Discrete Mathematics for Computer Science

Department of Computer Science

Lecturer: Nazeef Ul Haq

Reference Book: Discrete Mathematics and its applications BY Kenneth H. Rosen – 8th edition

Honor Code Of Class

- RESPECT YOURSELF!

- Maintain silence
- Use of mobile phones are not allowed
- Cheating/Plagiarism case will be dealt strictly
- Avoid cross talking
- Avoid copy paste submission of work

Course Introduction

- What we will cover in this course?
- Proofs and logics,
- Hashing function, Pseudorandom numbers
- Check Digits – UPCs, ISBNs, Airline ticket number
- Cryptography
- Mathematical Induction, Counting Techniques, Relations, Graphs and Trees

- Reference Book: Discrete Mathematics and its applications BY Kenneth H. Rosen – 8th edition

Grading Policy

Mid Exam	30%
Final Exam (Complete Syllabus)	40%
Quizzes (3 to 4)	10%
Assignments (2 to 3)	10%
Surprise Quiz/Class Participation	10%

- Minimum 75% attendance is MUST.

Lecture 1

Course Overview

Chapter 1. The Foundations

1.1 Propositional Logic

1-5

What is Mathematics, really?

- It's *not* just about numbers!
- Mathematics is *much* more than that:

> Mathematics is, most generally, the study of <u>any and all</u> *absolutely certain* truths about <u>any and all</u> *perfectly well-defined* concepts.

- These concepts can be *about* numbers, symbols, objects, images, sounds, *anything*!
- It is a way to interpret the world around you.

So, what's *this* class about?

What are "discrete structures" anyway?

- "***Discrete***" - Composed of distinct, separable parts. (Opposite of *continuous*.)

  discrete:*continuous* :: *digital*:*analog*

- "***Discrete Mathematics***" - concerns processes that consist of a sequence of individual steps.

Why Study Discrete Math?

- The basis of all of digital information processing is: <u>*Discrete manipulations of discrete structures represented in memory.*</u>
- It's the basic language and conceptual foundation for all of computer science.
- Discrete math concepts are also widely used throughout math, science, engineering, economics, biology, *etc.*, …
- A generally useful tool for rational thought!

Uses for Discrete Math in Computer Science

- Advanced algorithms & data structures
- Programming language compilers & interpreters
- Computer networks
- Operating systems
- Computer architecture
- Database management systems
- Cryptography
- Error correction codes
- Graphics & animation algorithms, game engines, *etc.*…
- *i.e.*, the whole field!

1.1 Propositional Logic

- Logic -- Logic is the study of the principles and methods that distinguishes between a valid and an invalid argument.
  - Focuses on the relationship among statements, not on the content of any particular statement.
  - Gives precise meaning to mathematical statements.
- ***Propositional Logic*** is the logic that deals with statements (propositions) and 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 simply:

- a *statement* (*i.e.*, a declarative sentence)
  - *with some definite meaning*,
    (not vague or ambiguous)
- having a *truth value* that's 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! (or in terms of **1** and **0**)

Examples of Propositions

- It is raining. (In a given situation)
- Beijing is the capital of China. (T)
- 2 + 2 = 5. (F)
- 1 + 2 = 3. (T)
- A fact-based declaration is a proposition, even if no one knows whether it is true
  - 11213 is prime.
  - There exists an odd perfect number.

Proposition

- Rule -- If the sentence is preceded by other sentences that make the pronoun or variable reference clear, then the sentence is a statement.

Example:

$x = 1$ and $x > 2$

$x > 2$ is a statement with truth-value FALSE.

COMPOUND STATEMENT:

Simple statements could be used to build a compound statement.

EXAMPLES: **LOGICAL CONNECTIVES**

1. "3 + 2 = 5" **and** "Lahore is a city in Pakistan"
2. "The grass is green" or " It is hot today"
3. "Discrete Mathematics is **not** difficult to me"

AND, OR, NOT are called LOGICAL CONNECTIVES.

Examples of Non-Propositions

The following are **NOT** propositions:

- Who's there? (interrogative, question)
- Just do it! (imperative, command)
- La la la la la. (meaningless interjection)
- Yeah, I sorta dunno, whatever... (vague)
- 1 + 2 (expression with a non-true/false value)
- x + 2 = 5 (declaration about semantic tokens of non-constant value)

Truth Tables

- An *operator* or *connective* combines one or more *operand* expressions into a larger expression. (*e.g.*, "+" in numeric expressions.)
- ***Unary*** operators take *one* operand (*e.g.,* −3);
  Binary operators take *two* operands (*e.g.* 3×4).
- ***Propositional*** or ***Boolean operators*** operate on propositions (or their truth values) instead of on numbers.
- The ***Boolean domain*** is the set {T, F}. Either of its elements is called a ***Boolean value***.
  An n-tuple $(p_1,\ldots,p_n)$ of Boolean values is called a ***Boolean*** ***n-tuple***.
- An n-operand truth table is a table that assigns a Boolean value to the set of all Boolean n-tuples.

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

- The unary ***negation*** *operator* "¬" (*NOT*) transforms a proposition into its logical *negation*.
- *E.g.* If p = "I have brown hair."
  then $\neg p$ = "It is not the case that I have brown hair" or "I do **not** have brown hair."
- The *truth table* for NOT:

p	$\neg p$
T	F
F	T

Operand column

Result column

The Conjunction Operator

- The binary ***conjunction*** *operator* "$\land$" (*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, $p \land q$ = "I will have salad for lunch **and** I will have steak for dinner."

Conjunction Truth Table

Operand columns

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

- 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

The Disjunction Operator

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

- *E.g.* If p = "My car has a bad engine." and
  q = "My car has a bad carburetor."

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

Meaning is like "and/or" in informal English.

Disjunction Truth Table

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

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.

The Exclusive-Or Operator

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

- *E.g.* If p = "I will earn an A in this course." and

  q = "I will drop this course.", then

  $p \oplus q$ = "I will **either** earn an A in this course,

  or I will drop it (**but not both**!)"

Exclusive-Or Truth Table

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

Note difference from OR.

- 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.

Natural Language is Ambiguous

- Note that the English "or" can be ambiguous regarding the "both" case!
- "Pat is a singer or Pat is a writer." - $\vee$
- "Pat is a man or Pat is a woman." - $\oplus$

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

- Need context to disambiguate the meaning!
- For this class, assume "or" means inclusive ($\vee$).

The Implication Operator

- The conditional statement (aka ***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 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)
  - p: *hypothesis* or *antecedent* or *premise*
  - q: *conclusion* or *consequence*

Implication Truth Table

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

The only False case!

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

Examples of Implications

- "If this lecture ever ends, then the sun will rise tomorrow." *True or False?* $(T \rightarrow T)$
- "If 1+1=6, then Joe Biden is president." *True or False?* $(F \rightarrow T)$
- "If the moon is made of green cheese, then I am richer than Bill Gates." *True or False?* $(F \rightarrow F)$
- "If Tuesday is a day of the week, then I am a penguin." *True or False?* $(T \rightarrow F)$

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***: $\neg q \rightarrow \neg p$.

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

- One of these three has the *same meaning* (same truth table) as $p \rightarrow q$. Can you figure out which?

Contrapositive

Examples

- p: Today is Easter
  q: Tomorrow is Monday

- $p \to q$:
  If today is Easter then tomorrow is Monday.
- ***Converse***: $q \to p$
  If tomorrow is Monday then today is Easter.
- ***Inverse***: $\neg p \to \neg q$
  If today is not Easter then tomorrow is not Monday.
- ***Contrapositive***: $\neg q \to \neg p$
  If tomorrow is not Monday then today is not Easter.

The Biconditional Operator

- The ***biconditional*** statement $p \leftrightarrow q$ states that p ***if and only if*** (*iff*) q.
- p = "It is below freezing."

  q = "It is snowing."

  $p \leftrightarrow q$ = "It is below freezing if and only if it is snowing."

  or

  = "That it is below freezing is necessary and sufficient for it to be snowing"

Biconditional Truth Table

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

- p is necessary and sufficient for q
- If p then q, and conversely
- p iff q

- $p \leftrightarrow q$ is equivalent to $(p \rightarrow q) \wedge (q \rightarrow p)$.
- $p \leftrightarrow q$ means that p and q have the **same** truth value.
- $p \leftrightarrow q$ does **not** imply that p and q are true.
- Note this truth table is the exact **opposite** of $\oplus$'s! Thus, $p \leftrightarrow q$ means $\neg(p \oplus q)$.

Boolean Operations Summary

- Conjunction: $p \wedge q$, (read p and q), "discrete math is a required course and I am a computer science major".
- Disjunction: , $p \vee q$, (read p or q), "discrete math is a required course or I am a computer science major".
- Exclusive or: $p \oplus q$, "discrete math is a required course or I am a computer science major but not both".
- Implication: $p \rightarrow q$, "if discrete math is a required course then I am a computer science major".
- Biconditional: $p \leftrightarrow q$, "discrete math is a required course if and only if I am a computer science major".

Boolean Operations Summary

- We have seen 1 unary operator and 5 binary operators. What are they? Their truth tables are below.

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

- For an implication $p \rightarrow q$
- Its ***converse*** is: $q \rightarrow p$
- Its ***inverse*** is: $\neg p \rightarrow \neg q$
- Its ***contrapositive***: $\neg q \rightarrow \neg p$

Compound Propositions

- A ***propositional variable*** is a variable such as p, q, r (possibly subscripted, e.g. p_j) over the Boolean domain.
- An ***atomic proposition*** is either Boolean constant or a propositional variable: e.g. T, F, p
- A ***compound proposition*** is derived from atomic propositions by application of propositional operators: e.g. $\neg p$, $p \vee q$, $(p \vee \neg q) \rightarrow q$
- Precedence of logical operators: $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$
- Precedence also can be indicated by parentheses.
  - e.g. $\neg p \wedge q$ means $(\neg p) \wedge q$, not $\neg(p \wedge q)$

An Exercise

- Any compound proposition can be evaluated by a truth table
- $(p \vee \neg q) \rightarrow q$

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

Translating English Sentence

- 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:

  $\neg p$ = "It didn't rain last night."

  $r \wedge \neg p$ = "The lawn was wet this morning, and it didn't rain last night."

  $\neg r \vee p \vee q$ = "The lawn wasn't wet this morning, or it rained last night, or the sprinklers came on last night."

Another Example

- Find the converse of the following statement.
  - "Raining tomorrow is a sufficient condition for my not going to town."
- **Step 1**: Assign propositional variables to component propositions.
  - p: It will rain tomorrow
  - q: I will not go to town
- **Step 2**: Symbolize the assertion: $p \rightarrow q$
- **Step 3**: Symbolize the converse: $q \rightarrow p$
- **Step 4**: Convert the symbols back into words.
  - "If I don't go to town then it will rain tomorrow" or
  - "Raining tomorrow is a *necessary condition* for my not going to town."

Logic 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".
- A ***bit string of length n*** is an ordered sequence 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.
  - What is the length of the above bit string?

Bitwise Operations

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

- Example:

  01 1011 0110

  <u>11 0001 1101</u>

  11 1011 1111 Bit-wise OR

  01 0001 0100 Bit-wise AND

  10 1010 1011 Bit-wise XOR

End of 1.1

You have learned about:

- Propositions: what they are
- Propositional logic operators'
  - symbolic notations, truth tables, English equivalents, logical meaning
- Atomic vs. compound propositions
- Bits, bit strings, and bit operations

- Next section:
  - Propositional equivalences
  - Equivalence laws
  - Proving propositional equivalences