2202 COL 352 Major

CHINMAY MITTAL

TOTAL POINTS

35 / 35

QUESTION 1

1 Quotient Turing Recognizable 7 / 7

✓ **+ 7 pts** *Correct*

+ 4 pts Partially Correct

+ 0 pts Incorrect

QUESTION 2

2 CFL XOR 7 / 7

+ 7 pts Correct

+ 0 pts Incorrect

✓ **+ 1 pts** *True/False(i)*

✓ **+ 1 pts** *Express XOR using logical operators*

✓ **+ 1 pts** *CFLs are !(closed under complementation)*

✓ **+ 1 pts** *Counter -example*

✓ **+ 1 pts** *True/False (ii)*

✓ **+ 2 pts** *Correct explanation*

+ 0.5 pts Partial answer

QUESTION 3

3 Kleene Star P 7 / 7

+ 7 pts Correct

✓ **+ 5 pts** *Dynamic programming algorithm*

✓ **+ 2 pts** *run-time analysis*

+ 0 pts Incorrect

QUESTION 4

4 Many-One Transitive 7 / 7

+ 0 pts Incorrect/ Not Attempted

✓ **+ 1 pts** *Definition of transitivity*

✓ **+ 2.5 pts** *Formal Proof*

✓ **+ 1 pts** *True/False*

✓ **+ 2.5 pts** *Counter Example*

QUESTION 5

5 REGPAL 7 / 7

+ 0 pts Incorrect

✓ **+ 2 pts** *Correct Grammar for Non-Palindromes*

+ 2.5 pts Some correct ideas (towards reduction to emptiness etc)

✓ **+ 5 pts** *Reduction to emptiness of CFL*

+ 7 pts Correct

gradescope

Name: CHIN MAY MITTAL Roll No: 2020CS10336

(COL 352) Introduction to Automata and Theory of Computation May 2, 2023

Major (A)

Duration: 2 hours **(35 points)**

- Be clear in your writing. If you use a statement proved in class or in the problem set, then write down the entire statement before using it.
- **You will not get a new sheet, so make sure you are certain when you write something.** Make a judicious decision of which tool(s) to use to get a clean and short answer that fits in the space.
- Every question in this paper is worth 7 points.
- This major paper contains two question papers in one: one **regualar paper and one easy paper** as follows. Every question has two parts: **A** and **B**. You can **either answer Part A for all questions** or **answer Part B for all questions**, i.e., you can either **answer 1A, 2A, 3A, 4A, 5A** or you can **answer 1B, 2B, 3B, 4B, 5B**. You **CANNOT mix**: for e.g., **if you have answered 1A, 2B, 3A, 4B, 5A, then your questions 2 and 4 will not be graded**, i.e, if you mix, I will assume you chose to answer part A or B based on whichever part the majority of your answers come from. **Part B of every question will be considerably easy**. So if you want to choose the easy paper, you should attempt just the questions from Part B for all the questions. **However if you choose to answer Part B, the maximum final grade you will be eligible for is a D, even if your pre-major + major score is eligible for a better grade.**
- Before you turn in your paper, indicate which part (A or B) you have attempted in this paper in the top of this page in the space provided, i.e., **Major ( )**
- If you cheat, you will surely get an F in this course.

Notation and some helpful information. You may assume that the following languages we studied in class are undecidable:

$$\begin{aligned} \mathrm{A_{TM}} &= \{\langle M, w\rangle \mid M \text{ accepts } w\}. \\ \mathrm{Halt} &= \{\langle M, w\rangle \mid M \text{ halts on } w\}. \\ \mathrm{E_{TM}} &= \{\langle M\rangle \mid L(M) = \emptyset\} \end{aligned}$$

- Let $L_1 \oplus L_2$ be defined as $\{w \mid (w \in L_1 \text{ and } w \notin L_2) \text{ OR } (w \notin L_1 \text{ and } w \in L_2)\}$. A class of languages $\mathcal{L}$ is said to be closed under $\oplus$ if for all $L_1, L_2 \in \mathcal{L}$, $L_1 \oplus L_2 \in \mathcal{L}$.
- For any two languages $A, B \subseteq \Sigma^*$, the quotient
$$\frac{A}{B} = \{x \in \Sigma^* \mid \exists y \in B, xy \in A\}$$
- A mapping (or many-one) reduction from a language A to B is a computable function f such that, for every string x, $x \in A \iff f(x) \in B$. We use the notation $A \leq_m B$ to denote the existence of a mapping reduction from A to B.
- 3-SAT $= \{\langle\phi\rangle \mid \phi$ encodes a satisfiable 3-CNF formula$\}$.
- A clique is a graph where every pair of vertices is connected by an edge.
- k-CLIQUE $= \{\langle G, k\rangle \mid G$ is an undirected graph containing a clique on k vertices$\}$

---

xy $\quad x \in A_B$

$\underset{z}{xy} \in A$ & $\underset{z}{y} \in B$.

1. **(A)** Show that if A, B are Turing recognizable languages, then so is the quotient $\frac{A}{B}$ (see Page 1).

   (B) Define Turing recognizable and decidable languages. Show that there exists a language that is not Turing recognizable using diagonalization.

Given x we need to find y such that $xy \in A$ and $y \in B$ if it exists. Consider a Turing Machine which does the following. first define an lexicographic ordering of all strings in Σ^* (Since this set is countable such an ordering exists) let this ordering be $w_1, w_2 \ldots\ldots$)

Let M_1 be the recognizers for A and M_2 be the recognizers for B.

A Turing Machine M which recognizes $\frac{A}{B}$ is as follows.

In the ith step.

it will simulate M_1 on $xw_1, xw_2 \ldots\ldots xw_i$, for at most i steps and it will simulate M_2 on $w_1, w_2 \ldots\ldots w_i$ for at most i steps. if any string w_j exists. in w_1 to $\ldots\ldots w_i$ such that within i steps both M_1 accept xw_j and M_2 accepts w_j then M will accept x. (M will keep running step1, step2 and so on till it accepts x or else keeps running)

if $x \notin \frac{A}{B}$ then there some string y such that $xy \notin A$ $y \notin B$. Let xy be the ith string and y be j^{th} string

let $M = \max(M_1, M_2)$ $\quad M_2 \to$ # of steps for M_2 recognizing y

$M_1 \to$ # of steps for M_1 recognizing xy

Consider. $\alpha = \max(M, i, j)$. In the αth iteration of the TM M we will definitely encounter y since $y = w_j$ and $j < M$. Hence will run M_1 on xy and M_2 on y for α steps $\geq M_1, M_2$ and hence M_1 will accept xy and M_2 will accept $y \Rightarrow M$ will accept $x \Rightarrow M$ recognizes $\{A/B\}$.

NOTE: (i) if $x \notin A/B$ then M will not Halt.

(ii) in the ith step. M will simulate M_1 on xw_j and M_2 on w_j parallely $\forall j$ one step of M_1 followed by one step of M_2.

We can use multitape TMs for M to simulate $M_1 | M_2$ and also maintain a count of which step we are in (i.e i) and which string we are simulating (ie j)

Name: CHINMAY MITTAL Roll No: 2020CS10336.

2. **(A)** One of the following statements is true and one is false. Correctly identify with a brief justification:
   (i) If L_1 is a CFL and L_2 is a regular language then $L_1 \oplus L_2$ (see Page 1) is a CFL.
   (ii) If L_1, L_2 are both decidable languages then $L_1 \oplus L_2$ (see Page 1) is also decidable.

   (B) Prove that $\{0^n 1^n \mid n \in \mathbb{N}\}$ is not a regular language, and show that it is a CFL by giving a CFG.

TRUE
(ii)

Consider two languages L_1 and L_2 both of which are decided by Turing Machines M_1 and M_2 respectively. Consider a machine M which decides $L_1 \oplus L_2$

for any string x simulate M_1 on x and M_2 on x. Since they are deciders both the turing Machines will halt.

Accept x only if the verdict of M_1 and M_2 is opposite $\Rightarrow$ accept if M_1 accepts and M_2 rejects or if M_1 rejects and M_2 accepts

$\Rightarrow$ M accepts x only if $x \in L_1$ and $x \notin L_2$ OR $x \notin L_1$ and $x \in L_2$

and M halts on all inputs since M_1 and M_2 halt on all inputs $\Rightarrow$ M decides $L_1 \oplus L_2$ (decidable language)

FALSE.

(i) We know that CFGs are not closed under complementation. (shown in class.) Hence there exists a Language L such that L is a CFL but $\bar{L}$ is not a CFL

take $L_1 = L$ and $L_2 = \Sigma^*$ (regular)

$$L_1 \oplus L_2 = (L_1 \cap \bar{L}_2) \cup (\bar{L}_1 \cap L_2)$$

$$= \phi \cup \bar{L}_1 = \bar{L}_1$$ not a CFL by our assumption..

$\Rightarrow$ if L_1 is a CFL and L_2 is a regular language then $L_1 \oplus L_2$ need not be a CFL.

3. **(A)** Show that **P** is closed under Kleene star operation (i.e, $L \in \mathbf{P} \implies L^* \in \mathbf{P}$).

(B) Give a DFA, CFG and Turing Machine for $\{w \mid w \text{ contains at least three 1s}\}$. What is the time complexity of this language?

Given a Turing Machine M that decides whether a string $x \in L$ in $|x|^c$ time for some constant $c > 0$. We want to build a TM which decides whether a string $w = w_1 w_2 w_3 \ldots w_n \in L^*$ in $|w|^{c'}$ time.

We will use the following dynamic programming approach. for each i from 0 to n We would compute if $w_1 \ldots w_i \in L^*$ and let this value be called $dp[i]$ (stored on the tape of the Turing Machine). w_0 is the string ϵ and is in L^* for any L hence $dp[0] = 1$.

The subroutine used is as follows. HELPER(i) computes and stores $dp[i]$ (We maintain a second tape which is used to store the dp values and initialize it 1 followed by n 0's.

HELPER(i) $\longrightarrow$ computes $dp[i]$

for $j = 0$ to i $\Rightarrow$ $w_0 w_1 \ldots w_j \in L^*$ (computed before)

if $dp[j]$ is 1.

check if $w_{j+1} w_{j+2} \ldots w_n \in L$

if YES $dp[j] = 1$

break.

The dp relation is as follows $w_1 w_2 \ldots w_n$ is in L^* if for any i $w_0 w_1 \ldots w_i \in L^*$ and $w_{i+1} w_{i+2} \ldots w_n \in L$

We can use HELPER from $i = 1$ to n to compute the entire dp array.

We accept a string if $dp[n] = 1$ else we reject it. We can build a TM M_0 which implements the above algorithm.

We initialize dp array on the second tape in $O(|w|)$ time.

Helper can use the third tape for checking $w_{j+1} w_{j+2} \ldots w_n \in L$ by simulating M

HELPER is called $O(|w|)$ times

each helper call takes $O(\overset{O(|w|)}{i} * \underbrace{|w|^c}_{\text{time to check } w_{j+1} \ldots w_n \in L})$

and hence the total algorithm is polynomial in $|w| \Rightarrow L^* \in P$

if a Multitape turing machine can decide in $O(t(|w|))$ time then a single tape turing machine can execute in $O(t^c |w|)$ time for some $c \Rightarrow$ the algorithm remains polynomial in $|w|$ on single tape TM

Name: Chinmay Mittal Roll No: 2020CS10336

4. **(A)** Show that $\leq_m$ (see Page 1) is a transitive relation. If $A \leq_m B$ and B is a regular language, does that imply that A is a regular language? Why or why not?

(B) Define the class **NP**. Show that 3-SAT and k-CLIQUE (see Page 1) are decidable in **NP**.

Clearly $\leq_m$ is reflexive. $A \leq_m A$ can be achieved by a Turing Machine that does nothing to the input and hence $f(w) = w$ follows trivially.

$$w \in A \iff f(w) \in A.$$

if $A \overset{f_1}{\leq_m} B$ and $B \overset{f_2}{\leq_m} C$. M_1 computes f_1 and M_2 computes f_2. Consider a Turing machine M that computes $f_2(f_1(x))$ by first simulating M_1 on x to convert it to $f_1(x)$ and then simulating M_2 on $f_1(x)$ to convert it to $f_2(f_1(x))$

$$x \in A \iff f(x) \in B \iff f_2 f_1(x) \in C$$

$\Rightarrow f = f_1 \circ f_2$ is the computible function such that $x \in A \iff f(x) \in C$ which can be implemented by a TM as shown above. $\Rightarrow A \leq_m C$ (transitivity)

Symmetric. if $A \leq_m B$ then $B \leq_m A$. This can be achieved by a turing machine that does exactly the reverse steps of the Machine which computes A to B. (invert the transition function)

Consider $A = \{0^n 1^n \mid n \geq 0\}$ and $B = \{0\}$. Consider a turing Machine that decides whether a string is of the form $0^n 1^n$ (Such a TM exists, because CFLs can be decided by Turing Machines)

if $w \in A$ then the Turing Machine writes 0 on the tape erasing everything else and otherwise it erases the entire tape $\Rightarrow$

$$w \in A \text{ iff } f(w) \in B$$

$\rightarrow f$ described above

$f(w) = 0$ if $w \in A$

else ϵ

B is regular since it is finite

but A is not a regular language (shown in class by the pumping lemma)

Hence by this counter example

if $A \leq_m B$ and B is regular then A need not be regular.

$L_1 \cap L_2$ $\quad L \cap L_{\text{pallindrom}}$

$L \cap L$ $\quad \in L$

aab bba $\quad (L(M) \cap \text{not pallindrome})$ is ϕ

5. **(A)** Prove that the following language is decidable.

$$\text{REGPAL} = \{\langle M \rangle \mid \text{M is a DFA and all strings accepted by M are palindromes}\}$$

(B) Show that $\text{ALL}_{\text{DFA}} = \{\langle M \rangle \mid M \text{ is a DFA and } L(M) = \Sigma^*\}$ is decidable.

— 0 — 1 —

— 0 0 1 —

Consider The problem of deciding REGPAL

if M is DFA and all strings accepted by M are pallindrome $\iff L(M) \cap L_{\text{not pallindrome}} = \phi$

language of all strings which are not pallindrome.

We first show that $L_{\text{not pallindrome}}$ is a CFL

$S \longrightarrow S_1$ (i)

$S \longrightarrow 0S0 \mid 1S1$ (ii)

$S_1 \longrightarrow 0S_2 1 \mid 1S_2 0$ (iii)

$S_2 \longrightarrow \epsilon \mid 0S_2 \mid 1S_2$ (iv)

Any string which is not a Pallindrome will be of the following form.

$\underbrace{\quad\omega\quad}\,0\,\underbrace{\quad\quad}\,1\,\underbrace{\quad\omega\quad}$

Any string $\longrightarrow$ made by S_2 (iv)

or $\underbrace{\quad\omega\quad}\,1\,\underbrace{\quad\quad}\,0\,\underbrace{\quad\omega\quad}$

Made by applications of (iii)

made by repetitive applications of (ii)

We have shown in a previous paper that if L_1 is a regular language and L_2 is a CFL then $L = L_1 \cap L_2$ is a CFL

$\Rightarrow$ Since $L(M)$ is regular given M is a DFA and $L_{\text{not pallindrome}}$ is a CFL $\quad L = L(M) \cap L_{\text{not pallindrome}}$. is a CFL.

We have reduced the problem to checking whether or not a CFL (L) is empty. E_{CFL} is a decidable problem. (as shown in class)

Hence the decider for REGPAL is as follows. reject $\langle M \rangle$ is not a valid encoding. otherwise find the Grammer G of the CFL such

$L(G) = L(M) \cap L_{\text{not pallindrome}}$.

accept if E_{CFL} accepts G otherwise reject.