INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Stamp / Signature of the Invigilator

EXAMINATION (Mid Semester)	**SEMESTER (Spring 2025-2026)**

Roll Number										**Section**		**Name**	

Subject Number	C	S	2	1	2	0	4	**Subject Name**	*FORMAL LANGUAGE AND AUTOMATA THEORY*

Department / Center of the Student		**Additional sheets**	

Important Instructions and Guidelines for Students

1. You must occupy your seat as per the Examination Schedule/Sitting Plan.
2. Do not keep mobile phones or any similar electronic gadgets with you even in the switched off mode.
3. Loose papers, class notes, books or any such materials must not be in your possession, even if they are irrelevant to the subject you are taking examination.
4. Data book, codes, graph papers, relevant standard tables/charts or any other materials are allowed only when instructed by the paper-setter.
5. Use of instrument box, pencil box and non-programmable calculator is allowed during the examination. However, exchange of these items or any other papers (including question papers) is not permitted.
6. Write on both sides of the answer script and do not tear off any page. **Use last page(s) of the answer script for rough work.** Report to the invigilator if the answer script has torn or distorted page(s).
7. It is your responsibility to ensure that you have signed the Attendance Sheet. Keep your Admit Card/Identity Card on the desk for checking by the invigilator.
8. You may leave the examination hall for wash room or for drinking water for a very short period. Record your absence from the Examination Hall in the register provided. Smoking and the consumption of any kind of beverages are strictly prohibited inside the Examination Hall.
9. Do not leave the Examination Hall without submitting your answer script to the invigilator. **In any case, you are not allowed to take away the answer script with you.** After the completion of the examination, do not leave the seat until the invigilators collect all the answer scripts.
10. During the examination, either inside or outside the Examination Hall, gathering information from any kind of sources or exchanging information with others or any such attempt will be treated as **'unfair means'**. Do not adopt unfair means and do not indulge in unseemly behavior.

Violation of any of the above instructions may lead to severe punishment.

Signature of the Student

To be filled in by the examiner

Question Number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Total**
Marks Obtained											

Marks obtained (in words)	**Signature of the Examiner**	**Signature of the Scrutineer**

Indian Institute of Technology Kharagpur
Department of Computer Science and Engineering

Formal Language and Automata Theory (CS21204) **Spring 2025-2026**

21-February-2026 (FN) **Mid-Semester Examination** Maximum Marks: 60

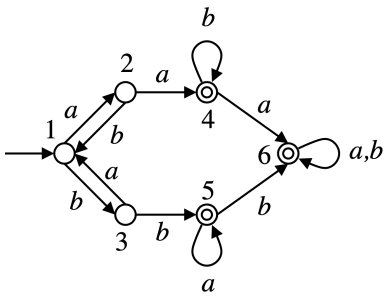
Instructions:

- Write your answers in the question paper itself. Be brief and precise. Answer *all* questions.
- Write the answers only in the respective spaces provided. The last three blank pages may be used for rough work or leftover answers.
- In case you may need more space/pages, please ask for additional sheets in the exam hall and attach the same with this booklet while submitting.
- If you use any algorithm / result / formula covered in the class, just mention it, do not elaborate (unless the same thing has been explicitly asked to answer in the question).

— Question Paper Starts from Next Page —

Q1. [Finite Automata] **15 marks**

(a)



Consider the deterministic finite automaton (DFA) in the left where the six states are numbered as $\{1,2,3,4,5,6\}$ and transitions are given over the alphabet $\Sigma = \{a,b\}$. Use the DFA state-minimization algorithm to convert this DFA to an equivalent DFA with the minimum possible number of states. Show all the steps in details. Also draw the minimum-state DFA or the quotient automaton. **(6)**

Solution:

Initialization: [1 mark]

	1	2	3	4	5	6
1	—					
2	—	—				
3	—	—	—			
4	—	—	—	—		
5	—	—	—	—	—	
6	—	—	—	—	—	—

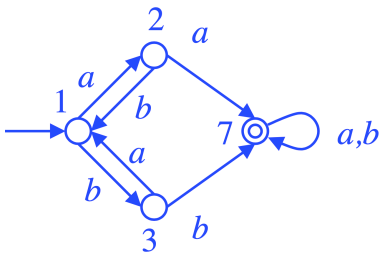
Initial Conflicts: [1 mark]

	1	2	3	4	5	6
1	—					
2	—	—				
3	—	—	—			
4	×	×	×	—		
5	×	×	×	—	—	
6	×	×	×	—	—	—

Iteration / Pass 1: [1 mark]

	1	2	3	4	5	6
1	—					
2	×	—				
3	×	×	—			
4	×	×	×	—		
5	×	×	×	—	—	
6	×	×	×	—	—	—

No further conflicts can be found. We see that the states $4,5,6$ are equivalent. Therefore, we collapse these three states into one, and obtain the quotient automaton as follows: [3 marks]



(b) Let L be the language, $L = \{w \in \{a,b\}^* \mid w$ contains an equal number of occurrences of ab and $ba\}$. For example, $ababa \in L$ (two occurrences of ab, and two of ba), whereas $bbaba \notin L$ (one occurrence of ab, and two of ba). Answer the following questions in parts (i) and (ii).

(i) Give a regular expression for the language L with proper justifications. **(4)**

Solution:

L is the language of the regular expression:

$$[\varepsilon \;+\; aa^*(bb^*aa^*)^* \;+\; bb^*(aa^*bb^*)^*] \quad \text{or} \quad [\varepsilon \;+\; a^+(b^+a^+)^* \;+\; b^+(a^+b^+)^*]$$

Justification:

$\underbrace{\varepsilon}_{\text{empty string has zero } ab \text{ and } ba}$

$+ \quad \underbrace{aa^*(bb^*aa^*)^*}_{\text{strings starting with } ab \text{ and ending with } ba \text{ or strings with only } a\text{'s}}$

$+ \quad \underbrace{bb^*(aa^*bb^*)^*}_{\text{strings starting with } ba \text{ and ending with } ab \text{ or strings with only } b\text{'s}}$

An Alternative:

L is the language of the regular expression:

$$[\varepsilon \;+\; a \;+\; b \;+\; a(a+b)^*a \;+\; b(a+b)^*b]$$

Justification:

$\underbrace{\varepsilon + a + b}_{\text{empty string or single symbol has zero } ab \text{ and } ba}$

$+ \quad \underbrace{a(a+b)^*a}_{\text{all valid strings starting with } a \text{ and ending with } a}$

$+ \quad \underbrace{b(a+b)^*b}_{\text{all valid strings starting with } b \text{ and ending with } b}$

Another Alternative:

L is the language of the regular expression:

$$[\varepsilon \;+\; a(bb^*a+a)^* \;+\; b(aa^*b+b)^*]$$

... many more variants possible!

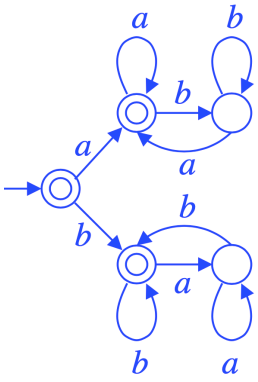
Deductions:

- $\frac{1}{2}$ mark for missing ε,
- 1 mark for missing a and b, and
- 2-3 marks for some/many string miss or allowing wrong strings

(ii) Design a minimum-state deterministic finite automaton (DFA) to accept L. Show only the state-transition diagram, not its mathematical definition. **(5)**

Solution:

The following minimum-state DFA accepts L.



Deductions:

- $\frac{1}{2}$ mark for missing initial state mark,
- 1 mark for missing final state,
- 2 marks for minor state/transition errors, and
- 3 marks for not producing min-state DFA

Q2. [Regular Languages] **15 marks**

For the following problems in parts (a), (b) and (c) under this question, we define the languages L over alphabet $\Sigma = \{0,1\}$. Prove or disprove whether $L \in \mathsf{RL}$. *No credits will be given without justifications.*

Note: RL denotes the class of regular languages. For proving $L \in \mathsf{RL}$, you must present a Regular-Expression / DFA / NFA / ε-NFA (any one of these) that accepts L. However, to disprove (i.e., proving $L \notin \mathsf{RL}$), you must use pumping lemma or closure properties for regular languages to contradict.

(a) $L = \left\{0^k w 1^k \;\middle|\; k \geq 1, w \in \{0,1\}^*\right\}$. **(5)**

Solution:

We prove that L is regular by providing a regular expression.

We observe that if $k = 1$, then L contains all strings that start with 0 and end with 1 (note that w includes all characters between the starting and ending characters; this is possible since $w \in \{0,1\}^*$). On the other hand, for any string $u = 0^k w 1^k \in L$, since $k \geq 1$, u must start with 0 and end with 1.

Therefore, L is the set of all strings that start with 0 and end with 1. Due to this observation, $L = \mathscr{L}\left(0(0+1)^*1\right)$, so L is regular, i.e., $L \in \mathsf{RL}$.

(b) $L = \{0^k 1 w 1^k \mid k \geq 1, w \in \{0,1\}^*\}$. **(5)**

Solution:

We prove that L is not regular, using the pumping lemma for regular languages.

Suppose that L is regular. Let p be the pumping-lemma constant for L. We choose the string $v = 0^p 1 1^p$, and it is clear that $v \in L$ and $|v| = 2p+1 \geq p$. For all partitions $v = xyz$, due to the conditions $|xy| \leq p$ and $|y| > 0$, we see that y must consist of at least one 0 and is composed only of 0s, or in other words, $y = 0^k$, $0 < k \leq p$. We then consider the pumped-up string $xy^2z = 0^{p+k}11^p$ and see that $xy^2z \notin L$ leading to the contradiction![1]

Therefore, L does not satisfy the pumping lemma for regular languages, so L is non-regular, i.e., $L \notin \mathsf{RL}$.

[1] By pumping up, we see the difference between this language L and the previous language $\{0^k w 1^k \mid k \geq 1,\ w \in \{0,1\}^*\}$: the fixed 1 allows for the distinction between the substring of the first occurrence of 0^k and w, which in turn allows us to pump up the string v in our solution.

(c) $L = \{ww \mid w \in \{0,1\}^* \text{ and } w \text{ contains at least one 0 and at least one 1}\}$. **(5)**

Solution:

Define the following language: $A = \{ww \mid w \in \{0,1\}^* \text{ and } w \text{ does not contain both a 0 and a 1}\}$.

We see that $A = \mathscr{L}\left((00)^* + (11)^*\right)$, as every string in A must have even length (due every string in A having form ww) and must be composed of only 0s or only 1s (in order to not contain both a 0 and a 1).

We then observe that $L \cup A = B$, where $B = \{ww \mid w \in \{0,1\}^*\}$.

Since A can be expressed as a regular expression, A is regular. Now assume, for the sake of contradiction, that L is also regular. Since the class of regular languages is closed under the union operation, if L and A are both regular, then B must also be regular.

However, B is not context-free and therefore not regular.[2]

Since B is not regular, then due to closure properties, it cannot be the case that both L and A are regular, so L must be non-regular, i.e., $L \notin \mathsf{RL}$.

[2] Since it is true that if a language is regular, it is also context-free; the contrapositive must be true which says that if a language is not context-free, then it is also not regular. It was proved in class that B is not context-free, so automatically B is not regular by the above argument.

Q3. [Context-free Grammars and Derivations] **8 marks**

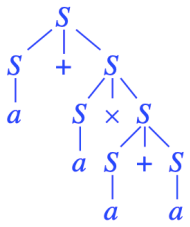
Consider the following context-free grammar (CFG) $G = (V, \Sigma, P, S)$ with $V = \{S\}, \Sigma = \{a, +, \times\}$ to generate arithmetic expressions in one variable a involving addition and multiplication operations only. Here, S is the start symbol and P is the set of following productions/rules:

$$S \rightarrow a \mid S+S \mid S \times S$$

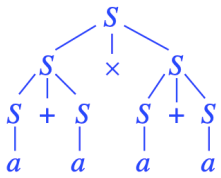
(a) Draw all possible derivation (parse) trees for the string $a+a \times a+a$ following this grammar. **(5)**

Solution:

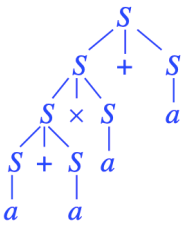
There are five derivation trees for $a+a \times a+a$. These trees are shown below. They respectively stand for the following interpretations of the given expression: $a+(a \times (a+a))$, $(a+a) \times (a+a)$, $((a+a) \times a)+a$, $a+((a \times a)+a)$ and $(a+(a \times a))+a$.



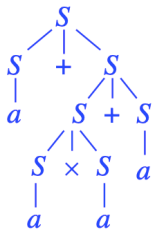
(a) First tree



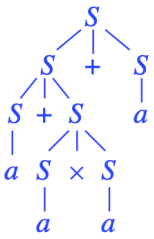
(b) Second tree



(c) Third tree



(d) Fourth tree



(e) Fifth tree

[1 mark for each tree]

(b) Design an *unambiguous* CFG to generate the same language. Force the operations to be evaluated from left to right. This means that $+$ and $\times$ are given the same precedence and left-to-right associativity. For example, $a+a \times a+a$ is to be interpreted as $((a+a) \times a)+a$. **(3)**

Solution:

The following productions form the unambiguous grammar that generates the same language, but forces left-to-right evaluation of the operations.

$$S \rightarrow a \mid S+a \mid S \times a$$

[1 mark corresponding to each rule (roughly)]

Q4. [Context-free Languages] **14 marks**

For the following problems in parts (a) and (b) under this question, we define languages L over alphabet $\Sigma = \{a, b, c\}$. Prove or disprove whether $L \in \mathsf{CFL}$. *No credits will be given without justifications.*

Note: CFL denotes the class of context-free languages. For proving $L \in \mathsf{CFL}$, you must present a context-free grammar (CFG) and a push-down automaton (PDA) (both are asked here) accepting L. To disprove (i.e., to prove $L \notin \mathsf{CFL}$), you must use pumping lemma for context-free languages.

(a) $L = \left\{ a^i b^j c^j \;\middle|\; i, j \geq 0 \text{ and } i \geq j \right\}$. **(7)**

Solution:

We prove that L is not context-free, using the pumping lemma for CFL's. Suppose that L is context-free. Let k be a pumping-lemma constant for L. By definition of L, the string $\beta = a^k b^k c^k \in L$ and is of length $\geq k$. The pumping lemma gives a decomposition $\beta = \beta_1\beta_2\beta_3\beta_4\beta_5$ such that

(i) $|\beta_2\beta_4| > 0$,

(ii) $|\beta_2\beta_3\beta_4| > 0$, and

(iii) $\beta_1\beta_2^i\beta_3\beta_4^i\beta_5 \in L$ for all $i \geq 0$.

We consider all possible cases for β_2 and β_4, and show that in each case, there is a contradiction. Condition (ii) implies that β_2 and β_4 belong either to the same block or to two adjacent blocks.

Case 1: β_2 or β_4 (or both) contain different symbols. In this case,$\beta_1\beta_2^2\beta_3\beta_4^2\beta_5$ is not of the form $a^*b^*c^*$.

Case 2: Both β_2 or β_4 belong to the block of a's. In this case, $\beta_1\beta_3\beta_5$ contains less a's than b's or c's.

Case 3: Both β_2 or β_4 belong to the block of b's. In this case, $\beta_1\beta_2^2\beta_3\beta_4^2\beta_5$ contains more b's than a's.

Case 4: Both β_2 or β_4 belong to the block of c's. In this case, $\beta_1\beta_2^2\beta_3\beta_4^2\beta_5$ contains more c's than a's.

Case 5: β_2 belongs to the block of a's, and β_4 belongs to the block of b's. If $\beta_4 \neq \varepsilon$, consider the string $\beta_1\beta_2^2\beta_3\beta_4^2\beta_5$ which contains more b's than c's. If $\beta_4 = \varepsilon$, then $\beta_2 \neq \varepsilon$, and $\beta_1\beta_3\beta_5$ contains less a's than c's.

Case 6: β_2 belongs to the block of b's, and β_4 belongs to the block of c's. In this case, $\beta_1\beta_2^2\beta_3\beta_4^2\beta_5$ contains more b's or c's (or both) than a's.

Hence, all the above cases lead to contradictions!

(b) $L = \{a^i (bc)^j \mid i, j \geq 0 \text{ and } i \geq j\}$. **(7)**

Solution:

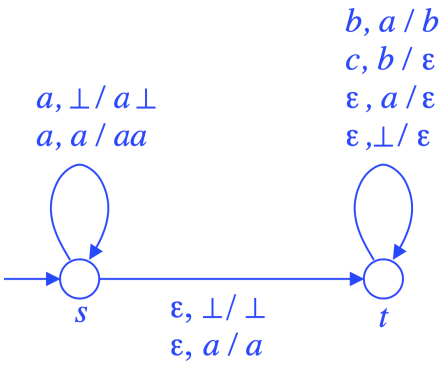
The following grammar generates L. [3 marks]

$$
\begin{array}{lcll}
S & \rightarrow & UV & [S \text{ is the start symbol}] \\
U & \rightarrow & \varepsilon \mid aU & [U \text{ generates the excess of } a\text{'s over } (bc)\text{'s}] \\
V & \rightarrow & \varepsilon \mid aVbc & [V \text{ generates equal number of } a\text{'s and } (bc)\text{'s}]
\end{array}
$$

A single start symbol S itself can generate L as follows:

$$S \rightarrow \varepsilon \mid aS \mid aSbc.$$

We can design a PDA $M = (Q, \Sigma, \Gamma, \delta, s, \perp, F)$ to accept L. M contains two states, $Q = \{s, t\}$ with the start state s and no final states ($F = \phi$). The machine accepts by empty stack. The input alphabets are, $\Sigma = \{a, b, c\}$ and the tape alphabets are, $\Gamma = \{a, b, \perp\}$. The transitions are described in the following figure. [4 marks]



In the start state s, the machine reads the initial block of a's. For every a read from the input, an a is pushed to the stack. M subsequently makes an ε-transition to the state t to read the block of (bc)'s, that follows the block of a's. Only if a matching a is found at the top of the stack, the reading of one occurrence of bc is initiated. The a at the top of the stack is replaced by b to indicate that the reading of bc is only half-way through. Only if a c is available at the input at this stage, this c is consumed, and the intermediate marker b is popped out of the stack exposing the next a to be matched against the next occurrence of bc.

When an input of L is fully read by M, the stack contains $(i - j)$ occurrences of a and the bottom marker $\perp$. M uses the transitions $[\varepsilon, a/\varepsilon]$ and $[\varepsilon, \perp/\varepsilon]$ against the loop at the state t, in order to pop the excess a's and the bottom marker $\perp$. If M uses the transition $[\varepsilon, a/\varepsilon]$ more than $(i - j)$ times before reading all the j occurrences of bc, the machine gets stuck before reading the entire input.

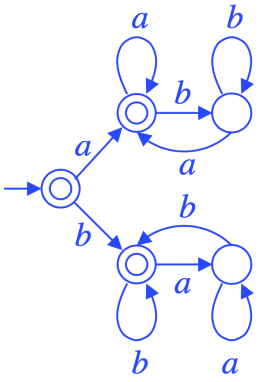
Q5. [Regular Grammars] **8 marks**

Consider (again) the language, $L = \{w \in \{a,b\}^* \mid w$ contains an equal number of occurrences of ab and $ba\}$, introduced in **Q1(b)**. Answer the following questions. *You may use your DFA presented in* **Q1b(ii)**.

(a) Construct a *right-linear grammar* which can generate L. **(4)**

Solution:

Let us use the DFA presented in the solution of **Q1b(ii)** to create the right-linear grammar (again shown below for convenience).



The following right-linear grammar, $G_1 = (N_1, \Sigma, P_1, S_0)$, where the non-terminal symbols are $N_1 = \{S_0, S_1, S_2, S_3, S_4\}$, the alphabet (terminal symbols) are $\Sigma = \{a,b\}$, the start symbol is S_0 and the productions P_1 are denoted as follows:

$$
\begin{aligned}
S_0 &\rightarrow aS_1 \mid bS_2 \mid a \mid b \mid \varepsilon \\
S_1 &\rightarrow aS_1 \mid bS_3 \mid a \\
S_2 &\rightarrow bS_2 \mid aS_4 \mid b \\
S_3 &\rightarrow aS_1 \mid bS_3 \mid a \\
S_4 &\rightarrow bS_2 \mid aS_4 \mid b
\end{aligned}
$$

This right-linear grammar G_1 can generate the language L.

(b) Construct a *left-linear grammar* which can generate L. **(4)**

Solution:

Let us first modify the DFA presented in the solution of **Q1b(ii)** to create an NFA as follows:

(i) Make all final states as initial states and the (only) initial state as the (only) final state.

(ii) Reverse all the transition directions.

In this newly formed NFA, we can derive the following right-linear grammar, $G_2' = (N_2, \Sigma, P_2', T_0)$, where the non-terminal symbols are $N_2 = \{T_0, T_1, T_2, T_3, T_4\}$, the alphabet (terminal symbols) are $\Sigma = \{a, b\}$, the start symbol is T_0 and the productions P_2' are denoted as follows:

$$
\begin{aligned}
T_0 &\rightarrow T_1 \mid T_2 \mid \varepsilon \\
T_1 &\rightarrow aT_0 \mid aT_1 \mid aT_3 \mid a \\
T_2 &\rightarrow bT_0 \mid bT_2 \mid bT_4 \mid b \\
T_3 &\rightarrow bT_1 \mid bT_3 \\
T_4 &\rightarrow aT_2 \mid aT_4
\end{aligned}
$$

Now, just flipping the right-side pair of the productions in P_2' will result the required left-linear grammar, $G_2 = (N_2, \Sigma, P_2, T_0)$, where the productions P_2 becomes:

$$
\begin{aligned}
T_0 &\rightarrow T_1 \mid T_2 \mid \varepsilon \\
T_1 &\rightarrow T_0a \mid T_1a \mid T_3a \mid a \\
T_2 &\rightarrow T_0b \mid T_2b \mid T_4b \mid b \\
T_3 &\rightarrow T_1b \mid T_3b \\
T_4 &\rightarrow T_2a \mid T_4a
\end{aligned}
$$

This left-linear grammar G_2 can generate the language L.

— Question Paper Ends Here —