Indian Institute of Technology Roorkee

CSN-353 Theory of Computation | End Semester Exam

Total Marks: 50 | **Time: 3 Hours**

True/False Questions (10 Marks)

1. $L = \{\alpha\beta\alpha\gamma \mid \alpha, \beta, \gamma \in \Sigma^*, \alpha = \epsilon, |\beta| = |\gamma|\}$ is a Context-Free Language.

   Solution. **True.** □

2. Let L be a context-free language (CFL), $x \in L$, and a proper prefix of x is also in L. L cannot be accepted by a deterministic pushdown automaton (DPDA) in empty stack mode.

   Solution. **True.** □

3. If L is a context-free language (CFL) and $x \in L$ with $|x| \geq p$, where p is the pumping constant, then the number of strings in L is infinite.

   Solution. **True.** □

4. If L_1 and L_2 are recognized by Turing machines (TMs) M_1 and M_2, then there exists a TM that recognizes L_1L_2.

   Solution. **True.** □

5. Given a grammar G of length n, we can find an equivalent Chomsky-Normal-Form grammar for G in time $O(n)$ and the resulting grammar has length $O(n)$.

   Solution. **False.** □

6. Neither the language TOTAL $= \{M \mid M$ halts on all inputs$\}$ nor its complement is recursively enumerable.

   Solution. **True.** □

7. The class of recursively enumerable sets is closed under union and intersection.

   Solution. **True.** □

8. A multi-tape Turing Machine can recognize a language that no single tape TM can recognize.

Solution. **False.** □

9. There exists a Language L for which there is an NDTM M to accept it, but there is no DTM to accept the same language L.

Solution. **False.** □

10. A context-free grammar is said to be linear if, in each production rule, at most, one non-terminal occurs on the right-hand side.

Solution. **True.** □

Multiple Choice Questions (20 Marks)

1. Consider the symmetric difference of two languages A and B (over the same alphabet), denoted by $A \triangle B$. Which of the following statements is/are **TRUE**?

   (a) If A and B are both context-free languages (CFLs), then $A \triangle B$ must be a CFL.

   (b) If A is a CFL and B is not a CFL, then $A \triangle B$ must be a CFL.

   (c) If A is a CFL and B is regular, then $A \triangle B$ must be a CFL.

   (d) $\boxed{\text{If } A \text{ and } B \text{ are regular languages, then } A \triangle B \text{ is always context-free.}}$

2. Consider the languages:

$$L_1 = \{a^m b^m c^{m+n} \mid m, n > 1\},$$
$$L_2 = \{a^m b^n c^{m+n} \mid m, n > 1\}.$$

   Which of the following statements is **TRUE**?

   (a) Both L_1 and L_2 are context-free languages (CFLs).

   (b) Neither L_1 nor L_2 is a context-free language.

   (c) $\boxed{L_1 \text{ is not a CFL, but } L_2 \text{ is a CFL.}}$

   (d) L_1 is a CFL, but L_2 is not a CFL.

3. Consider the two grammars G and G' with the start symbols S and S', and with the following productions:

   - Productions of G:

$$S \to aS \mid B, \quad B \to bB \mid b.$$

- Productions of G':

$$S' \to aA' \mid bB', \quad A' \to aA' \mid B', \quad B' \to bB' \mid \epsilon.$$

Which of the following statements is **TRUE**?

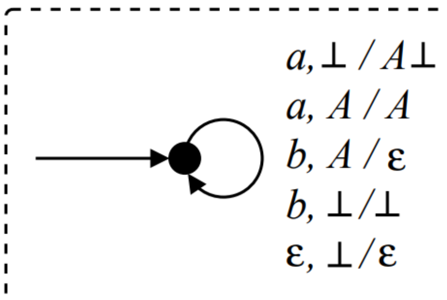
(a) $L(G) = L(G')$.

(b) $L(G)$ is strictly contained in $L(G')$.

(c) $L(G')$ is strictly contained in $L(G)$.

(d) Neither $L(G)$ is contained in $L(G')$ nor $L(G')$ is contained in $L(G)$.

4. What is the language over the alphabet $\{a, b\}$ that is accepted by the following PDA? The PDA accepts by empty stack. Here, $\perp$ is the initial bottom marker for the stack.



(a) $\{a^n b^n \mid n > 0\}$

(b) $\{a^m b^n \mid m, n \geq 0\}$

(c) $\{a^m b^n \mid m, n \geq 1\}$

(d) $L\{(a + b)^* b\}$

5. Let Σ_1 and Σ_2 be disjoint alphabets, $\Sigma = \Sigma_1 \cup \Sigma_2$, and $L \subseteq \Sigma^*$. Denote by L_1 the language over Σ_1 obtained by deleting all symbols of Σ_2 from the strings in L. Likewise, let L_2 denote the language over Σ_2 obtained by deleting all symbols of Σ_1 from the strings in L.

For example, if $\Sigma_1 = \{a\}$, $\Sigma_2 = \{b\}$, and $L = \{abab^2ab^3...ab^n, \mid n \geq 1\}$, then we have:

$$L_1 = \{a^n \mid n \geq 1\}, \quad L_2 = \{b^{n(n+1)/2} \mid n \geq 1\}.$$

Which of the following statements is/are **FALSE**?

(a) If L is a DCFL, then both L_1 and L_2 must be DCFL.

(b) If both L_1 and L_2 are DCFL, then L must be a DCFL.

(c) If L_1 is a regular language and L_2 is a DCFL, then L must be a DCFL.

(d) If L is a regular language, then both L_1 and L_2 must be regular languages.

6. Let M be a Turing machine over the alphabet Σ with $L(M) = L$. Let M' be the Turing machine obtained from M by swapping the roles played by the accept and reject states of M. Finally, let $L' = L(M')$, and $\sim L$ denote the complement of L (in Σ^*).

   Which of the following statements is/are always **TRUE**?

   (a) $L' =\sim L$

   (b) $L' \neq\sim L$

   (c) $L' \subseteq\sim L$

   (d) $\sim L \subseteq L'$

7. Which of the following statements about multi-tape Turing machines is **TRUE**?

   (a) Multi-tape Turing machines can recognize a strictly larger class of languages than single-tape Turing machines.

   (b) Every multi-tape Turing machine can be simulated by a single-tape Turing machine with only a quadratic increase in time complexity.

   (c) Multi-tape Turing machines require exponentially more states than single-tape Turing machines to recognize the same language.

   (d) The language classes recognized by single-tape and multi-tape Turing machines are fundamentally different.

8. Which of the following statements is/are **FALSE**?

   (a) For every non-deterministic Turing machine, there exists an equivalent deterministic Turing machine.

   (b) Turing recognizable languages are closed under union and complementation.

   (c) Turing decidable languages are closed under intersection and complementation.

   (d) Turing recognizable languages are closed under union and intersection.

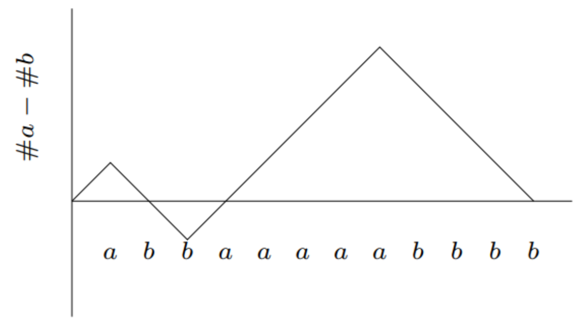
9. The graph below shows the value $\#a - \#b$ plotted against prefixes of a word $x \in \{a, b\}^*$. Analyze the graph carefully and identify the language represented by it.

   (a) $L = \{x \in \{a, b\}^* \mid \#a(x) > \#b(x)\}$

   (b) $L = \{x \in \{a, b\}^* \mid \#a(x) < \#b(x)\}$

   (c) $L = \{x \in \{a, b\}^* \mid \#a(x) = \#b(x)\}$

   (d) $L = \{x \in \{a, b\}^* \mid \#a(x) + \#b(x) \text{ is even}\}$



10. What language is generated by the unrestricted grammar $G = (\{S, B, a, b, c\}, \{a, b, c\}, R, S)$, where R consists of the following productions?

$$S \to aBSccc \mid aBccc$$

$$Ba \to aB, \quad Bc \to bbc, \quad Bb \to bbb$$

(a) $\{a^n b^{3n} c^{3n} \mid n \geq 0\}$

(b) $\{a^n b^{2n} c^{3n} \mid n \geq 0\}$

(c) $\{a^n b^n c^n \mid n > 0\}$

(d) $\boxed{\{a^n b^{2n} c^{3n} \mid n > 0\}}$

1. (a) Define a Turing Machine formally. **[2]**

   (b) Explain how a multitape Turing Machine can be simulated using a single-tape Turing Machine. **[3]**

Solution.

A probable outline of the solution:

(a) A Turing Machine (TM) is defined as a 7-tuple:

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}}),$$

where:

- Q: A finite set of states.
- Σ: The input alphabet (does not include the blank symbol $\sqcup$).
- Γ: The tape alphabet ($\Sigma \subseteq \Gamma$, and $\sqcup \in \Gamma$).
- δ: The transition function $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$, where L and R indicate moving the tape head left or right, respectively.
- q_0: The start state ($q_0 \in Q$).
- q_{accept}: The accept state ($q_{\text{accept}} \in Q$).
- q_{reject}: The reject state ($q_{\text{reject}} \in Q$), where $q_{\text{reject}} \neq q_{\text{accept}}$.

A Turing Machine operates by reading the input from the tape, modifying the tape according to the transition function, and moving the tape head until it reaches either q_{accept} or q_{reject}.

(b) A multitape Turing Machine (MTM) can be simulated by a single-tape Turing Machine (STM) as follows:

- **Encoding the Tapes:** Represent the contents of all k tapes of the MTM on a single tape of the STM. Use special delimiter symbols (e.g., #) to separate the contents of each tape. For example, if there are 3 tapes with contents $aabb$, bba, and ab, the single tape representation could be:

  $$\#aabb\#bba\#ab\#$$

- **Simulating Tape Heads:** Use markers or pointers to keep track of the positions of the tape heads for each of the k tapes. For example, you can encode the tape head position by placing a special symbol or annotation near the respective character.
- **Simulating Transitions:** The STM simulates the MTM by:
  i. Scanning the entire single tape to read the symbols under the virtual tape heads for each tape.
  ii. Applying the transition function of the MTM based on the current state and the symbols read.
  iii. Updating the symbols under the virtual tape heads and moving the virtual tape heads left or right by scanning and modifying the tape as necessary.
- **Time Complexity:** The STM requires extra steps to simulate the k-tape MTM because it must scan the single tape to locate the positions of the virtual tape heads. This increases the time complexity by a quadratic factor, resulting in an overall time complexity of $O(T^2)$, where T is the runtime of the MTM.

□

2. Consider the language $L = \{a^n b^{n^2} \mid n \geq 0\}$. Use the Pumping Lemma for CFLs to determine whether L is a context-free language or not. Clearly explain your assumptions. **[5]**

Solution.
A probable outline of the solution:

Answer: $L = \{a^n b^{n^2} \mid n \geq 0\}$ is **not** a context-free language.

Explanation:
We will use the Pumping Lemma for context-free languages (CFLs) to prove that L is not context-free. The Pumping Lemma states:
If L is a context-free language, then there exists a pumping length $p \geq 1$ such that any string $z \in L$ with $|z| \geq p$ can be split into $z = uvwxy$ such that:

(a) vwx has a length $|vwx| \leq p$.

(b) $vx \neq \epsilon$ (at least one of v or x is non-empty).

(c) For all $i \geq 0$, the string $uv^iwx^iy \in L$.

We will attempt to show that L does not satisfy these conditions.

Assumptions: - Let p be the pumping length guaranteed by the Pumping Lemma. - Consider the string $z = a^p b^{p^2} \in L$, where $n = p$.

Splitting $z = uvwxy$: - The substring vwx has $|vwx| \leq p$, so vwx consists of either: - Only a's, - Only b's, or - A combination of a's and b's.

Case Analysis:

(a) Case 1: vwx contains only a's. - Pumping v and x changes the number of a's in z. - For $i \neq 1$, the number of a's in uv^iwx^iy is no longer p, but the number of b's remains p^2. - Hence, $uv^iwx^iy \notin L$, as the number of a's does not match the square root of the number of b's.

(b) Case 2: vwx contains only b's. - Pumping v and x changes the number of b's in z. - For $i \neq 1$, the number of b's in uv^iwx^iy is no longer p^2, but the number of a's remains p. - Hence, $uv^iwx^iy \notin L$, as the number of b's is no longer the square of the number of a's.

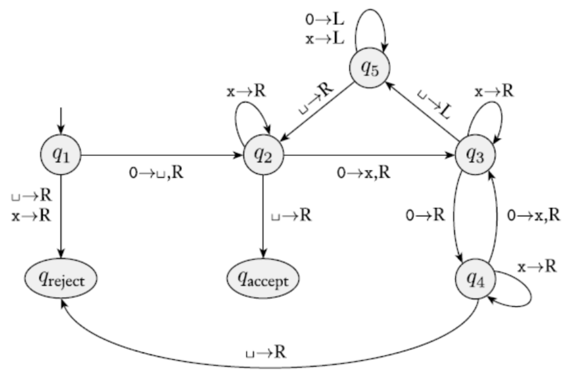
(c) Case 3: vwx contains both a's and b's. - Pumping v and x disrupts the order of a's and b's in z. - For $i \neq 1$, uv^iwx^iy is no longer of the form $a^n b^{n^2}$. - Hence, $uv^iwx^iy \notin L$.

Conclusion: In all cases, pumping v and x causes the resulting string uv^iwx^iy to fall outside L. Therefore, L does not satisfy the Pumping Lemma for CFLs, and we conclude that L is **not a context-free language**.

□

3. Design a Turing Machine (TM) M that decides the language:

$$L = \{0^{2^n} \mid n \geq 0\}.$$



- A Turing Machine (TM) M that **decides** L = { 0^{2^n} | n ≥ 0 }.
- A TM M is (Q, Σ, Γ, δ, q_1, q_{accept}, q_{reject})
  - Q = {q_1,…,q_5, q_{accept}, q_{reject} }
  - Σ = {0}
  - Γ = {0,x,⊔}

Solution.
A probable outline of the solution:

Example Execution:
For $w = 00000000$ (8 0's):

- q_0: Validate input; all symbols are 0, proceed to q_{mark}.
- q_{mark}: Mark every second 0: $0X0X0X0X$.
- q_{scan}: Scan back and verify remaining 0's. Repeat the marking process.
- Second iteration: Mark every second 0: $XXXXXX0X$.
- Final iteration: Verify that only one 0 remains; transition to q_{accept}.

Conclusion: The Turing Machine M accepts strings whose lengths are powers of 2 and rejects all others, thus deciding the language $L = \{0^{2^n} \mid n \geq 0\}$. □

Clearly explain the steps your Turing Machine takes to decide if the given string belongs to L. **[5]**

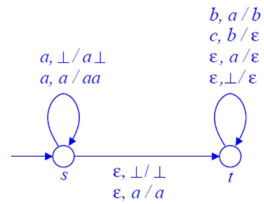
4. Consider the following language over $\Sigma = \{a, b, c\}$:

$$L_1 = \{a^i(bc)^j \mid i, j > 0 \text{ and } i > j\}.$$

(a) Design a PDA $M = (Q, \Sigma, \Gamma, \delta, s, \perp, F)$ to accept L_1. M must contain at most two states and clearly mention whether it accepts by final state, empty stack, or both. **[3]**

(b) Provide a detailed explanation of the transition function δ of your PDA, and describe how it ensures that $i > j$. **[2]**

Solution We take $Q = \{s, t\}$ with the start state s and no final states ($F = \emptyset$). The machine accepts by empty stack. The alphabets are $\Sigma = \{a, b, c\}$ and $\Gamma = \{a, b, \perp\}$. The transitions are described in the following figure.



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_1 is fully read by M, the stack contains $i - j$ occurrences of a and the bottom marker $\perp$. M uses the transitions $\epsilon, a \,/\, \epsilon$ and $\epsilon, \perp \,/\, \epsilon$ 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 $\epsilon, a \,/\, \epsilon$ more than $i - j$ times before reading all the j occurrences of bc, the machine gets stuck before reading the entire input.

Solution.
A probable outline of the solution:

Transition Function δ: **[2]**

- Push a's onto the stack:

$$\delta(s, a, \perp) = (s, A \perp), \quad \delta(s, a, A) = (s, AA)$$

  When reading a's, push A onto the stack. This counts the number of a's (i).

- Pop A for each bc pair:

$$\delta(s, b, A) = (s, A), \quad \delta(s, c, A) = (s, \epsilon)$$

  When reading a b, leave the stack unchanged (waiting for c), and when reading a c, pop A from the stack. Each bc pair corresponds to one A being popped, effectively matching j with i.

- Transition to final state when $i > j$:

$$\delta(s, \epsilon, A) = (f, \epsilon)$$

  When no more input remains, and there is at least one A on the stack, transition to the final state f. This ensures $i > j$, as there are unmatched A's remaining.

- Reject otherwise:

$$\delta(s, \epsilon, \perp) = \text{reject}$$

  If the stack is empty ($\perp$) but more input is expected, reject the string.

How the PDA Ensures $i > j$:
The PDA maintains a balance between i and j using the stack:

- Each a pushes an A onto the stack, counting i.
- Each bc pair pops an A, reducing the stack count by 1 for each j.
- After processing the input, if the stack is non-empty (A's remain), it means $i > j$. The PDA transitions to the final state and accepts the string.
- If the stack becomes empty before all bc pairs are processed, or if bc pairs remain unmatched, the PDA rejects the string.

□