Theory Assignment I

Automata Theory Monsoon 2024, IIIT Hyderabad

September 19, 2024

Total Marks: 35 points

Due date: **22/08/24 11:59 pm**

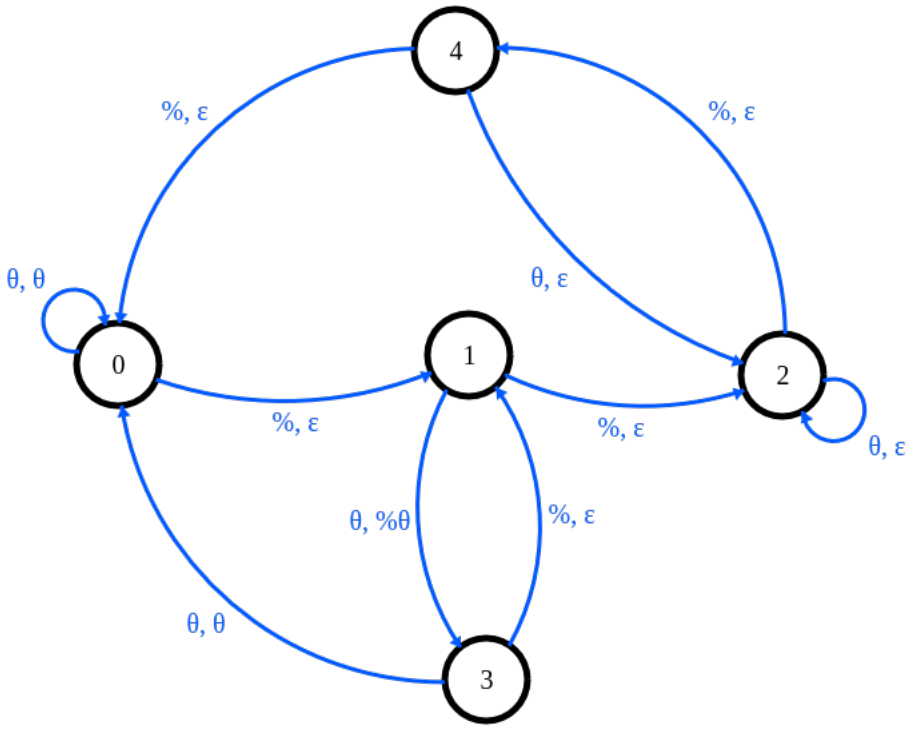
General Instructions: All symbols have the usual meanings (example: $\mathbb{R}$ is the set of reals, $\mathbb{N}$ the set of natural numbers, and so on). FSM stands for finite state machine. DFA stands for deterministic finite automata. NFA stands for non-deterministic finite automata. a^* is the Kleene Star operation.

---

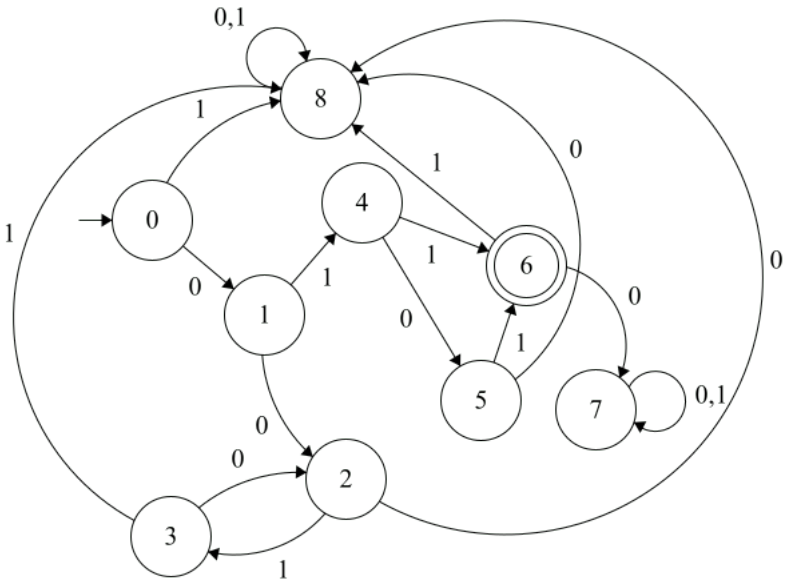
1. [2 points]

   $$\text{Max transitions in NFA} = (\text{No. of states})^2 \times (\text{size of alphabet} + 1) = 5^2 \times 4 = 100[]$$

2. [2 points] FST attached below



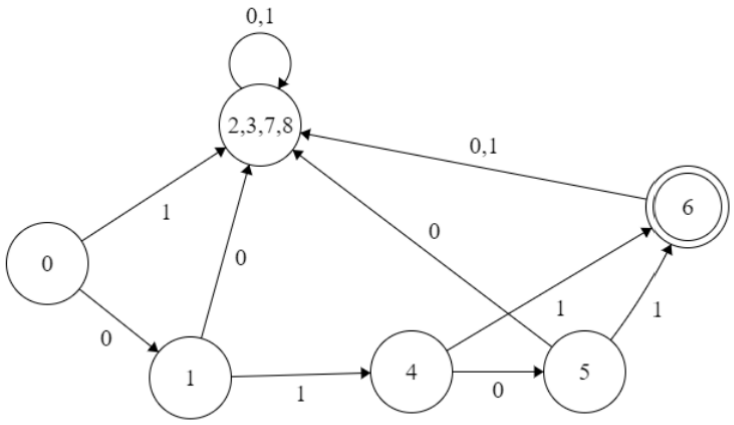
3. [3 points] Minimize the following DFA.



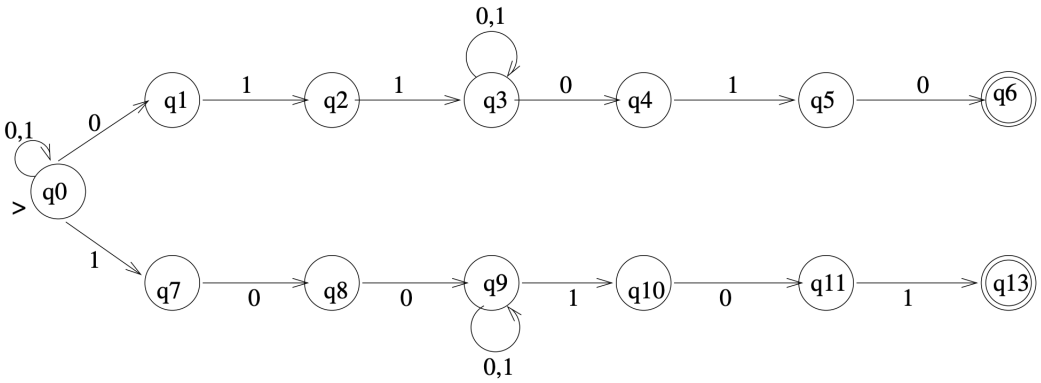
Solution: Equivalence Classes are as follows

$$
\begin{aligned}
&\text{Equivalence 0:}\{0,1,2,3,4,5,7,8\},\{6\}\\
&\text{Equivalence 1:}\{0,1,2,3,7,8\},\{4,5\},\{6\}\\
&\text{Equivalence 2:}\{0,2,3,7,8\},\{1\},\{4\},\{5\},\{6\}\\
&\text{Equivalence 3:}\{0\},\{2,3,7,8\},\{1\},\{4\},\{5\},\{6\}\\
&\text{Equivalence 3 = Equivalence 4}
\end{aligned}
\tag{1}
$$

The final DFA looks as follows



0 is start state here.



4. [2 points]

5. [1 point] 1. a(bb + bba)*ba or ab(bb + bab)*a
   2. (b + ab + aab)*aaa(baa + ba + b)*

6. [3 points] Let $A = \{0^*\#1^*\}$, which is regular. Consider the language $A_{1/3-1/3} \cap \{0^*1^*\}$. This intersection results in the language $\{0^n1^n \mid n \geq 0\}$. Since regular languages are closed under intersection and $\{0^*1^*\}$ is a regular language, the resulting language must also be regular if $A_{1/3-1/3}$ were regular. However, the language $\{0^n1^n \mid n \geq 0\}$ is known to be non-regular. Therefore, $A_{1/3-1/3}$ is not necessarily regular even if A is regular.

7. [3 points] Solution: Consider a DFA $D = (Q, \Sigma, \delta, q_0, F)$ such that $L(D) \neq \emptyset$. Therefore, if the final state is reachable, transitioning from D's start state to a final state requires at most $|Q| - |F|$ transitions. Since $U \neq \emptyset$, then either $L(M_1) \neq \emptyset$ or $L(M_2) \neq \emptyset$ (or both). We have the following cases:

   1. $L(M_1) = \emptyset, L(M_2) \neq \emptyset$. This implies that M_1 has no reachable accept states, and M_2 has at least one reachable accept state. Also, $U = L(M_2)$. From our observation above, if M_2 accepts a string s, then $|s| \leq k_2 - |F_2| < k_2 \leq \max(k_1, k_2)$ where F_2 is the set of accept states of M_2. Therefore, $s \in L(M_2) = U$.
   2. $L(M_1) \neq \emptyset, L(M_2) = \emptyset$. This is equivalent to Case 1.
   3. $L(M_1), L(M_2) \neq \emptyset$. Therefore, M_1 and M_2 have at least one reachable accept state. Let s_1 be the string of minimal length accepted by M_1, and s_2 for M_2. Let $s \in U$ be such that $|s| = \min(|s_1|, |s_2|)$. We have the following 2 cases:
      (a) $|s_1| \neq |s_2|$. This implies $|s| = \min(|s_1|, |s_2|) < \max(|s_1|, |s_2|)$. There are two possibilities. If $\max(|s_1|, |s_2|) = |s_1|$, then $|s| = |s_2| < |s_1| < k_1 \leq \max(k_1, k_2)$. Therefore, we are done. The same conclusion is reached if we consider when $\max(|s_1|, |s_2|) = |s_2|$.
      (b) $|s_1| = |s_2|$. This implies $|s| = \min(|s_1|, |s_2|) = \max(|s_1|, |s_2|)$. From our observation above, we have that $|s| = \min(|s_1|, |s_2|) = \max(|s_1|, |s_2|) = |s_1| < k_1 \leq \max(k_1, k_2)$. Therefore, we are done.

8. [3 points] 1. Suppose we select w from L such that $w = (^p)^p$ where p is the pumping length. Now we know that xy can only contain opening parenthesis since $|xy| \leq p$. When we pump on this string, its easy to see that we will only increase the number of opening parenthesis, and since the pumped string contains a different number of opening and closing parenthesis, they cannot possibly be balanced and hence $\notin$ L.
   2. Lets assume that L is context-free and hence satisfies the pumping lemma for some p.
   Consider string s = $a^{p!}$, $s \in L$ & $|s| \geq p$. Therefore, by the pumping lemma, s can be written as

s = xyz where $|xy| \leq p,\ |y| > 0$ & $xy^i z \in L\ \forall\ i \geq 0$.
Clearly y = a^j.
For i = 0, xy^0z = xz = $a^{p!-j}$.
For xz ∈ L, p! - j = s! for some s.
But by pumping lemma, j ≤ p.
Therefore, p! - j > (p - 1)! and hence xz ∉ L. Therefore, the pumping lemma is not satisfied and L is not regular.

9. [2 points] Solution: To determine the terminal and variable symbols, we use the fact that a terminal symbol cannot appear on the left-hand side of any production. By examining the production rules, we observe the following:

   - S, U, W, Y, and Z only appear on the right-hand side of the productions. Therefore, S, U W, Y, and Z must be terminal symbols. - R, T, V, and X appear on the left-hand side of the productions and may also appear on the right-hand side. Therefore, R, T, V, and X are variables. - Additionally, the start symbol must be one of the variables that appears on the left-hand side of a production rule. Given that R, T, V, and X are variables and R appears on the left-hand side but not on the right-hand side of the productions, we can deduce that R is the start symbol.

   Thus, the terminals are S, W, Y, and Z, and the variables are R, T, V, and X. The start symbol is R.

10. [3 points] Idea: this language is simply the union of $A_1 = \{a^i b^j c^k \mid i, j, k \geq 0, i = j\}$ and $A_2 = \{a^i b^j c^k \mid i, j, k \geq 0, j = k\}$. We can create simple grammars for the separate languages and union them: $S \to S_1 \mid S_2$ For A_1, we simply ensure that the number of a's equals the number of b's:

$$S_1 \to S_1 c \mid A \mid \epsilon$$
$$A \to aAb \mid \epsilon$$

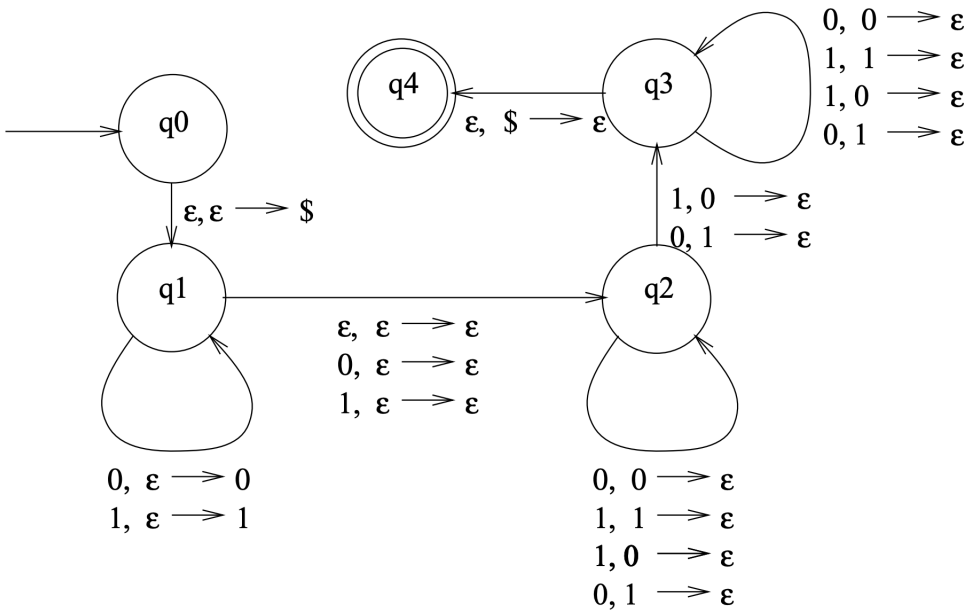
    Similarly for ensuring that the number of b's equals the number of c's:

$$S_2 \to aS_2 \mid B \mid \epsilon$$
$$B \to bBc \mid \epsilon$$

    This grammar is ambiguous. For $x = a^n b^n c^n$, we may use either S_1 or S_2 to generate x. To generate the language $A = \{a^i b^j c^k \mid i, j, k \geq 0$ and either $i = j$ or $j = k\}$, we can use the following CFG:

$$S \to S_1 \mid S_2$$
$$S_1 \to S_1 c \mid A \mid \epsilon$$
$$A \to aAb \mid \epsilon$$
$$S_2 \to aS_2 \mid B \mid \epsilon$$
$$B \to bBc \mid \epsilon$$

11. [4 points] We can use the PDA for recognizing palindromes to create a PDA for this language. To change the PDA accepting all palindromes into one that accepts all non-palidromes, we simply insist in the new machine that there is at least one inconsistency between the first and second half of input string w. So the new PDA can essentially be the same, except when we are popping symbols off the stack and matching them with inputs, we must make sure that there is at least one a where there should have been a b or vice versa. Here is the PDA:



We first mark the bottom of the stack with a \$, push the first half of the string (excepting the middle symbol if the string has odd length) onto the stack in q1, guess nondeterministically where the middle of the string is and switch to state q2. If rev(w) != w, then at some point there will be a mismatch between what is on the stack and in the input; when this is true, the machine can take the transition from q2 to q3. Otherwise, any match or mismatch of inputs symbols to symbols on the stack is allowed. Finally, the machine accepts when 1) the input is exhausted and 2) the stack is empty.

12. [4 points]
- PDAs accept strings either by emptying the stack or by reaching the final state after reading the entire input tape. In this example, the transitions from initial state q_0 to q_1 is marked by pushing # into the stack and is the only transition that involves # and can't be popped out again. So, there is no chance of acceptance by emptying the stack rather it's only by reaching the final state.
- a's can only be accepted in state q_1 and b's only by state q_2 this implies the string must be of the form

$$\{a^m b^n\}$$

for now.
- To read b's from the input tape, we've to pop-out a's each time which's obtainable from accepting a's. So, the number of b's $<$ number of a's for all the b's to be accepted in this state.
- To reach the final state we need at least one a to make the transition to the final state.

The given automaton accepts strings of the form

$$\{a^m b^n\}$$

s.t n+1 $\leq$ m And, it's given the question that m+n = 100 So the possible set values of m and n in the form of (m,n) are (100,0),(99,1) ... (51,49) which in total there are 50