Welcome to

CS220/MATH 320 – Applied Discrete Mathematics

Summer 2020

Instructor: Ramin Dehghanpoor

SI: Bella Baidak

TA: Diana Alisevich

I got help from and used Dr. Pomplun's and Dr. Haspel's slides

UMASS BOSTON

Instructor – Ramin Dehghanpoor

- Office: M-201-35
- Office Hours: Mondays 3:00-5:00 PM
- Wednesdays 3:00-5:00 PM
- E-Mail: ramin.dehghanpoor001@umb.edu
- Website: http://www.cs.umb.edu/~ramin/cs220/

Textbook and Course Website

- Discrete Mathematics and Its Applications

Kenneth H. Rosen

- We will study around 400 pages of this book.
- Important! Course homepage:
- *http://www.cs.umb.edu/~ramin/cs220/*
- Contains all kinds of course information and also my slides

Your Evaluation

- Homework assignments(4-5) 30%
- Midterm exam 30%
- Final exam 40%

Grading

For the assignments, exams and your course grade, the following scheme will be used to convert percentages into letter grades:

- $90\% < P$: A
- $85\% < P \leq 90\%$: A-
- $80\% < P \leq 85\%$: B+
- $75\% < P \leq 80\%$: B
- $70\% < P \leq 75\%$: B-
- $65\% < P \leq 70\%$: C+
- $60\% < P \leq 65\%$: C
- $55\% < P \leq 60\%$: C-
- $50\% < P \leq 55\%$: D+
- $45\% < P \leq 50\%$: D
- $40\% < P \leq 45\%$: D-
- $P \leq 40\%$: F

Course Requirements

- Enroll in the course on Gradescope with Entry code **9GBK2Z**
- Enroll in the course on Piazza with Access code **9GBK2Z** or the link: piazza.com/umb/summer2020/cs220math320
- Your final grade should be at least 40% to pass.
- You also have to pass the final exam.
- Your TA will grade homework assignments
- Your SI will work with you on a regular basis to answer your questions and do some practice.
- Always ask your questions on Piazza please.

Academic Dishonesty

- You are allowed to discuss problems regarding your homework with other students in the class.
- However, you have to do the actual work (computing values, writing algorithms, drawing graphs, etc.) by yourself.
- You cannot copy anything from other sources (Wikipedia, other students' work, etc.)
- The first violation will result in zero points for the entire homework or exam (and official notification).
- The second violation will result in failing the course.

Complaints about Grading

- If you think that the grading of your **homework** was unfair, please talk to the TA (Diana).
- If you are still unhappy afterwards, please talk to me.
- If you think that the grading of your **exam** was unfair, please talk to me

Why Care about Discrete Math?

- Digital computers are based on discrete “atoms” (bits).
- Therefore, both a computer’s
  - **structure (circuits) and**
  - **operations (execution of algorithms)**

  can be described by discrete math.
- Most importantly, software engineers need to have a solid background in discrete mathematics in order to develop appropriate algorithms for given problems.

Syllabus

- A discrete mathematics course has more than one purpose.
- We will learn about five important themes:

❑ Mathematical reasoning, Combinatorial analysis, Discrete structures, Algorithmic thinking, And Applications and Modeling

- Let us just take a look at the syllabus on the course homepage…

▸ *http://www.cs.umb.edu/~ramin/cs220/sillabus.pdf*

We will cover these parts of the book (8th edition):

1.1
1.3.1-1.3.4
1.4.1-1.4.10

Logic

- Crucial for mathematical reasoning
- Used for designing electronic circuitry
- Logic is a system based on propositions.
- A proposition is a declarative statement (that declares a fact) that is either true or false (not both).
- We say that the truth value of a proposition is either true (T) or false (F).
- Corresponds to 1 and 0 in digital circuits
- We use letters (p,q,r,…) to denote propositional variables

The Statement/Proposition Game

- "Elephants are bigger than mice."

Is this a statement?	yes
Is this a proposition?	yes
What is the truth value of the proposition?	true

The Statement/Proposition Game

- "520 < 111"

Is this a statement? yes

Is this a proposition? yes

What is the truth value of the proposition? false

The Statement/Proposition Game

- "y > 5"

Is this a statement? yes

Is this a proposition? no

Its truth value depends on the value of y, but this value is not specified.
We call this type of statement a propositional function or open sentence.

The Statement/Proposition Game

- "Today is January 23 and 99 < 5."

Is this a statement? yes

Is this a proposition? yes

What is the truth value of the proposition? false

The Statement/Proposition Game

- "Please do not fall asleep."

Is this a statement? no

It's a request.

Is this a proposition? no

Only statements can be propositions.

The Statement/Proposition Game

- "If elephants were red,
- they could hide in cherry trees."

Is this a statement? yes

Is this a proposition? yes

What is the truth value of the proposition? probably false

The Statement/Proposition Game

- "x < y if and only if y > x."

Is this a statement? yes

Is this a proposition? yes

… because its truth value does not depend on specific values of x and y.

What is the truth value of the proposition? true

Combining Propositions

- As we have seen in the previous examples, one or more propositions can be combined to form a single compound proposition.

- Propositions that cannot be expressed in terms of simpler propositions, are atomic.

- When two compound propositions always have the same truth values, regardless of the truth values of its propositional variables, we call them equivalent

Combining Propositions

- We formalize this by denoting propositions with letters such as p, q, r, s, and introducing several logical operators.

- The area of logic that deals with propositions is called the propositional calculus or propositional logic.

Logical Operators (Connectives)

- We will examine the following logical operators:

- Negation (NOT)
- Conjunction (AND)
- Disjunction (OR)
- Exclusive or (XOR)
- Implication (if – then)
- Biconditional (if and only if)

- Truth tables can be used to show how these operators can combine propositions to compound propositions.

Negation (NOT)

- Unary Operator, Symbol: ¬

P	¬P
true	false
false	true

"It is not the case that P"

Conjunction (AND)

- Binary Operator, Symbol: $\wedge$

P	Q	P∧Q
true	true	true
true	false	false
false	true	false
false	false	false

Disjunction (OR)

- Binary Operator, Symbol: ∨

P	Q	P∨Q
true	true	true
true	false	true
false	true	true
false	false	false

Exclusive Or (XOR)

- Binary Operator, Symbol: $\oplus$

P	Q	P⊕Q
true	true	false
true	false	true
false	true	true
false	false	false

Implication (if - then)

- Binary Operator, Symbol: $\rightarrow$

P	Q	P→Q
true	true	true
true	false	false
false	true	true
false	false	true

Implication (if - then)

- An implication is only false if the left side (called hypothesis) is True and the right side (called conclusion) is false.
- It is therefore equivalent to : $\neg P \vee Q$
- Example from class: If today is Monday, you have a class.
- The only way the entire statement can be false is if today is Monday and you don't have a class (P is true, Q is false).
- Notice that if P is false, the entire statement is true regardless of the value of Q: If today is not Monday then you either have a class or not.

P → Q terminologies

- If P, then Q
- If P, Q
- P is sufficient for Q
- Q if P
- P implies Q
- Q whenever P
- Q follows from P
- P only if Q
- Q unless ¬P

Converse, Contrapositive, and Inverse of P → Q

- Converse: $Q \rightarrow P$
- Contrapositive: $\neg Q \rightarrow \neg P$
  - Always has the same truth value as $P \rightarrow Q$
- Inverse: $\neg P \rightarrow \neg Q$

Biconditional (if and only if sometimes written as iff)

- Binary Operator, Symbol: $\leftrightarrow$

P	Q	P↔Q
true	true	true
true	false	false
false	true	false
false	false	true

- True if both P and Q have the same truth value
- It is equivalent to $(P \rightarrow Q) \wedge (Q \rightarrow P)$

Statements and Operations

- Statements and operators can be combined in any way to form new statements.

P	Q	P∧Q	¬ (P∧Q)	(¬P)∨(¬Q)
true	true	true	false	false
true	false	false	true	true
false	true	false	true	true
false	false	false	true	true

Equivalent Statements

P	Q	$\neg(P \wedge Q)$	$(\neg P) \vee (\neg Q)$	$\neg(P \wedge Q) \leftrightarrow (\neg P) \vee (\neg Q)$
true	true	false	false	true
true	false	true	true	true
false	true	true	true	true
false	false	true	true	true

- The statements $\neg(P \wedge Q)$ and $(\neg P) \vee (\neg Q)$ are logically equivalent, because $\neg(P \wedge Q) \leftrightarrow (\neg P) \vee (\neg Q)$ is always true.

Logic and Bit operations

- Computers use bits. A bit is a symbol with two values 0 (F) and 1 (T)
- Computer bit operations correspond to the logical connectives.
- We have bitwise OR, bitwise AND, and bitwise XOR
- Example for two bit strings 0110110110 and 1100011101

01 1011 0110	
11 0001 1101	
11 1011 1111	Bitwise OR
01 0001 0100	Bitwise AND
10 1010 1011	Bitwise XOR

Tautologies and Contradictions

- A tautology is a statement that is always true.
- Examples:
- $R \vee (\neg R)$
- $\neg(P \wedge Q) \leftrightarrow (\neg P) \vee (\neg Q)$

- If $S \rightarrow T$ is a tautology, we write $S \Rightarrow T$.
- If $S \leftrightarrow T$ is a tautology, we write $S \Leftrightarrow T$.

Tautologies and Contradictions

- A contradiction is a statement that is always false.
- Examples:
- $R \wedge (\neg R)$
- $\neg(\neg(P \wedge Q) \leftrightarrow (\neg P) \vee (\neg Q))$
- The negation of any tautology is a contradiction, and the negation of any contradiction is a tautology.

Some Logical Equivalences

Equivalence	Name
$p \wedge T \equiv p$ $p \vee F \equiv p$	Identity laws
$p \wedge F \equiv F$ $p \vee T \equiv T$	Domination laws
$p \vee p \equiv p$ $p \wedge p \equiv p$	Idempotent laws
$\neg(\neg p)) \equiv p$	Double negation law
$p \vee q \equiv q \vee p$ $p \wedge q \equiv q \wedge p$	Commutative laws

Some Logical Equivalences

$p \vee (p \wedge q) \equiv p$ $p \wedge (p \vee q) \equiv p$	Absorption laws
$p \vee \neg p \equiv T$ $p \wedge \neg p \equiv F$	Negation laws
$(p \vee q) \vee r \equiv p \vee (q \vee r)$ $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$	Associative laws
$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$ $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$	Distributive laws
$\neg(p \wedge q) \equiv \neg p \vee \neg q$ $\neg(p \vee q) \equiv \neg p \wedge \neg q$	De Morgan's laws

Predicates and Quantifiers

- The statement P(x): "*x* is greater than 3" has two parts. The first part, the variable *x*, is the subject of the statement. The second part, (P) the predicate, "is greater than 3" refers to a property that the subject of the statement can have.
- P(x) is the value of the propositional function P at x. Once a value has been assigned to the variable *x*, the statement P(x) becomes a proposition and has a truth value.
- Quantification expresses the extent to which a predicate is true over a range of elements

Universal Quantification

- Let P(x) be a propositional function.

- **Universally quantified sentence**:
- For all x in the universe of discourse (domain) P(x) is true.

- Using the universal quantifier ∀:
- ∀x P(x) “for all x, P(x)” or “for every x, P(x)”

- (Note: ∀x P(x) is either true or false, so it is a proposition, not a propositional function.)

Universal Quantification

- Example:
- S(x): x is a UMB student.
- G(x): x is a genius.

- What does $\forall x\,(S(x) \rightarrow G(x))$ mean ?

- "If x is a UMB student, then x is a genius."

or

- "All UMB students are geniuses."

Existential Quantification

- **Existentially quantified sentence**:
- There exists an x in the universe of discourse (domain) for which P(x) is true.

- Using the existential quantifier ∃:
- ∃x P(x) "There is an x such that P(x)."
- "There is at least one x such that P(x)."
- (Note: ∃x P(x) is either true or false, so it is a proposition, but no propositional function.)
- Uniqueness quantifier: The notation $\exists! \, xP(x)$ [or $\exists_1 xP(x)$] states "There exists a unique x such that P(x) is true."

Existential Quantification

- Example:
- P(x): x is a UMB professor.
- G(x): x is a genius.

- What does $\exists x\,(P(x) \wedge G(x))$ mean ?

- "There is an x such that x is a UMB professor and x is a genius."

or

- "At least one UMB professor is a genius."

Quantification

- Another example:
- Let the universe of discourse be the real numbers.

- What does $\forall x \exists y\ (x + y = 320)$ mean ?

- "For every x there exists a y so that x + y = 320."

Is it true? yes

Is it true for the natural numbers? no

Negation

- $\neg(\forall x\ P(x))$ is logically equivalent to $\exists x\ (\neg P(x))$.
- $\neg(\exists x\ P(x))$ is logically equivalent to $\forall x\ (\neg P(x))$.

Quantification

- Introducing the **universal quantifier** $\forall$ and the **existential quantifier** $\exists$ facilitates the translation of world knowledge into predicate calculus.

- **Examples:**

- Paul beats up all professors who fail him.
- $\forall x([Professor(x) \wedge Fails(x, Paul)] \rightarrow BeatsUp(Paul, x))$

- All computer scientists are either rich or crazy, but not both.
- $\forall x\ (CS(x) \rightarrow [Rich(x) \wedge \neg Crazy(x)] \vee [\neg Rich(x) \wedge Crazy(x)])$

- Or, using XOR:
- $\forall x\ (CS(x) \rightarrow [Rich(x) \oplus Crazy(x)])$

More Practice for Predicate Logic

- **Important points:**

- Define propositional functions in a useful and reusable manner, just like functions in a computer program.

- Make sure your formalized statement evaluates to “true” in the context of the original statement and evaluates to “false” whenever the original statement is violated.

More Practice for Predicate Logic

- **More Examples:**
- Jenny likes all movies that Peter likes (and possibly more).
- $\forall x\ [Movie(x) \wedge Likes(Peter, x) \rightarrow Likes(Jenny, x)]$

- There is exactly one UMass professor who won a Nobel prize
- $\exists x[UMBProf(x) \wedge Wins(x, NobelPrize)] \wedge$
$\neg\exists y,z[y \neq z \wedge UMBProf(y) \wedge UMBProf(z) \wedge$
$Wins(y, NobelPrize) \wedge Wins(z, NobelPrize)]$