Theory of Computation, CSCI 438, Spring 2021
Exam 1, Feb. 14 **Name ________________**
Happy Valentine's Day

Essay Questions

(20 pts.)

Definitions

1. Give the complete definition of a non-deterministic finite automaton.

(5 pts.)

Nondeterministic finite automaton, NFA, $M = (Q,\Sigma,\delta,q_0,F)$

* Q - finite set of states
* Σ - finite set of symbols, input alphabet
* δ: $Q \times \Sigma\varepsilon \rightarrow \mathscr{P}(Q)$, transition function ($\Sigma_\varepsilon$ is Σ augmented by ε which indicates that the machine can move forward without an input symbol)
* $q_0 \in Q$, initial state
* $F \subseteq Q$, set of accept states

(Definition 1.37, page 53)

2. Describe, without using formulas, what it means for a string to be accepted by an NFA. (5 pts.)

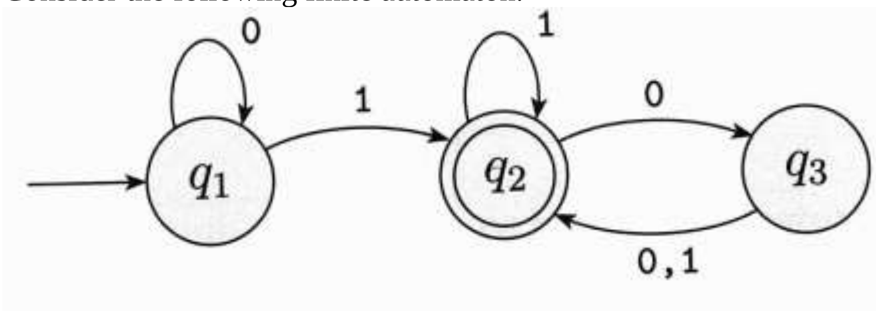
A string is accepted by an NFA if there is some sequence of possible moves that will put the machine in a final state at the end of the string. (Intuitively, the right move is always chosen.)
(Page 54)

3. Give the formal definition of what it means for a string to be accepted by an NFA. (5 pts.)

Given the definition of an NFA M from question 1,
M accepts string $w \in \Sigma^*$ iff there exists some sequence such that $\delta^*(q_0, w) \in F$

Problem Solving

4. Consider the following finite automaton.



Write the definition of this machine using the linear format. (5 pts.)

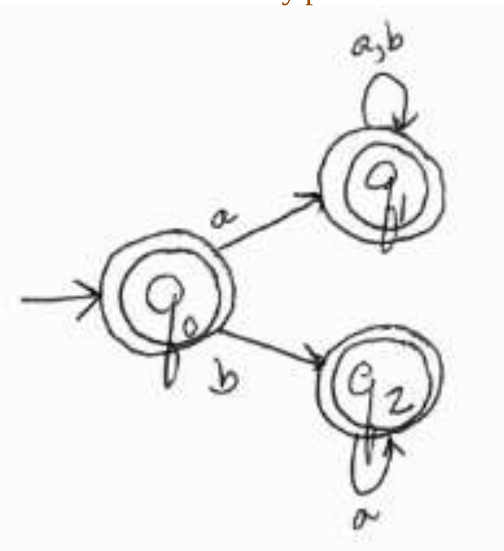
M = (…….)

M= ({q_1, q_2, q_3}, {0, 1}, δ, q_0, {q_2}) where δ is defined by:

$\delta(q_1, 0) \rightarrow q_1$
$\delta(q_1, 1) \rightarrow q_2$
$\delta(q_2, 0) \rightarrow q_3$
$\delta(q_2, 1) \rightarrow q_2$
$\delta(q_3, 0) \rightarrow q_2$
$\delta(q_3, 1) \rightarrow q_2$

5. Give an NFA for the language L on Σ={a, b} where
L = {w | w starts with an a or has at most one b } (5 pts.)

There are many possible answers. The following answer is almost a DFA.



6. Give a regular expression for all strings on the alphabet Σ={a,b} which end with three a's, or contain an even number of a's. (5 pts.)

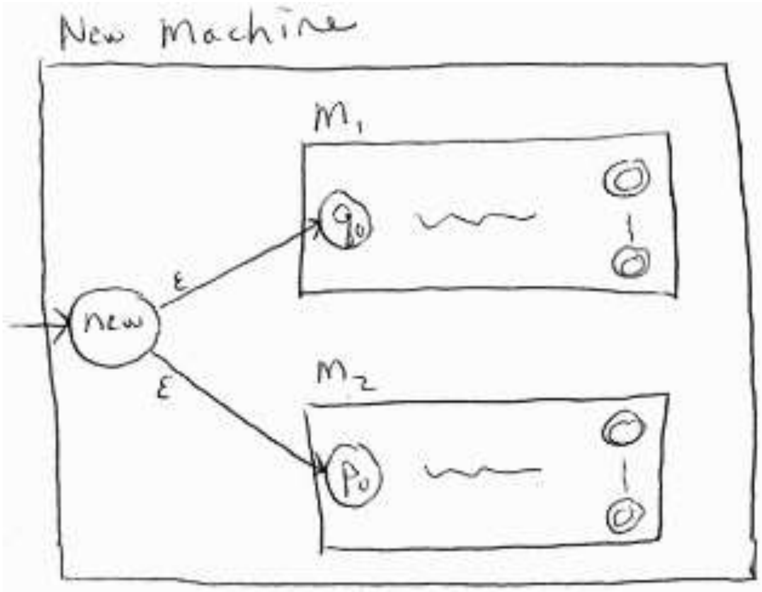
Possible answer:

$[(a \cup b)^*aaa] \cup [b^*ab^*ab^*]^*$

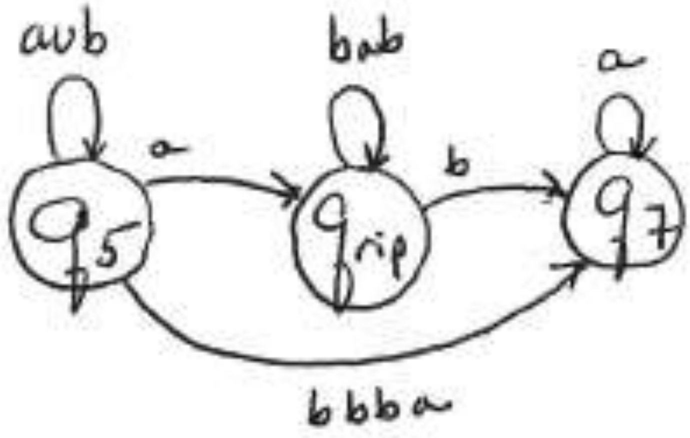
7. Show that regular languages are closed under union by beginning with a DFA for each of the regular languages, and constructing an NFA for the union of the languages. Show this by drawing a picture of the construction. You do NOT need to formally write the construction. (You will need to do this later in the exam.)

(10 pts.)

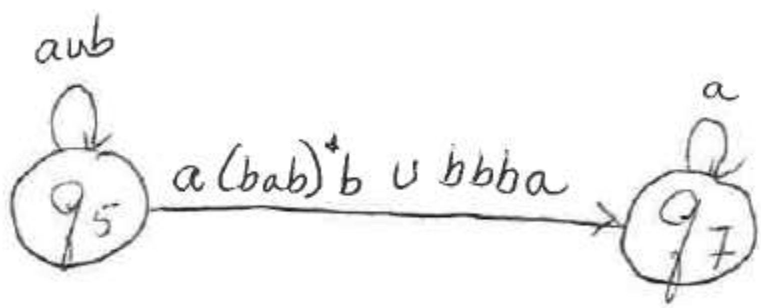
Answer:



8. Give the result of ripping the state q_{rip} from the following Generalized NFA using the algorithm discussed in class and given in the text. (10 pts.)



Answer:



9. Prove that regular languages are closed under union by beginning with a DFA for each of the regular languages, and constructing an NFA for the union of the languages. (This is the same question as problem 6, only there you only needed to show what to do. Here you need to prove the statement.) (10 pts.)

Suppose that languages L_1 and L_2 are regular. Then there must be DFAs
$M_1 = (Q, \sum, \delta_1, q_0, F_1)$ where $L_1 = \mathcal{L}(M_1)$ and
$M_2 = (P, \sum, \delta_2, p_0, F_2)$ where $L_2 = \mathcal{L}(M_2)$.

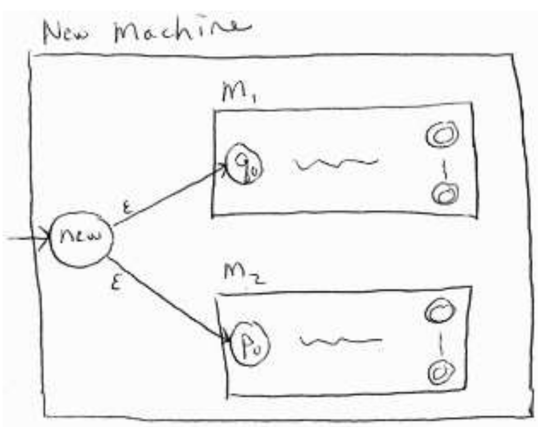
Define the NFA

$$M' = (Q \cup P \cup \{new\}, \sum, \delta', new, F_1 \cup F_2) \text{ where}$$

$$\delta'(q,a) = \begin{cases} \{\delta_1(q,a)\} & \text{for } q \in Q \text{ and } a \in \sum \\ \{\delta_2(p,a)\} & \text{for } p \in P \text{ and } a \in \sum \\ \{q_0, p_0\} & \text{for } q = new \text{ and } a = \varepsilon \end{cases}$$

Clearly, the NFA M' recognizes the union of L1 and L2. Thus, regular languages are closed under union.

Here is a picture of the new machine.



10. Prove that L is regular by giving an NFA or regular expression for L, or prove that it is not regular using the pumping lemma for regular languages. (10 pts.)

$L = \{w \mid (n_a(w) = n_b(w))\}$
where $n_a(w)$ is the number of a's in w and $n_b(w)$ is the number of b's in w.

L is not regular.

Proof
Suppose, by way of contradiction, that L is regular. Then the pumping lemma must hold for L. Let p be the pumping length. Consider the string $s=a^p b^p$. Note that $s \in L$ and $|s| \geq p$. The pumping lemma guarantees that s can be divided into three pieces s=xyz, with $|y|>0$, $|xy| \leq p$ and where $xy^i z \in L$ for all $i \geq 0$.

For any breakdown s=xyz with $|xy| \leq p$, y must lie entirely in the a's. Thus we have $x=a^n$, $n \geq 0$, $y = a^m$, $m>0$, and $z=a^{p-n-m}b^p$.

Consider string $s_0 = xy^0 z = xz$. This string will be $a^n a^{p-n-m} b^p = a^{p-m} b^p$. This string cannot be L because in order to be in L it would have to have the same number of a's and b's. Thus $s_0 \notin L$ which means that the pumping lemma does not hold for L, so L must not have been regular.
■

11. Consider the language F= $\{a^i b^j c^k \mid i, j, k \geq 0$ and if $i=1$ then $j=k\}$. Note that this language is not regular, because any string beginning with a single 'a', must have an equal number of b's and c's.

a. Attempt to show that F is not regular using the pumping lemma, and explain why this can't be done. (5 pts.)

Proof that L is not regular:
Suppose, by way of contradiction, that F is regular. Then the pumping lemma must hold for F. Let p be the pumping length. Consider the string $s=ab^Pc^P$. Clearly $s \in F$ and $|s| \geq p$. Therefore, the pumping lemma guarantees that s can be divided into three pieces s=xyz, with $|y|>0$, $|xy| \leq p$, and where $xy^iz \in F$ for all $i \geq 0$.

There are 3 possible breakdowns s=xyz with $|xy| \leq p$.

Case 1. x is empty and y=a.
Case 2. x is empty and $y=ab^n$.
Case 3. y lies somewhere in the bs.

Case 1. x is empty and y=a.
In this case, all s_i's are in F. Thus, the pumping lemma holds for L and this method can't be used to show that F is not regular. Bummer!

(Case 2 & Case 3 do not allow pumping. However, the pumping lemma only says that one decomposition can be pumped, not that all of them can.)

Thus, the string can be pumped, and the pumping lemma does not show that F is not a regular language.

b. Prove that F is not regular. (5 pts.)

Hint: It can be helpful to remember that regular languages are closed under the operations of complementation, concatenation, union, intersection, and Kleene closure.

Consider the language R = {ab*c*}. Clearly R is regular, as it is described by the regular expression ab*c*.

Recall that regular languages are closed under intersection.
$F \cap R = \{ab^n c^n \mid n \geq 0\}$.

Suppose, by way of contradiction, that F is regular. Then $F \cap R = \{ ab^n c^n \mid n \geq 0\}$ is regular. However, we know from earlier examples that $\{ ab^n c^n \mid n \geq 0\}$ is not regular. Thus, F must not have been regular.
■