Discrete Structures

Credits

Michael P. Frank

Mehrdad Nojoumian

Husni Al-Muhtaseb

Course Outline (as per Rosen)

Ch	Topics	Included Sections	Wks
1	**Logic and Proofs**	**1.1 – 1.6, 1.8**	**4**
2	Sets, Functions, Sequences and Sums	2.1 – 2.4, 2.5 up to theorem 3 p.174	3
5	Induction and Recursion	5.1, 5.2, 5.3 (only tree examples. Generalized Induction is excluded)	2
6	Counting	6.1 – 6.4	2
7	Discrete Probability	7.1, 7.2 (up to page 449)	1.5
8	Advanced Counting Techniques	8.1 (no dynamic programming), 8.2 (up to page 504)	1.5

Course Objectives

- Upon completion of this course, the student should be able to:
  1. Formulate and derive propositional/ predicate logic expressions
  2. Apply proving methods
  3. Apply counting techniques to solve combinatorial problems

Think!

Ch	Topics	Included Sections	Wks
1	Logic and Proofs	1.1 – 1.6, 1.8	4
2	Sets, Functions, Sequences and Sums	2.1 – 2.4, 2.5 up to theorem 3 p.174	3
5	Induction and Recursion	5.1, 5.2, 5.3 (only tree examples. Generalized Induction is excluded)	2
6	Counting	6.1 – 6.4	2
7	Discrete Probability	7.1, 7.2 (up to page 449)	1.5
8	Advanced Counting Techniques	8.1 (no dynamic programming), 8.2 (up to page 504)	1.5

The Foundations: Logic and Proofs

1 The Foundations: Logic and Proofs

Foundations of Logic

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

- A formal 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

- Propositional logic (1.1-1.3):
  - Propositional logic (1.1)
  - Applications of Propositional logic (1.2)
  - Propositional Equivalences (1.3)

Propositional Logic (1.1)

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

Definition of a *Proposition*

Definition: A *proposition* (denoted *p*, *q*, *r*, …) is:

- a *statement* (*i.e.*, a declarative sentence)
  - *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
- Later, we will study *probability theory,* in which we assign *degrees of certainty* ("between" **T** and **F**) to propositions.
  - But for now: think True/False only!

Examples

- Propositions
- “It is raining.” (In a given situation.)
- “Riyadh is the capital of Saudi Arabia.”
- “1 + 2 = 3”
- الحر شديد (It is very hot)

But, the following are **NOT** propositions:

- “Who is there?” (interrogative, question)
- ركز لو سمحت (Concentrate, please) (imperative, command أمر)
- “La la la la la.” (meaningless interjection)
- “Just do it!” (imperative, command)
- “Yeah, I sorta dunno, whatever...” (vague)
- “1 + 2” (expression with a non-true/false value)

Operators / Connectives

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

- *Unary* operators take 1 operand (*e.g.*, −3)
- *binary* operators take 2 operands (*e.g.*, 3 × 4).
- *Propositional* or *Boolean* operators operate on propositions (or their truth values) instead of on numbers.

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	$\leftrightarrow$

The Negation Operator (*NOT*) "¬"

The unary *negation operator* "¬" (*NOT*) transforms a proposition into its logical *negation*.

E.g. If p = "I have brown hair."

then $\neg p$ = "I do **not** have brown hair."

The *truth table* for NOT:

p	$\neg p$
T	F
F	T

Operand column

Result column

T :≡ True; F :≡ False

":≡" means "is defined as"

The Conjunction Operator (AND) "$\wedge$"

The binary *conjunction operator* "$\wedge$" (*AND*) combines two propositions to form their logical *conjunction*.

E.g. If p = "I will have salad for lunch." and q = "I will have steak for dinner.", then $p \wedge q$ = "I will have salad for lunch **and** I will have steak for dinner."

AND

Remember: "$\wedge$" points up like an "A", and it means "AND"

Conjunction Truth Table

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

Operand columns

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

# rows = 2^2

# rows = $2^{\text{\# of operands}}$

- ¬ and ∧ operations together are sufficient to express *any* Boolean truth table (universal)!

Conjunction $p_1 \wedge p_2 \wedge \ldots \wedge p_n$ of n prop ll have 2^n rows in its truth table.

- # rows = $2^{\text{# of operands}}$

p_1	p_2
F	F
F	T
T	F
T	T

# rows = $2^2 = 4$

p_1	p_2	p_3
F	F	F
F	F	T
F	T	F
F	T	T
T	F	F
T	F	T
T	T	F
T	T	T

# rows = $2^3 = 8$

p_1	p_2	p_3	p_4
F	F	F	F
F	F	F	T
F	F	T	F
F	F	T	T
F	T	F	F
F	T	F	T
F	T	T	F
F	T	T	T
T	F	F	F
T	F	F	T
T	F	T	F
T	F	T	T
T	T	F	F
T	T	F	T
T	T	T	F
T	T	T	T

# rows = $2^4 = 16$

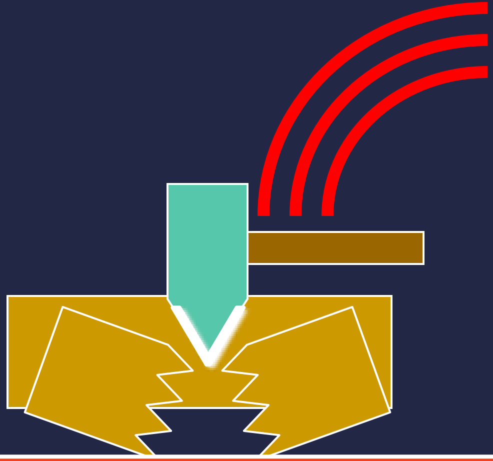
The Disjunction Operator (*OR*) "$\vee$"

The binary *disjunction operator* "$\vee$" (*OR*) combines two propositions to form their logical *disjunction*.

p = "My car has a bad engine."

q = "My car has a bad carburetor."

$p \vee q$ ="My car has a bad engine **or** my car has a bad carburetor."



After the downward-pointing "axe" of "$\vee$" splits the wood, you can take 1 piece OR the other, or both.

Meaning is like "and/or" in English.

Disjunction Truth Table

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

$\left.\begin{array}{l} \\ \\ \\ \end{array}\right\}$ Note difference from AND

- Note that $p \vee q$ means that p is true, or q is true, **or both** are true!
- So, this operation is also called *inclusive* OR, because it **includes** the possibility that both p and q are true.
- "¬" and "∨" together are also universal (together are sufficient to express *any* Boolean truth table).

Nested Propositional Expressions

- Use parentheses to *group sub-expressions*: "I just saw my old **f**riend, and he's **g**rown or I've **s**hrunk."
- $= f \land (g \lor s)$
  - $(f \land g) \lor s$ 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 المرشات came on last night,"
r = "The lawn العشب was wet this morning."

Translate each of the following into English:

$\neg p$ =	"It didnot rain last night."
$r \wedge \neg p$ =	"The lawn was wet this morning, and it didnot rain last night."
$\neg r \vee p \vee q$ =	"The lawn wasnot wet this morning, or it rained last night, or the sprinklers came on last night."

The *Exclusive Or* Operator (*XOR*) "$\oplus$"

The binary *exclusive-or operator* "$\oplus$" (*XOR*) combines two propositions to form their logical "exclusive or"

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!)"

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.
- "¬" and "⊕" together are **not** universal.

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

$\}$ Note difference from OR.

Natural Language is Ambiguous

Note that English "OR" can be ambiguous regarding the "both" case!

"Noor is a teacher or Noor is a writer." - $\vee$

"Noor is a man or Noor is a woman." - $\oplus$

p	q	p "OR" q
F	**F**	F
F	**T**	T
T	**F**	T
T	**T**	?

Need context to disambiguate the meaning!

For this course, assume **"or"** means inclusive

The *Implication* Operator (implies) "→"

The *implication* $\overbrace{p}^{\text{antecedent}} \rightarrow \overbrace{q}^{\text{consequent}}$ 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 study hard."
q = "You will get a good grade."

$p \rightarrow q$

= "If you study hard, then you will get a good grade."
(else, it could go either way)

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**!

p	***q***	***p→q***
F	**F**	**T**
F	**T**	**T**
T	**F**	**F**
T	**T**	**T**

The only False case!

- *E.g.* "$(1 = 0) \rightarrow$ cats can fly" is TRUE!

Examples of Implications

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

- "If this lecture ever ends, then the sun will rise tomorrow."

  True

- "If Tuesday is a day of the week, then I am a penguin."

  False

- "If 1 + 1 = 6, then Trump is president." (year 2018)

  True

- "If the moon is made of green cheese, then I am richer than Bill Gates."

  True

English phrases meaning $p \to q$

p	q	$p \to q$
F	F	T
F	T	T
T	F	F
T	T	T

- “*p* implies *q*”
- “if *p*, then *q*”
- “if *p*, *q*”
- “when *p*, *q*”
- “whenever *p*, *q*”
- “*p* only if *q*” *
- “*p* is sufficient for *q*”

- “*q* is implied by *p*”
- “*q* if *p*”
- “*q* when *p*”
- “*q* whenever *p*”
- “*q* follows from *p*”
- “*q* is necessary for *p*”

We will see some equivalent logic expressions later.

* “*p* only if *q*” says that *p* cannot be true when *q* is not true. That is, the statement is false if *p* is true, but *q* is false. When *p* is false, *q* may be either true or false, because the statement says nothing about the truth value of *q*.

Example

- If you get 100% on the final you will get an A+
- You get 100% on the final is sufficient to get an A+
- A sufficient condition to get an A+ is to get 100% on the final
- You get an A+ is necessary for you to get 100% on the final (but not sufficient)
- A necessary condition for you get 100% is you get an A+
- Imagine that you know your letter grade and you are trying to guess your grade in the final:
  - If you didnot get A+ then for sure you didn't get 100%
  - If you get A+ then you may or may not get 100%

Example

- If you show up on Monday you will get the job
- You show up on Monday is sufficient for you to get the job
- A sufficient condition for you to get the job is to show up on Monday
- You get the job is necessary for you have shown up on Monday.
- A necessary condition for you have shown up on Monday is you got the job.

You walk 8 miles is necessary to get to the top

- Which is equivalent?
- If you walk 8 miles then you get to the top
- If you got to the top then you have walked 8 miles

- The first statement is not equivalent. Walking 8 miles is necessary (but other things might be also necessary). suppose you walked 8 miles in the wrong direction ! But if you got to the top then you are sure that you must have walked 8 miles.

Converse, Inverse, Contrapositive

Some terminology, for an implication $p \to q$:

- The *converse* of $p \to q$ is: $q \to p$.
- The *inverse* of $p \to q$ is: $\neg p \to \neg q$.
- *The contrapositive of $p \to q$ is:* $\neg q \to \neg p$. *(if not q then not p)*
- One of these three has the *same meaning* (same truth table) as $p \to q$. Can you figure out which?

Contrapositive $\neg q \to \neg p$

A conditional statement is logically equivalent to its contrapositive.

How do we know?

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

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

The *biconditional* operator "$\leftrightarrow$"

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

p = "Ali wins the club election."

q = "Ali will be the president of the club for this year."

$p \leftrightarrow q$ = "If, and only if, Ali wins the club election, Ali will be the president of the club for this year."

Biconditional Truth Table

p	q	$p\leftrightarrow q$
F	F	T
F	T	F
T	F	F
T	T	T

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

Ex: This program is correct if, and only if, it produces the correct answer for all possible sets of input data.

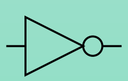
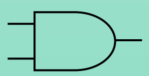
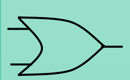
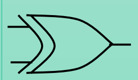
- This is equivalent to:
  - If this program is correct, then it produces the correct answer for all possible sets of input data and if it produces the correct answer for all possible sets of input data then this program is correct.
- This is equivalent to:
  - This program is correct is necessary and sufficient condition for it to produce correct answer for all possible sets of input data.
- This is equivalent to:
  - This program produces the correct answer for all possible sets of input data is necessary and sufficient condition for the program to be correct

Boolean Operations Summary

- We have seen 1 unary operator and 5 binary operators.

p	q	$\neg p$	$p \land q$	$p \lor q$	$p \oplus q$	$p \to q$	$p \leftrightarrow q$
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

Some Alternative Notations

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

Precedence of Logical Operators

Operator	Precedence
$\neg$	1
$\wedge$ $\vee$	2 3
$\rightarrow$ $\leftrightarrow$	4 5

$p \vee q \rightarrow \neg r$ is equivalent to $(p \vee q) \rightarrow \neg r$
If the intended meaning is $p \vee (q \rightarrow \neg r)$
then parentheses must be used.

Bits and Bit Operations

- A *bit* is a **b**inary (base 2) dig**it**: 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".

Bit Strings

- A *Bit string* of *length n* is an ordered sequence (series, tuple) of $n \geq 0$ bits.
- By convention, bit strings are (sometimes) written left to right:
  - *e.g.* the “first” bit of the bit string “1001101010” is 1.
  - Another common convention is that the rightmost bit is bit #0, the 2nd-rightmost is bit #1, etc.
- When a bit string represents a base-2 number, by convention, the first (leftmost) bit is the *most significant* bit. *Ex.* $1101_2=8+4+0+1=13$.

Counting in Binary

- We 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 1011 1111	Bit-wise OR
01 0001 0100	Bit-wise AND
10 1010 1011	Bit-wise XOR