Homework 1: Regular Languages: Automata and Expressions

CS 212 Nature of Computation
Habib University
Homework 1 Team 11

Fall 2023

General instructions

- For drawing finite automata, see this TikZ guide or the JFLAP tool. Hand drawn diagrams will not be accepted.
- Please ensure that your solutions are neatly formatted and organized, and use clear and concise language.
- Please consult Canvas for a rubric containing the breakdown of points for each problem.
- For all the problems below, $\Sigma = \{a, b\}$.
- Some of the problems below make use of the following count function.

$$n_a(w) = \text{the number of occurrences of } a \text{ in } w, \text{ where } a \in \Sigma, w \in \Sigma^*.$$

Problems

1. $\boxed{\text{15 points}}$ List 2 members and 2 non-members of the language, $(a \cup ba \cup bb)\Sigma^*$.

2. $\boxed{\text{20 points}}$ Provide the state diagram of a simplified DFA that recognizes the language,

$$A = \{w \in \Sigma^* \mid n_a(w) \geq 2, n_b(w) \leq 1\}.$$

3. $\boxed{\text{30 points}}$ Given the languages, A and B, we derive the language, $C = \{w \in A \mid w \in B\}$. Prove or disprove the following claim.

   Claim 1. *If A and B are regular languages, then so is C.*

4. $\boxed{\text{35 points}}$ Given the languages, A and B, we define the following operation.

$$A \smile_a B = \{u \in A \mid \exists v \in B \ni n_a(u) = n_a(v)\}$$

   Prove or disprove the following claim.

   Claim 2. *The class of regular languages is closed under $\smile_a$.*

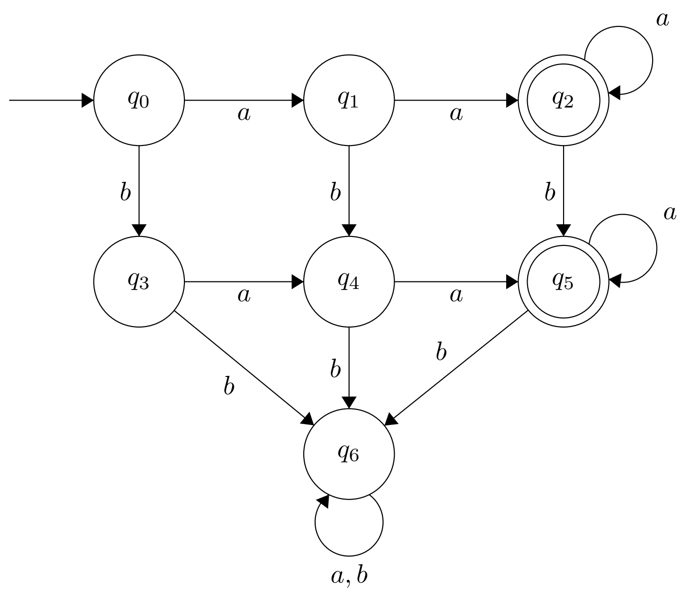
Solution:

Problem 1

Members: a, ba
Non-members: ε, b

Problem 2

$A = \{w \in \Sigma^* \mid n_a(w) \geq 2, n_b(w) \leq 1\}$ describes the language that contains at least 2 occurences of 'a', and at most one occurence of 'b'. The DFA for this language is as follows:



The above DFA shows that for any string that contains more than 1 'b', then machine goes to a regular state and stays there. If the string contains 1 or less 'b', and 2 or more 'a', the machine goes to an accept state.

Problem 3

Given the languages A and B, and $C = \{w \in A \mid w \in B\}$, we need to prove or disprove that if A and B are regular languages, then C is also a regular language.

By definition, C is the language that consists of all the strings w that belong to both A and B. So $C = A \cap B$.

If A and B are regular languages, then there must exist some finite automata that recognizes them. Let M_1 and M_2 be the DFAs that recognize A and B respectively. Then:
- $M_1 = (Q, \Sigma, \delta_A, q_o, F_A)$ recognizes A
- $M_2 = (R, \Sigma, \delta_B, r_o, F_B)$ recognizes B

Then we can construct a DFA M_3 for C that keeps track of the pair of states (q_i, r_j), where $q_i \in Q$ and $r_j \in R$. In this construction, the states of the new DFA are pairs of states from the A and B, that is, the Cartesian Product of the two languages

$$Q \times R = \{(q, r) \mid q \in Q \text{ and } r \in R\}$$

Then $M_3 = (Q \times R, \sum, \delta, (q_o, r_o), F_A \times F_B)$.
For M_3,

- Set of states $Q \times R$, the Cartesian Product of Q and R to include all pairs of states in both the machines as mentioned above.
- $\sum$ remains the same as the input alphabet.
- We have the transition function δ defined as
$$\delta((q, r), a) = (\delta_A(q, a), \delta_B(r, a)) \;\; \forall q, r \in Q, R \text{ and } \forall a \in \sum$$
- The start state is (q_o, r_o) which is the pair of start states of M_1 and M_2.
- The set of accept states is again the Cartesian Product of the accept states of M_1 and M_2, that is, all pairs of states (q_f, r_f) where $q_f \in F_A$ and $r_f \in F_B$.

Therefore, by this construction of M_3, we have created a DFA that would only reach an accepting state if both, M_1 and M_2 reach the accept state. Therefore, the string would have to be accepted by both M_1 and M_2. Hence, for any arbitrary string w that is accepted by M_3, $w \in A$ and $w \in B$.

Hence, if A and B are regular languages, then we can construct a DFA to recognize $C = \{w \in A \mid w \in B\}$, therefore, C is also a regular language.

If there are no strings common to both A and B, then C will be an empty language, meaning it will only consist of the empty string ε. Then $C = \{\varepsilon\}$ which is also a regular language as again a DFA can be constructed with only 1 state as both the start and accept state.

Therefore C is a regular language.

■

Problem 4

Given the regular languages A and B, $A \smile_a B = \{u \in A \mid \exists v \in B \ni n_a(u) = n_a(v)\}$. This operation takes two languages A, and B, and produces a new language that contains all strings $u \in A$ that have the same number of occurrences of a as some string $v \in B$.

Let C be the language obtained by the operation $A \smile_a B$. Then $C = A \smile_a B$.

If A and B are two regular langauges, then there must exist some finite automata that recognizes them. Let M_1 and M_2 be the DFAs that recognize A and B respectively. Then:
- $M_1 = (Q, \Sigma, \delta_A, q_o, F_A)$ recognizes A
- $M_2 = (R, \Sigma, \delta_B, r_o, F_B)$ recognizes B

Then we can construct an NFA M_C that recognizes C as follows. To decide whether $u \in A$ for some arbitrary string u, and in parallel nondeterministically guess some string $v \in B$ such that v contains the same number of a's in u. Then $M_C = \{Q, \sum, \delta, q_o, F\}$ where:

1. $Q = Q_1 \times Q_2$, the Cartesian Product of the states of M_1 and M_2.
2. $\sum$ remains the same as the input alphabet.
3. $q_o = q_o$
4. $F = F_A \times F_B$
5. $\delta((q, r), s)$ where $q \in Q, r \in R, s \in \sum_\varepsilon$,

$$\delta((q,r)s) = \begin{cases} \{\delta_A(q,s), \delta_B(r,s)\}, & s = a \\ \{\delta_A(q,s), r\}, & s = b \\ \{q, \delta_B(r,s)\}, & s = \varepsilon \end{cases}$$

The machine must have a parir of all states that exist in A and B, that is, the Cartesian Product of the two languages, the start state same as q_o, accept states $F_A \times F_B$.
The transition function is defined such that it has a pair of states, and an alphabet;

1. if the current alphabet is a, then it moves on to the next state in A, and the next state in B, which means that the NFA is now trying to guess a string $v \in B$ that has the same number of occurrences of a as the current input string and the previous input alphabet
2. if the current alphabet is b, then the NFA moves on to the next state in A while remaining in the same state in B which again means that the NFA is trying to guess a string $v \in B$ that has the same number of occurrences of a as the current input string and the previous input alphabet, however, it does not change its state in B since we are concerned with the number of occurrences of a
3. if we have an ε or an empty string, then it moves on to the next state in B while remaining in the same state in A

Then the NFA M_C accepts a string if and only if $\forall u \in A$, $\exists v \in B$ such that $n_a(u) = n_a(v)$. Again this claim can be proved by showing that the language of M_C is a subset of the language of $A \smile_a B$ and vice versa.

Suppose M_C accepts a string s, meaning that it reaches the accept state when processing s. We defined the accept states F as a pair of accept states from M_A and M_B, i.e., (q_f, r_f) where $q_f \in F_A, r_f \in F_B$. For each alphabet s_i in s, the transition function ensures that the NFA moves to the next state in A and B if $s_i = a$, and moves to the next state in A while remaining in the same state in B if $s_i = b$. This means that the NFA is guessing a string $v \in B$ that has the same number of occurrences of a as the current input string and the previous input alphabet. If the NFA reaches the accept state, it implies that M_A also reaches the accept state $q_f \in F_A$ after processing a substring u of s, and M_B reaches an accept state $r_f \in F_B$ after processing a substring v of s. Crucially, since M_C accepts s, there must be a sequence of transitions in M_C that correspond to reading the alphabets in s such that $n_a(u) = n_a(v)$ since for each alphabet 'a' in s, M_C consumes one 'a' in both machines, and for each alphabet 'b' in s, M_C consumes one 'b' in M_B. Hence $n_a(u) = n_a(v)$, therefore, $L(M_C) \subseteq A \smile_a B$.

Now consider strings $u \in A$ and $v \in B$, such that $n_a(u) = n_a(v)$. Then we can construct a string s from u and v such that $s \in A \smile_a B$ by the definition of the smile operation. Then s is some concatenation of u and v. When M_C processes s, for each alphabet 'a' in s, M_C consumes one 'a' in both machines, and for each alphabet 'b' in s, M_C consumes one 'b' in M_B. Hence M_C reaches the accept state in the final state of M_A and M_B both which corresponds to $q_f \in F_A$ and $r_f \in F_B$ respectivel. Therefore, $A \smile_a B \subseteq L(M_C)$.

Hence, the class of regular languages is closed under $\smile_a$.

■