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.

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 = "This statement is true" (assert p = **T**)
  - Yes a proposition; consistent ***T/F*** assignment
  - If p = ***T*** then statement is true
  - If p = ***F*** then ¬p and the statement is not true

- p = "This statement is false" (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 ∧ ¬p is F

Some Popular Boolean Operators

Formal Name	**Nickname**	**Arity**	**Symbol**
Negation operator	NOT	Unary	$\neg$
Conjunction operator	AND	Binary	$\wedge$
Disjunction operator	OR	Binary	$\vee$
Exclusive-OR operator	XOR	Binary	$\oplus$
Implication operator	IMPLIES	Binary	$\rightarrow$
Biconditional operator	IFF	Binary	$\leftrightarrow$

Truth tables

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

p	$\neg p$
T	F
F	T

p	q	$p \wedge q$
F	F	F
F	T	F
T	F	F
T	T	T

p	q	$p \vee q$
F	F	F
F	T	**T**
T	F	**T**
T	T	T

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 just saw my old friend"
  - g = "he's grown"
  - s = "I've shrunk"

- $= f \wedge (g \vee s)$
  - $(f \wedge g) \vee s$ would mean something different
  - $f \wedge g \vee s$ would be ambiguous

- By convention, "¬" takes *precedence* over both "∧" and "∨".
  - $\neg s \wedge f$ means $(\neg s) \wedge f$, **not** $\neg (s \wedge f)$

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:
- $\neg p$ =
- $r \wedge \neg p$ =
- $\neg r \vee p \vee q$ =

The *Exclusive Or* Operator

- The binary *exclusive-or operator* "$\oplus$" (*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**!

p	q	$p \oplus q$
F	F	F
F	T	T
T	F	T
T	T	**F**

Note difference from OR.

- This operation is called *exclusive or,* because it **excludes** the possibility that both p and q are true.
- Remark. "¬" and "⊕" together are **not** universal.

Natural Language is Ambiguous

- Note that <u>English</u> "or" can be <u>ambiguous</u> regarding the "both" case!
- "Pat is a singer or Pat is a writer." - $\vee$
- "Pat is alive or Pat is deceased." - $\oplus$

p	q	p "or" q
F	F	F
F	T	T
T	F	T
T	T	?

- Need context to disambiguate the meaning!
- For this class, assume "or" means <u>inclusive</u>.

The *Implication* Operator

antecedent consequent

- 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 = "You master ECS20."
  q = "You will get a good job."

- $p \rightarrow q$ = "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** <u>only</u> when p is true but q is **not** true.
- $p \rightarrow q$ does **not** say that p <u>causes</u> q!
- $p \rightarrow q$ does **not** require that p or q **<u>are ever true</u>**!
- *E.g.* "(1=0) → pigs can fly" is TRUE!

p	q	$p \rightarrow q$
F	F	T
F	T	T
T	F	**F**
T	T	T

The <u>only</u> False case!

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).

Contrapositive

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

p	q	$\neg q$	$\neg p$	$p \rightarrow q$	$\neg q \rightarrow \neg p$
F	F	T	T	T	T
F	T	F	T	T	T
T	F	T	F	F	F
T	T	F	F	T	T

The *biconditional* operator

- The *biconditional* $p \leftrightarrow q$ 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."
- $p \leftrightarrow q$ = "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	q	$p \leftrightarrow q$
F	F	T
F	T	F
T	F	F
T	T	T

- $p \leftrightarrow q$ 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

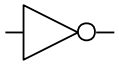
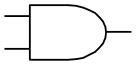
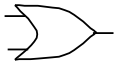
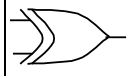
p	q	$\neg p$	$p \wedge q$	$p \vee q$	$p \oplus q$	$p \rightarrow q$	$p \leftrightarrow q$
F	F	T	F	F	F	T	T
F	T	T	F	T	T	T	F
T	F	F	F	T	T	F	F
T	T	F	T	T	F	T	T

Order of operation: $\neg$, $\wedge$, $\vee$, $\oplus$, $\rightarrow$, $\leftrightarrow$

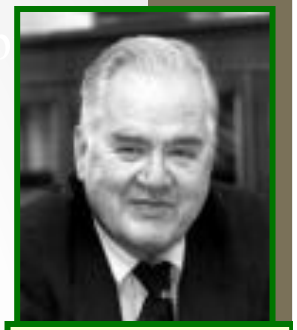
i.e., $p \vee \neg q \rightarrow p \wedge q$ means $(p \vee (\neg q)) \rightarrow (p \wedge q)$

(Note, precedence of $\vee$, $\oplus$ is ambiguous and often depends on the programming language)

Some Alternative Notations

Name:	not	and	or	xor	implies	iff
Propositional logic:	$\neg$	$\wedge$	$\vee$	$\oplus$	$\rightarrow$	$\leftrightarrow$
Boolean algebra:	$\overline{p}$	pq	$+$	$\oplus$		
C/C++/Java (wordwise):	!	&&	||	!=		==
C/C++/Java (bitwise):	~	&	|	^		
Logic gates:						

Bits and Bit Operations



John Tukey
(1915-2000)

- 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.* $p \vee \neg p$ [What is its truth table?]

- A *contradiction* is a compound proposition that is **false** no matter what! *Ex.* $p \wedge \neg p$ [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 $p \Leftrightarrow q$, **IFF** the compound proposition $p \leftrightarrow q$ is a tautology.
- Note, $\Leftrightarrow$ 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 <u>all</u> rows of their truth tables.

Proving Equivalence via Truth Tables

- Prove that $p \vee q \Leftrightarrow \neg(\neg p \wedge \neg q)$.

Proving Equivalence via Truth Tables

- *Ex.* Prove that $p \vee q \Leftrightarrow \neg(\neg p \wedge \neg q)$.

p	q	$p \vee q$	$\neg p$	$\neg q$	$\neg p \wedge \neg q$	$\neg(\neg p \wedge \neg q)$
F	F	F	T	T	T	F
F	T	T	T	F	F	T
T	F	T	F	T	F	T
T	T	T	F	F	F	T

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*: $p \wedge \mathbf{T} \Leftrightarrow p$ $\quad p \vee \mathbf{F} \Leftrightarrow p$
- *Domination*: $p \vee \mathbf{T} \Leftrightarrow \mathbf{T}$ $\quad p \wedge \mathbf{F} \Leftrightarrow \mathbf{F}$
- *Idempotent*: $p \vee p \Leftrightarrow p$ $\quad p \wedge p \Leftrightarrow p$
- *Double negation:* $\neg\neg p \Leftrightarrow p$
- *Commutative:* $p \vee q \Leftrightarrow q \vee p$ $\quad p \wedge q \Leftrightarrow q \wedge p$
- *Associative:* $(p \vee q) \vee r \Leftrightarrow p \vee (q \vee r)$

  $(p \wedge q) \wedge r \Leftrightarrow p \wedge (q \wedge r)$

More Equivalence Laws

- *Distributive*: $p \vee (q \wedge r) \Leftrightarrow (p \vee q) \wedge (p \vee r)$

  $p \wedge (q \vee r) \Leftrightarrow (p \wedge q) \vee (p \wedge r)$

- *De Morgan's*:

  $\neg(p \wedge q) \Leftrightarrow \neg p \vee \neg q$

  $\neg(p \vee q) \Leftrightarrow \neg p \wedge \neg q$

- *Trivial tautology/contradiction*:

  $p \vee \neg p \Leftrightarrow \mathbf{T}$ $\quad$ $p \wedge \neg p \Leftrightarrow \mathbf{F}$

Augustus De Morgan (1806-1871)

Defining Operators via Equivalences

- Using equivalences, we can *define* operators in terms of other operators.
- Exclusive or: $p \oplus q \Leftrightarrow (p \vee q) \wedge \neg(p \wedge q)$

  $p \oplus q \Leftrightarrow (p \wedge \neg q) \vee (q \wedge \neg p)$
- Implies: $p \rightarrow q \Leftrightarrow \neg p \vee q$
- Biconditional: $p \leftrightarrow q \Leftrightarrow (p \rightarrow q) \wedge (q \rightarrow p)$

  $p \leftrightarrow q \Leftrightarrow \neg(p \oplus q)$

An Example Problem

- Check using a symbolic derivation whether $(p \wedge \neg q) \rightarrow (p \oplus r) \Leftrightarrow \neg p \vee q \vee \neg r$.

- $(p \wedge \neg q) \rightarrow (p \oplus r)$
- $\Leftrightarrow \neg(p \wedge \neg q) \vee (p \oplus r)$ [Expand definition of $\rightarrow$]
- $\Leftrightarrow \neg(p \wedge \neg q) \vee ((p \vee r) \wedge \neg(p \wedge r))$ [Expand defn. of $\oplus$]
- $\Leftrightarrow (\neg p \vee q) \vee ((p \vee r) \wedge \neg(p \wedge r))$ [DeMorgan's Law]
- *cont.*

Example Continued...

- $\Leftrightarrow (\neg p \vee q) \vee ((p \vee r) \wedge \neg(p \wedge r))$
- $\Leftrightarrow (q \vee \neg p) \vee ((p \vee r) \wedge \neg(p \wedge r))$ [$\vee$ commutes]
- $\Leftrightarrow q \vee (\neg p \vee ((p \vee r) \wedge \neg(p \wedge r)))$ [$\vee$ associative]
- $\Leftrightarrow q \vee (((\neg p \vee (p \vee r)) \wedge (\neg p \vee \neg(p \wedge r)))$ [distrib. $\vee$ over $\wedge$]

$\Leftrightarrow q \vee (((\neg p \vee p) \vee r) \wedge (\neg p \vee \neg(p \wedge r)))$ [assoc.]

$\Leftrightarrow q \vee ((\mathbf{T} \vee r) \wedge (\neg p \vee \neg(p \wedge r)))$ [trivail taut.]

$\Leftrightarrow q \vee (\mathbf{T} \wedge (\neg p \vee \neg(p \wedge r)))$ [domination]

$\Leftrightarrow q \vee (\neg p \vee \neg(p \wedge r))$ [identity]

cont.

End of Long Example

$\Leftrightarrow q \vee (\neg p \vee \neg(p \wedge r))$

$\Leftrightarrow q \vee (\neg p \vee (\neg p \vee \neg r))$ [DeMorgan's]

$\Leftrightarrow q \vee ((\neg p \vee \neg p) \vee \neg r)$ [Assoc.]

- $\Leftrightarrow q \vee (\neg p \vee \neg r)$ [Idempotent]
- $\Leftrightarrow (q \vee \neg p) \vee \neg r$ [Assoc.]

$\Leftrightarrow \neg p \vee q \vee \neg r$ [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: $p, q, r, \ldots$
- Boolean operators: $\neg \ \wedge \ \vee \ \oplus \ \rightarrow \ \leftrightarrow$
- Compound propositions: $s :\equiv (p \wedge \neg q) \vee r$
- Equivalences: $p \wedge \neg q \Leftrightarrow \neg(p \rightarrow q)$
- Proving equivalences using:
  - Truth tables.
  - Symbolic derivations. $p \Leftrightarrow q \Leftrightarrow r \ldots$

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

Kurt Gödel
1906-1978

- 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*... denote objects/entities; uppercase variables *P*, *Q*, *R*... 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."