Discrete Mathematics, KOM1062 Lecture #1

Instructor: Dr. Yavuz Eren

Spring 2024

1

Course Content

- Introduction, Propositional Logic
- Propositional Equivalences, Predicates
- Quantifiers, Nested Quantifiers
- Rules of Inference and Examples and introduction to proofs
- Sets, Sets operations and Functions
- Sequences and Summations, Induction
- Relations and Their Properties, Representing Relations, Closures of Relations, Equivalence Relations and Partial Ordering
- Graphs and Graph Models, Graph Terminology and Types of Graphs, Representing Graphs
- Connectivity, Euler and Hamilton Paths
- Shortest Path Problems, Plannar Graphs, Graph Coloring
- Introduction to Trees, Application of Trees, Tree Traversal
- Spanning Trees, Minimum Spanning Trees

***Lecture Book*: Discrete Mathematics, Seventh Edt., Kenneth H. Rosen, 2007, McGraw Books**
Discrete Mathematics and Applications, Susanna S. Epp, Brooks, 4th Edt., 2011.

2

Grading Policy

- Midterm 1- %30
- Midterm 2- %30
- Final- %40

Office Hours

- Monday – 15.00-17.00- A217
- Wendesday – 11.00-13.00- A217

3

MOTIVATION

- The area of logic that deals with propositions is called the **propositional calculus or propositional logic. It was first developed systematically by the Greek philosopher Aristotle more** than 2300 years ago.
- The current methods were discussed by the English mathematician George Boole in 1854 in his book *The Laws of Thought*
- A discrete mathematics course has more than one purpose. Students should learn a particular set of mathematical facts and how to apply them; more importantly, such a course should teach students how to think logically and mathematically.
- To achieve these goals, this lecture stresses mathematical reasoning and the different ways problems are solved. Five important themes which are mathematical reasoning, combinatorial analysis, discrete structures, algorithmic thinking, applications and modeling.
- *Mathematical Reasoning- Students must understand mathematical reasoning in order to* read, comprehend, and construct mathematical arguments.
- *Combinatorial Analysis- An important problem-solving skill is the ability to count or enumerate* objects. The stress is on performing combinatorial analysis to solve counting problems and analyze algorithms, not on applying formula.

4

MOTIVATION

- *Discrete Structures: A course in discrete mathematics should teach students how to work* with discrete structures, which are the abstract mathematical structures used to represent discrete objects and relationships between these objects. These discrete structures include sets, permutations, relations, graphs, trees, and finite-state machines.
- *Algorithmic Thinking: Certain classes of problems are solved by the specification of an* algorithm. After an algorithm has been described, a computer program can be constructed to implement it.
- *Applications and Modeling: Discrete mathematics has applications to almost every conceivable* area of study. There are many applications to computer science and data networking, as well as applications to such diverse areas as chemistry, biology, linguistics, geography, business, and the Internet. These applications are natural and important uses of discrete mathematics and are not contrived. Modeling with discrete mathematics is an extremely important problem-solving skill, which students have the opportunity to develop by constructing their own models in some of the exercises.

5

Propositional Logic and Proofs

- The rules of logic specify the meaning of mathematical statements. For instance, these rules help us understand and reason with statements such as

"There exists an integer that is not the sum of two squares" and

"For every positive integer *n, the sum of the positive integers* not exceeding *n is n(n + 1)/2."*

- To understand mathematics, we must understand what makes up a correct mathematical argument, that is, a proof. Once we prove a mathematical statement is true, we call it as theorem. A collection of theorems on a topic organize what we know about this topic. To learn a mathematical topic, a person needs to actively construct mathematical arguments on this topic, and not just read exposition. Moreover, knowing the proof of a theorem often makes it possible to modify the result to fit new situations.
- In fact, proofs are used to verify that computer programs produce the correct output for all possible input values, to show that algorithms always produce the correct result, to establish the security of a system, and to create artificial intelligence.
- In this part, mathematical arguments and strategies will be taught for constructing proofs of theorems and conjectures.

6

Propositional Logic

- The rules of logic give precise meaning to mathematical statements. These rules are used to distinguish between valid and invalid mathematical arguments.
- Besides the importance of logic in understanding mathematical reasoning, logic has numerous applications to computer science. These rules are used in the design of
  - computer circuits,
  - the construction of computer programs,
  - the verification of the correctness of programs, and
  - in many other ways.

Proposition: A proposition is a declarative sentence (that is, a sentence that declares a fact) that is either true or false, but not both.

Ex 1: The following declarative sentences are propositions.

1. Toronto is the capital of Canada.
2. 1 + 1 = 2.
3. 2 + 2 = 3.

Proposition 1 is true; 2 and 3 are false.

- Some sentences that are not propositions are given in Example 2.

Ex. 2: Consider the following sentences.

1. What time is it?
2. Read this carefully.
3. x+1=2.
4. x+y=z.

7

Sentences 1 and 2 are not propositions because they are not declarative sentences. Sentences 3 and 4 are not propositions because they are neither true nor false. Note that, 3 and 4 can be turned into a proposition if we assign values to the variables.

- The area of logic that deals with propositions is called the **propositional calculus or propositional logic.** New proposition can be generated by the proposition that have been already defined.
- **Propositional variables (or statement variables)** represent propositions by using the letters {p,q,r,s, …}.
- The **truth value of a** proposition is true, denoted by T, if it is a true proposition, and the truth value of a proposition is false, denoted by F, if it is a false proposition.

Definition: Let *p be a proposition. The negation of p(not p), denoted by* $\neg p$ *(or* $\overline{p}$*), is the statement of "It is not the case that p."*

Truth value of the $\neg p$*, is the oppsite of the truth value of p.*

Ex 3: Find the negation of the following proposition and express this in simple English.

"Michael's PC runs Linux"

Solution:

"It is not the case that Michael's PC runs Linux."

This negation can be more simply expressed as

"Michael's PC does not run Linux."

8

Ex. 4: Find the negation of the following proposition and express this in simple English.

"Vandana's smartphone has at least 32GB of memory"

Solution:

"It is not the case that Vandana's smartphone has at least 32GB of memory."

This negation can also be expressed as

"Vandana's smartphone does not have at least 32GB of memory"

or even more simply as

"Vandana's smartphone has less than 32GB of memory."

- Following table displays the **truth table for the negation of a proposition *p. This table has a row*** for each of the two possible truth values of a proposition *p. Each row shows the truth value of¬p corresponding to the truth value of p for this row.* The negation operator constructs a new proposition froma single existing proposition.

Truth table for negation of a proposition

p	¬p
T	F
F	T

9

Definition: Let *p and q be propositions. The conjunction of p and q, denoted by p ∧ q, is the proposition "p and q." The conjunction p ∧ q is true when both p and q are true and is false otherwise.*

- Following table displays the truth table of *p ∧ q. This table has a row for each of the four possible* combinations of truth values of *p and q. The four rows correspond to the pairs of truth values* TT, TF, FT, and FF, where the first truth value in the pair is the truth value of *p and the second* truth value is the truth value of *q.*

Truth table for conjunction with "and " operator of two proposition

p	q	*p ∧ q*
T	T	T
T	F	F
F	T	F
F	F	F

10

Ex 5: Find the conjunction of the propositions *p and q where p is the proposition "Rebecca's PC has* more than 16 GB free hard disk space" and *q is the proposition "The processor in Rebecca's* PC runs faster than 1 GHz."

Solution:

"Rebecca's PC has more than 16 GB free hard disk space, and the processor in Rebecca's PC runs faster than 1GHz." or

"Rebecca's PC has more than 16 GB free hard disk space, and its processor runs faster than 1 GHz."

Definition: Let *p and q be propositions. The disjunction of p and q, denoted by p ∨ q, is the proposition "p or q." The disjunction p ∨ q is false when both p and q are false and is true otherwise.*

Truth table for conjunction with "or " operator of two proposition

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

11

- The use of the connective *or in a disjunction corresponds to one of the two ways the word or is used in English, namely, as an* ***inclusive or. A disjunction is true when at least one of the*** two propositions is true. For instance, the inclusive or is being used in the statement

  "Students who have taken calculus or computer science can take this class."

- Here, we mean that students who have taken both calculus and computer science can take the class, as well as the students who have taken only one of the two subjects. On the other hand, we are using the **exclusive or when we say**

  "Students who have taken calculus or computer science, but not both, can enroll in this class."

- Here, we mean that students who have taken both calculus and a computer science course can not take the class. Only those who have taken exactly one of the two courses can take the class.

Definition: Let *p and q be propositions. The exclusive or of p and q, denoted by p ⊕ q, is the proposition* that is true when exactly one of *p and q is true and is false otherwise.*

Truth table for conjunction with "exclusive or " operator of two proposition

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

12

Definition: Let *p and q be propositions. The conditional statement p → q is the proposition "if p, then q." The conditional statement p → q is false when p is true and q is false, and true otherwise.* In the conditional statement *p → q, p is called the hypothesis (or antecedent or premise)* and *q is called the conclusion (or consequence).*

- The statement *p → q is called a conditional statement because p → q asserts that q is true* on the condition that *p holds. A conditional statement is also called an* ***implication.***
- The statement *p → q is true when both p and q are true and when p is false (no matter what truth* value *q has).*

Truth table for conjunction with "conditional " operator of two proposition

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

- Note that, a variety of terminology is used to express *p → q as follows:*

"if p, then q"
"if p, q"
"p is sufficient for q"
"q if p"
"q when p"
"a necessary condition for p is q"
"q unless $\neg p$"
"p implies q"
"p only if q"
"a sufficient condition for q is p"
"q whenever p"
"q is necessary for p"
"q follows from p"

13

- Following example explain the conditional statement precisely:

"If the politician is elected, voters would expect this politician to lower taxes. Furthermore, if the politician is not elected, then voters will not have any expectation that this person will lower taxes, although the person may have sufficient influence to cause those in power to lower taxes. It is only when the politician is elected but does not lower taxes that voters can say that the politician has broken the campaign pledge. This last scenario corresponds to the case when *p* is true but *q is false in p → q.*"

14

Ex 7: Let *p be the statement "Maria learns discrete mathematics" and q the statement "Maria will* find a good job." Express the statement *p → q as a statement in English.*

Solution:

"If Maria learns discrete mathematics, then she will find a good job."

"Maria will find a good job when she learns discrete mathematics."

"For Maria to get a good job, it is sufficient for her to learn discrete mathematics."

- The if-then construction used in many programming languages is different from that used in logic. Most programming languages contain statements such as **if *p then S, where p is a*** proposition and *S is a program segment (one or more statements to be executed). When execution* of a program encounters such a statement, *S is executed if p is true, but S is not executed if p* is false, as illustrated in the following example.

Ex 8: What is the value of the variable *x after the statement "***if 2 + 2 = 4 then *x := x + 1",*** if *x = 0 before this statement is encountered ? (The symbol := stands for assignment. The* statement *x := x + 1 means the assignment of the value of x + 1 to x.)*

Solution:

Because 2 + 2 = 4 is true, the assignment statement x := x + 1 is executed. Hence, x has the value 0 + 1 = 1 after this statement is encountered.

15

CONVERSE, CONTRAPOSITIVE, AND INVERSE

- We can form some new conditional statements starting with a conditional statement *p → q. In particular, there are three related* conditional statements that occur so often that they have special names. The proposition *q → p* is called the **converse of *p → q. The contra-positive of p → q is the proposition ¬q → ¬p.*** The proposition *¬p → ¬q is called the* ***inverse of p → q. We will see that of these three*** conditional statements formed from *p → q, only the contra-positive always has the same truth* value as *p → q.*
- When two compound propositions always have the same truth value we call them **equivalent,** so that a conditional statement and its contra-positive are equivalent. The converse and the inverse of a conditional statement are also equivalent.

Ex. 9: What are the contra-positive, the converse, and the inverse of the conditional statement

"The home team wins whenever it is raining?"

Solution:

Because "q whenever p" is one of the ways to express the conditional statement p → q, the original statement can be rewritten as "If it is raining, then the home team wins."

Consequently, the contrapositive of this conditional statement is "If the home team does not win, then it is not raining."

The converse is

"If the home team wins, then it is raining."

The inverse is

"If it is not raining, then the home team does not win."

Only the contrapositive is equivalent to the original statement.

16

Definition: Let *p and q be propositions. The biconditional statement p ↔ q is the proposition "p if* and only if *q." The biconditional statement p ↔ q is true when p and q have the same truth* values, and is false otherwise. Biconditional statements are also called *bi-implications.*

- ✓ Note that the statement *p ↔ q is true when both* the conditional statements *p → q and q → p are true and is false otherwise. That is why we use* the words "if and only if" to express this logical connective and why it is symbolically written by combining the symbols→ and ←. There are some other common ways to express *p ↔ q:* "*p is necessary and sufficient for q",* "if *p then q, and conversely" and "p iff q."*

- ✓ The last way of expressing the biconditional statement *p ↔ q uses the abbreviation "iff" for* "if and only if." Note that *p ↔ q has exactly the same truth value as (p → q) ∧ (q → p).*

Truth table for conjunction with "biconditional " operator of two proposition

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

17

Ex. 10: Let *p be the statement "You can take the flight," and let q be the statement "You buy a ticket."* Then *p ↔ q is the statement* "You can take the flight if and only if you buy a ticket."

Solution:

This statement is true if p and q are either both true or both false, that is, if you buy a ticket and can take the flight or if you do not buy a ticket and you cannot take the flight.

It is false when p and q have opposite truth values, that is, when you do not buy a ticket, but you can take the flight (such as when you get a free trip) and when you buy a ticket but you cannot take the flight (such as when the airline bumps you).

Truth Tables of Compound Propositions

- There are four important logical connectives (and their negations)which are **"conjunctions", "disjunctions", "conditional statements",** and **"biconditional statements".**
- These connectives are used to build up complicated compound propositions involving any number of propositional variables.

Ex. 11: Construct the truth table of the compound proposition $(p \vee \neg q) \rightarrow (p \wedge q)$.

Solution:

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

18

Precedence of Logical Operators

- Compound propositions can be constructed by using the negation operator and the logical operators
- Parentheses are used to specify the order in which logical operators in a compound proposition are to be applied. For instance, *(p ∨ q) ∧ (¬r) is the conjunction* of *p ∨ q and ¬r.*
- *However, to reduce the number of parentheses, we specify that the negation* operator is applied before all other logical operators. This means that *¬p ∧ q is the conjunction* of *¬p and q, namely, (¬ p) ∧ q, not the negation of the conjunction of p and q, namely ¬(p ∧ q).*
- Conjunction operator takes precedence over the disjunction operator, so that *p∧ q ∨ r means (p ∧ q) ∨ r rather than p ∧ (q ∨ r). (But, parantheses are used to make clear the statements.)*
- Conditional and biconditional operators → and ↔ have lower precedence than the conjunction and disjunction operators, ∧ and ∨. *p ∨ q → r is the same as (p ∨ q) → r.)*
- Conditional operator has precedence over the biconditional operator.

Precedence of operators	
Opr.	*Rank*
¬	1
∧	2
∨	3
→	4
↔	5

Logic and Bit Operations

• A bit is a symbol with two possible values, namely, 0 (zero) and 1 (one). This meaning of the word bit comes from *binary digit, because zeros and* ones are the digits used in binary representations of numbers(represents T (true), 0 represents F (false)).

• A variable is called a Boolean variable(represented by bits) if its value is either true or false.

Truth Value/ Bit	
T	1
F	0

19

- Computer bit operations correspond to the logical connectives. By replacing true by a one and false by a zero in the truth tables for the operators ∧, ∨, and ⊕.
- The notation of *OR, AND, and XOR for the operators ∨, ∧, and ⊕ can be used, respectively, in various programming languages.*

Bit operators of ∧, ∨, and ⊕				
a	b	a∨b	a∧b	a⊕b
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0

Definition 7: A *bit string is a sequence of zero or more bits. The length of this string is the number of bits* in the string.

Ex. 12: 101010011 is a bit string of length nine.

•Bitwise *OR, bitwise AND, and* bitwise *XOR of two strings of the same length to be the strings that have as their bits the OR, AND, and XOR of the corresponding bits in the two strings, respectively.*

Ex. 13: Find the bitwise "or", bitwise "and", and bitwise "xor" of the bit string 0110110110 and 1100011101.

Solution:

$$\begin{array}{l} 01\ 1011\ 0110 \\ \underline{11\ 0001\ 1101} \\ 11\ 1011\ 1111 \quad \text{bitwise } OR \\ 01\ 0001\ 0100 \quad \text{bitwise } AND \\ 10\ 1010\ 1011 \quad \text{bitwise } XOR \end{array}$$

20

20

Application of Propositional Logic

- Logic has many important applications to mathematics, computer science, and numerous other disciplines.
- Some examples of application areas:
  - Designing of computer circuits
  - For specifications of software and hardware
  - To construct computer program
  - To verify the correctness of programs
- Statements in mathematics and the sciences and in natural language often are imprecise or ambiguous.
- To make such statements precise, they can be translated into the language of logic.
- Translating sentences into compound statements removes the ambiguity (as making a set of reasonable assumptions based on the intended meaning of the sentence).

21

Application of Propositional Logic

Ex. 1: How can this English sentence be translated into a logical expression?

"You can access the Internet from campus only if you are a computer science major or you are not a freshman."

Solution: We let a, c, and f represent "You can access the Internet from campus," "You are a computer science major," and "You are a freshman," respectively. Noting that "only if" is one way a conditional statement can be expressed, this sentence can be represented as:

$$a \rightarrow (c \vee \neg f)$$

Ex. 2: How can this English sentence be translated into a logical expression?

"You cannot ride the roller coaster if you are under 4 feet tall unless you are older than 16 years old."

Solution: Let q, r, and s represent "You can ride the roller coaster," "You are under 4 feet tall," and "You are older than 16 years old," respectively.

$$(r \wedge \neg s) \rightarrow \neg q.$$

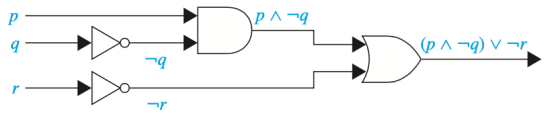
22

Logic Gates: A logic circuit (or digital circuit) receives input signals *p1, p2, . . . , pn, each a bit [either* 0 (off) or 1 (on)], and produces output signals *s1, s2, . . . , sn, each a bit.*

Complicated digital circuits can be constructed from three basic circuits, called gates, shown in Figure 1.

- The inverter, or NOT gate, takes an input bit *p, and produces as output* $\neg p$.
- *The* OR gate takes two input signals *p and q, each a bit, and produces as output the signal* $p \vee q$.
- The AND gate takes two input signals *p and q, each a bit, and produces as output the* signal $p \wedge q$.

Ex.3 : Determine the output for the combinatorial circuit for the following Figure.
Solution:



Ex. 4: Build a digital circuit that produces the output $(p \vee \neg r) \wedge (\neg p \vee (q \vee \neg r))$ *when given input* bits *p, q, and r.*
Solution: