Introduction to the Theory of Computation 2021 — Midterm 1

Solutions

Problem 1 (35 pts). Consider the following languge

$$\{w \in \{0,1\}^* \mid \text{the sum of } w \bmod 6 = 0\}, \tag{1}$$

where the sum of w means the sum of the symbols in w. Note that we define

$$\text{the sum of } w = 0 \text{ if } w = \epsilon.$$

(a) (10 pts) Recall Example 1.13 in our slides and textbook that is an example of the sum is 0 modulo 3. Follow that example to construct a DFA for (1), and give the **formal definition**.

(b) (5 pts) Consider the section "Regular Operations" in our slides "chap1_DFA3.pdf" (Theorem 1.25 in textbook). We demonstrate how to construct the DFA of $A \cup B$ from the DFAs of two languages A and B. The same technique can be slightly modified for $A \cap B$. Describe the procedure in detail by modifying Theorem 1.25.

(c) (10 pts) Clearly, the language (1) is equivalent to

$$\{w \in \{0,1\}^* \mid \text{the sum of } w \bmod 2 = 0\} \cap \{w \in \{0,1\}^* \mid \text{the sum of } w \bmod 3 = 0\}.$$

Let us take

$$A = \{w \in \{0,1\}^* \mid \text{the sum of } w \bmod 2 = 0\}$$

and

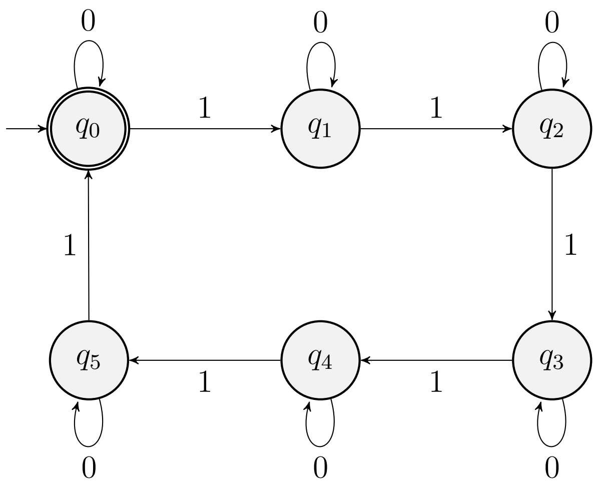
$$B = \{w \in \{0,1\}^* \mid \text{the sum of } w \bmod 3 = 0\}.$$

Give DFAs for A and B first, and then use the procedure of (b) to construct a DFA for $A \cap B$.

(d) (10 pts) Convert the DFA in (a) to GNFA, and generate a regular experssion for this language. You need to show the GNFA after each state is removed.

Solution.

(a) The DFA can be



Its formal definition is

$$
\begin{aligned}
M =& (Q, \Sigma, \delta, q_0, F) \\
Q =& \{q_i \mid i = 0, \ldots, 5\} \\
\Sigma =& \{0, 1\} \\
F =& \{q_0\} \\
\delta =& \begin{array}{c|cc}
 & 0 & 1 \\
\hline
q_0 & q_0 & q_1 \\
q_1 & q_1 & q_2 \\
q_2 & q_2 & q_3 \\
q_3 & q_3 & q_4 \\
q_4 & q_4 & q_5 \\
q_5 & q_5 & q_0
\end{array}
\end{aligned}
$$

(b) The definition of $A \cap B$ is the same as $A \cup B$, except for the accepted states. That is, if

$$M_A = (Q_A, \Sigma, \delta_A, q_A, F_A)$$

and

$$M_B = (Q_B, \Sigma, \delta_B, q_B, F_B)$$

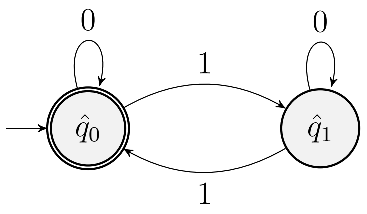
recognize A and B respectively, the formal definition of M for $A \cap B$ is

$$
\begin{aligned}
Q =& \{(r_A, r_B) \mid r_A \in Q_A, r_B \in Q_B\} \\
& \Sigma \text{ is the same one} \\
\delta((r_A, r_B), a) =& (\delta_A(r_A, a), \delta_B(r_B, a)) \\
q_0 =& (q_A, q_B) \\
F =& \{(r_A, r_B) \mid r_A \in F_A \textbf{ and } r_B \in F_B\}.
\end{aligned}
$$

(c) The DFA of

$$\{w \in \{0,1\}^* \mid \text{the sum of } w \bmod 2 = 0\}$$

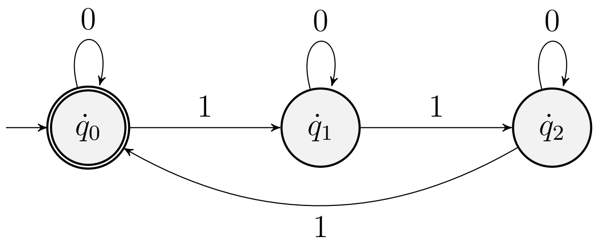
is



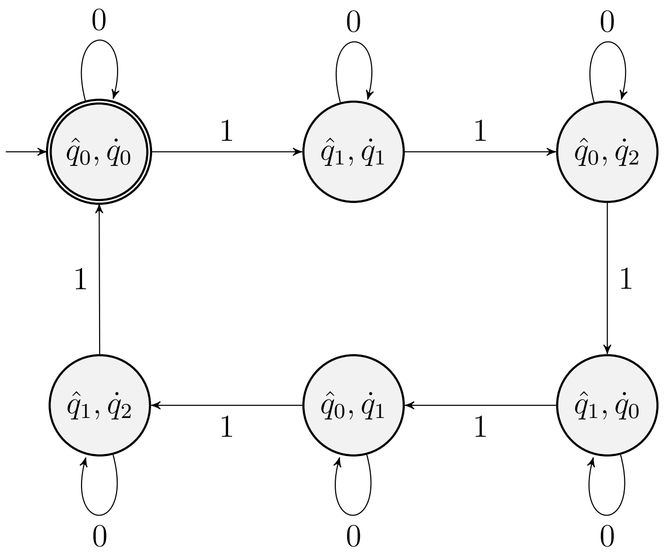
The DFA of

$$\{w \in \{0,1\}^* \mid \text{the sum of } w \bmod 3 = 0\}$$

is

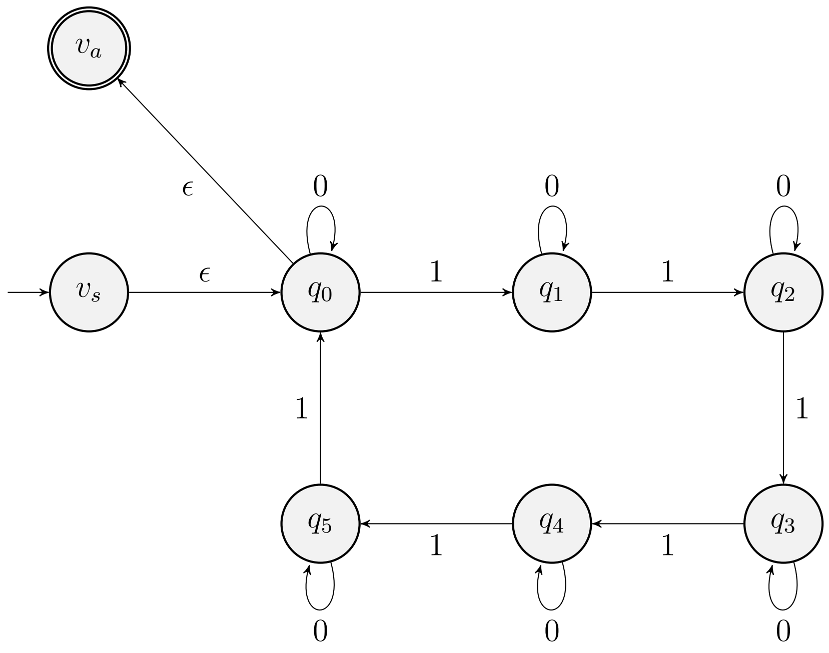


By the procedure of (b), we get

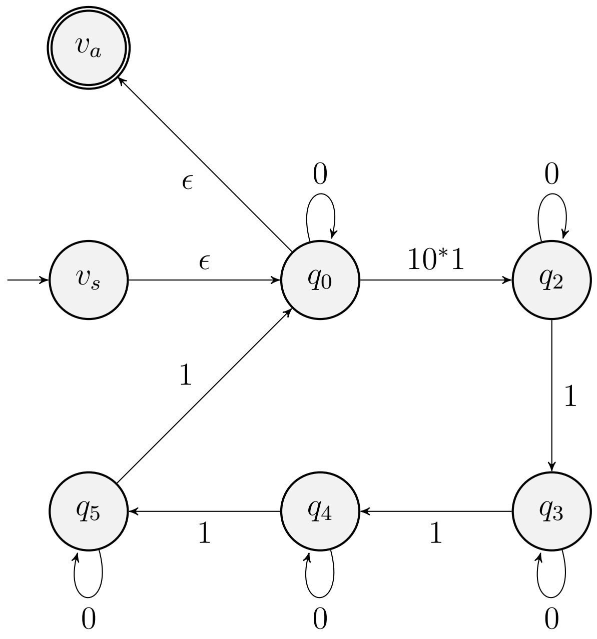


(d) We show the details by the following steps.

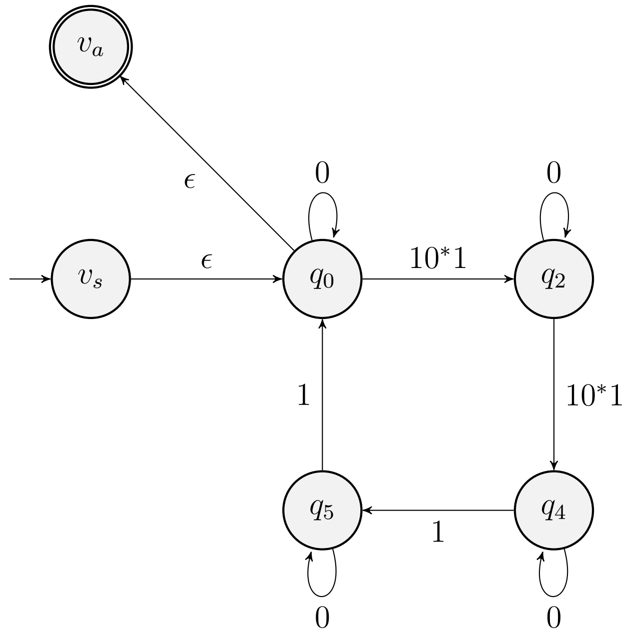
Step 1: Let us begin from adding phaton initial and accepted states



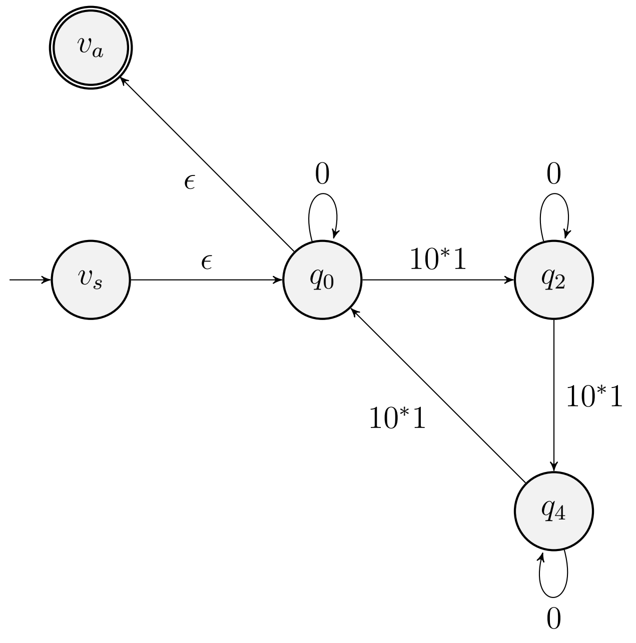
Step 2: Remove state q_1



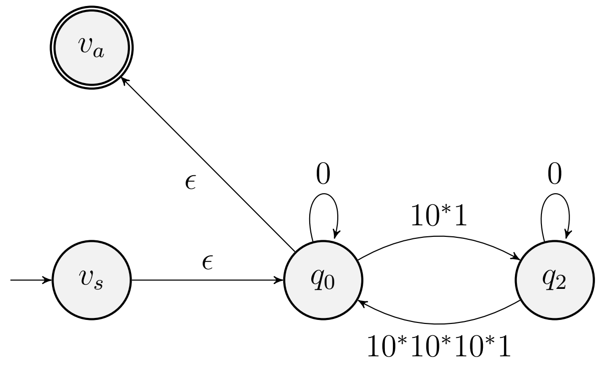
Step 3: Remove state q_3



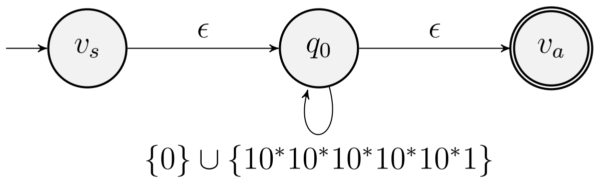
Step 4: Remove state q_5



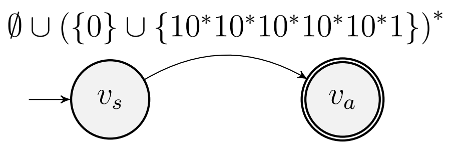
Step 5: Remove state q_4



Step 6: Remove state q_2



Step 7: Remove state q_0



Therefore, the regular expression of (a) is

$$\emptyset \cup (\{0\} \cup \{10^*10^*10^*10^*10^*1\})^*$$

Problem 2 (20 pts). Consider the alphabet

$$\Sigma = \{0, 1\}$$

and two language

$$A = \{0\} \text{ and } B = \{1\}.$$

(a) (5 pts) Construct NFAs for A and B.

(b) (5 pts) Follow the process of Theorem 1.47 in the textbook ("Closed Under Concatenation" in the slides chap1_NFA4.pdf) to have an NFA for

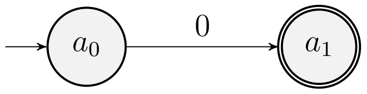
$$A \circ B.$$

Give the **formal definition** of this NFA.

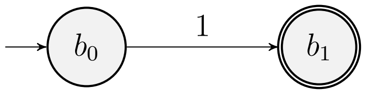
(c) (10 pts) Convert the NFA in (b) to DFA by the procedures of Example 1.41. Note that you need to draw all the possible states in the beginning, and explain why some states are never used. You need not to draw the edges that connect to unused states.

Solution.

(a) The NFA for A can be shown as



Similarly, the NFA for B is



(b) We can concatenate two NFAs as

$$\begin{tikzcd} \arrow[r] & a_0 \arrow[r, "0"] & a_1 \arrow[r, "\epsilon"] & b_0 \arrow[r, "1"] & \boxed{b_1} \end{tikzcd}$$

where the formal definition is

$$\begin{aligned} Q =& \{a_0, a_1, b_0, b_1\} \\ \Sigma =& \{0, 1\} \\ q_0 =& a_0 \\ F =& \{b_1\} \end{aligned}$$

$$\delta = \begin{array}{c|ccc} & 0 & 1 & \epsilon \\ \hline a_0 & \{a_1\} & \emptyset & \emptyset \\ a_1 & \emptyset & \emptyset & \{b_0\} \\ b_0 & \emptyset & \{b_1\} & \emptyset \\ b_1 & \emptyset & \emptyset & \emptyset \end{array}$$

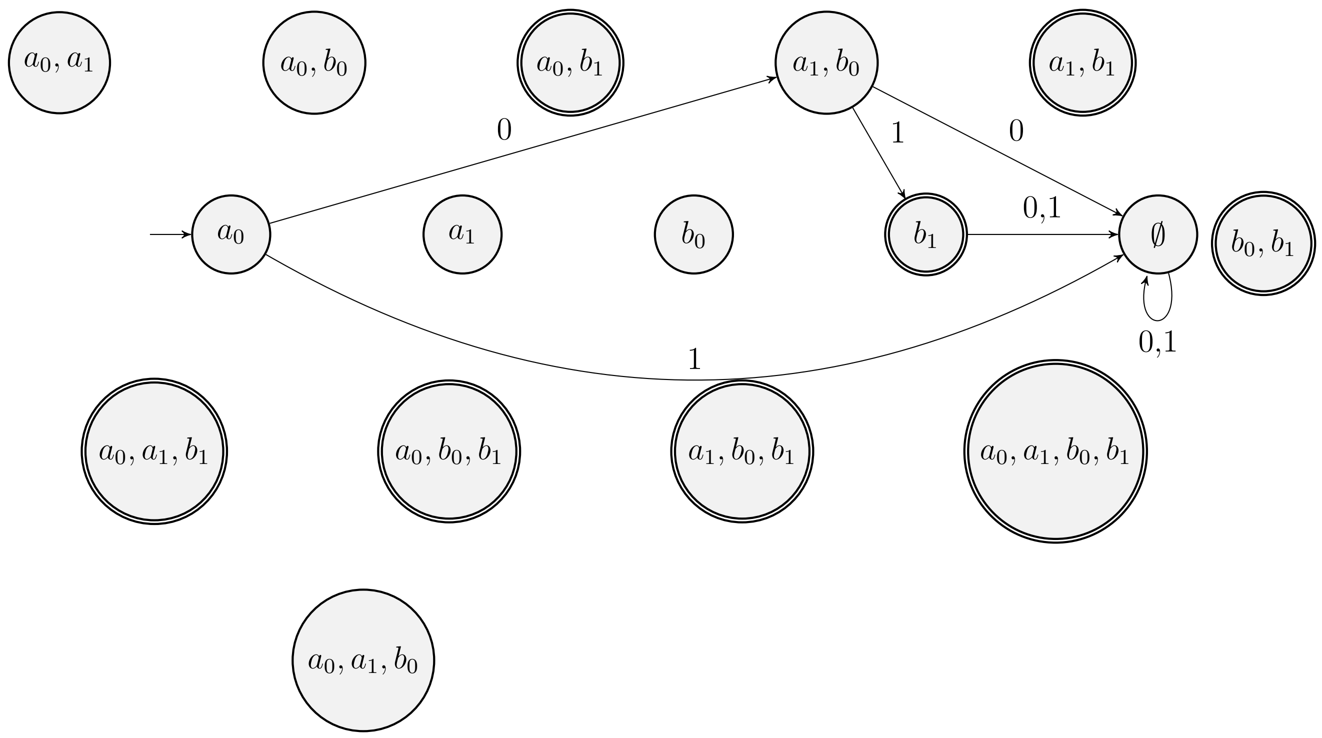
Common mistake: the output of δ should be a set.

(c) We show the procedures of Example 1.41 by the following steps.

Step 1: We have the conbination of the states as

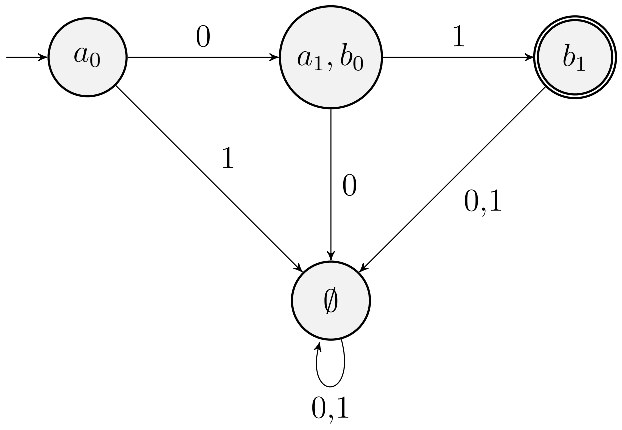
$$\emptyset, a_0, a_1, b_0, b_1, \{a_0, a_1\}, \{a_0, b_0\}, \{a_0, b_1\}, \{a_1, b_0\}, \{a_1, b_1\}, \{b_0, b_1\},$$
$$\{a_0, a_1, b_0\}, \{a_0, a_1, b_1\}, \{a_0, b_0, b_1\}, \{a_1, b_0, b_1\}, \{a_0, a_1, b_0, b_1\}$$

Step 2: Using the states of (c1), we have the DFA as



The start state should be a_0. From it, the input 1 goes to $\emptyset$, while input 0 goes to a_1 or b_0. Thus, $\{a_1, b_0\}$ becomes another reachable state. In the state $\{a_1, b_0\}$, we know that b_0 can reach b_1 by the input 1, but a_1 goes to nowhere. Therefore, when the input 1 comes, the state $\{a_1, b_0\}$ goes to $\{b_0\}$. Moreover, the state $\{a_1, b_0\}$ goes to $\emptyset$ if the input 0 comes. At $\{b_1\}$ from the NFA, we go to $\emptyset$ for either input 0 or 1. Then at $\emptyset$, the input 0 or 1 leads back to $\emptyset$. Thus, we have checked all reachable states.

Step 3: After we remove the unused states, we have



Problem 3 (30 pts). Consider these two languages defined by regular expression:

$$A = \emptyset, B = 1^*$$

and assume $\Sigma = \{0, 1\}$.

(a) (5 pts) First, let's construct an NFA for B. Give an NFA for the regular expression 1 and follow the procedure of star operation[1] on NFAs to obtain the NFA for B.

(b) (5 pts) Show that B can actually be recognized by a single state NFA.

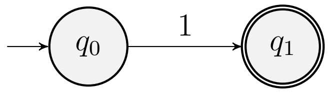
(c) (5 pts) Construct a one state NFA for language A. Then, concatenate[2] it with the NFA from (b) to give the NFA for $A \circ B$. Do not remove unreachable states so that you should have two states.

(d) (5 pts) Convert the NFA for $A \circ B$ obtained in (c) to a DFA.[3] Do not remove unreachable states so that you should have 4 states.

(e) (10 pts) For the DFA obtained in (d), find a corresponding regular expression by first converting[4] it to a GNFA. Then, sequentially remove the states, starting from the unreachable states. You must show the GNFA at each step.

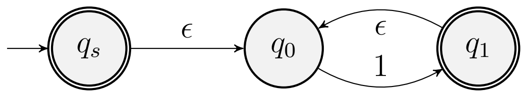
Solution.

(a) The NFA for the regualr expression 1 is

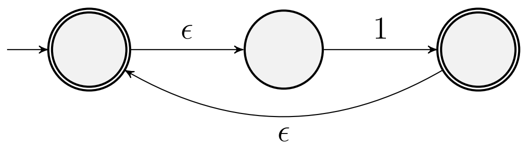


[1]page 10 of slide chap1_NFA4 or page 62, theorem 1.49 in the text book
[2]page 7 of slide chap1_NFA4 or page 60, theorem 1.47 in the text book
[3]page 2 of slide chap1_NFA3 or page 55, theorem 1.39 in the text book
[4]slide chap1_regexp3,4 or page 69, lemma 1.60 in the text book

The NFA for 1^* is then

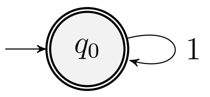


Common mistake: Some give the following figure:

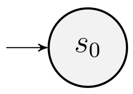


Though this works for the language, we specifically asked you to follow the slides/book. Therefore, you won't get all the points by giving this answer.

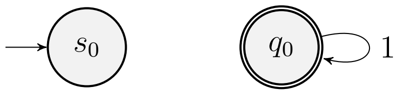
(b) $B = 1^*$ can be recognized by this NFA:



(c) Firstly, the NFA for $\emptyset$ is:



Concatenating it with the NFA from (b) leads to

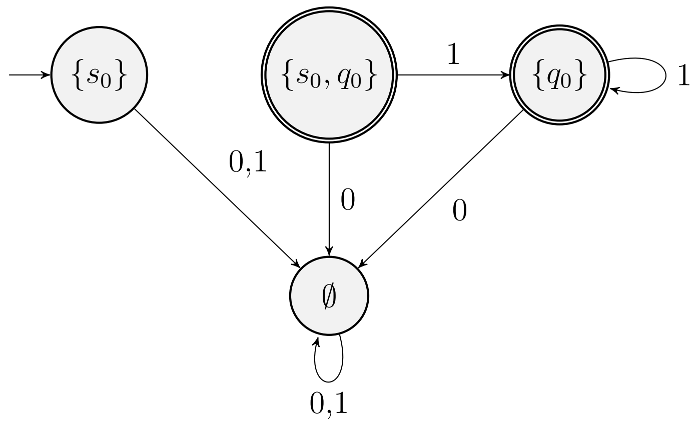


(d) The DFA would be $(Q, \Sigma, \delta, s_0, F)$ where $Q = \{\emptyset, \{s_0\}, \{q_0\}, \{s_0, q_0\}\}$, $F = \{\{q_0\}, \{s_0, q_0\}\}$ and the transition function is given by the table

	0	1
$\{s_0\}$	$\emptyset$	$\emptyset$
$\{q_0\}$	$\emptyset$	$\{q_0\}$
$\{q_0, s_0\}$	$\emptyset$	$\{q_0\}$
$\emptyset$	$\emptyset$	$\emptyset$

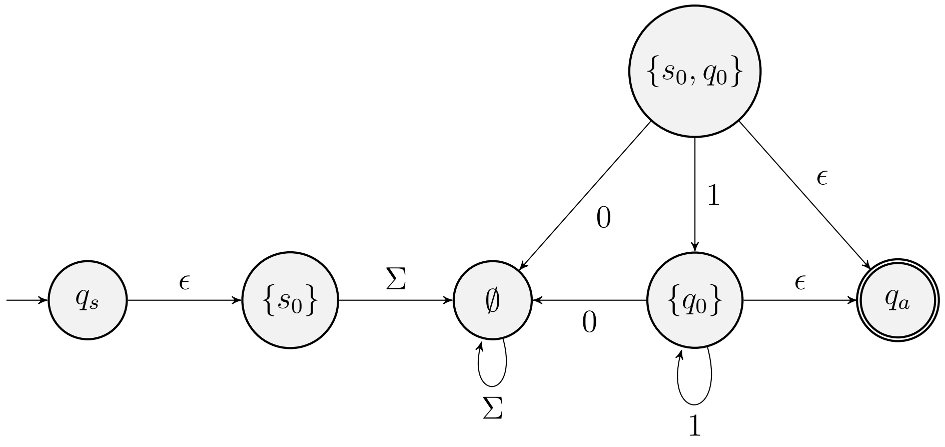
.

The state diagram:

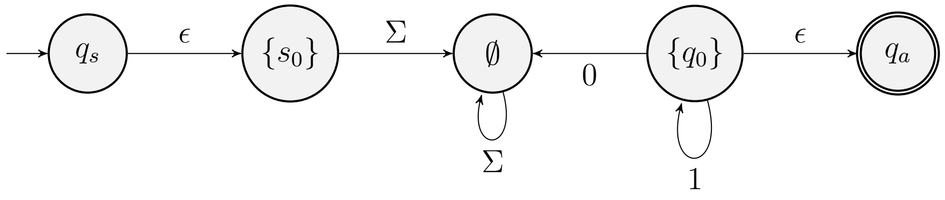


Common mistake: $\{s_0, q_0\}$ should be an accept state.

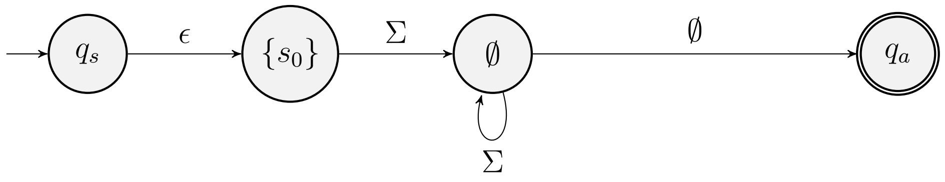
(e) First we convert the DFA to a GNFA by adding a new start state and an accept state. Also, edges labeled with $0, 1$ are changed to $0 \cup 1 = \Sigma$ ($\emptyset$ edges are not shown):



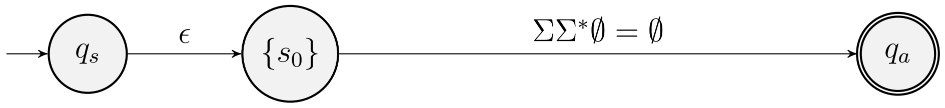
Removing $\{s_0, q_0\}$ does not affect other edges since all the edges going to it are labeled $\emptyset$:



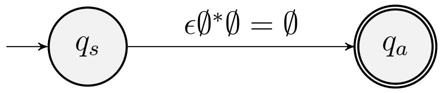
Removing $\{q_0\}$ also does not affect the edges (an $\emptyset$ edge is shown to make things clearer):



Then, remove the $\emptyset$ state:



Finally, remove the $\{s_0\}$ state:



Thus, the final regular expression is equivalent to $\emptyset$.

Problem 4 (15 pts). Consider the following language

$$A = \{w \mid w \in \{0,1,2\}^*, \text{ sum of } w \bmod 3 = 0\}.$$

Is it possible to have a DFA with two states for this language? If yes, give a DFA. If not, rigorously prove the result. Your proof must be complete and include all the details.

Solution.

Proof. We prove the result by contradiction. Suppose there is a DFA that recognizes A with two states, say $M = (Q = \{q_0, q_1\}, \Sigma = \{0,1,2\}, \delta, q_0, F)$. We can assume without loss of generality that q_0 is the starting state. We know that the string 111 is in A. Thus, there exists a sequence of states

$$q_0, w_1, w_2, w_3$$

such that $w_1 = \delta(q_0, 1), w_2 = \delta(w_1, 1), w_3 = \delta(w_2, 1)$ and $w_3 \in F$. Since w_1, w_2 and w_3 must be chosen from Q, we know that there is a repeating state in them by pigeonhole principle. We can discuss three cases:

1. $w_1 = w_2$: In this case, we can remove w_2 in the sequence and obtain another accepting sequence of states. We can then conclude that M accepts 11. This is a contradiction since $11 \notin A$.

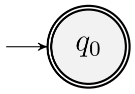
2. $w_1 = w_3$: In this case, we can see that q_0, w_1 is also a accepting sequence of states. This tells us that M accepts 1, which is also a contradiction.

3. $w_2 = w_3$: Similarly, we can conclude that M accepts 11 and run into a contradiction.

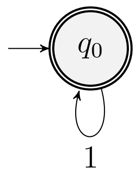
Since all possible cases lead to a contradiction, there does not exists DFA that recognizes A with only two states. □

Here's an alternative proof by drawing diagrams.

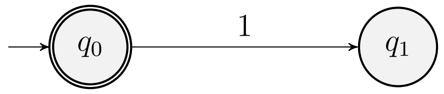
Proof. Because $\epsilon \in A$, the start state must be an accept state. Therefore, we have the following subgraph:



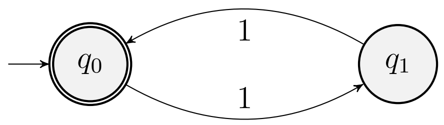
Consider an input 1 at this state. It is impossible to have



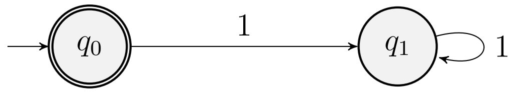
Otherwise, $1 \notin A$ would be accepted. Thus, we must have



and q_1 is not an accept state. Next, consider input 1 at state q_1. We can not have



Otherwise, $11 \notin A$ would be accepted. Since we can use only two states, it must be



To accept $111 \in A$, q_1 must be an accept state. However, we would then have $1 \in A$, a contradiction. This shows that we can not use only two states. $\square$

Common mistake: Your description must be clearly written. For example, some wrote "if the sum is at the accept state," but how can a sum be at a state? Your every statement must be clear and logical. Some wrote : "if there are two states q_0, q_1 and assume q_0 is the start state, then q_0 is an accept state." You must give reasons why q_0 is an accept state (because ϵ is accepted).