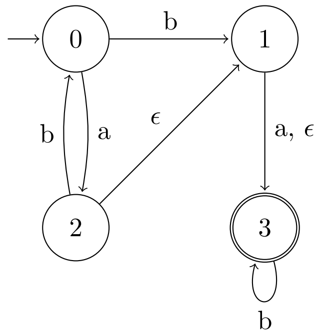
Homework 2 Solutions

Formal Languages and Automata (CS322)

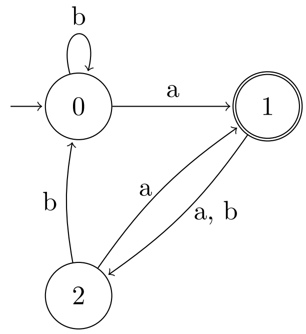
Last update: October 17, 2025

1. Answer the following questions.

δ_1:



δ_3:



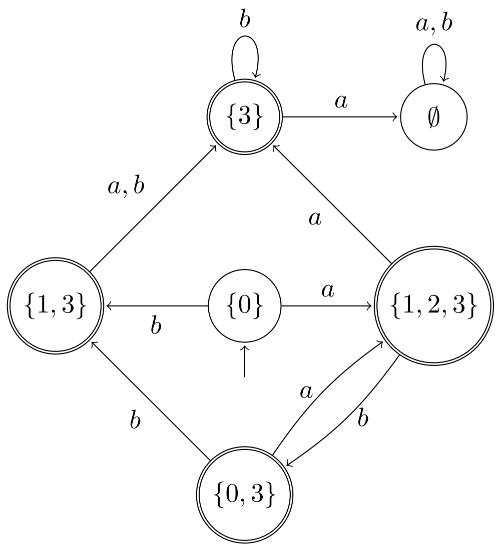
(a) Convert the NFA $M_1 = (\{0, 1, 2, 3\}, \{a, b\}, \delta_1, 0, \{3\})$ into an equivalent DFA.

(b) Convert the regular expression $(1 \cup 0^+)1^*(10)^+$ over the alphabet $\Sigma_2 = \{0, 1\}$ into an equivalent NFA.

(c) Convert the DFA $M_3 = (\{0, 1, 2\}, \{a, b\}, \delta_3, 0, \{1\})$ into an equivalent regular expression. Give a generalized nondeterministic finite automaton (GNFA) for each conversion step in which one state is reduced.

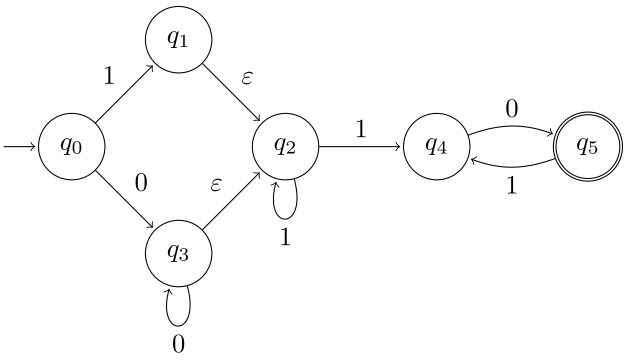
Solution.

(a) $M_1' = (\{\{0\}, \{3\}, \{1, 3\}, \{0, 3\}, \{1, 2, 3\}, \emptyset\}, \{a, b\}, \delta_1', \{0\}, \{\{3\}, \{0, 3\}, \{1, 3\}, \{1, 2, 3\}\})$, where δ_1' is given by the following transition diagram.

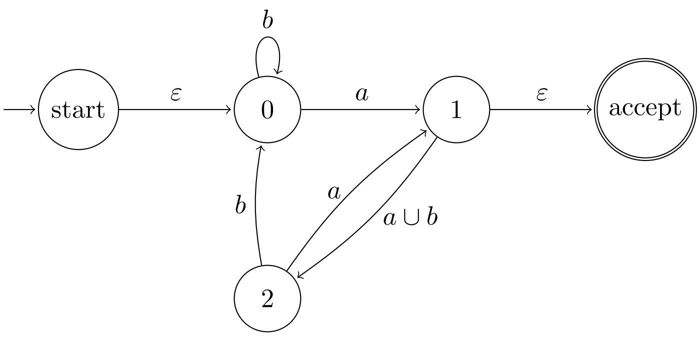


There are several answers to (b) and (c). We provide a possible answer for each.

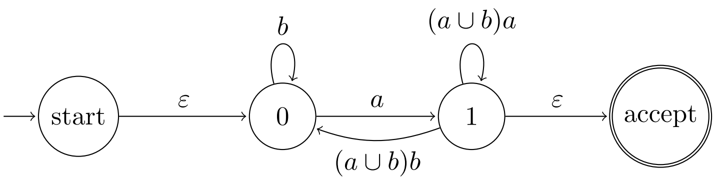
(b) $M_2' = (Q, \{0,1\}, \delta_2, q_0, \{q_5\})$, where $Q = \{q_i : 0 \le i \le 5\}$, and δ_2 is given by the following transition diagram.



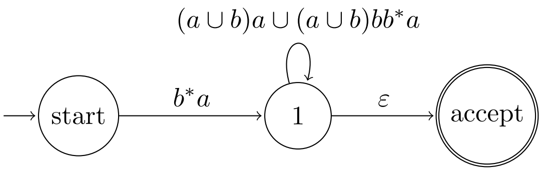
(c) 1) Add a new start state with an ε arrow to the old start state and a new accept state with an ε arrow from the old accept state, and replace arrows with multiple labels with a single arrow whose label is the union of the previous labels. We omit the arrows labeled $\emptyset$ between states that had no arrows.



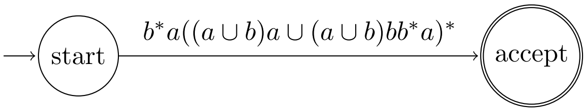
2) We remove state 2.



3) We remove state 0.



4) We remove state 1.



We have an equivalent regular expression $b^*a((a \cup b)a \cup (a \cup b)bb^*a)^*$, which can be abbreviated into $b^*a((a \cup b)b^*a)^*$.

$\square$

2. Show that the class of regular languages is closed under the following operations. We say x is a *prefix* of a string y if there is a string z such that $xz = y$, and x is a *proper prefix* of y if additionally, we have $x \neq y$. Let $L \subseteq \Sigma^*$.

(a) $\mathsf{REVERSE}(L) = \{w^R : w \in L\}$;

(b) $\mathsf{NOEXTEND}(L) = \{w \in L : w \text{ is not a proper prefix of any string in } L\}$;

(c) $\mathsf{HALVES}(L) = \{w \in \Sigma^* : \text{there is } u \in \Sigma^* \text{ such that } |u| = |w| \text{ and } wu \in L\}$.

Solution.

(a) If L is regular, then there exists an NFA $M = (Q, \Sigma, \delta, q_0, F)$ recognizing L. To construct an automaton for $\mathsf{REVERSE}(L) = \left\{w^R : w \in L\right\}$, we define an NFA $M' = (Q, \Sigma, \delta', F, q_0)$ obtained by reversing all transitions of M and swapping the roles of the start and accepting states. Formally, for every $p, q \in Q$ and $a \in \Sigma$, set $p \in \delta'(q, a)$ iff $q \in \delta(p, a)$. Then M' basically starts from any state in F (which is effectively achievable by adding a dummy start state and adding ϵ-transitions from the dummy start state to all the states in F) and accepts if it can reach q_0 after reading the input. It is not very difficulty to see that a word w is accepted by M' iff M accepts w^R, so $L(M') = \mathsf{REVERSE}(L)$. Because NFAs recognize exactly the regular languages, $\mathsf{REVERSE}(L)$ is regular.

(b) Since L is regular, there is a DFA $M = (Q, \Sigma, \delta, q_0, F)$ recognizing L. Consider the DFA $N = (Q, \Sigma, \delta, q_0, F')$ with F' defined as follows:

$$F' = \{q \in F : \text{there is no } w \in \Sigma^+ \text{ such that } \hat{\delta}(q, w) \in F\}.$$

We claim that $L(N) = \mathsf{NOEXTEND}(L)$.

Proof of the claim. Suppose N accepts $w \in \Sigma^*$. Then $q := \hat{\delta}(q_0, w) \in F'$, hence $q \in F$ and, by the definition of F', for every $u \in \Sigma^+$ we have $\hat{\delta}(q, u) \notin F$. Equivalently, for every $u \in \Sigma^+$, $\hat{\delta}(q_0, wu) = \hat{\delta}(\hat{\delta}(q_0, w), u) \notin F$. Thus $w \in L$ (since $q \in F$) and no proper extension wu with $u \in \Sigma^+$ lies in L; hence $w \in \mathsf{NOEXTEND}(L)$.

Conversely, suppose $w \in \mathsf{NOEXTEND}(L)$. Then $q := \hat{\delta}(q_0, w) \in F$ (since $w \in L$). If there were $u \in \Sigma^+$ with $\hat{\delta}(q, u) \in F$, we would have $\hat{\delta}(q_0, wu) \in F$, contradicting that w is not a proper prefix of any string in L. Therefore no such u exists and $q \in F'$, so N accepts w. Hence $L(N) = \mathsf{NOEXTEND}(L)$. □

We have *defined* a DFA recognizing $\mathsf{NOEXTEND}(L)$. However, we are not done yet. The caveat lies in the nature of the definition. From the definition itself, it is not immediately clear how we will be actually computing F' from a given M (or whether we will be able to compute such F' at all). This leaves some gap in our argument: If somehow we can show that that function that is supposed to compute F' from a given M is not computable, then we do not necessarily have a well-defined DFA N to end up with, leading for this approach to not work. However in this specific case, we show below that F' is computable from a given M, and we will be done.

Form the directed graph $G = (Q, E)$ with $E = \{(p, \delta(p, a)) : p \in Q,\ a \in \Sigma\}$, Compute the set $R = \{\, p \in Q : \exists v \in \Sigma^*\ \hat{\delta}(p, v) \in F \,\}$, the set of states that can reach some accepting state (allowing length-0 paths). Practically this is obtained by doing a single breadth-first (or depth-first) search on the *reverse* graph $G_{\text{rev}} = (Q, E_{\text{rev}})$ with $E_{\text{rev}} = \{(q, p) : (p, q) \in E\}$, starting from all nodes in F; the visited nodes are exactly R. Then $F' = \{\, q \in F : \forall a \in \Sigma\ \hat{\delta}(q, a) \notin R \,\}$, because $q \in F'$ iff no *nonempty* word from q leads to F, i.e. none of the immediate successors of q can reach F. The whole computation takes time $O(|Q| + |E|) = O(|Q| \cdot |\Sigma|)$.

(c) First, since L is regular, it has a DFA $M = (Q, \Sigma, \delta, q_0, F)$ such that $L(M) = L$.

The intuition is that we construct a DFA $N = (Q \times (2^Q)^Q, \Sigma, \delta', (q_0, f_0), F')$, so that $L(N) = \mathsf{HALVES}(L)$, whose states are pairs keeping track of some information as the machine N reads any input string:

- The first component q tracks $\delta(q_0, w)$ while reading the input w (the first half we see in the language description).

- the second component is a function $f\colon Q \to 2^Q$. After reading k input symbols (k equals the number of symbols N has read to arrive at current q), $f(a)$ will exactly be the set of states reachable from a by *some* word of length k (we name such (unique) f the "k-seeking function" henceforth). Thus, when the input ends (length $n = |w|$), $f(q)$ is the set of states reachable from $q = \delta(q_0, w)$ by *some* word u of length n.

As N reads one input symbols, we update the first component q precisely in the same we as in M using δ and update the second component, which is supposed to be a k-seeking function if N has read k symbols, to be the $(k+1)$-seeking function.

After N finishes reading the input string and lands on q in the first component (note that $q = \hat{\delta}(q_0, w)$ with w being the input string), we check if seeking $|w|$ times from q lands M in a state in F; that is, whether the $|w|$-seeking function f (which can be retrieved by looking at the second component of the state N finally sits on) admits $f(q) \cap F \neq \emptyset$.

Before finishing up, we figure out, given the k-seeking function $f\colon Q \to 2^Q$, for $k \geq 0$, how we can compute the $(k+1)$-seeking function $\mathcal{N}(f) = f'$. This is given by the following easily computable formula (latter two equalities briefly and partly argues correctness):

$$f'(a) = \bigcup_{y \in \Sigma} \bigcup_{b \in f(a)} \{\delta(b, y)\} = \left\{\delta(\delta(a, v), y) : v \in \Sigma^k, y \in \Sigma\right\} = \left\{\delta(a, vy) : vy \in \Sigma^{k+1}\right\}.$$

Note that the 0-seeking function $f\colon Q \to 2^Q$ is just $f(a) = \{a\}$.

Putting it all together, we give a concrete description of $N = (Q \times (2^Q)^Q, \Sigma, \delta', (q_0, f_0), F')$ recognizing $\mathsf{HALVES}(L)$:

- $f_0\colon Q \to 2^Q$ is given by $f_0(a) = \{a\}$ for all $a \in Q$;
- for $x \in \Sigma$, the transition function δ' is given by $\delta'((q, f), x) = (\delta(q, x), \mathcal{N}(f))$, following the definition of $\mathcal{N}$ above; and
- the set of accepting states given by $F' = \{(q, f) \in Q \times (2^Q)^Q : f(q) \cap F \neq \emptyset\}$.

$\square$

3. Use the pumping lemma to show that the following languages are not regular. We denote the numerical value of the binary sequence w by $[w]_2$, e.g., $[110]_2 = 6$.

(a) $L_1 = \{a^n b^m : n \neq m\}$;

(b) $L_2 = \{w \in \{a,b\}^* : w = w^R\}$;

(c) Consider the alphabet $\Sigma_3 = \{0,1\}^3$.

$$L_3 = \left\{ \begin{pmatrix} a_1 \\ b_1 \\ c_1 \end{pmatrix} \begin{pmatrix} a_2 \\ b_2 \\ c_2 \end{pmatrix} \cdots \begin{pmatrix} a_n \\ b_n \\ c_n \end{pmatrix} \in \Sigma_3^* : [a_1 a_2 \dots a_n]_2 \cdot [b_1 b_2 \dots b_n]_2 = [c_1 c_2 \dots c_n]_2 \right\}.$$

Solution.

(a) For every positive number $\forall p$, consider $\exists w = a^p b^{p+p!} \in L_1$. For every proper split $xyz = w$ with $|y| \geq 1$ and $|xy| \leq p$, let $|x| = s, |y| = t$. Notice that $x = a^s, y = a^t$ since xy is prefix of w. Then, consider $xy^i z$ where $i = 1 + \frac{p!}{t}$. $i \in \mathbb{N}$ because $t \leq p$. So, $xy^i z = (a^s)(a^t)^{1+\frac{p!}{t}}(a^{p-s-t} b^{p+p!}) = (a^s)(a^{t+p!})(a^{p-s-t} b^{p+p!}) = a^{p+p!} b^{p+p!}$. It means that there exists $i \geq 0$ with $xy^i z \notin L_1$. Therefore, L_1 is not regular by contrapositive of pumping lemma.

(b) For every positive number $\forall p$, consider $\exists w = a^p b a^p \in L_2$. For every proper split $xyz = w$ with $|y| \geq 1$ and $|xy| \leq p$, let $|x| = s, |y| = t$. Notice that $x = a^s, y = a^t$ since xy is prefix of w. Then, consider $xy^2 z = a^{p+t} b a^p$. According rule of split, $|y| = t \geq 1$ and $p + t \neq p$. Therefore, $(xy^2 z) \neq (xy^2 z)^R$. It means that there exists $i = 2 \geq 0$ with $xy^i z \notin L_2$. Therefore, L_2 is not regular by contrapositive of pumping lemma.

(c) For every positive number $\forall p$, consider $\exists w \in L_3$ where $|w| = 2p+1$, and $a_1 a_2 \dots a_n = b_1 b_2 \dots b_n = 0^p 1 0^p$, $c_1 c_2 \dots c_n = 10^{2p}$. For every proper split $xyz = w$ with $|y| \geq 1$ and $|xy| \leq p$, let $|x| = s, |y| = t$. Then, consider $xy^0 z = xz$.

- If $|x| = s = 0$

  Then, $xy^0 z = z = \begin{pmatrix} a_{t+1} \\ b_{t+1} \\ c_{t+1} \end{pmatrix} \cdots \begin{pmatrix} a_n \\ b_n \\ c_n \end{pmatrix}$. We can observe that $[a_{t+1} \dots a_n]_2 = [b_{t+1} \dots b_n]_2 = [0^{p-t} 1 0^p]_2 = 2^p$, and $[c_{t+1} \dots c_n]_2 = [0^{2p-(t-1)}]_2 = 0$. So, $[a_{t+1} \dots a_n]_2 \cdot [b_{t+1} \dots b_n]_2 \neq [c_{t+1} \dots c_n]_2$ and $xy^0 z \notin L_3$.
- If $|x| = s \neq 0$

  Then, $xy^0 z = \begin{pmatrix} a_1 \\ b_1 \\ c_1 \end{pmatrix} \cdots \begin{pmatrix} a_s \\ b_s \\ c_s \end{pmatrix} \begin{pmatrix} a_{s+t+1} \\ b_{s+t+1} \\ c_{s+t+1} \end{pmatrix} \cdots \begin{pmatrix} a_n \\ b_n \\ c_n \end{pmatrix}$. We can observe that $[a_1 \dots a_s a_{s+t+1} \dots a_n]_2 = [b_1 \dots b_s b_{s+t+1} \dots b_n]_2 = [0^{p-t} 1 0^p]_2 = 2^p$, and $[c_1 \dots c_s c_{s+t+1} \dots c_n]_2 = [10^{2p-t}]_2 = 2^{2p-t} < 2^{2p}$. So, $xy^0 z \notin L_3$.

It means that there exists $i = 0 \geq 0$ with $xy^i z \notin L_3$. Therefore, L_3 is not regular by contrapositive of pumping lemma.

$\square$

4. Let Σ be an alphabet. The *vocabulary for string over* Σ, denoted by τ_Σ, consists of a binary relation $<$, and a unary relation P_a for each $a \in \Sigma$. A string $w = w_1 \dots w_n$ over Σ is seen as a τ_Σ-structure $(U, <, (P_a)_{a\in\Sigma})$ with the universe $U = \{1, 2, \dots, n\}$ corresponding to the positions in the string. The binary relation $<$ is the usual linear order on the natural numbers, which expresses "the position i precedes the position j in the string", and the unary relation P_a expresses "the i-th position in the string contains the symbol a", i.e., $P_a = \{i \in [n] : w_i = a\}$. For instance, the string $aabb$ over the alphabet $\Sigma = \{a, b\}$ is a τ_Σ-structure $(U, <, (P_x)_{x\in\Sigma})$ with $U = \{1, 2, 3, 4\}$, $P_a = \{1, 2\}$, and $P_b = \{3, 4\}$.

We say that an MSO-sentence φ over τ_Σ expresses the language $L \subseteq \Sigma^*$ if for every string w over Σ, it holds that

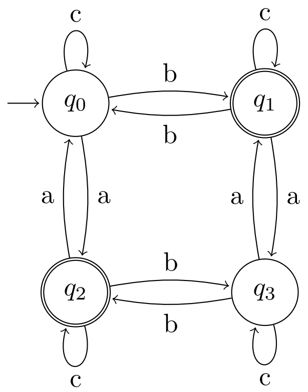
$$w \in L \text{ if and only if } w \models \varphi.$$

For example, Let $\Sigma = \{a, b\}$. The following MSO sentence φ expresses the language described by the regular expression a^*b^*.

$$\varphi \coloneqq \forall y[P_a(y)] \vee \exists x \forall y[(y < x \to P_a(y)) \wedge ((x < y \vee x = y) \to P_b(y))]$$

Give MSO-sentences over τ_Σ expressing the following languages over $\Sigma = \{a, b, c\}$.

δ:



(a) $L(R)$, where $R = (aa)^*(b \cup c)^*a^+$;

(b) $L(M)$, where $M = (Q, \Sigma, \delta, q_0, \{q_1, q_2\})$ with $Q = \{q_0, q_1, q_2, q_3\}$, and δ is given by the state diagram above.

Solution. There are several answers to this question. We provide a possible answer.

(a) We use the notation $x \leq y \coloneqq x = y \vee x < y$. A string in $L(R)$ can be divided into three parts. The first part has an even number of a's, the second part contains b's and c's, and the last part consists only of a's with at least one a. Note that the first and the second parts can be empty.

- $\varphi_{min}(x, A) \coloneqq x \in A \wedge \forall y(y \in A \to x \leq y)$
- $\varphi_{max}(x, A) \coloneqq x \in A \wedge \forall y(y \in A \to y \leq x)$
- $\varphi_{disjoint}(A_1, A_2) \coloneqq \forall x\Big((x \in A_1) \leftrightarrow \neg(x \in A_2)\Big)$
- $\varphi_{succ}(x, y) \coloneqq x < y \wedge \forall z\Big[(x \leq z \wedge z \leq y) \to (z = x \vee z = y)\Big]$
- $\varphi_{partition}(A, B, C) \coloneqq \forall x(x \in A \vee x \in B \vee x \in C) \wedge$
  $\exists x_1 \exists x_2\Big(\forall y(y < x_1 \leftrightarrow y \in A) \wedge \forall y\big((x_1 \leq y \wedge y < x_2) \leftrightarrow y \in B\big) \wedge \forall y(x_2 \leq y \leftrightarrow y \in C)\Big)$
- $\varphi_{first}(A) \coloneqq \forall x\big(x \in A \to P_a(x)\big) \wedge \exists A_1 \exists A_2\Bigg[\forall x\big(x \in A \leftrightarrow (x \in A_1 \vee x \in A_2)\big) \wedge$
  $\varphi_{disjoint}(A_1, A_2) \wedge \forall x(\varphi_{min}(x, A) \to x \in A_1) \wedge \forall x(\varphi_{max}(x, A) \to x \in A_2) \wedge$
  $\forall x \forall y\Bigg(\Big(x \in A \wedge y \in A \wedge \varphi_{succ}(x, y)\Big) \to \Big((x \in A_1 \wedge y \in A_2) \vee (x \in A_2 \wedge y \in A_1)\Big)\Bigg)\Bigg]$

- $\varphi_{second}(B) \coloneqq \forall x\Big(x \in B \to (P_b(x) \vee P_c(x))\Big)$
- $\varphi_{last}(C) \coloneqq \exists x(x \in C \wedge P_a(x)) \wedge \forall x(x \in C \to P_a(x))$

The language $L(R)$ can be expressed by the following MSO-sentence φ.

$$\varphi \coloneqq \exists A \exists B \exists C \Big[\varphi_{partition}(A, B, C) \wedge \varphi_{first}(A) \wedge \varphi_{second}(B) \wedge \varphi_{last}(C)\Big]$$

(b) $L(M)$ is the language, where the number of a's and b's is odd. You can express this language with an MSO-sentence similar to the MSO-sentence provided in Example 1 in `https://github.com/ssimplexity/CS492_spring2025/blob/main/01-02.Intro-MSO-DFA.pdf`.

$\square$

5. For a DFA $M = (Q, \Sigma, \delta, q_0, F)$, a string $w \in \Sigma^*$ is called a **synchronizer** if there exists a $r \in Q$ such that for all $q \in Q$ we have that $\hat{\delta}(q, w) = r$. A DFA is called **synchronizing** if it has a synchronizer.

(a) Assume you are given a complete description of a DFA $M = (Q, \Sigma, \delta, q_0, F)$. Propose a DFA M' on the same alphabet as M such that the number of states in M' only depends on that of M and that M is synchronizing if and only if $L(M')$ is non-empty. (Hint: Try to employ ideas similar to the ones we have seen for the proof of NFA-DFA equivalence in the class.) Conclude that there is an algorithm that determines given the description of a DFA whether it is a synchronizing one or not in time depending only on $|Q|$ and $|\Sigma|$.

(b) For a fixed $n \geq 2$, consider the DFA $C_n = (Q, \Sigma, \delta, q_0, F)$, where

- $Q = \{0, 1, \ldots, n-1\}$, $\Sigma = \{\mathtt{a}, \mathtt{b}\}$,
- $\delta(i, \mathtt{a}) = (i+1) \bmod n$ for all i,
- $\delta(0, \mathtt{b}) = \delta(1, \mathtt{b}) = 1$, and $\delta(i, \mathtt{b}) = i$ for $i \geq 2$.

Show, for each $n \geq 2$, that C_n is synchronizing and the length of shortest synchronizer is $(n-1)^2$.

Solution.

(a) Construct a DFA $M' = \left(Q', \Sigma, \delta', q_0', F'\right)$ over the same alphabet Σ as follows:

$$Q' := 2^Q, \quad q_0' = Q, \quad \text{for all } S \subseteq Q, a \in \Sigma, \delta'(S, a) := \{\delta(q, a) : q \in S\}, \text{ and } \quad F' := \left\{ \{r\} \subseteq Q : r \in Q \right\}$$

For any word $w \in \Sigma^*$ we have, by construction, $\hat{\delta}'(Q, w) = \{\hat{\delta}(q, w) : q \in Q\}$. Hence w is accepted by M' iff $\hat{\delta}'(Q, w) \in F'$, i.e., iff $\{\hat{\delta}(q, w) : q \in Q\}$ is a singleton, which is precisely the definition that w is a synchronizer for M. Therefore, $L(M') \neq \emptyset$ iff M is synchronizing.

The number of states of M' is at most $|Q'| = 2^{|Q|}$, which depends only on $|Q|$. A graph search (e.g. BFS) from the start state Q in the directed graph of the subset automaton, with edges labeled by letters of Σ, decides whether a singleton is reachable. This takes time polynomial in the size of that graph, i.e. $O\big(2^{|Q|} \cdot |\Sigma| \cdot |Q|\big)$, which depends only on $|Q|$ and $|\Sigma|$. Consequently, there is an algorithm that decides, given the description of M, whether M is synchronizing, with running time depending only on $|Q|$ and $|\Sigma|$.

(b) We begin with the easy part. Consider

$$w_n \; := \; \left(\mathtt{b}\,\mathtt{a}^{\,n-1}\right)^{\,n-2} \mathtt{b}.$$

This word has length $(n-1)^2$, and it is relatively easy to see that this is a synchronizer for C_n. This establishes that the shortest synchronizer is at most of length $(n-1)^2$.

It remains to show that any synchronizer is of length at least $(n-1)^2$. Let us agree on some notations first. For $i, j \in \{0, 1, \ldots, n-1\}$, denote by $[i, j]$ the cyclic interval obtained by starting at i and moving forward (adding $1 \bmod n$) until reaching j, inclusive. Examples (for $n = 6$): $[2, 4] = \{2, 3, 4\}, [5, 1] = \{5, 0, 1\}$. Every nonempty proper subset that is a cyclic interval has a unique such representation. The full set Q can be represented in n ways as $[i, i-1]$. Note additionally that we take modulo n when talking about i, j in the notation $[i, j]$; so, if i (or j) is apparently not in $\{0, \ldots, n-1\}$, we understand it to be the unique integer in $\{0, \ldots, n-1\}$ equal to i modulo n.

For any word prefix τ_i (first i letters of a synchronizing word τ), set

$$S_i := \hat{\delta}\left(\tau_i, Q\right), \quad \ell(i) := \text{ length of the shortest cyclic interval containing } S_i.$$

Thus $\ell(0) = n$ and, if $|\tau| = m$, then $\ell(m) = 1$. For $j \in \{1, \ldots, n\}$, let

$$t(j) := \min\{i \geq 0 : \ell(i) \leq j\}$$

So $t(n) = 0, t(n-1) \geq 1$, and $t(1) = |\tau|$.

We reason backwards from S_i to S_{i-1} depending on the last letter of τ_i. Observe:

(a) If the i-th letter is $\mathtt{a}$ and $S_i \subseteq [j, k]$, then

$$S_{i-1} \subseteq [j-1, k-1]$$

(Reason: $\mathtt{a}$ is a rotation by $+1$; going backward applies rotation by -1.)

(b) If the i-th letter is $\mathtt{b}$ and $S_i \subseteq [j, k]$ with $j \neq 1$, then

$$S_{i-1} \subseteq [j, k]$$

(Reason: The only nontrivial preimage under $\mathtt{b}$ is that 1 has preimage $\{0, 1\}$. If $1 \notin [j, k]$, nothing changes going backward. If $1 \in [j, k]$ and $j \neq 1$, then the interval $[j, k]$ necessarily wraps (across $n - 1 \to 0$) and already contains 0 as well, so adding 0 does not enlarge the interval.)

From multiple iterations of these two points, it follows that (and we will refer to this property as "backward propagation" henceforth): If $S_i \subseteq [j, k]$ with $j \neq 1$, then for any $r \in [0, j-1]$ there exists a cyclic interval I_r of the same length $|[j, k]|$ such that $S_{i-r} \subseteq I_r$. The reasoning is that while going back $r \leq j - 1$ symbols, a final $\mathtt{a}$ just shifts the interval's endpoints by -1, and a final $\mathtt{b}$ keeps the interval the same (because its left endpoint is not 1); in either case, the length of the containing interval does not decrease.

We move onto investigating where the length drops of the shortest cyclic interval containing the states happen. First, $\mathtt{a}$ is a rotation, hence it preserves ℓ. So, every time ℓ strictly decreases, the symbol $\mathtt{b}$ must be used. In particular, the letter $t(j)$ is $\mathtt{b}$ for every $j \in \{1, \ldots, n-1\}$.

Furthermore, note that for $j \in \{1, \ldots, n-2\}$,

$$S_{t(j)} \subseteq [1, j] \quad \text{and} \quad S_{t(j)-1} \subseteq \mathtt{b}^{-1}([1, j]) = [0, j]. \tag{$*$}$$

Now, starting from the index $i = t(j) - 1$ (so that the current containing the interval is $[0, j]$ with left endpoint $\neq 1$), apply the backward-propagation property for $r = n - 1$ steps. We then obtain that

$$S_{t(j)-n} \subseteq I$$

for some cyclic interval I of length $|[0, j]| = j + 1$. Therefore,

$$t(j+1) \leq t(j) - n \quad \text{for all } j \in \{1, \ldots, n-2\}.$$

Finally, note that $t(n-1) \geq 1$ (we need at least one symbol—necessarily $\mathtt{b}$). Summing the $n - 2$ inequalities,

$$t(1) \;=\; |\tau| \;\geq\; \underbrace{1}_{\text{first drop } n \to n-1} + (n-2) \cdot n \;=\; n^2 - 2n + 1 \;=\; (n-1)^2.$$

$\square$

6. Let Σ_1 and Σ_2 be alphabets. A function $h : (\Sigma_1)^* \to (\Sigma_2)^*$ is a *homomorphism from* $(\Sigma_1)^*$ *to* $(\Sigma_2)^*$ if for any $w_1, w_2 \in (\Sigma_1)^*$, the equality $h(w_1w_2) = h(w_1)h(w_2)$ holds (observe that this is just an example of monoid homomorphisms from Homework 1). Let $h : (\Sigma_1)^* \to (\Sigma_2)^*$ be a homomorphism.

(a) For $L \subseteq (\Sigma_1)^*$, define the *image* of L under h is defined as

$$h(L) := \{h(w) \in (\Sigma_2)^* : w \in L\}.$$

Suppose that L is regular. Provide a regular expression of $h(L)$ by using that of L. This shows that the (homomorphic) image of a regular language is always regular.

(b) Prove or disprove that the converse of (a) is true, i.e., if $h(L)$ is regular, then so is L.

(c) For $M \subseteq (\Sigma_2)^*$, the *inverse image* of M under h is defined as

$$h^{-1}(M) := \{w \in (\Sigma_1)^* : h(w) \in M\}.$$

By using the Myhill-Nerode theorem, prove that $h^{-1}(M) \subseteq (\Sigma_1)^*$ is regular if $M \subseteq (\Sigma_2)^*$ is regular.

(d) Prove or disprove that the converse of (c) is true, i.e., if $h^{-1}(M)$ is regular, then so is M.

Solution.

(a) The idea is very simple: you replace every symbol in the regular expression by its homomorphic image! To write more formally, let R be a regular expression of L and let r be the number of operations that are used in R. We construct the regular expression $h(R)$ of $h(L)$ by induction on r.

When $r = 0$, then either $R = \emptyset$ or R consists of a single symbol $x \in \Sigma \cup \{\varepsilon\}$. In each of these cases, $\emptyset$ and $h(x)$ are the regular expressions of $h(L)$, respectively. Now, assume $r \geq 1$. Then either $R = R_1 \cup R_2$, $R = R_1 \circ R_2$, or $R = (R_1)^*$ for some regular expressions R_1, R_2 over Σ both using less than r operations. Now, if $R = R_1 \cup R_2$, take $h(R) = h(R_1) \cup h(R_2)$; if $R = R_1 \circ R_2$, take $h(R) = h(R_1) \circ h(R_2)$; if $R = (R_1)^*$, take $h(R) = h(R_1)^*$.

(b) The statement is **False**. For example, let $\Sigma_1 = \Sigma_2 = \{\mathsf{a}, \mathsf{b}\}$ and let $L = \{\mathsf{a}^n\mathsf{b}^n : n \in \mathbb{N}\}$. Consider the homomorphism $h : (\Sigma_1)^* \to (\Sigma_2)^*$ defined by $h(w) = \varepsilon$ for any $w \in (\Sigma_1)^*$. Then $h(L) = \{\varepsilon\}$ so $h(L)$ is regular, but L is not regular.

(c) Let $L := h^{-1}(M)$. Since M is regular, there are finitely many equivalence classes $M_1, \dots, M_t$ of $\equiv_M$. For each $i \in \{1, \dots, t\}$, let $L_i := h^{-1}(M_i)$. We claim that for each i, any two strings in L_i are indistinguishable by L. Suppose to the contrary that there is $i \in \{1, \dots, t\}$ and two strings $x, y \in L_i$ such that $x \not\equiv_L y$. Then there is $z \in (\Sigma_1)^*$ such that $xz \in L$ and $yz \notin L$ or vice versa. Without loss of generality, assume that $xz \in L$ and $yz \notin L$.

Observe that $h(L_i) = h(h^{-1}(M_i)) \subseteq M_i$ and, similarly, $h(L) \subseteq M$. Since $x, y \in L_i$, we have $h(x), h(y) \in M_i$ and, in particular, $h(x) \equiv_M h(y)$. Since $xz \in L$, $h(xz) = h(x)h(z) \in M$. Thus, $h(y)h(z) \in M$. However, then $h(yz) \in M$ so $yz \in L$ and we reach a contradiction. This shows that the number of equivalence classes of $\equiv_L$ is no larger than that of $\equiv_M$. Therefore, by the Myhill-Nerode theorem, we conclude that L is regular.

(d) The statement is **False**. We use a similar counterexample given in part (b): Let $\Sigma_1 = \Sigma_2 = \{\mathsf{a}, \mathsf{b}\}$, let $M = \{\mathsf{a}^n\mathsf{b}^n : n \in \mathbb{N}\}$, and let $h : (\Sigma_1)^* \to (\Sigma_2)^*$ be a homomorphism defined by $h(w) = \varepsilon$ for every $w \in (\Sigma_1)^*$. Since $\varepsilon \in M$, we have $h^{-1}(M) = (\Sigma_1)^*$. Thus, $h^{-1}(M)$ is regular whereas M is not regular.

□