Final Exam Solutions

Automata Theory Monsoon 2025, IIIT Hyderabad

October 1, 2025

1. [3 points] [Q1] each part given equal weightage (1.5) - correct solution for part 1 - is lexicographic - Binary is wrong due to many to one mapping - 01 and 1

   - correct solution to part 2 - is cantors diagonalization - solutions using power set given 0.5 - solution using real is 1 - Question had asked for proof via bijection

2. [1+1 points] [Q2]*A grammar is defined as ambiguous if a string in its language has two or more distinct parse trees or, equivalently, two or more distinct leftmost derivations.*
   The language $L = \{w \in \{0,1\}^* \mid w \text{ has an equal number of 0s and 1s}\}$ can be generated by the following Context-Free Grammar (CFG):

   $$G : S \to SS \mid 0S1 \mid 1S0 \mid \epsilon$$

   This grammar is ambiguous. Your task is to prove this.

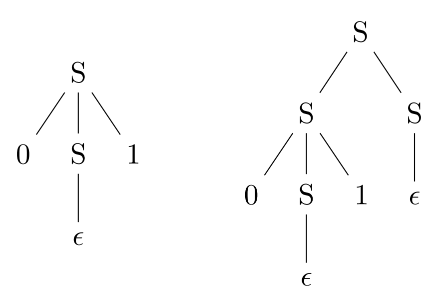
   (i) Find the shortest non-empty string $w \in L(G)$ that demonstrates the ambiguity of the grammar.

   (ii) Formally prove that the grammar G is ambiguous by providing two distinct parse trees for the string w you identified in part (a).

Solution:

(i) The shortest non-empty strings that G can generate are of length 2 – 01 and 10. These also happen to have multiple parse trees and hence can demonstrate the ambiguity of the grammar.

(ii) Both 01 and 10 have multiple parse trees. Here are two parse trees for 01 as an example:



Marking scheme

(i)
- Choosing either 01 or 10 as your string $\rightarrow 1$
- Choosing any other string $\rightarrow 0$

(ii)
- Providing two VALID and DISTINCT parse trees (or leftmost/rightmost derivations) for the chosen string $\rightarrow 1$
- Providing only one VALID parse tree (or leftmost/rightmost derivation) for the chosen string $\rightarrow 0.5$
- Providing invalid parse trees (or derivations) $\rightarrow 0$

Writing down one leftmost and one rightmost derivation will still fetch you only 0.5 points. Points were awarded if your parse trees proved ambiguity regardless of whether you chose the shortest string or not.

3. [4 points] [Q3]

Solution:

- 4 marks – the answer is complete, handles the case using the operations of TM while not oversimplifying it.
- 3 marks – NTM but the state definition and the split logic is oversimplified
- 2 marks – split by half is written and count of length is written without explaining in TM operations
- 1 mark – written code that explains the algorithm with no reference to TM
- 0 mark - completely incorrect

4. [2 points] [Q4] Describe the language generated by this grammar:

- $S \longrightarrow aS \mid bX$
- $X \longrightarrow aX \mid bY$
- $Y \longrightarrow aY \mid bZ$
- $Z \longrightarrow aZ \mid \epsilon$

Solution:

- All strings made up of a and b, with exactly 3b's
- 2 marks is said directly exact 3bs or drew the correct dfa/nfa/regex
- 1 mark if said atleast 3bs with no correct diagram or other close but incorrect answers

- 0.5 for describing grammar instead of language and leaving it there
- 0 otherwise

5. [4 points] [Q5]

Solution:

- Claim RegularTM is undecidable.
- Reduction: Reduce from $A_{TM} = \langle M, w \rangle \mid M$ accepts w.
- Construction: Given $\langle M, w \rangle$, build a TM N that on input x: • If x is not of the form 0^n1^n, accept. • If x is of the form 0^n1^n, simulate M on w; accept iff that simulation accepts.
- Correctness: • If M accepts w: N accepts every string, so $L(N) = \Sigma^*$ (which is regular). • If M does not accept w: N accepts exactly $\Sigma^* \setminus 0^n1^n$. Since 0^n1^n is non-regular and regular languages are closed under complement, $\Sigma^* \setminus 0^n1^n$ is non-regular. So $L(N)$ is non-regular.
- Conclusion: A decider for RegularTM would decide A_{TM}, which is a contradiction. Hence RegularTM is undecidable.

6. [3 + 3 points] The language L consisting of all strings having an equal number of 0's and 1's is context-free. State whether the following languages are regular or context-free. Draw the corresponding automaton (DFA/PDA) to support your answer.

   (i) $L_1 = \{w| \ |\#0's - \#1's| \leq 1\}$
   (ii) $L_2 = \{w| \ |\#0's - \#1's| \leq 1, \forall$ prefixes of $w\}$

   Note: The string 00001111 will belong to L_1 but not L_2 since 00001, a prefix of 00001111, does not satisfy $|\#0's - \#1's| \leq 1$.

Solution: The correct grammar for part 1 is

- $S \longrightarrow X0X \mid X1X \mid X$
- $X \longrightarrow XX \mid 0X1 \mid 1X0 \mid \epsilon$

1. For stating the correct language - 0.5 mark
2. For specifying only grammar in part 1 - 1 mark
3. For drawing the correct pda in part 1 - 2.5 marks
4. For drawing the correct DFA in part 2 - 2.5 marks

7. [5 points] Oblivious Turing Machine Question: Refer to this link: [LINK]. Question has been graded very leniently. Please read this before approaching.

8. [4 points] Let

$$\Sigma_3 = \left\{ \begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \ldots, \begin{bmatrix}1\\1\\1\end{bmatrix} \right\}$$

Σ_3 contains all size 3 columns of 0s and 1s. A string of symbols in Σ_3 gives three rows of 0s and 1s. Consider each row to be a binary number and let

$$B = \{w \in \Sigma_3^* \mid \textit{the bottom row of } w \textit{ is the sum of the top two rows}\}.$$

For example, $\begin{bmatrix}0\\0\\1\end{bmatrix}\begin{bmatrix}1\\0\\0\end{bmatrix}\begin{bmatrix}1\\1\\0\end{bmatrix} \in B$ but $\begin{bmatrix}0\\0\\1\end{bmatrix}\begin{bmatrix}1\\0\\1\end{bmatrix} \notin B$

Show that B is regular. **[CO1, CO2, CO3, CO4]**

Hint: Working with B^R is easier. Here L^R is defined as: $L^R = \{w^R \mid \forall\ w \in L\}$, where w^R is the string w, but in reverse. You can assume that if A^R is regular, then A is also regular.

Solution:

- For correct DFA/NFA - 4 marks
- For DFA with missing transitions - maximum 2 marks if logic is correct
- For DFA/NFA that accept wrong strings - maximum 1 mark if logic is correct
- If the alphabet of DFA/NFA is {0,1} - 0 marks

9. [4 points] [Q9]

Solution:

- Formulating a TM to simulate C(x) - 1 mark
- Recognizing that it will never reject (infinite loop) - 1 mark
- reducing this to Atm - 1 mark
- recognizing that to prove collatz conjecture, we will have infinite inputs - 1 mark
- reducing this to Atm/the machine above - 1 mark
- Other approches have also been given marks if found correct.

10. [4 points] Bonus 1

Solution: Proof idea: introduce a new character a' for all characters a', and update according to the transition function. If a is read then that cell has already been written once. When we need to write more than twice then copy the tape contents after a set delimiter.

Bonus 2

Solution:

L_k is regular,we provide the following NFA to prove it.

Set of States

$Q = Q_A * Q_B * 0, 1, .., n$, where Q_A and Q_B are the sets of states of DFAs for A and B respectively, and n is the number of bits in the binary representation of k.

Initial State

$q_0 = q_{A0}, q_{B0}, 0$ where q_{A0} and q_{B0} are the initial states of A and B respectively, and we start reading the first bit of k.

Final States

$F = (q_A, q_B, n) | q_A \epsilon F_A, q_B, \epsilon F_B$, where F_A and F_B are the sets of final states of A and B respectively.

Transition Function ρ

For each state (q_A, q_B, i) and input symbol $w \epsilon 0, 1$, does ρ as follows:

$$\rho((q_A, q_B, i), w) = (\rho_A(q_A, k_i), \rho_B(q_B, k_i \oplus w), i + 1)$$

Here, k_i, represents i-th bit of the binary representation of k.

Given that the states are now represented as (q_A, q_B, i), here are the transition cases:

For $w = 0$ and $k = 0$:

$$\rho((q_A, q_B, i), 0) = (\rho_A(q_A, 0), \rho_B(q_B, 0), i + 1), (\rho_A(q_A, 1), \rho_B(q_B, 1), i + 1)$$

For $w = 1$ and $k = 0$:

$$\rho((q_A, q_B, i), 1) = (\rho_A(q_A, 0), \rho_B(q_B, 1), i + 1), (\rho_A(q_A, 1), \rho_B(q_B, 0), i + 1)$$

For $w = 0$ and $k = 1$:

$$\rho((q_A, q_B, i), 0) = (\rho_A(q_A, 0), \rho_B(q_B, 1), i + 1), (\rho_A(q_A, 1), \rho_B(q_B, 0), i + 1)$$

For $w = 1$ and $k = 1$:

$$\rho((q_A, q_B, i), 1) = (\rho_A(q_A, 0), \rho_B(q_B, 0), i + 1), (\rho_A(q_A, 1), \rho_B(q_B, 1), i + 1)$$

Here, i represents the index of the binary representation of k, and ρ_A and ρ_B are the transition functions for the DFAs of A and B respectively.