neg (s \land f)\)
A Simple Exercise

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

• Translate each of the following into English:
  • \( \neg p \) =
  • \( r \land \neg p \) =
  • \( \neg r \lor p \lor q \) =
The **Exclusive Or** Operator

- The binary *exclusive-or operator* “⊕” (XOR) combines two propositions to form their logical "exclusive or" (exclusive disjunction)

- \( p = \) “I will earn an A in this course,”
- \( q = \) “I will drop this course,”
- \( p \oplus q = \) “I will either earn an A in this course, or I will drop it (but not both!)”

- A more common phrase: “Your entrée comes with either soup of salad”
Exclusive-Or Truth Table

- 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.

- Remark. “−” and “⊕” together are not universal.

\[
\begin{array}{c|c|c}
 p & q & p \oplus q \\
 F & F & F \\
 F & T & T \\
 T & F & T \\
 T & T & F \\
\end{array}
\]
Natural Language is Ambiguous

• Note that English “or” can be ambiguous regarding the “both” case!
  • “Pat is a singer or Pat is a writer.”
  • “Pat is alive or Pat is deceased.”
• Need context to disambiguate the meaning!
• For this class, assume “or” means inclusive.

\[
P \quad q \quad p \quad "or" \quad q
\]

\[
\begin{array}{ccc}
F & F & F \\
F & T & T \\
T & F & T \\
T & T & ?
\end{array}
\]
The *Implication* Operator

- The *implication* $p \rightarrow q$ states that $p$ implies $q$.

- *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 $p = \text{“You master ECS20.”}$
  
  $q = \text{“You will get a good job.”}$

- $p \rightarrow q = \text{“If you master ECS20, then you will get a good job.”}$

(else, it could go either way; some great jobs do not require discrete math)

*Let’s build the truth table for $p \rightarrow q$*
Implication Truth Table

• \( p \rightarrow q \) is **false only** when 
  \( p \) is true but \( q \) is **not** true.

• \( p \rightarrow q \) does **not** say 
  that \( p \) **causes** \( q \)!

• \( p \rightarrow q \) does **not** require 
  that \( p \) or \( q \) are **ever true**!

• *E.g.* “\((1=0) \rightarrow \text{pigs can fly}\)” **is TRUE**!
Examples of Implications

• “If this lecture ever ends, then the sun will rise tomorrow.” True or False?

• “If Tuesday is a day of the week, then I am a penguin.” 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?
Examples of Implications

• “If this lecture ever ends, then the sun will rise tomorrow.” True or False?

• “If Tuesday is a day of the week, then I am a penguin.” 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 global peace will prevail”

• In logic, we consider the sentence **True** so long as either I don’t wear a red shirt, or global peace is not achieved.

• But, in normal English conversation, if I were to make this claim, you would think that I was crazy.
  
  • Why this discrepancy between logic & language?
    
    • **Logic is about consistency.**
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 $p$”

• We will see some equivalent logic expressions later.
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?
  (Let’s work it out on the board).
• Proving the equivalence of $p \rightarrow q$ and its contrapositive using truth tables:

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

- The *biconditional* \( p \iff q \) states that \( p \rightarrow q \) and \( q \rightarrow p \)
- In other words, \( p \) is true *if and only if (IFF) q* is true.

- \( p = “Clinton wins the 2016 election.””
- \( q = “Clinton will be president for all of 2017.””
- \( p \iff q = “If, and only if, Clinton wins the 2016 election, Clinton will be president for all of 2017.””

Biconditional Truth Table

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

- **Remark.** This truth table is the exact opposite of $\oplus$’s!
  - Thus, $p \leftrightarrow q$ means $\neg(p \oplus q)$

- $p \leftrightarrow q$ does not imply that $p$ and $q$ are true, or that either of them causes the other, or that they have a common cause.
### Boolean Operations Summary

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

**Order of operation:** ¬, ^, v, ⊕, →, ↔

i.e., p v ¬q → p ^ q means (p v (¬q)) → (p ^ q)

*(Note, precedence of v, ⊕ is ambiguous and often depends on the programming language)*
## 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>p̅</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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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”.
  - See module 23 (chapter 10) for more details.
Propositional Equivalence
Propositional Equivalence (§1.2)

• 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**. (i.e. most propositions are contingencies)
Logical Equivalence

• Compound proposition $p$ is *logically equivalent* to compound proposition $q$, written $p \iff q$, **IFF** the compound proposition $p \leftrightarrow q$ is a tautology.

• Note, $\iff$ is often denoted by $\equiv$
  
  (We will use both notations in this class)

• Compound propositions $p$ and $q$ are logically equivalent to each other **IFF** $p$ and $q$ contain the same truth values as each other in all rows of their truth tables.
Proving Equivalence via Truth Tables

• Prove that $p \lor q \iff \neg (\neg p \land \neg q)$. 
Proving Equivalence via 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>T</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>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</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>
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**: \( p \land T \iff p \) \( p \lor F \iff p \)
- **Domination**: \( p \lor T \iff T \) \( p \land F \iff F \)
- **Idempotent**: \( p \lor p \iff p \) \( p \land p \iff p \)
- **Double negation**: \( \neg \neg p \iff p \)
- **Commutative**: \( p \lor q \iff q \lor p \) \( p \land q \iff q \land p \)
- **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:
\[ p \lor \neg p \iff T \quad p \land \neg p \iff F \]
Defining Operators via Equivalences

• Using equivalences, we can \textit{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: \[ p \rightarrow q \iff \neg p \lor q \]

• Biconditional: \[ p \leftrightarrow q \iff (p \rightarrow q) \land (q \rightarrow p) \]
  \[ p \leftrightarrow q \iff \neg(p \oplus q) \]
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 \neg(p \land \neg q) \lor (p \oplus r) \quad \text{[Expand definition of \(\rightarrow\)]}\]
• \[\iff \neg(p \land \neg q) \lor ((p \lor r) \land \neg(p \land r)) \quad \text{[Expand defn. of \(\oplus\)]}\]
• \[\iff \neg(p \land \neg q) \lor ((p \lor r) \land \neg(p \land r)) \quad \text{[DeMorgan’s Law]}\]
• \[\text{cont.}\]
Example Continued...

• \( \Leftrightarrow (-p \lor q) \lor ((p \lor r) \land \neg (p \land r)) \)
• \( \Leftrightarrow (q \lor \neg p) \lor ((p \lor r) \land \neg (p \land r)) \) [\( \lor \) commutes]
• \( \Leftrightarrow q \lor (-p \lor ((p \lor r) \land \neg (p \land r))) \) [\( \lor \) associative]
• \( \Leftrightarrow q \lor (((-p \lor (p \lor r)) \land (-p \lor \neg (p \land r)))) \) [distrib. \( \lor \) over \( \land \)]
\( \Leftrightarrow q \lor (((-p \lor p) \lor r) \land (-p \lor \neg (p \land r))) \) [assoc.]
\( \Leftrightarrow q \lor ((T \lor r) \land (-p \lor \neg (p \land r))) \) [trivial taut.]
\( \Leftrightarrow q \lor (T \land (-p \lor \neg (p \land r))) \) [domination]
\( \Leftrightarrow q \lor (-p \lor \neg (p \land r)) \) [identity]

cont.
End of Long Example

\[ q \lor (\neg p \lor \neg (p \land r)) \]
\[ \iff q \lor (\neg p \lor (\neg p \lor \neg r)) \text{ [DeMorgan’s]} \]
\[ \iff q \lor ((\neg p \lor \neg p) \lor \neg r) \text{ [Assoc.]} \]
\[ \iff q \lor (\neg p \lor \neg r) \text{ [Idempotent]} \]
\[ \iff (q \lor \neg p) \lor \neg r \text{ [Assoc.]} \]
\[ \iff \neg p \lor q \lor \neg r \text{ [Commut.]} \]

Q.E.D.

Remark. Q.E.D. (quod erat demonstrandum)

(Which was to be shown.)
Review: Propositional Logic
(§§1.1-1.2)

• Atomic propositions: $p$, $q$, $r$, ...

• 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. $p \iff q \iff r$ ...
Predicate Logic
Predicate Logic (§1.3)

- *Predicate logic* is an extension of propositional logic that permits concisely reasoning about whole *classes* of entities.

- Propositional logic (recall) treats simple *propositions* (sentences) as atomic entities.

- In contrast, *predicate* logic distinguishes the *subject* of a sentence from its *predicate*.

- Remember these English grammar terms?
Applications of Predicate Logic

• It is *the* formal notation for writing perfectly clear, concise, and unambiguous mathematical *definitions, axioms, and theorems* (more on these later) 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 always remain some true statements that will never be proven by that procedure.

- i.e., we can’t discover all mathematical truths, unless we sometimes resort to making guesses.
Practical Applications of Predicate Logic

- It is the basis for clearly expressed formal specifications for any complex system.

- It is basis for *automatic theorem provers* and many other Artificial Intelligence systems.
  - *E.g.* automatic program verification systems.

- Predicate-logic like statements are supported by some of the more sophisticated *database query engines* and *container class libraries*
  - these are types of programming tools.
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 \( x, y, z \ldots \) denote objects/entities; uppercase variables \( P, Q, R \ldots \) denote propositional functions (predicates).

- **Remark.** 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 \( P(x,y,z) = \) “\( x \) gave \( y \) the grade \( z \)” , then if
  
  \[ x = \) “Mike” , \( y = \) “Mary”, \( z = \) “A”, then \( P(x,y,z) = \) “Mike gave Mary the grade A.”