INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR
End-Spring Semester 2022-23

Date of Examination: **21/04/23** Session(FN/AN) **AN** **Duration 3 hrs, Marks = 80**
Sub No: **CS21004/CS21204** Sub Name: **Formal Language & Automata Theory**
Department/Centre/School : **Computer Science and Engineering**
Specific charts, graph paper,log book etc. required **NO**
Special Instructions (if any) In case of reasonable doubt, make assumptions and state them upfront. Marks will be deducted for sketchy proofs and claims without proper reasoning. All parts of a single question should be done in the same place.

ANSWER 8 out of 10 questions. Fill in the blanks wherever asked for in the answer script and not in the question paper!

Q1-Q5: AM, Q6-Q10: SD

Question 1 (a) Consider the DFA M whose state diagram is given in Figure 1.

(i) Give an informal description of the language $L(M)$ accepted by M. Justify the answer. **[2]**

(ii) Are there strings $x \in L(M)$ (say $x \geq 3$) such that if the bits of x are flipped (0 changed to 1 and 1 changed to 0), the resulting string is also in $L(M)$. Why or why not? Is this true for every $x \in L(M)$? **[2+2]**

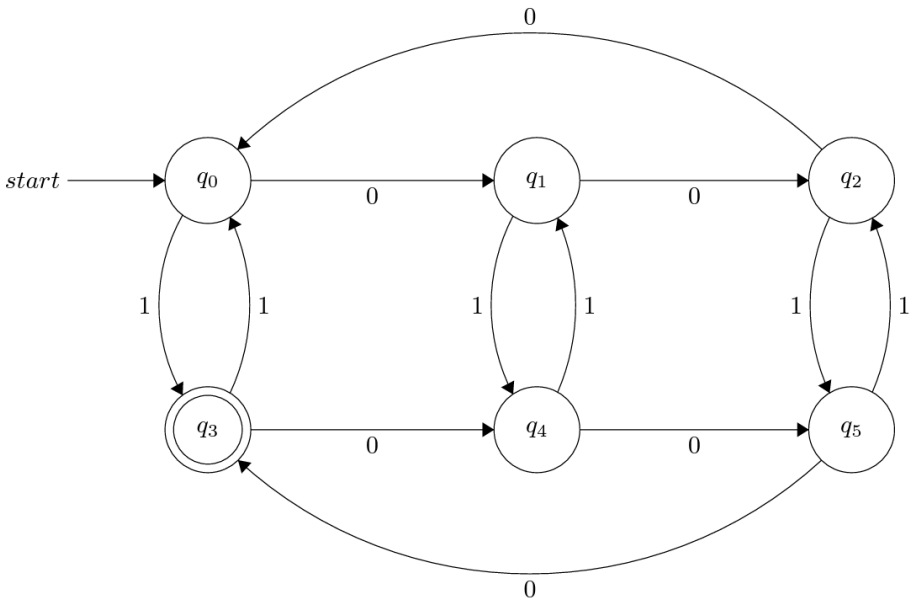

Figure 1: DFA M.

(b) Consider the language $L = \{tu \mid t \text{ and } u \text{ are strings over } \{0,1\} \text{ with the same number of 1's}\}$. Explain and correct the error below:
Proof that L is not regular using the Pumping Lemma: Assume (toward a contradiction) that L is regular. Then the Pumping Lemma applies to L. Let p be the pumping length of L. Choose s to be the string $1^p0^p1^p0^p$, which is in L because $t = 1^p0^p$ and $u = 1^p0^p$ each have p 1's. Since this string is in L and has a length greater than or equal to p, the Pumping Lemma guarantees s can be divided into parts $s = xyz$ such that for any $i \geq 0$, xy^iz is in L, and that $|y| > 0$ and $|xy| \leq p$. Since the first p characters of s are all 1's and we have $|y| > 0$ and $|xy| \leq p$, we know that y must be nonempty and made up of all 1's. But if we let $i = 2$, we get a string xy^iz that

The language recognized by M is the set of all strings with an odd number of $1's$ and a number of $0's$ that's a multiple of three. This is because horizontally, the states represent the number of $0's$ mod 3 and vertically, the states represent the number of $1's$ mod 2 (so even or odd). The bottom left state thus represents both an odd number of $1's$ and a number of $0's$ that's a multiple of three (i.e. number of 1's mod $3 = 0$).

Yes, there does exist such a string. For example, choose string 111000. Since it has an odd number of $1's$ and exactly three $0's$ it is in the language $L(M)$. If the bits are flipped, it still has an odd number of $1's$ and exactly three $0's$, so it is still in the language.

This is not true for every string, because 111 is in $L(M)$ but when the bits are flipped, we get 000 which is not, since it does not have an odd number of $1's$.

Figure 2: Answer 1(a)

The error is that actually when $s = 1^p0^p1^p0^p$ and $i = 2$, we can still have xy^2z be in the language L. It is correct that y is nonempty and made up of all 1s, but it is still possible that the string xy^2z can be written as the concatenation of two strings with the same number of 1's. For example, if $y = 11$, then the pumped string $xy^2z = 1^{p+2}0^p1^p0^p$ which is still in L because it can be written as tu where $t = 1^{p+1}$ and $u = 10^p1^p0^p$, which both have the same number of 1's.
In fact, the language L is regular, because any string that can be written as the concatenation of two strings with the same number of 1's must have an even number of 1's. Similarly, any string with an even number of 1's can be written as the concatenation of two strings with an equal number of 1's. So L is just the language of strings over $\{0, 1\}$ with an even number of 1's. This is easily shown to be regular by the following DFA that recognizes L.

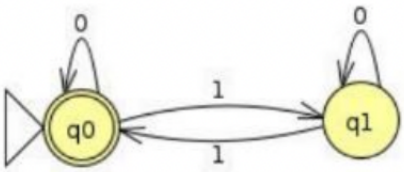

Figure 3: Answer 1(b)

is not in L because repeating y twice adds 1's to t but not to u, and strings in L are required to have the same number of 1's in t as in u. This is a contradiction. Therefore the assumption is false, and L is not regular. **[4]**

Question 2 Consider the three languages in Figure 4. In each case, $\sum = \{0, 1\}$. S is the start symbol for each CFG.

L_1:

$S \to S01A|A$
$A \to B10A|B$
$B \to 00|11$

L_2:

$S \to 10V01S|V$
$V \to 00Z11$
$Z \to \epsilon|0Z$

L_3:

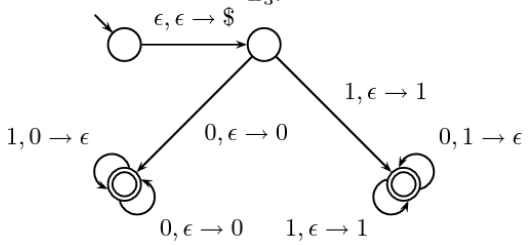

Figure 4: The three languages L_1, L_2 and L_3.

(i) Fill in the blanks with either $\in$ or $\notin$. 0001000100 $\in$ L_1, 1000001101100011010011 $\in$ L_2, 011000 $\notin$ L_3, 100100 $\notin$ L_2. **[1]**

(ii) For each language above, give a description in English of the language. **[1.5*3=4.5]**

(iii) Give a derivation and a parse tree showing that 00011110000111 $\in L_1$. **[1.5+1.5=3]**

(iv) Give the contents of the stack after running the PDA for L_3 on 11001101. Be sure to clearly indicate which end of the stack is the top and which end is the bottom. **[1.5]**

L_1 consists of strings of alternating types of pairs of bits. The odd pairs (first, third, etc.) must match (00 or 11), and the even pairs (second, forth, etc.) must not match (01 or 10). The total number of pairs must be odd.

L_2 consists of strings beginning with 0 or more copies of the sequence 1000, 0^*, 1101 and ending with the sequence 00, 0^*, 11. This is equivalent to the regular expression $10000^*1101)^*000^*11$.

L_3 consists of all strings beginning with 0 where every prefix of the string has more 0's than 1's and all strings beginning with 1 where every prefix of the string has more 1's than 0's.

$S \Rightarrow S01A \Rightarrow S01S01A \Rightarrow A01S01A \Rightarrow B01S01A \Rightarrow 0001S01A \Rightarrow 0001A01A \Rightarrow 0001B10A01A \Rightarrow 00011110A01A \Rightarrow 00011110B01A \Rightarrow 000111100001A \Rightarrow 000111100001B \Rightarrow 00011110000111$

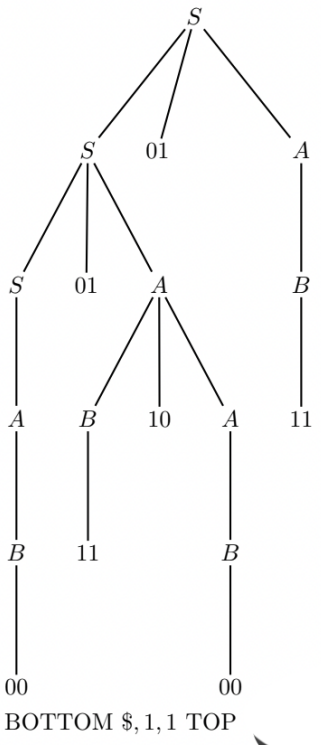

Figure 5: Answer 2

Question 3 (a) Let P be a PDA defined as follows. $Q = \{q_0, q_1, q_2\}$, $\sum = \{0, 1\}$, $\Gamma = \{R\}$, $F = \{q_1, q_2\}$.

$\delta(q_0, 0, \epsilon) = \{(q_0, R)\}$
$\delta(q_0, \epsilon, \epsilon) = \{(q_1, \epsilon)\}$
$\delta(q_0, 1, R) = \{(q_2, \epsilon)\}$
$\delta(q_1, \epsilon, R) = \{(q_1, \epsilon)\}$
$\delta(q_2, 1, R) = \{(q_2, \epsilon)\}$
$\delta(q_2, \epsilon, R) = \{(q_2, \epsilon)\}$

(i) Describe the language accepted by P using set-theoretic notations. **[2]**

(ii) Give the state diagram of P. **[2]**

(iii) Write down the traces of computations (all computation paths) of 001 in P using state transition tables (Current State | Current String | Current Stack). **[2]**

(iv) Show that $0011 \in L(P)$. **[1]**

Solution: 1. Describe the language accepted by P.

The language is $\{0^m 1^n \mid 0 \leq n \leq m\}$.

Figure 6: Answer 3a(i)

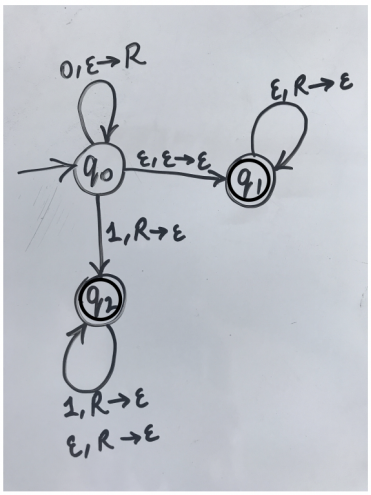

Figure 7: Answer 3a(ii)

State	String	Stack
q_0	001	ϵ
q_1	001	ϵ

State	String	Stack
q_0	001	ϵ
q_0	01	R
q_1	01	R
q_1	1	ϵ

State	String	Stack
q_0	001	ϵ
q_0	01	R
q_0	1	RR
q_1	1	RR
q_1	1	R
q_1	1	ϵ

State	String	Stack
q_0	001	ϵ
q_0	01	R
q_0	1	RR
q_2	ϵ	R
q_2	ϵ	ϵ

Figure 8: Answer 3a(iii)

To show that $0011 \in L(P)$, it's enough to give one computation path that halts at an accept state.

State	String	Stack
q_0	0011	ϵ
q_0	011	R
q_0	11	RR
q_2	1	R
q_2	ϵ	ϵ

Figure 9: Answer 3a(iv)

(b) Let A be a language. Define the operation *splittable* as follows:
$A^{splittable} = \{w_1 w_2 \mid w_1 \in A, w_2 \in A, \text{ and } w_1 w_2 \in A\}$. Show that the class of regular languages is closed under the *splittable* operation. **[3]**

If $A^{splittable}$ were defined as $\{w_1w_2 \mid w_1 \in A, w_2 \in A\}$ then it would be equivalent to $A \circ A$. Instead, we have the additional condition that every string in $A^{splittable}$ is also in A. Thus, $A^{splittable}$ is exactly the set of string that are both in $A \circ A$ and in A, or equivalently $A^{splittable} = (A \circ A) \cap A$.

If A is regular then $A \circ A$ is regular (the concatenation of two regular languages is regular, Sipser Thm. 1.26). It then follows that $(A \circ A) \cap A$ is regular (the intersection of two regular languages is regular, Sipser p. 46, footnote 3).

Figure 10: Answer 3(b))

Question 4 (a) Draw a parse tree for the string $f(f(x))$ using the CFG below. The terminals are $\{f, x, (,)\}$ and V is the start symbol. **[2]**

$V \to x|f|FL$
$F \to f($
$L \to V)$

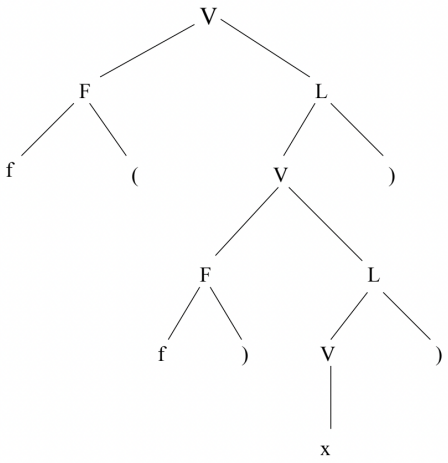

Figure 11: Answer 4(a)

(b) Complete the following high level description of a Turing machine recognising the following language.

$L = \{w \mid w \in \{a, b, c\}^*$ and w has equal number of a's, b's, and c's$\}$

"On input w:

1. If $w =$ ___, accept.
2. Scan the input and find the ___ a. Replace it by X. If ___, reject.
3. Scan the input and find the ___ b. Replace it by Y. If ___, reject.
4. Scan the input and find the ___ c. Replace it by Z. If ___, reject.
5. Scan the input to check if any ___. If none, then ___. Otherwise ___." **[3]**

On input w:

(a) If $w = \epsilon$, then Accept.

(b) Scan the input and find the leftmost a. Replace it by X. If no a is found, Reject.

(c) Scan the input and find the leftmost b. Replace it by Y. If no b is found, Reject.

(d) Scan the input and find the leftmost c. Replace it by Z. If no c is found, Reject.

(e) Scan the input and check if any a's, b's, or c's are left. If none, then Accept
Otherwise go to Step (b)

Figure 12: Answer 4(b)

(c) Prove/disprove that a language (L) is Turing-decidable *if and only if* some enumerator enumerates the strings of this language in length increasing lexicographic order. **Please do not write an essay here!** In the forward direction give a high-level stepwise description of the Turing machine (if it exists) and in the backward direction give a high-level stepwise description of the enumerator (if it exists). (**Hint:** The language for which the enumerator enumerates could be either finite or infinite). **[5]**

Solution:
Claim 1: If a language is Turing decidable, then some enumerator enumerates the strings of the language in lexicographic order
Proof:
Let L be a Turing decidable language. There exists a Turing machine M = (Q, Σ, Γ, δ, q_0, q_{accept}, q_{reject}) which decides L.
We will construct an enumerator E for this language which will enumerate the strings of L in lexicographic order.
Construction of E:
E generates each string in Σ* in lexicographic order and then runs the string on M. If M accepts, it prints the string. Otherwise, it does not.

Claim 2: If there exists some enumerator E which enumerates the strings on the language in lexicographic order, then the language is decidable
Proof:
Let L be the language for which E enumerates the strings in lexicographic order. We will construct a Turing machine M for L such that M halts on every input, either in the accept state or in the reject state.
Case 1 : L is finite.

Every finite language is decidable as we can construct a Turing machine M for L in which each string of L is hardwired in M.

Case 2: L is infinite
On receiving input w, M runs E to enumerate all strings in L in lexicographic order until some string lexicographically after w appears. If w has appeared in the enumeration already, then accept. Otherwise reject.
M is a decider as a string which is lexicographically after w will be printed by E as L is infinite. Therefore, M will halt in accept or reject state for every input.

Figure 13: Answer 4(c)

Question 5 (a) Let $L = \{< M > | M$ is a DFA and there exists some string w such that both w and w^R are in $L(M).\}$ Show that E_{DFA} can be used to prove that L is decidable. **[4]**

We know that regular languages are closed under both intersection and reverse. A the description of a DFA $\langle M \rangle$ is in L iff $L(M) \cap L(M)^{\mathcal{R}} \neq \emptyset$ because the intersection is exactly the set of strings w such that both w and $w^{\mathcal{R}}$ are in $L(M)$. We can use these properties to construct a TM to decide L:

Let T be a TM that decides E_{DFA}.

"On input w:

1. If w is not of the form $\langle M \rangle$, reject. Otherwise extract M from the input.
2. Construct a DFA N such that $L(N) = L(M)^{\mathcal{R}}$.
3. Construct a DFA P such that $L(P) = L(M) \cap L(N)$.
4. Run T on $\langle P \rangle$.
5. Accept if T rejects, and reject if T accepts."

We know that this TM always halts because T always halts and we can do the DFA transformations using the algorithms from Chapter 1. This TM decides L because it accepts iff $L(M) \cap L(M)^{\mathcal{R}} \neq \emptyset$.

Figure 14: Answer 5(a)

(b) (i) The following body of statements tries to prove/disprove that decidable languages are closed under intersection. Fill in the blanks and **finally argue** whether you have proved or disproved it.

Let L_1 and L_2 be decidable languages decided by TMs M_1 and M_2 respectively. It should be possible to construct the following TM for $L_1 \cap L_2$:

"On input w:

1. Run M_1 on w. If M_1 ___ then ___.

2. Run M_2 on w. If M_2 ___ then ___.

3. Otherwise ___." **[1.5+1]**

"On input w:

1. Run M_1 on w. If M_1 rejects then reject.
2. Run M_2 on w. If M_2 rejects then reject.
3. Otherwise accept."

This TM accepts if and only if M_1 and M_2 both accept. Since M_1 and M_2 both decide their respective languages, they must halt on every input. Thus this TM must halt on every input and therefor decides the intersection of L_1 and L_2.

Figure 15: Answer 5(b-i)

(ii) The following body of statements tries to prove/disprove that decidable languages are closed under concatenation. Fill in the blanks and **finally argue** whether you have proved or disproved it.

Let L_1 and L_2 be decidable languages decided by TMs M_1 and M_2 respectively. It should be possible to construct the following TM for $L_1 \circ L_2$:

"On input w:

1. For each x between 0 and $|w|$:

a. Let w_1 be the ___ symbols of w.

b. Let w_2 be the ___ w.

c. Run M_1 on w_1. If M_1 ___ then continue with ___.

d. Run M_2 on w_2. If M_2 ___ then continue with ___.

e. If ___ M_1 and M_2 ___ then ___.

2. ___." **[2.5+1]**

"On input w:

1. For each x between 0 and $|w|$:
2. Let w_1 be the first x symbols of w.
3. Let w_2 be the rest of w.
4. Run M_1 on w_1. If M_1 rejects then continue with the next value for x.
5. Run M_2 on w_2. If M_2 rejects then continue with the next value for x.
6. If both M_1 and M_2 accept then accept.
7. Reject."

Figure 16: Answer 5(b-ii)

Question 6 (a) Consider the following CFG for some alphabet $\Sigma = \{a, b\}$:

$$S \longrightarrow AB$$
$$A \longrightarrow BB \,|\, a$$
$$B \longrightarrow AB \,|\, b$$

Construct the CYK parsing table for the string *aabbb* in the following format.

A				
$\underline{\phi}$	A			
$\underline{S, B}$	$\underline{S, B}$	B		
$\underline{A}$	$\underline{A}$	$\underline{A}$	B	
$\underline{S, B}$	$\underline{S, B}$	$\underline{S, B}$	$\underline{A}$	B

[6]

(b) Show that if G is a CFG in CNF then for any string $w \in L(G)$ of length $n \geq 1$, exactly $2n - 1$ steps are required for any derivation of w. **[4]**

Ans: Note that with CNF grammar, every production will increase the string length by 1. Hence, starting from S, we need $n - 1$ productions for generating a string of size n. Further, generating each non-terminal needs one production rule. So, n number of productions are required to produce the n terminals in the final string. Hence, in total $2n - 1$ steps are required.

Question 7 For any two strings $w, t \in \{0, 1\}^*$, let $w \sim t$ denote that the symbols of w are a permutation of the symbols of t. In other words, $w \sim t$ if w and t have the same symbols in the same quantities, but possibly in a different order. For example, $0011 \sim 1010$. For any string w, define $SCRAMBLE(w) = \{t \,|\, t \sim w\}$. For any language A, let

$$SCRAMBLE(A) = \{t \,|\, t \in SCRAMBLE(w) \text{ for some } w \in A\}$$

(a) Using properties of regular languages, prove that $SCRAMBLE((01)^*)$ is not regular. **[2]**

Ans: Assume that $SCRAMBLE((01)^*)$ is regular. Then

$$SCRAMBLE((01)^*) \cap (0^*1^*) = \{0^n 1^n \,|\, n \geq 0\}$$

will also be regular, which is not true. Hence, $SCRAMBLE((01)^*)$ is not regular.

(b) For $\Sigma = \{a, b, c\}$, show that $SCRAMBLE((abc)^*)$ is not context free. **[2]**

Ans: Assume that $SCRAMBLE((abc)^*)$ is context free. Then,

$$SCRAMBLE((abc)^*) \cap (a^*b^*c^*) = \{a^n b^n c^n \,|\, n \geq 0\}$$

will also be context free (as intersection of context free with regular is context free). However, $\{a^n b^n c^n \,|\, n \geq 0\}$ is context sensitive (can be easily proved by Pumping Lemma for CFLs). Hence, $SCRAMBLE((abc)^*)$ is not context free.

(c) Prove that for $\Sigma = \{0, 1\}$, the SCRAMBLE of any regular language is always context-free.
Ans: This problem uses an interesting property of stack, which is "generating permutations of a given sequence". Consider an input being fed to a system. The system nondeterministically decides whether to push it to a stack, or pass it directly to output. When the system decides to not pass the original input to output directly, it further decides whether to pop something from stack and output or whether to not output anything at all. Take the string input 1100. Possible output permutations are 1001, 0110, 0011,..... These can be generated by push ($\downarrow$) pop ($\uparrow$) and pass $\rightarrow$ sequences like
$\rightarrow\downarrow\rightarrow\rightarrow\uparrow$
$\downarrow\downarrow\rightarrow\uparrow\uparrow\rightarrow$
$\downarrow\downarrow\rightarrow\rightarrow\uparrow\uparrow$
For any regular language L, some DFA M will exist. We modify the machine M with a stack and some new transition rules to create an NPDA as follows.
As long as input has not ended and stack is not empty, make choices from the following,

i. Push input to stack, do not make any state change in M,
ii. Pop stack and with that popped element simulate M, push input in stack,
iii. Simulate M directly with input, no stack manipulation,
iv. Simulate M with popped stack symbol and do not process any input (ϵ-transitions).

Note that as long as some permutation of the original input is accepted by M, this NPDA will accept. Hence, for any $\sigma \in L$, any permutation of σ when given as input to the NPDA will be accepted (due to some branch of computation generating σ on-the-fly in the NPDA). **[6]**

Question 8 (a) Consider the alphabets $\Delta = \{a, b\}$ and $\Sigma = \{0, 1\}$. Construct a DFA for $L = (00 + 01 + 1)^*$. Let $h : \Delta^* \mapsto \Sigma^*$ be a *homomorphism* given by $h(a) = 001$, $h(b) = 110$. Construct the DFA for $h^{-1}(L)$. **[3+3]**
Ans:

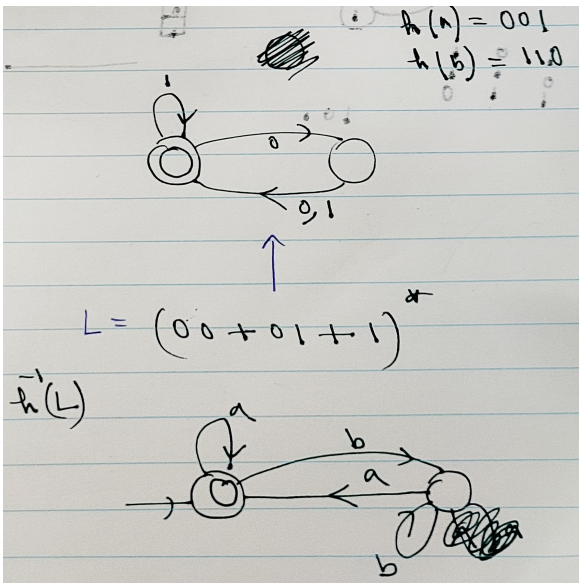

Figure 17: Q8(a)

(b) Prove using Pumping Lemma that $\{a^i b^j c^i d^j \mid i, j \geq 0\}$ is not a CFL. **[4]**
Ans: Let p be the pumping length and consider $z = a^p b^p c^p d^p$. Since $|z| > p$, there are u, v, w, x, y such that $z = uvwxy$, $|vwx| \leq p, |vx| > 0$ and $uv^i wx^i y \in L$, $\forall i \geq 0$. Since $|vwx| \leq p$, v, x cannot contain both a-s and c-s, nor can it contain both b-s and d-s. Further $|vx| > 0$. Now $uv^0 wx^0 y = uwy \notin L$, because it either contains fewer a-s than c-s, or fewer c-s than a-s, or fewer b-s than d-s, or fewer d-s than b-s.

Question 9 (a) Prove that $A_{TM} = \{M \text{ is a Turing Machine that accepts } w\}$ is not decidable. Explain the diagonalization in your argument. **[6]**

(b) Let A and B be two disjoint languages. We say that a language C **separates** A and B if $A \subseteq C$ and $B \subseteq \overline{C}$. Let $\overline{A}$ and $\overline{B}$ be Turing-recognizable. Prove that A and B are separable by some decidable language. **[4]**

Ans:

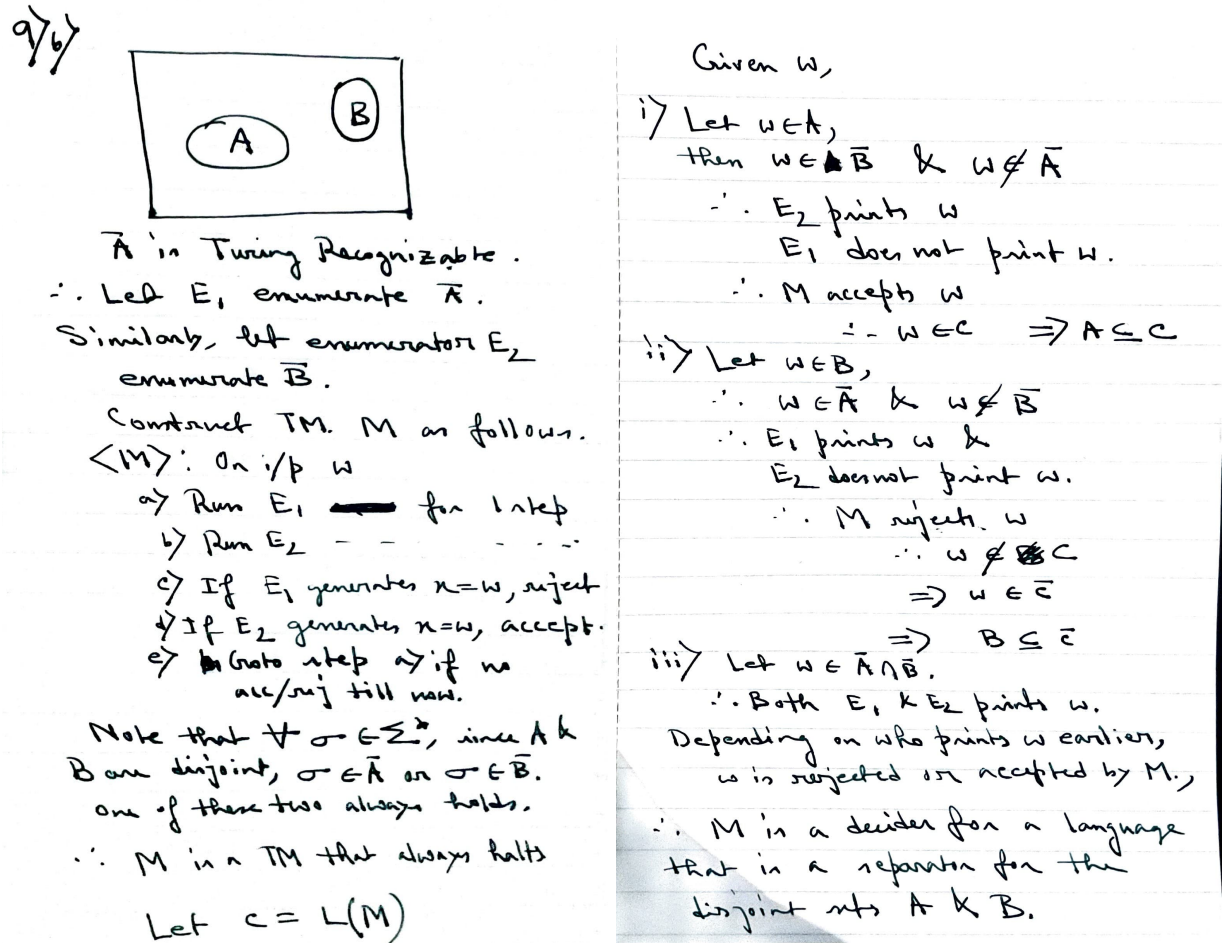

Figure 18: Ans 9(b)

Question 10 (a) Let $X = \{\langle M, w\rangle \mid M \text{ is a single-tape TM that never modifies the portion of the tape that contains the input } w\}$. Show that X is undecidable. (**Hint**: Assume that X has some decider and use it to build a decider for A_{TM}). **[6]**

Ans: We show that X is undecidable by reducing from A_{TM}. Let R be a TM that decides X. We use R to construct a TM S that decides A_{TM}.

TM S: On input $\langle M, w\rangle$,

1. Generate description of TM M_X as follows:
   $\langle M_X\rangle$: On input $\langle M, w\rangle$
   a. Mark the right end of the input with a symbol $\notin$ tape alphabet of M.
   b. Copy w to the part of the tape after the new symbol. Call this part w'
   c. Simulate M on w'
   d. If M accepts, write any character on the first cell of the input tape and accept.
   e. If M rejects, reject.
2. Input $\langle M_X, w\rangle$ to R.
3. If R accepts, accept. Otherwise reject.

Note that M_X is such a machine that writes something on its input (which is M, w) if M is a machine that accepts w. Otherwise, i.e, if M rejects or loops on w, M_X does not write anything on its input. Hence, if R accepts, then M accepts w and very interestingly, if R rejects (it is assumed as a decider), then M rejects/loops on w. So, S basically decides A_{TM}. Note that S always accepts/reject halts by design, as it is generating the description of M_X; it is never running M_X or trying to simulated M on w inside.

Since an A_{TM} decider cannot exist, R does not exist. So, X must be undecidable.

(b) Let $ALL_{DFA} = \{\langle A\rangle \mid \text{A is a DFA and } L(A) = \Sigma^*\}$. Show that ALL_{DFA} is decidable. **[4]**

Ans: Note that the equivalence problem of regular languages is decidable. Hence, the decider X of ALL_{DFA} works as follows.

X: On input A

1. Create an automaton B such that $L(B) = \Sigma^*$. (B has one initial/accept state only with a self-loop for all input symbols.

2. Call the decider EQ_{DFA} on (A, B). If it accepts, accept, otherwise reject.