CSCE 222 Section 501
Discrete Structures for Computing

Review for the Final

Hyunyoung Lee
Final Exam

Tuesday, December 13, 2016
starting at 3:30pm (until 5:30pm)
in our classroom (ETB 2005)
Topics

- Logic and Proofs (Chapter 1)
- Sets and Functions (Chapter 2)
- Algorithms and their Complexity (Chapter 3)
- Induction and Recursion (Chapter 5)
- Counting (Chapter 6)
- Solving Recurrences (Chapter 8)
- Relations (Chapter 9)
- Models of Computation (Chapter 13)
Strategy for Exam Preparation

- Start studying now (unless you have already started)!
- Study lecture notes, make sure you know the definitions, and study the examples!
- Review the midterm exams, quizzes/in-class exercises, and homework
- Study the examples in the textbook
- Do odd numbered exercises
- Bring a scantron: 8x11 gray
Logical Connectives - Summary

Let $B = \{ \texttt{t}, \texttt{f} \}$. Assign to each connective a function $M: B \to B$ that determines its semantics.

<table>
<thead>
<tr>
<th>$P$</th>
<th>$M_{\neg}(P)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\texttt{f}$</td>
<td>$\texttt{t}$</td>
</tr>
<tr>
<td>$\texttt{t}$</td>
<td>$\texttt{f}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$P$</th>
<th>$Q$</th>
<th>$M_{\wedge}(P, Q)$</th>
<th>$M_{\vee}(P, Q)$</th>
<th>$M_{\oplus}(P, Q)$</th>
<th>$M_{\rightarrow}(P, Q)$</th>
<th>$M_{\leftrightarrow}(P, Q)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\texttt{f}$</td>
<td>$\texttt{f}$</td>
<td>$\texttt{f}$</td>
<td>$\texttt{f}$</td>
<td>$\texttt{f}$</td>
<td>$\texttt{t}$</td>
<td>$\texttt{t}$</td>
</tr>
<tr>
<td>$\texttt{f}$</td>
<td>$\texttt{t}$</td>
<td>$\texttt{f}$</td>
<td>$\texttt{t}$</td>
<td>$\texttt{t}$</td>
<td>$\texttt{t}$</td>
<td>$\texttt{f}$</td>
</tr>
<tr>
<td>$\texttt{t}$</td>
<td>$\texttt{f}$</td>
<td>$\texttt{f}$</td>
<td>$\texttt{t}$</td>
<td>$\texttt{t}$</td>
<td>$\texttt{f}$</td>
<td>$\texttt{f}$</td>
</tr>
<tr>
<td>$\texttt{t}$</td>
<td>$\texttt{t}$</td>
<td>$\texttt{t}$</td>
<td>$\texttt{t}$</td>
<td>$\texttt{f}$</td>
<td>$\texttt{t}$</td>
<td>$\texttt{t}$</td>
</tr>
</tbody>
</table>
Perhaps the most important logical connective is the **conditional**, also known as **implication**:

\[ p \rightarrow q \]

The statement asserts that \( q \) holds on the condition that \( p \) holds. We call \( p \) the **hypothesis** or **premise**, and \( q \) the **conclusion** or **consequence**. Typical usage in proofs:

“If \( p \), then \( q \)”; “\( p \) implies \( q \)”; “\( q \) when \( p \)”; “\( q \) follows from \( p \)”

“\( p \) is sufficient for \( q \)”; “a sufficient condition for \( q \) is \( p \)”;
“a necessary condition for \( p \) is \( q \)”; “\( q \) is necessary for \( p \)”
Logical Equivalence

Two statements involving quantifiers and predicates are **logically equivalent** if and only if they have the same truth values no matter which predicates are substituted into these statements and which domain is used.

We write $A \equiv B$ for logically equivalent $A$ and $B$.

You use logical equivalences to derive more convenient forms of statements.

Example: De Morgan’s laws.
De Morgan's Laws

\[\neg \forall x P(x) \equiv \exists x \neg P(x)\]

\[\neg \exists x P(x) \equiv \forall x \neg P(x)\]

\[\neg(p \land q) \equiv \neg p \lor \neg q\]

\[\neg(p \lor q) \equiv \neg p \land \neg q\]
An argument in propositional logic is a sequence of propositions that end with a proposition called conclusion. The argument is called valid if the conclusion follows from the preceding statements (called premises).

In other words, in a valid argument it is impossible that all premises are true but the conclusion is false.
Modus Ponens

The tautology \((p \land (p \rightarrow q)) \rightarrow q\) is the basis for the rule of inference called “modus ponens”.

\[
p
\]
\[
p \rightarrow q
\]
\[
\therefore \quad q
\]
Modus Tollens

\[-q\]
\[p \rightarrow q\]

\[\therefore \neg p\]

“The University will not close on Wednesday.”

“If it snows on Wednesday, then the University will close.”

Therefore, “It will not snow on Wednesday”
Example Formal Argument

\[ \neg p \land q \]
\[ r \rightarrow p \]
\[ \neg r \rightarrow s \]
\[ s \rightarrow t \]

\[ \therefore \; t \]

1) \( \neg p \land q \) Hypothesis
2) \( \neg p \) Simplification of 1)
3) \( r \rightarrow p \) Hypothesis
4) \( \neg r \) Modus tollens using 2) and 3)
5) \( \neg r \rightarrow s \) Hypothesis
6) \( s \) Modus ponens using 4) and 5)
7) \( s \rightarrow t \) Hypothesis
8) \( t \) Modus ponens using 6) and 7)
Proofs

Direct Proof

Proof by Contradiction

Proof by Induction
- weak induction
- strong induction
- structural induction
- transfinite induction (= induction over well-ordered sets)
You should be familiar with

- the set builder notation
- subsets, equality of sets
- union, intersection, set difference, complement
- cartesian product, power set
- cardinality of sets (e.g., when is $|A| \leq |B|$?)
De Morgan's Laws

\[ \overline{A \cap B} = \overline{A} \cup \overline{B} \]

**Proof:**

\[ \overline{A \cap B} = \{x \mid x \notin A \cap B\} \text{ by definition of complement} \]
\[ = \{x \mid \neg(x \in A \cap B)\} \]
\[ = \{x \mid \neg(x \in A \land x \in B)\} \text{ by definition of intersection} \]
\[ = \{x \mid \neg(x \in A) \lor \neg(x \in B)\} \text{ de Morgan's law from logic} \]
\[ = \{x \mid (x \notin A) \lor (x \notin B)\} \text{ by definition of } \notin \]
\[ = \{x \mid x \in \overline{A} \lor x \in \overline{B}\} \text{ by definition of complement} \]
\[ = \{x \mid x \in \overline{A \cup B}\} \text{ by definition of union} \]
\[ = \overline{A} \cup \overline{B} \]
Functions

Let $f: A \rightarrow B$ be a function.

We call

- $A$ the **domain** of $f$ and

- $B$ the **codomain** of $f$.

The **range** of $f$ is the set

$$f(A) = \{ f(a) \mid a \text{ in } A \}$$
Functions

Let $A$ and $B$ be sets. Consider a function $f: A \to B$.
- When is $f$ surjective?
- When is $f$ injective?
- When is $f$ bijective?
Floor and Ceiling Functions

The **floor** function \( \lfloor x \rfloor : \mathbb{R} \rightarrow \mathbb{Z} \) assigns to a real number \( x \) the largest integer \( \leq x \).

The **ceiling** function \( \lceil x \rceil : \mathbb{R} \rightarrow \mathbb{Z} \) assigns to a real number \( x \) the smallest integer \( \geq x \).

\[ \lfloor 3.2 \rfloor = 3 \text{ and } \lceil 3.2 \rceil = 4 \]

\[ \lfloor -3.2 \rfloor = -4 \text{ and } \lceil -3.2 \rceil = -3 \]
Basic Facts

We have \([ x ] = n\) if and only if \(n \leq x < n+1\).

We have \([ x ] = n\) if and only if \(n-1 < x \leq n\).

We have \([ x ] = n\) if and only if \(x-1 < n \leq x\).

We have \([ x ] = n\) if and only if \(x \leq n < x+1\).
Example

Prove or disprove:

\[ \lfloor \sqrt{\lceil x \rceil} \rfloor = \lfloor \sqrt{x} \rfloor \]
Example 3

Let $m = \lfloor \sqrt{x} \rfloor$

Hence, $m \leq \sqrt{\lfloor x \rfloor} < m + 1$

Thus, $m^2 \leq \lfloor x \rfloor < (m + 1)^2$

It follows that $m^2 \leq x < (m + 1)^2$

Therefore, $m \leq \sqrt{x} < m + 1$

Thus, we can conclude that $m = \lfloor \sqrt{x} \rfloor$

This proves our claim.
You need to know

- sequence notations
- important sequences
- summation notation
- geometric sum
- sum of first n positive integers
If $a$ and $r \neq 0$ are real numbers, then

$$
\sum_{j=0}^{n} ar^j = \begin{cases} 
\frac{ar^{n+1} - a}{r-1} & \text{if } r \neq 1 \\
(n + 1)a & \text{if } r = 1 
\end{cases}
$$

Proof:
The case $r = 1$ holds, since $ar^j = a$ for each of the $n + 1$ terms of the sum.

The case $r \neq 1$ holds, since

$$(r - 1) \sum_{j=0}^{n} ar^j = \sum_{j=0}^{n} ar^{j+1} - \sum_{j=0}^{n} ar^j$$

$$= \sum_{j=1}^{n+1} ar^j - \sum_{j=0}^{n} ar^j$$

$$= ar^{n+1} - a$$

and dividing by $(r - 1)$ yields the claim.
For all $n \geq 1$, we have

$$\sum_{k=1}^{n} k = \frac{n(n + 1)}{2}$$

We prove this by induction.

Basis step: For $n = 1$, we have

$$\sum_{k=1}^{1} k = 1 = 1(1 + 1)/2.$$
Induction Hypothesis: We assume that the claim holds for $n - 1$.

Induction Step: Assuming the Induction Hypothesis, we will show that the claim holds for $n$.

$$\sum_{k=1}^{n} k = n + \sum_{k=1}^{n-1} k$$

$$= 2n/2 + (n - 1)n/2 \text{ by Induction Hypothesis}$$

$$= \frac{2n+n^2-n}{2}$$

$$= \frac{n(n+1)}{2}$$

Therefore, the claim follows by induction on $n$.  


Infinite Geometric Series

Let $x$ be a real number such that $|x| < 1$. Then

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1-x}.$$
Asymptotic Notations and Complexity

1. You need to know
   - big Oh notation
   - big Omega notation
   - big Theta notation
   - be able to prove that \( f = O(g), \ldots \)
   - be able to show that \( O(g) = O(h) \)

2. You need to be able to find the time complexity of given code
Let $f, g: \mathbb{N} \to \mathbb{R}$ be functions from the natural numbers to the set of real numbers.

We write $f \in O(g)$ if and only if there exists some real number $n_0$ and a positive real constant $U$ such that

$$|f(n)| \leq U|g(n)|$$

for all $n$ satisfying $n \geq n_0$.
We define \( f(n) = \Omega(g(n)) \) if and only if there exists a constant \( L \) and a natural number \( n_0 \) such that

\[
L |g(n)| \leq |f(n)|
\]

holds for all \( n \geq n_0 \).

In other words, \( f(n) = \Omega(g(n)) \) if and only if \( g(n) = O(f(n)) \).
We define \( f(n) = \Theta(g(n)) \) if and only if there exist constants \( L \) and \( U \) and a natural number \( n_0 \) such that
\[
L|g(n)| \leq |f(n)| \leq U|g(n)|
\]
holds for all \( n \geq n_0 \).

In other words, \( f(n) = \Theta(g(n)) \) if and only if \( f(n) = \Omega(g(n)) \) and \( f(n) = O(g(n)) \).
def bubble_sort(list)  # O(1) for parameter assignment
    list = list.dup   # O(n)
    for i in 0..(list.length-2) do  # O(n²)
        for j in 0..(list.length - i - 2) do
            list[j], list[j + 1] = list[j + 1], list[j] if list[j + 1] < list[j]
        end
    end
    return list  # O(1)
end  # O(1) + O(n) + O(n²) + O(1) = O(n²)
Induction Principle

Let $A(n)$ be an assertion concerning the integer $n$. If we want to show that $A(n)$ holds for all positive integer $n$, we can proceed as follows:

**Induction basis:** Show that the assertion $A(1)$ holds.

**Induction step:** Show that $[A(n) \implies A(n+1)]$ holds for all positive integers $n$. 
Strong Induction Principle

Let $A(n)$ be the assertion concerning the integer $n$.

If we want to prove it for all $n \geq 1$, we can do the following:

**Induction basis:**

Show that $A(1)$ is true.

**Induction step:**

Show that $[(A(1) \land ... \land A(n)) \rightarrow A(n+1)]$ holds for all $n \geq 1$. 
Inductively Defined Sets

An inductive definition of a set $S$ has the following form:

(a) **Basis**: Specify one or more “initial” elements of $S$.

(b) **Induction**: Give one or more rules for constructing “new” elements of $S$ from “old” elements of $S$.

(c) **Closure**: The set $S$ consists of exactly the elements that can be obtained by starting with the initial elements of $S$ and applying the rules for constructing new elements of $S$.

The closure condition is usually omitted, since it is always assumed in inductive definitions.
In structural induction, the proof of the assertion that every element of an inductively defined set $S$ has a certain property $P$ proceeds by showing that

Basis: Every element in the basis of the definition of $S$ satisfies the property $P$.

Induction: Assuming that every argument of a constructor has property $P$, show that the constructed element has the property $P$. 
Counting

You need to know

- the basic counting principles such as product rule, sum rule, subtraction rule (inclusion-exclusion principle), ...
- generalized pigeonhole principle
- permutations and combinations
- binomial theorem and identities
- Pascal’s identity and Pascal’s triangle
In any cocktail party $n \geq 2$ people, there must be at least two people who have the same number of friends (assuming that the friends relation is symmetric and irreflexive).

The number of friends of each person ranges between 0 and $n-1$.

Case 1: Everyone has at least one friend.

If everyone has at least one friend, then each person has between 1 to $n-1$ friends. Since we have $n$ people, and just $n-1$ different values, there must be two partygoers that have the same number of friends by the pigeonhole principle.

Case 2: Someone has no friends.

If someone lacks any friends, then that person is a stranger to all other guests. Because friend is symmetric, the highest value anyone else could have is $n - 2$, that is, everyone has between 0 to $n - 2$ friends. Since we have $n$ people, and just $n-1$ different values, there must be two partygoers that have the same number of friends by the pigeonhole principle.
Solving Recurrences

You need to be able to solve recurrences

- by solving characteristic equations (e.g., for homogeneous linear recurrences of degree 2)
- by using generating functions
- by using iteration method
- by inspecting, guessing, and verifying a solution
Generating Functions

Given a sequence \((a_0, a_1, a_2, a_3, \ldots)\) of real numbers, one can form its generating function, an infinite series given by

\[
\sum_{k=0}^{\infty} a_k x^k
\]

The generating functions is a formal power series, meaning that we treat it as an algebraic object, and we are not concerned with convergence questions of the power series.
Example 1

Suppose that the sequence is given by

\[(k+1)_{k \geq 0}\]

Then its generating function is given by

\[
\sum_{k=0}^{\infty} (k + 1)x^k = 1 + 2x + 3x^2 + \cdots
\]
Example 2

The generating function of the sequence \((1, 1, 1, 1, \ldots)\)
is given by

\[
1 + x + x^2 + x^3 + \cdots = \frac{1}{1 - x}.
\]
Shifting Sequences

Let $G(x)$ be the generating function of the sequence $(a_k)_k$. Then

$$xG(x) = \sum_{k=0}^{\infty} a_k x^{k+1} = \sum_{k=1}^{\infty} a_{k-1} x^k.$$
Solving a Recurrence

Suppose we have the recurrence system with initial condition $a_0 = 2$ and recurrence $a_k = 3a_{k-1}$ for $k \geq 1$.

$$G(x) - 3xG(x) = \sum_{k=0}^{\infty} a_k x^k - 3 \sum_{k=1}^{\infty} a_{k-1} x^k$$

$$= a_0 + \sum_{k=1}^{\infty} (a_k - 3a_{k-1}) x^k$$

$$= 2.$$  

Thus, $G(x) - 3xG(x) = (1 - 3x)G(x) = 2$. Hence,

$$G(x) = \frac{2}{1 - 3x}.$$
Solving a Recurrence

Since we know that $1/(1-ax)=1+ax+a^2x^2+...$,

we have

$$G(x) = 2(1+3x+3^2x^2+...).$$

Therefore, a sequence solving the recurrence is given by

$$(2, 2x3, 2x3^2, ...)=(2x3^k)_{k \geq 0}$$
Characteristic Polynomial

Let

\[ a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k} \]

be a linear homogeneous recurrence relation. The polynomial

\[ x^k - c_1 x^{k-1} - \cdots - c_{k-1} x - c_k \]

is called the characteristic polynomial of the recurrence relation.

**Remark:** Note the signs!
Theorem

Let $c_1, c_2$ be real numbers. Suppose that

\[ r^2 - c_1 r - c_2 = 0 \]

has two distinct roots $r_1$ and $r_2$. Then a sequence $(a_n)$ is a solution of the recurrence relation

\[ a_n = c_1 a_{n-1} + c_2 a_{n-2} \]

if and only if

\[ a_n = \alpha_1 r_1^n + \alpha_2 r_2^n \]

for $n \geq 0$ for some constants $\alpha_1, \alpha_2$. 
You need to

- know the basic properties of relations (reflexive, symmetric, antisymmetric, transitive, ...)

- be able to show that a relation is an equivalence relation, and consequently, find its equivalence classes

- know/understand the “congruence mod m” relation

- be able to show that a relation is a partial order

- be able to show that an order is a lattice
Equivalence Relation

A relation $R$ on a set $A$ is called an equivalence relation if and only if $R$ is reflexive, symmetric, and transitive.

- Reflexive: For all $a$ in $A$, we have $(a,a)$ in $R$.
- Symmetric: $(a,b)$ in $R$ implies that $(b,a)$ in $R$.
- Transitive: $(a,b)$ in $R$ and $(b,c)$ in $R$ implies that $(a,c)$ in $R$. 
Example: Congruence mod m

Let \( m \) be a positive integer. For integers \( a \) and \( b \), we write

\[ a \equiv b \pmod{m} \]

if and only if \( m \) divides \( a-b \).

For all \( a \) in \( \mathbb{Z} \), we have \( m \mid (a-a) \), since \( m \cdot 0 = 0 = a-a \). Thus, \( a \equiv a \pmod{m} \) holds for all integers \( a \). Thus, the relation is reflexive.

For \( a, b \) in \( \mathbb{Z} \), if \( a \equiv b \pmod{m} \), then this means that there exists an integer \( k \) such that \( mk = a-b \). Thus, \( m(-k) = b-a \), which implies \( b \equiv a \pmod{m} \). Thus, the relation is symmetric.
Example: Congruence mod m

If \( a \equiv b \pmod{m} \) and \( b \equiv c \pmod{m} \) holds, then this means that there exist integers \( k \) and \( l \) such that

\[
mk = a-b \quad \text{and} \quad ml = b-c
\]

Hence, \( m(k+l) = a-b + b-c = a-c \)

This shows that \( a \equiv c \pmod{m} \) holds.

Therefore, the relation is transitive.

We can conclude that \( a \equiv b \pmod{m} \) is an equivalence relation.
Equivalence Classes

Let R be an equivalence relation on a set A. For an element \( a \) in A, the set of elements

\[
[a]_R = \{ b \in A \mid a \mathbin{R} b \}
\]

is called the equivalence class of \( a \).
Example

Let us consider the equivalence relation $a \equiv b \pmod{4}$ on the set of integers. Thus, two integers $a$ and $b$ are related whenever their difference is a multiple of 4. Thus, the equivalence classes are:

$[0] = \{ \ldots, -8, -4, 0, 4, 8, \ldots \}$

$[1] = \{ \ldots, -7, -3, 1, 5, 9, \ldots \}$

$[2] = \{ \ldots, -6, -2, 2, 6, \ldots \}$

$[3] = \{ \ldots, -5, -1, 3, 7, \ldots \}$

A relation $R$ on a set $A$ is called a **partial order** if and only if it is reflexive, antisymmetric, and transitive.

A set $A$ with a partial order is called a partially ordered set (**poset**).
Lattices

A partially ordered set in which every pair has both a least upper bound and a greatest lower bound is called a lattice.
Example

Consider the set $(\mathbb{N},|)$ of positive integers that is partially ordered with respect to the divisibility relation.

Let $a$ and $b$ be two distinct positive integers. Then the least upper bound of $\{a,b\}$ is the least common multiple of $a$ and $b$. The greatest lower bound is the greatest common divisor of $\{a,b\}$. Therefore, $(\mathbb{N},|)$ is a lattice.
You need to

- be able to explain/remember the Chomsky Hierarchy

- be able to tell what type a given language or grammar is

- be able to determine the language associated with a grammar (or vice versa)

- be able to construct finite state machines and finite state automata

- know the characterization of regular languages as languages accepted by finite state automata or languages described by regular expressions
Formal Languages (Cont.)

You need to

- understand the configuration of the Turing machine
- be able to explain the execution of the Turing machine
- be able to determine which language can be recognized by which machine (FSA or TM)
Chomsky Hierarchy (Computing View)

Type 0 – Phrase-structure Grammars
Type 1 – Context-Sensitive
\[ a^n b^n c^n \text{ (Turing Machines)} \]
Type 2 – Context-Free
\[ a^n b^n \text{ (Pushdown automata)} \]
Type 3 – Regular
\[ a^* b^* \text{ (FSA)} \]
Defining the PSG Types

Type 0: Any PSG

Type 1: Context-Sensitive PSG:

Productions are of the form $lAr \rightarrow lwr$ where $A$ is a nonterminal symbol, and $w$ a nonempty string in $V^*$. Can contain $S \rightarrow \lambda$ if $S$ does not occur on RHS of any production.

Type 2: Context-Free PSG:

Productions are of the form $A \rightarrow B$ where $A$ is a nonterminal symbol.

Type 3: Regular PSGs:

Productions are of the form $A \rightarrow aB$ or $A \rightarrow a$ where $A,B$ are nonterminal symbols and $a$ is a terminal symbol. Can contain $S \rightarrow \lambda$. 

59
A finite state machine $M$ is given by $M=(S, I, O, f, g, s_0)$, where:

- $S$ is the set of states
- $I$ is input alphabet
- $O$ is output alphabet
- $f$ is the transition function that assigns each (state, input) pair a new state
- $g$ is output function that assigns each (state, input) pair an output.
- $s_0$ is the initial state
A finite state automaton $M = (S, I, f, s_0, F)$ consists of a finite set $S$ of states, a finite input alphabet $I$, a transition function $f: S \times I \rightarrow S$ that assigns to a given current state and input the next state of the automaton, an initial state $s_0$, and a subset $F$ of $S$ consisting of accepting (or final) states.
Operations on Languages: Concatenation

Suppose that \( V \) is an alphabet.
Let \( A \) and \( B \) be subsets of \( V^* \).
Denote by \( AB \) the set \( \{ xy \mid x \text{ in } A, y \text{ in } B \} \).
Example: \( A = \{0,11\} \) and \( B = \{1, 10, 110\} \)
Then \( AB = \{ 01, 010, 0110, 11, 110, 1110 \} \)
Suppose that $V$ is an alphabet.
Let $A$ be a subset of $V^*$.
Define $A^0 = \{ \lambda \}$ and $A^{n+1} = A^n A$
Example: $A = \{1, 00\}$
$A^2 = \{11, 100, 001, 0000\}$
Operations on Languages: Kleene Closure

Let $V$ be an alphabet, and $A$ a subset of $V^*$. The Kleene closure $A^*$ of $A$ is defined as

$$A^* = \bigcup_{k=0}^{\infty} A^k$$
The union of two formal languages $L_1$ and $L_2$ is the formal language $\{ x \mid x \in L_1 \text{ or } x \in L_2 \}$. 
A Turing Machine

\[ M = (Q, \Sigma, \Gamma, \delta, q_0, \lambda, F) \]
Hints

Study the lecture slides and read the textbook, carefully studying the examples.

Study quizzes/in-class exercises, homework problems, two midterm exams.

Drill using odd numbered exercises.

Get enough sleep!!