

Homework 1 Solutions

CS 321

1.2.6

Let L be any language on a non-empty alphabet. Show that L and \overline{L} cannot both be finite.

There are two possible cases: 1) when L is infinite, and 2) when L is finite. The first case is trivial since if L is infinite then it is by definition not finite and hence it cannot be the case that L and \overline{L} are both finite.

Now consider the case when L is finite. By the definition of language complementation we know that,

$$\overline{L} = \Sigma^* - L$$

where Σ is the non-empty alphabet. We know that since Σ is non-empty that Σ^* has an infinite number of strings. Which means that $\Sigma^* - L$ must contain an infinite number of strings (since removing a finite number of elements from an infinite set still leaves us with an infinite set). Thus \overline{L} is infinite. This shows that in each of the cases either L or \overline{L} is infinite (perhaps both).

1.2.7

Is there any language L such that $\overline{L^*} = \overline{L}^*$?

The answer is that there is no such language. The reason is that for any language L we have that $\lambda \in L^*$. This means that $\overline{L^*}$ will not contain the empty string for any L . Also, this means that \overline{L}^* will contain the empty string for any L . Thus the two languages will always differ on the string λ and hence never be equal.

Additional Problem

14b. $L_2 = \{a^n b^{2n} : n \geq 0\}$

L_2 is the language containing all strings that begin with a sequence of zero or more a's followed by a sequence of b's such that the number of b's is twice

the number of a's.

15a. $L = \{w : |w| \bmod 3 = 0\}$

L is the language of strings that contain only a's and that have lengths that are a multiple of 3.

18b. $L = \{w : n_a(w) > n_b(w)\}$

L is the set of all strings that contain only a's and b's such that there are strictly more a's than b's.

18d. $L = \{w : |n_a(w) - n_b(w)| = 1\}$

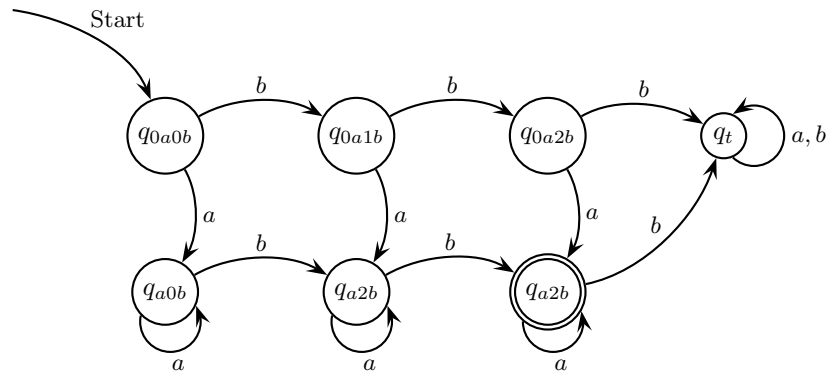
L is the set of all strings that contain only a's and b's such that the number of a's and b's differ by exactly 1.

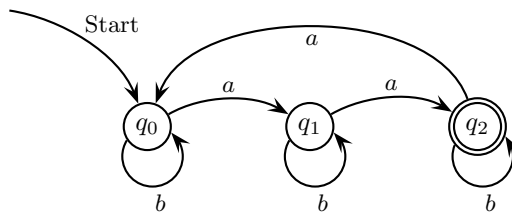
2.1.2(d)

Give a DFA that accepts the set of strings that contain at least one a and exactly two b 's.

Below the states in the DFA are labelled according to how many a's and b's were necessary in order to reach the particular node. For example, state q_{0a0b} is a state where exactly zero a's and zero b's were observed; q_{0a2b} is a state that is reached after observing zero a's and two b's; and q_{a1b} is a state that is reached after observing a positive number of a's and exactly one b. The state q_t is a terminal state from which no string will ever be accepted and is entered only after observing more than two b's.

After labelling the states in this way according to their meaning it is relatively straightforward to determine where to connect the a and b arcs coming out of each node.





2.1.7(c)

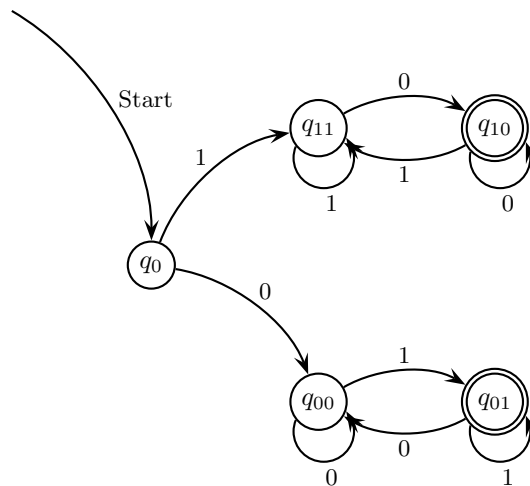
The above diagram shows a DFA for $L = \{w : n_a(w) \bmod 3 > 1\}$.

Problem 2.1.9(c)

Give a DFA that accepts all strings of 0's and 1's whose leftmost and rightmost symbols are different.

In the below DFA, a node labelled q_{xy} denotes that the node is reached whenever the input string began with an x (here x is either 0 or 1) and the most recent symbol was y (again y is either 0 or 1).

Again after assigning such meaningful labels it is relatively straightforward to determine the destination of arcs.

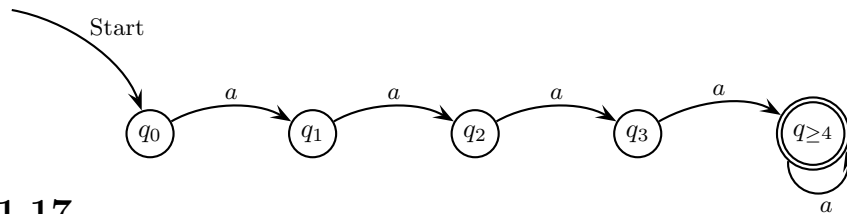


Problem 2.1.12

Show that the language $L = \{a^n : n \geq 4\}$ is regular.

For this problem, we will assume that the alphabet is equal to $\Sigma = \{a\}$. Make sure that you understand how the machine would be different if the alphabet were equal to $\Sigma = \{a, b\}$.

In order to show that the language is regular we must show that there is a DFA that accepts the language. Below is such a DFA.



2.1.17

Show that if L is regular, so is $L - \{\lambda\}$.

This problem is more abstract than your previous problems because it is asking you to show a result about all regular languages. To do this we will use the fact that any regular language L has a corresponding DFA. We will describe a generic way for transforming a DFA for L to a new DFA whose language is $L - \{\lambda\}$, which will prove that $L - \{\lambda\}$ is indeed regular provided that L is regular.

Let $M = (Q, \Sigma, q_0, \delta, F)$ be a DFA for L , i.e. $L = L(M)$. There are two cases to consider: 1) L does not contain λ , and 2) L does contain λ . The first case is trivial since we know that $L(M) = L - \{\lambda\}$ and hence is regular.

For the second case, we know that $\lambda \in L(M)$ and thus we must have $q_0 \in F$, otherwise the machine would not be able to accept λ . A naive (and incorrect) idea is to simply remove q_0 from the set of final states. However, this approach does not work since it may cause a non-empty string w in $L(M)$ to be rejected. This, for example, will happen for strings such that $\delta^*(q_0, w) = q_0$ (make sure you understand this point).

Instead, we will create a new DFA M' by adding a new initial state q'_0 to M . In addition, we will draw an arc with label a from q'_0 to state q in Q if and only if there is an arc with label a from q_0 to q in M . Everything else about M' will be identical to M . The result of doing this is that λ will no longer be accepted by M' since q'_0 is not a final state of M' . However, after reading the first character into M' the behavior of the machine will be exactly the same as if it had started in q_0 . Thus, M' accepts a non-empty string w if and only if M accepts w showing that M' accepts the language $L - \{\lambda\}$.

More formally, we define M' as the tuple $(Q \cup \{q'_0\}, \Sigma, q'_0, \delta', F)$, where q'_0 is the new initial state, and $\delta'(q, a) = \delta(q, a)$ for all $q \in Q$ and $a \in \Sigma$ and $\delta'(q'_0, a) = \delta(q_0, a)$ for all $a \in \Sigma$.