Final Exam

Automata Theory Monsoon 2024, IIIT Hyderabad

23rd September, 2023

Total Points: 35 Time: 90 mins

General Instructions: FSM stands for finite state machine. DFA stands for deterministic finite automata. NFA stands for non-deterministic finite automata. PDA stands for Push Down Automata. CFG stands for context-free grammar while CFL stands for Context-free Language.

---

1. [5 points] (a) CFG for $\{\omega \mid \omega$ contains at least three 1s$\}$ is given by:

$$S \to P1P1P1P$$

$$P \to 0P \mid 1P \mid \epsilon$$

(b) CFG for $\{\omega \mid$ the length of ω is odd$\}$ is given by:

$$S \to 0 \mid 1 \mid 00S \mid 01S \mid 10S \mid 11S$$

(or equivalently)

$$S \to 0 \mid 1 \mid 0S0 \mid 0S1 \mid 1S0 \mid 1S1$$

2. [4 points] Assume for the sake of contradiction that there exists an FST T that outputs w^R on input w. Consider the following two input strings: 00 and 01.

(i) On input 00, FST T must produce the output 00, because $00^R = 00$.

(ii) On input 01, FST T must produce the output 10, because $01^R = 10$.

Notice that in both cases, the first input bit is the same, i.e., 0. However, the first output bits differ: 0 in the first case and 1 in the second case.

This means that after reading the first input bit (0), the FST T would have to decide what the first output bit should be. However, in our cases, this first output bit must be different depending on the second bit of the input string (0 or 1). This is impossible for an FST because its decision for the first output bit cannot depend on a symbol it has not yet read.

Thus, our assumption that an FST T can output w^R for every input w is incorrect.

Therefore, no FST can output w^R for every input w over the alphabets $\{0, 1\}$.

3. [2 points] 1. pumping length = 1
   2. pumping length = 3

4. [3 points] Solution: First notice that State 6 is unreachable. Therefore we can prune it out. Among the remaining states, 1 and 5 are the final states, and the rest are non-final. Therefore the initial partition is given by

$$\Pi_0 = \{\{1,5\},\{2,3,4\}\} \tag{1}$$

Since on input 1 state 2 goes to state 1, state 3 goes to state 4 and 1 and 4 are in different sets in Π_0 , states 2 and 3 are going to be separated from each other in the next partition. Also since on 0 state 4 goes to state 4, state 3 goes to state 5 and 4 and 5 are in different sets in Π_0 , states 3 and 4 are going to be separated from each other in the next partition.Further, since on 1 state 2 goes to state 1, state 4 goes to state 4 and 1 and 4 are in different sets in Π_0 , 2 and 4 are separated from each other in the next partition. State 1 and state 5 are not separated yet. Therefore the new partition Π_1 is given by.

$$\Pi_1 = \{\{1,5\},\{2\},\{3\},\{4\}\} \tag{2}$$

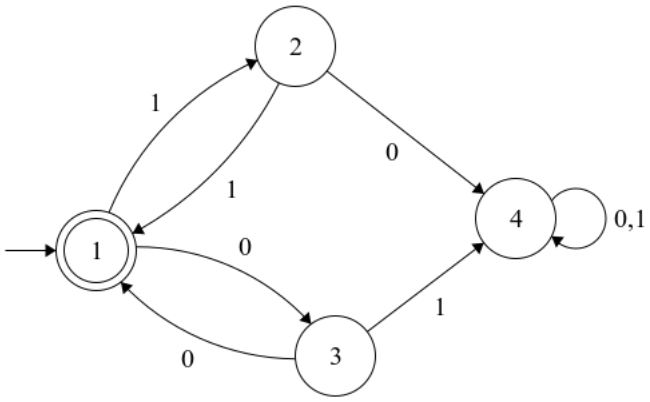
We notice that states 1 and 5 have the same transitions (goes to state 3 on input 0 and goes to state 2 on input 1). Therefore they are equivalent states. Therefore our final partition is

$$\Pi_{final} = \{\{1,5\},\{2\},\{3\},\{4\}\} \tag{3}$$

Therefore the set of minimized states is

$$Q_{final} = \{1,2,3,4\} \tag{4}$$

And the final minimized DFA is.



The regular language for this DFA is $R = \{\{00\}^* + \{11\}^*\}^*$

5. [4 points] For an ambiguous grammar, we can use:

$$S \rightarrow aSbS \mid aS \mid \epsilon$$

For an unambiguous grammar, we can use:

$$S \rightarrow Sa \mid Ab \mid \epsilon$$

$$A \rightarrow Sa \mid AAb$$

6. [4 points] **Decidable language:** A language is decidable if and only if some Turing machine decides it.

   We call a Turing machine a decider if all branches halt on all inputs.

   The given data is, A is the language containing only the single string s, where

$$s = \begin{cases} 0 & \text{if life never will be found on Mars} \\ 1 & \text{if life will be found on Mars someday.} \end{cases}$$

   So, the language A may contain either 0 or 1 but not both. Thus,

$$A = \{0\} \quad \text{(or)} \quad A = \{1\}$$

   In both these cases, A is finite and the string s is fixed, so we know the same finite language is always decidable.

   We are not able to determine whether $A = \{0\}$ or $A = \{1\}$,in this case, we need to give two Turing machines for both cases. So, definitely, one of them will be the decider of A.

7. [4 points] Consider the Language $L = \{0^m 1^n \mid m \neq n\}$.

   Assume that L is regular language and string $S = 0^P 1^{P+P!}$. Divide the string into three pieces x, y and z. So, $S = 0^P 1^{P+P!} = xyz \in L$ and $|S| \geq P$ where, P is the pumping length.

   Note: $P!$ is divisible by all integers from 1 to P, where $P! = P \times (P-1) \times (P-2) \times ... \times 1$

   Assume that $x = 0^a, y = 0^b$ and $z = 0^c 1^{P+P!}$, where $b \geq 1$ and $a + b + c = P$.

   Now take string $s' = xy^{i+1}z$, where $i = \frac{P!}{b}$.

   Then $y^i = 0^{P!}$ so $y^{i+1} = 0^{b+Pi}$, and

   So $xyz_i = 0^{a+b+c+P!} 1^{P+P!}$.

   That gives $xyz = 0^{P+P!} 1^{P+P!} \notin L$, a contradiction.

   Here, $m = P + P!, n = P + P!$ and $m = n$. This is a contradiction to the assumption because $m = n$. Thus, by using pumping lemma it is proved L is not regular.

8. [5 points] PDA with queue is equivalent to Turing Machine? Prove it. Solution: To show that a queue automaton is equivalent to Turing machines, we need to show that one can be simulated by the other.

   First we'll show that a queue automaton can be simulated by a 2-tape Turing machine which we know is equivalent to a single-tape turing machine. The first tape would just be a read-only input tape similar to the queue automaton. The second tape would be the work tape which is empty initially. A "write" operation would involve the Turing machine moving its write head to the leftmost end and then moving the tape contents by one cell to the right and then writing the required symbol on the first cell. The "read" operation would be to move the write-head to the rightmost cell, real the contents and then overwrite it with a blank symbol. This effectively simulates the queue automaton.

   Now, to show that a queue automaton can simulate a general Turing machine. Say that the write head of the queue automaton denotes the read/write head of the Turing machine. To simulate a move of one cell to the left, we can pop a symbol from the right, and simultaneously write it to the front of the queue effectively performing a right cyclic shift on the queue. Similarly, to move one cell to the right, we pop every symbol from the right $n-1$ times where n is the current tape length and

simultaneously write it onto the left-end. This brings the required symbol to the right-end of the queue where we can read it.

This shows that both the machines are equivalent.

9. [4 points] We can construct a Turing machine that decides L. It will do the following:

   (a) On input $\langle M, k\rangle$ where M is a Turing machine:

   (b) For all strings w_i where $\text{len}(w_i) \leq k+1$:

   - Run M on w_i for k steps. If M doesn't terminate on w_i within k steps, accept.

   (c) If we're finished enumerating and M terminated within k steps every time, reject.

Proof that this is a decider:

Important observation is that we do not need to iterate over all possible strings (which is potentially infinite). Only strings of length $\leq$ k+1 need to be checked because a TM that runs for k steps cannot explore more than first k characters of the input string.

- If $x \in L$, then M runs for at least k steps on some string. Therefore, there is a string w_i such that $\text{len}(w_i) \leq k+1$ where the computation of M on w_i will not have terminated after k steps. Therefore, the program will accept on some iteration.
- If $x \notin L$, then M terminates within k steps on every input string. Since we are testing a finite number of strings and M terminates on all of them, we will reach step (c) and then reject.

BONUS
Let USELESS_TM $= \{\langle M, q\rangle \mid q$ is a useless state in $M\}$. Suppose that USELESS_TM was decidable, and let U be its decider. We will construct a decider E for E_{TM}:

$$E = \text{"On input } \langle M\rangle \text{ where } M \text{ is a TM:}$$

1. Run U on $\langle M, q_{\text{accept}}\rangle$, where q_{accept} is M's accept state.
2. Output what U outputs.

Since E_{TM} is undecidable, USELESS_TM is also undecidable.