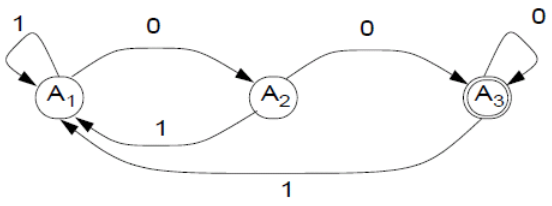
Theory of Computation

Spring 2025, Homework # 1 (Solutions)

1. (10 pts) A string u is an *anagram* of a string w if u is obtained from w by rearranging the symbols. For example, *listen* is an anagram of *silent*. Formally, if $w = w_1 w_2 \cdots w_n$, then u is its anagram if $u = w_{\sigma(1)} w_{\sigma(2)} \cdots w_{\sigma(n)}$ for some permutation σ. Given a regular language L, is $L_A = \{u | u \text{ is an anagram of } w \in L\}$ always regular? Justify your answer.

   Answer. L_A is **not** always regular. Let $\Sigma = \{0, 1\}$. Consider regular language $L = (01)^*$, we have $L_A = \{w \in \Sigma^* |$ w has an equal number of 0's and 1's$\}$, and we know L_A is not regular (Ch.1 Corollary 21).

2. (20 pts) Consider the following DFA,



   we associate the DFA with the following system of equations:

$$A_1 = 1A_1 + 0A_2$$
$$A_2 = 0A_3 + 1A_1$$
$$A_3 = 0A_3 + 1A_1 + \epsilon$$

   The solution of A_i $(1 \leq i \leq 3)$ corresponds to set of strings that can lead the DFA from state A_i to accepting state A_3. Solve the system of equations to obtain the regular expression corresponding to the solution of A_1, which is the language accepted by the DFA as A_1 is the initial state.
   Hint: you may use the fact that the solution of $X = \alpha X + \beta$ is $\alpha^* \beta$. For instance, $A_3 = 0A_3 + 1A_1 + \epsilon$ implies $A_3 = 0^*(1A_1 + \epsilon)$. Also use the idea similar to Gaussian elimination in solving systems of linear equations.

   Answer.

$$\begin{aligned} A_1 &= 1A_1 + 0A_2 \\ &= 1A_1 + 0(0A_3 + 1A_1) \\ &= (1 + 01)A_1 + 00A_3 \\ &= (1 + 01)A_1 + 000^*(1A_1 + \epsilon) \\ &= (1 + 01 + 000^*1)A_1 + 000^* \\ &= (0^*1)A_1 + 000^* \\ &= (0^*1)^*0^*00 \\ &= (0 + 1)^*00 \end{aligned} \tag{1}$$

3. (10 pts) Use the pumping lemma to prove that $L = \{0^{n!} | n \geq 0\}$ is not regular, where $n!$ denotes the factorial of n.

   Answer. Suppose L is regular, and let $p \geq 1$ be the pumping length. Let $s = 0^{p!} \in L$, notice that

$|s| \geq p$. Now, let xyz be the partition in pumping lemma. Write $|x| = a, |y| = b, |z| = p! - a - b$. Note that $0 < b \leq p$.

$$|xy^{p!}z| = a + bp! + p! - a - b = p!(b+1) - b < p!(b+1) \leq (p+1)! \tag{2}$$

$$|xy^{p!}z| > p! \tag{3}$$

The two inequalities give that $|xy^{p!}z| \notin L$.
Proof by Contradiction, L is not regular.

4. (15 pts) Let $\Sigma = \{0, 1\}$. For each of the following languages, decide whether it is regular or not. Justify your answers.

   (a) $L_1 = \{1^k y \mid y \text{ contains at least } k\ 1's, k \geq 1.\}$

   (b) $L_2 = \{1^k 0 y \mid y \text{ contains at least } k\ 1's, k \geq 1.\}$.

   (c) $L_3 = \{1^k y \mid y \text{ contains at most } k\ 1's, k \geq 1.\}$

   Answer.
   (a) L_1 is regular.
   Notice that for any $w = 1^k y$ where y contains at least k $1's$, w can be written as $11^{k-1}y$ and $1^{k-1}y$ is a string contains at least one 1.
   So $L_1 = \{1y | y \text{ contains at least one } 1\} = 10^*1\Sigma^*$

   (b) L_2 is not regular.
   Prove by pumping lemma. Let p be the pumping length, and $s = 1^p 01^p \in L_2$ and xyz be the partition in pumping lemma. We have $|xy| \leq p$ and $|y| > 0$ so $x = 1^i$ $y = 1^j$ for some $j > 0$. Then $xy^2z = 1^{p+j}01^p \notin L_2$, a contradiction.

   (c) L_3 is not regular.
   Notice that $L_3 \bigcap 1^*01^* = \{1^i 01^j | i \geq j\}$ is not regular.

5. (15 pts) If L is a language over the alphabet $\Sigma = \{0, 1\}$, define L_{ERROR} so that a string is in L_{ERROR} iff it is the result of flipping a bit in a string in L; i.e.,

$$L_{ERROR} = \{w \mid w = uxv, u, v \in \Sigma^*, x \in \{0, 1\},\ u\overline{x}v \in L\}$$

   where $\overline{0} = 1, \overline{1} = 0$.
   Prove that if L is regular, then L_{ERROR} is also regular.
   Hint: Suppose M is a DFA accepting L, construct an NFA M' so that $L(M') = L_{ERROR}$.

   Answer. Suppose DFA $M(Q, \Sigma, \delta, q_0, F)$ accept L. The goal is to construct an NFA M' that accepts L_{ERROR}. We start by making a copy of the finite automaton (FA) M. Let the original automaton be denoted as M_A and the copied automaton as M_B.

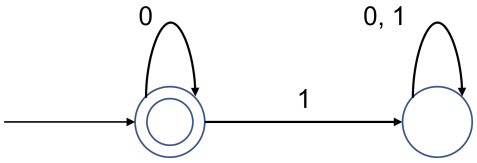
   - The start state of M' is the start state of M_A
   - The accept state of M' is the start state of M_B
   - The transition function δ' of M' behaves as follows:

     Within M_A and M_B, the transitions remain the same as the original transition function. Additionally, if $\delta(q_1, 0(resp.1)) = q_2$ then $\delta'(q_1 \text{ in } M_A, 1(resp.0)) = q_2$ in M_B.

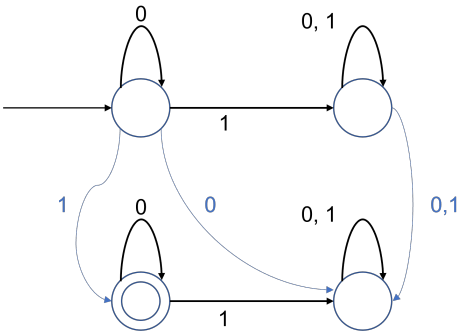
   In other words, δ' mirrors δ within M_A and M_B, while also allowing bit-flipped transitions from M_A to M_B. We can observe that M' accepts L. Below is an example for $L = 0^*$.

6. (20 pts) For a language L over an alphabet Σ, define

$$CYCLE(L) = \{x_1x_2 \mid \exists x_1, x_2 \in \Sigma^* \text{ such that } x_2x_1 \in L\}.$$



(a) DFA accepts 0^*.



(b) NFA accepts 0^*_{error}.

For example, if $abc \in L$, abc, bca, $cab \in CYCLE(L)$. Prove that if L is regular, then so is $CYCLE(L)$.
Hint: Given an FA $M = (Q, \Sigma, \delta, q_0, F)$ accepting L, construct an NFA $M' = (Q', \Sigma, \delta', q_0', F')$ to accept $CYCLE(L)$. You may let a state of Q' be of the form (p, q, d), $d \in \{1, 2\}$. The d is used to indicate whether the current input symbol is a part of x_1 or x_2.

Answer. Given a regular language L and a FA M accepts L.
Based on the Hint, we represent states in the form (p, q, d), where:

- p records the beginning state of this computation
- q represents the current state in Q
- d is used to indicate whether the current input symbol is a part of x_1 or x_2.

In M, the computations are $q_0 \xrightarrow{x_1} q_i \xrightarrow{x_2} q_f$. In M', the computations are $q_i \xrightarrow{x_2} q_f \xrightarrow{\epsilon} q_0 \xrightarrow{x_1} q_i$.

The states setQ' of M' is $Q \times Q \times \{1, 2\} \bigcup \{q_0'\}$. q_0' is the start state and it transitions to beginning states via epsilon transitions.
The accept statesF' set of M' is $\{(p, p, 2) | p \in Q\}$.
The transition function δ' in the NFA M' is constructed by inheriting the transitions from the original automaton M, with additional epsilon transitions.

- $\delta'((p, q, d), a) = \{(p, \delta(q, a), d)\}$
- $\delta'(q_0', \epsilon) = \{(p, p, 1) | p \in Q\}$
- $\delta'((p, q_f, 1), \epsilon) = \{(p, q_0, 2)\} \forall q_f \in F$

We have constructed a NFA $M' = (Q', \Sigma, \delta', q_0', F')$ accepts $CYCLE(L)$.

7. (10 pts) Consider a new kind of finite automata called $\forall$-NFA. A $\forall$-NFA M is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ that recognizes x if *every* possible computation of M on x ends in a state from F. Note, in contrast, that an ordinary NFA accepts a string if *some* computation ends in an accept state. Prove that $\forall$-NFA recognize the class of regular languages.

   Answer. Claim that $\forall$-NFA accepts $L \implies L$ is regular. We know that a $\forall$-NFA $M = (Q, \Sigma, \delta, q_0, F)$ accepts L means $\forall w \in L$, all computations of w ends in F. Now constructing a NFA $M' = (Q, \Sigma, \delta, q_0, Q - F)$. M' shares the same states, alphabet, transition function, and start state with M. Notice that M' is a NFA accepts $\overline{L}$ so $\overline{L}$ is regular, which gives that L is regular.