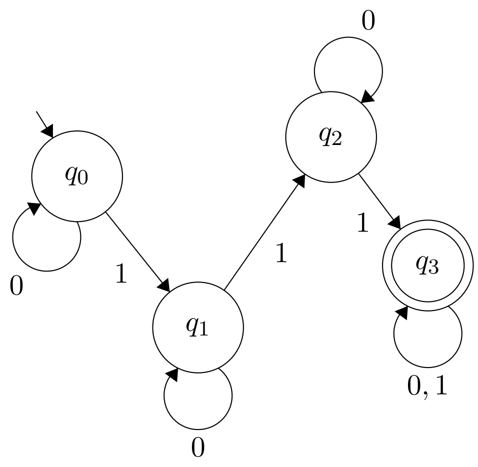
Homework 1 Solutions

COMS W3261, Summer B 2021

This homework is due **Tuesday, 7/6/2021, at 11:59PM EST**. (Monday is off: happy university holiday.) Submit to GradeScope (course code: X3JEX4).

Grading policy reminder: LaTeX is preferred, but neatly typed or handwritten solutions are acceptable. Your TAs may dock points for indecipherable writing. Proofs should be complete; that is, include enough information that a reader can clearly tell that the argument is rigorous.

1 Problem 1 (12 points)



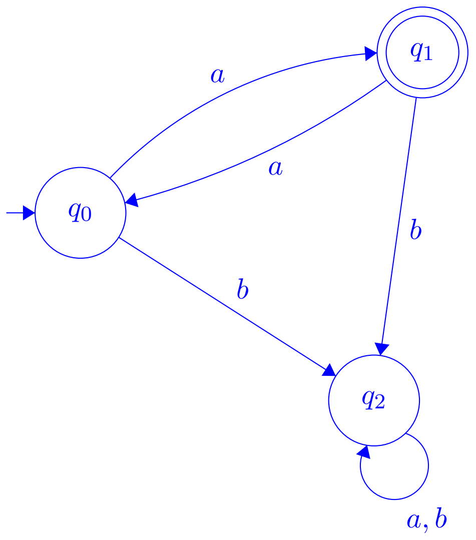
1. (4 points.) The state diagram above is defined on the alphabet $\Sigma = \{0, 1\}$. Write out its formal definition (as a 5-tuple) and describe the language that it recognizes in one sentence.

   The state diagram represents a DFA $M = (Q, \Sigma, \delta, q_0, F)$, where $Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{0, 1\}$, $F = \{q_3\}$. δ can be summarized as follows: $\delta(q, 0) = q$ for all $q \in Q$. $\delta(q_i, 1) = q_{i+1}$ for $i \in \{0, 1, 2\}$ and $\delta(q_3, 1) = q_3$.

   M recognizes the language of strings over $\{0, 1\}$ that contain at least three 1's.

2. (4 points.) Consider the DFA $M = (Q, \Sigma, \delta, q_0, F)$, where $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{a, b\}$, $F = \{q_1\}$, and δ is defined as in the following table. Draw the state diagram and describe the language that it recognizes in one sentence. (To draw state diagrams, you may wish to use the tool `http://madebyevan.com/fsm/`.)

	a	b
q0	q1	q2
q1	q0	q2
q2	q2	q2

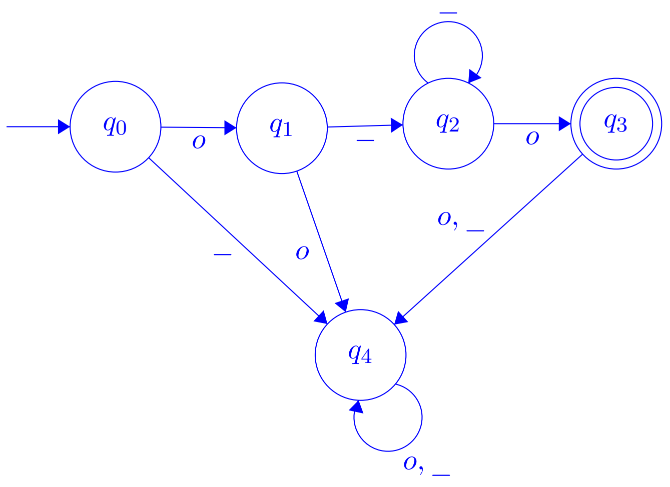


M recognizes the language of strings over $\{a, b\}$ that contain an odd number of a's and no occurrences of b.

3. (4 points.) Draw a state diagram for a DFA **with at most 5 states** that recognizes the following language over the alphabet $\Sigma = \{o, \_\}$:

$$L := \{w \mid \text{consists of one } o\text{, followed by one or more } \_ \text{ symbols, and a final } o.\}.$$

Explain in words why your DFA recognizes the language specified.



(Other equivalent state diagrams may work.)

Our state diagram contains a 'terminal state', q_4, at which computation gets stuck: if we ever find ourselves at q_4 during the execution of our DFA on a string, we read the remaining symbols and ultimately reject. The states q_0, q_1, q_2 and q_3 check to see if an input string has the form specified and send the execution to q_4 otherwise.

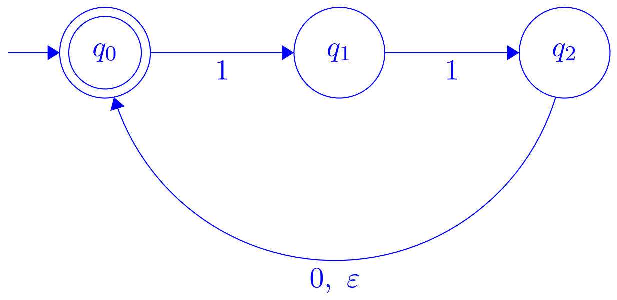
First, q_0 checks if the leading input is an o. q_1 then checks for at least one _ symbol. If either of these checks fails, our computation goes to q_4. The state q_2 reads zero or more additional _ symbols. Finally, if a second o is reached before the computation terminates, we proceed to q_3, the accept state. If at this point there are any more symbols, our string is not in the language and we go to q_4.

2 Problem 2 (4 points)

1. (0 points). Draw a state diagram for an NFA **with at most 3 states** that recognizes the regular expression

$$((1 \circ 10) \cup 11)^*.$$

Explain in words why your NFA recognizes the language specified.

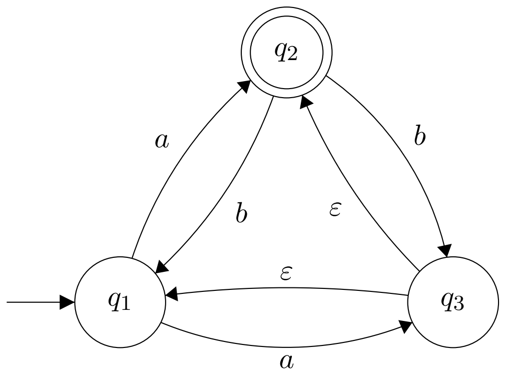


(Other equivalent state diagrams may work.)

$(1 \circ 10)$ evaluates to 110, so our regular expression is equivalent to the language of all strings that can be decomposed into '110' and '11' substrings. First note that our NFA state diagram accepts ε. If the string contains more symbols, the NFA nondeterministically 'guesses' a decomposition into '110' and '11' substrings.

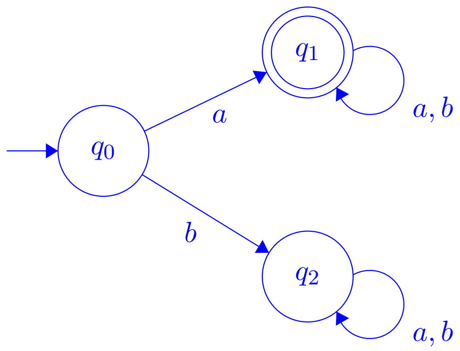
In particular, each branch of computation dies if the next substring in its decomposition does not begin with '11'. It then nondeterministically guesses both that the next substring is '11' (proceeding back to q_0 on the ε edge) or reads a '0' if this is possible, implicitly guessing that the next substring is '110'. Any branch of computation that terminates at q_0 has successively read zero or more substrings '11' or '110'.

2. (4 points). Draw the state diagram of a DFA (alphabet $\Sigma = \{a, b\}$) **with at most 3 states** that recognizes the same language as the NFA whose state diagram is pictured below. Explain in words why your DFA captures the same language as the original NFA.



Solving this question requires simplifying this NFA, which in turn requires understanding exactly what it does. You could do so by testing strings, reasoning about its structure, or following the procedure for turning any NFA into a DFA outlined in lecture (this is the proof of Theorem 1.39 in the textbook).

Either procedure reveals that if the input string is ε or starts with b, all branches of computation die (and the NFA rejects). However, if the computation starts with a, branches of computation reach all three states q_1, q_2, and q_3. If the string ends at this point, the computation accepts. Moreover, any new input symbol, whether a or b, results in some branch of computation at all three states. Thus this machine accepts all strings that begin with a. The state diagram below recognizes the same language.



3 Problem 3 (9 points)

1. (9 points.) Given languages A and B, define the XOR operation $\oplus$ as follows:

$$A \oplus B := \{x \mid x \in A \text{ or } x \in B, \text{ but } x \notin A \cap B\}.$$

Prove that the class of regular languages is closed under $\oplus$.

One way to prove this is to modify our original proof that the regular languages are closed under $\cup$. (See the proof of Theorem 1.25 in the text.)

Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ be DFAs that recognize the languages A_1 and A_2, respectively. We'll construct a new machine $M = (Q, \Sigma, \delta, q_0, F)$ that recognizes $A_1 \oplus A_2$.

As in our previous proof, let $Q = \{(r_1, r_2) \mid r_1 \in Q_1, r_2 \in Q_2\}$, i.e., the set of all pairs of states in Q_1 and Q_2, otherwise known as the Cartesian product $Q_1 \times Q_2$. We let Σ be unchanged, set $q_0 = (q_1, q_2)$, and define our transition function so that it simulates both transition functions δ_1 and δ_2: for each $(r_1, r_2) \in Q$ and $a \in \Sigma$, let $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$.

The difference from our previous construction is in the set of accept states: we define

$$F = \{(r_1, r_2) \mid (r_1 \in F_1, r_2 \notin F_2) \text{ OR } (r_1 \notin F_1, r_2 \in F_2)\}.$$

As a result, M accepts a string w precisely when exactly one of M_1 and M_2 accepts w: in other words, when $w \in A_1 \oplus A_2$.

4 Problem 4 (10 points)

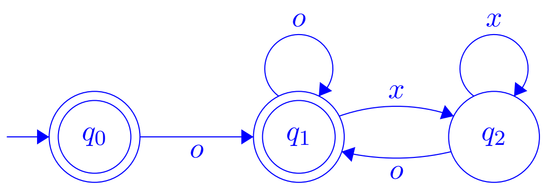
1. (10 points). Consider the language L on the alphabet $\{x, o\}$ defined as follows:

$$L := \{w \mid w \text{ contains the same number of `} xo\text{' and `} ox\text{' substrings}\}.$$

For example, the string 'xoxxo' contains two 'xo' substrings and one 'ox' substring. Prove that L is regular. (To prove this, you may draw one or more state diagrams and/or use previously proven facts about the closure of regular languages under regular operations.)

The trick here is the realization that L is the same as the language that contains (1) all strings that start and end with x, (2) all strings that start and end with o, and (3) the empty string. To see this, consider (for example) any string that starts with x, and consider reading it from left to right, counting each xo substring and each ox substring. Whenever we encounter our first o, this creates our first xo substring. We then read o's until we see an x, which is our first ox substring. Continuing this logic, we can see that the number of xo and ox substrings is the same if and only if our string ends in x. (We also accept the input string ε, which contains no xo and ox substrings.)

To prove that L is regular, it suffices to show a NFA and prove that it recognizes L, the language of all strings that start and end with the same character. Consider the following NFA N_0:



This NFA accepts the empty string ε and all strings that start and end with o. (To see this, observe that after reading the first o, our NFA is in an accept state after reading every o and a reject state after reading every x.) Define L_0 to be $L(N_0)$. By reversing the labels, we can create a NFA N_1 that recognizes the language L_1, defined as ε and all strings that start and end with x.

Thus L_0 and L_1 are regular. Moreover, $L = L_0 \cup L_1$, so L is regular by the closure of regular languages under union.

5 Problem 5 (1 point)

1. What in-class topic or problem did you find most confusing this week?

   (Any coherent response.)

2. What in-class topic or problem did you find most interesting this week?

   (Any coherent response.)

3. (Optional) Any other thoughts? Thank you!