

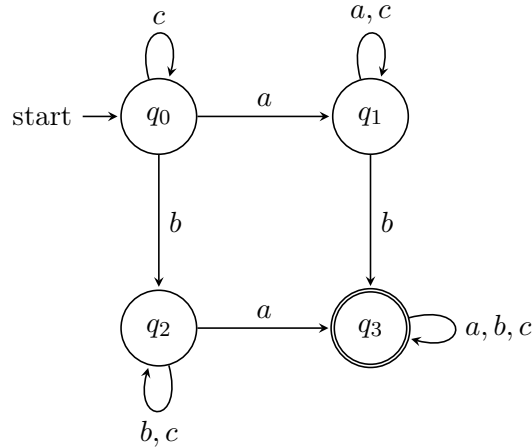
Handout 1: DFAs and Regular Languages

Ruby Chang, Keir Dorchon, Zachary Thayer

3. Practice problems with Solutions

Problem 1

Let D be the following DFA over the alphabet $\Sigma = \{a, b, c\}$.



- Write out the formal definition of D (as a 5-tuple). Describe the transition function in a table.
- What is the language recognized by D ? Give your answer using set notation (such as $L(D) = \{x \in \{a, b\}^* \mid x \text{ starts and ends with an } a\}$).

Solution.

(a) $D = (Q, \Sigma, \delta, q_0, F)$ where $Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{a, b, c\}$, q_0 is the start state, and $F = \{q_3\}$. The transition function δ is given by:

δ	a	b	c
q_0	q_1	q_2	q_0
q_1	q_1	q_3	q_1
q_2	q_3	q_2	q_2
q_3	q_3	q_3	q_3

(b)

$$L(D) = \{w \in \{a, b, c\}^* \mid w \text{ contains at least one } a \text{ and at least one } b\}.$$

Explanation:¹ q_0 tracks that neither an a nor a b has been seen yet, q_1 tracks that at least one a , but no b , has been seen, q_2 tracks that at least one b , but no a , has been seen, and q_3 (the accepting state) tracks that both have been seen. Since c never affects this, all four states self-loop on c , and q_3 is a sink once reached, D accepts exactly the strings containing both an a and a b .

¹Not required in the problem, but included for your understanding/intuition

Problem 2

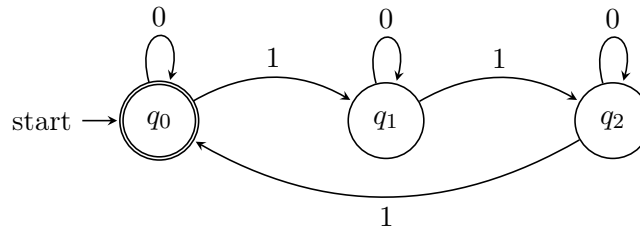
Give a DFA D recognizing the language

$$L = \{w \in \{0, 1\}^* \mid \text{the number of 1's in } w \text{ is divisible by 3}\}$$

over $\Sigma = \{0, 1\}$. For instance, 000 has zero 1's and is in the language, and 111011 has five 1's and is not in the language, while 111 has three 1's and is in the language.

Draw the state diagram. Give your states useful names, and for each state, write down what must be true of a string for its computation to end in that state.

Solution.



What each state represents. State q_i (for $i \in \{0, 1, 2\}$) is for strings whose number of 1's is $\equiv i \pmod{3}$ (that is, the computation on a string ends in state q_i if the number of 1s in the string is $i \pmod{3}$). In particular, q_0 is for strings with a number of 1's divisible by 3, which is exactly what we want to accept.

(Bonus: a proof that this is correct.) As D reads w , its state always equals q_i where i is the number of 1's read so far, mod 3 — every 0 leaves the state (and the count) unchanged, and every 1 advances the state from q_i to $q_{(i+1) \bmod 3}$, exactly matching the count increasing by one. So after reading all of w , the DFA lands in q_j , where j is the total number of 1's in w , mod 3.

- If $w \in L$, its number of 1's is divisible by 3, so $j = 0$, and the computation ends in $q_0 \in F$.
- If $w \notin L$, its number of 1's is not divisible by 3, so $j \neq 0$, and the computation ends in q_1 or q_2 , neither in F .

Hence $L(D) = L$.

Problem 3

Draw a state diagram for a DFA that recognizes the following language.

$$L = \{w \in \{a, b\}^* \mid w \text{ has even length and an odd number of } a\text{'s}\}.$$

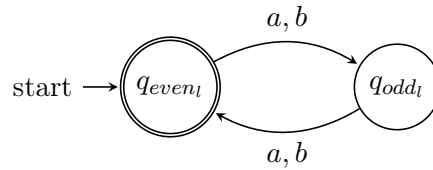
Solution.

Rather than directly come up with a DFA, we will first split the condition into its two simpler parts to write L as the intersection of two languages. Let

$$L_1 = \{w \mid w \text{ has even length}\}, \text{ and } L_2 = \{w \mid w \text{ has an odd number of } a\text{'s}\}.$$

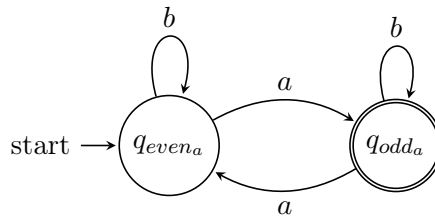
Then $L = L_1 \cap L_2$. We can draw separate DFAs for L_1 and L_2 , and then construct a DFA for the intersection of the two, as seen in class.

For L_1 :



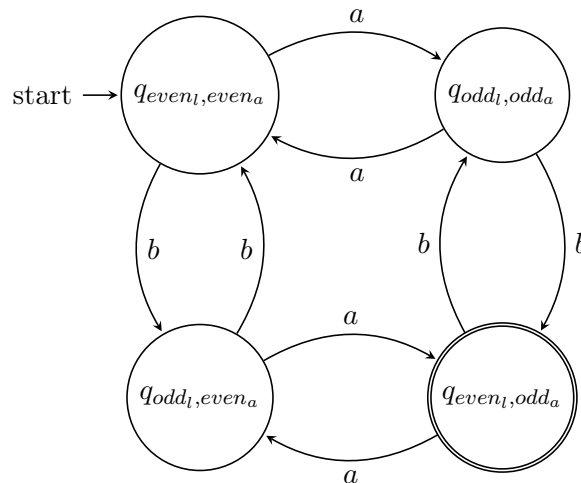
The state q_{even_l} represents an even length, and q_{odd_l} represents an odd length. The transition for any character alternates between the two states, so this DFA accepts if and only if a string has an even number of characters.

For L_2 :



The state q_{even_a} represents an even number of a 's, and q_{odd_a} represents an odd number of a 's. A b will not change the current state, so it does not affect the result. An a will alternate between the two states, so this DFA accepts if and only if there are an odd number of a 's.

DFA recognizing the intersection, using the cartesian product of their states:



We recommend looking at the intersection construction section of this handout to see how this directly exemplifies the general form given there. For some more details though, observe that we named our states using pairs of states from the above two DFAs. Each transition follows what each of the two states would do. For example $\delta((q_{even_l}, q_{even_a}), a)$ will apply the transition function of L_1 on (q_{even_l}, a) and the transition function of L_2 on (q_{even_a}, a) to see where each DFA would have gone, and then we transition to the state labeled with both of those names (in this case q_{odd_l}, q_{odd_a}).

Problem 4

Prove that the following languages are not regular.

- (a) $L = \{w \in \{a, b\}^* \mid w \text{ is a palindrome, } |w| > 0\}$
- (b) (Challenge) $L = \{xy \in \{0, 1\}^* \mid |x| = |y|, x \text{ has an even number of 1's, } y \text{ has an even number of 0's}\}$

Solution.

- (a) **Myhill–Nerode:** Let $S = \{a^n b^n \mid n > 0\}$. Since $a^n b^n \neq a^m b^m$ for any $n < m$, there are infinitely many strings in S . Given $x = a^n b^n$ and $y = a^m b^m$ for $n < m$, let $z = b^n a^n$. This z is a distinguishing extension for x and y , since $xz = a^n b^n b^n a^n$ is a palindrome with nonzero length, but $yz = a^m b^m b^n a^n$ is not a palindrome. This means that xz is in L but yz is not, so x and y are distinguishable. Since each pair of strings in S is distinguishable, there are infinitely many equivalence classes in $\sim_L$, and by Myhill–Nerode L is not regular.

Closure Properties: As an alternative, we can notice that this L is very similar to a language we mentioned in class and proved in the notes:

$$L' = \{w \mid w \text{ is a palindrome}\}.$$

Assume for the sake of contradiction that L is regular, and let $L_1 = \{\varepsilon\}$. L_1 is finite, and therefore regular. We have

$$L' = L \cup \{\varepsilon\}.$$

This equality is true because the extra condition $|w| > 0$ only excludes ε , so it is the only string in L' that is missing from L . Since we assumed L is regular, L' must also be regular, since regular languages are closed under unions. But we have already seen in the lecture notes that L' is not regular, through a proof using Myhill–Nerode like above. This is a contradiction, so L must not be regular.

- (b) (Challenge) **Myhill–Nerode:** Let $S = \{0^{2n} \mid n > 0\}$. Since $0^{2n} \neq 0^{2m}$ for any $m < n$, there are infinitely many strings in S .

Given $x = 0^{2n}$ and $y = 0^{2m}$ for $m < n$, let $z = 1(0)^{2n-2}1$. We choose this z specifically so that xz will be in L , but yz will not. Notice that

$$|x| = |(0)^{2n}| = 2n$$

and

$$|z| = |1(0)^{2n-2}1| = 1 + |(0)^{2n-2}| + 1 = 1 + 2n - 2 + 1 = 2n.$$

This shows that the two concatenated parts of xz have equal length, and we can see that the first part will have an even number of 1's (exactly zero), and the second part will have an even number of 0's ($2n-2$).

On the other hand, yz is not in L . To see this, expand yz :

$$yz = (0)^{2m}1(0)^{2n-2}1.$$

This string can still be split into two equal-length parts. However, since $m < n$, the midpoint of the string is greater than $2m$. Because of this, the first part of the string will contain the first 1 in z , but not the other. Since the first part has an odd number of 1's, yz cannot be in L .

We have shown that $xz \in L$ but $yz \notin L$, so x and y are distinguishable by L . Since this argument works for any pair of strings in S , there are infinitely many equivalence classes in $\sim_L$, and by Myhill–Nerode L is not regular.

Problem 5

Let $\Sigma = \{a, b\}$. For each part, give an example of two languages $L, L' \subseteq \Sigma^*$ satisfying the conditions below. Don't forget to prove your claims.

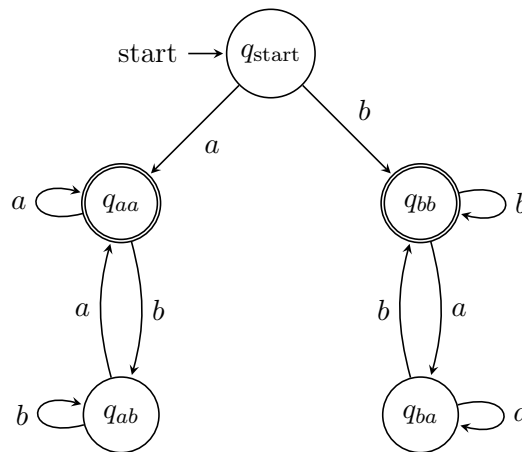
(a) L is regular, L' is not regular, but $L \cup L'$ is regular.

Solution:

We choose $L = \{w \in \{a, b\}^* \mid w \text{ starts and ends with the same character and } |w| > 0\}$, and $L' = \{w \in \{a, b\}^* \mid w \text{ is a palindrome and } |w| > 0\}$.

In the above problem we just proved L' is not regular.

We prove L is regular by providing a DFA below:



Observe that our DFA splits strings based on if they start with an a or a b . The states labeled q_{ij} indicate the first character of the string was i and the last, or most recently seen character was j . Hence why q_{aa} and q_{bb} are accept states.

Observe that $L' \subseteq L$, since all palindromes of nonzero length start and end with the same character necessarily. Thus $L' \cup L = L$ which we've already proven to be regular.

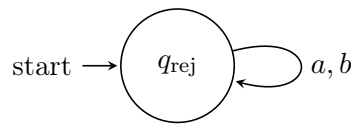
Note that simpler examples are also possible (can you think of one?)

(b) L is regular, L' is not regular, and $L \cup L'$ is not regular.

Solution:

Let $L' = \{a^n b^n \mid n \geq 0\}$ and let $L = \{\}$, that is, L contains no strings whatsoever. We've seen in class that L' is not regular.

We prove L is regular by providing a DFA for it below:

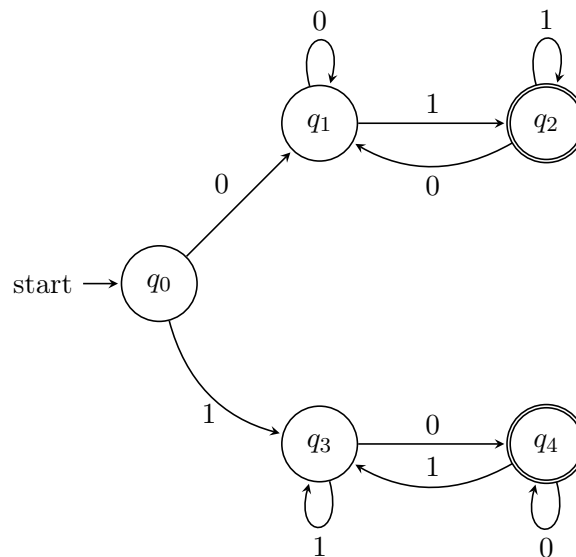


As you can see, the above DFA remains in a reject state regardless of the input string given to it, and thus rejects all strings, correctly recognizing L .

Observe that $L' \cup L = L'$ since L contains no elements, since we know L' is not regular already we are done!

Extra Practice Problem 1

Let D' be the following DFA over the alphabet $\Sigma = \{0, 1\}$.



What is the language recognized by D' ? Give your answer in set notation, then a short English description explaining your answer.

Solution.

$$L(D') = \{w \in \{0, 1\}^* \mid |w| \geq 1 \text{ and the first and last characters of } w \text{ are different}\}.$$

The start state q_0 represents the empty prefix. After the first character has been read, each state records the first character and the most recently read character. States q_1 and q_2 represent strings beginning with 0, while q_3 and q_4 represent strings beginning with 1. Reading the same character as the current last character leaves the state unchanged. Reading the other character switches between the matching and mismatching states within the relevant pair. The first character never changes. Since q_2 and q_4 are exactly the states where the first and last characters differ, these are the accepting states.

Extra Practice Problem 2

Give a DFA D recognizing the language

$$L = \{w \in \{0, 1, \dots, 9\}^+ \mid w \text{ is the decimal representation of a number divisible by 3}\}$$

(we allow leading zeros; for instance 012 represents the number 12; note w must be nonempty). For instance, 12 and 102 are in the language, but 13 is not.

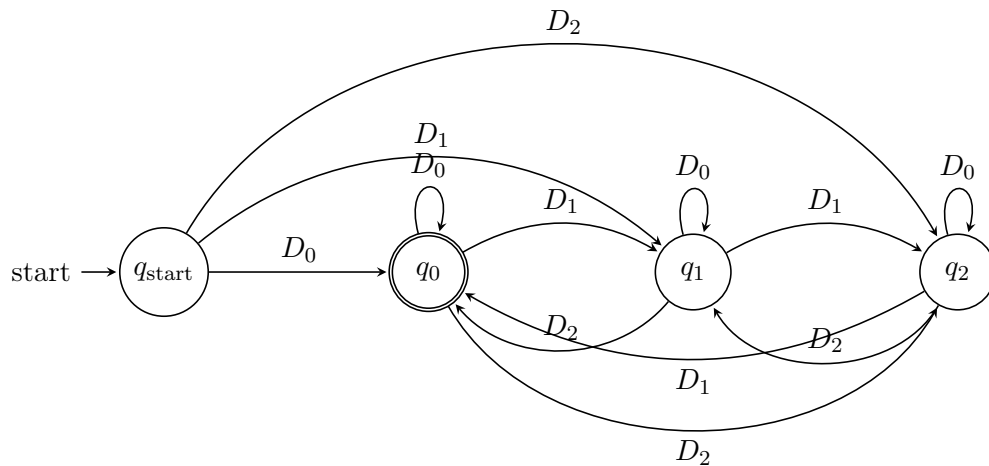
Draw the state diagram. Give your states useful names, and for each state, write down what must be true of a string for its computation to end in that state.

Solution.

Key idea. Group the digits $0, \dots, 9$ by their remainder mod 3:

$$D_0 = \{0, 3, 6, 9\}, \quad D_1 = \{1, 4, 7\}, \quad D_2 = \{2, 5, 8\}.$$

If the digits read so far form a number v , and we read one more digit d , the new number is $10v + d$. Since $10 \equiv 1 \pmod{3}$, we get $10v + d \equiv v + d \pmod{3}$ — so only $v \bmod 3$ and $d \bmod 3$ matter, and we can track everything with a few states.



What each state represents. State q_{start} is for the empty prefix, before any digit has been read; it is not accepting, since ε is not a decimal representation of any number. State q_i (for $i \in \{0, 1, 2\}$) is for nonempty strings that represent a number $\equiv i \pmod{3}$. In particular, q_0 is for nonempty strings representing a number divisible by 3, which is exactly what we want to accept.

Transitions. From q_{start} , reading a digit d moves directly to $q_{d \bmod 3}$: this is the value of the single digit read so far, mod 3. From any q_i , reading a digit from D_r (i.e. a digit $\equiv r \pmod{3}$) moves the DFA from q_i to $q_{(i+r) \bmod 3}$, matching the update rule $v \mapsto v + d \pmod{3}$ derived above.

(Bonus: a proof that this is correct.) On the empty string, D stays in q_{start} , which is not accepting, matching $\varepsilon \notin L$. For a nonempty string w , once the first digit is read, the DFA's state after reading any prefix always equals q_i , where i is the value of the number formed by that prefix, mod 3 — this is exactly what the transitions above are built to maintain. So after reading all of w , the DFA lands in state q_j , where j is the value of w (as a decimal number) mod 3.

- If $w \in L$, the number represented by w is divisible by 3, so $j = 0$, and the computation ends in $q_0 \in F$.
- If $w \notin L$ (and $w \neq \varepsilon$), the number represented by w is not divisible by 3, so $j \neq 0$, and the computation ends in q_1 or q_2 , neither in F .

Hence $L(D) = L$.