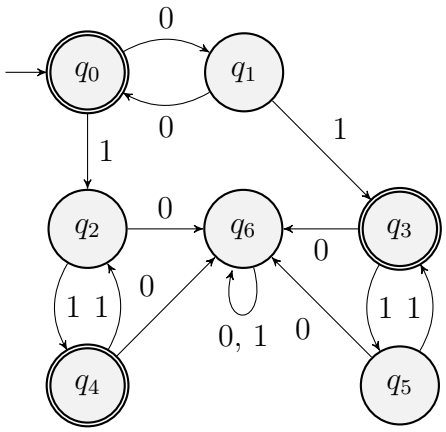
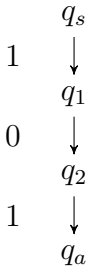
Introduction to the Theory of Computation 2024 — Midterm 1

Solutions

Problem 1 (5 pts). Let $\Sigma = \{0, 1\}$. Consider the following DFA



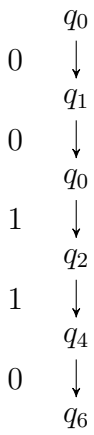
Please simulate the computation on the input strings **0111** and **00110**. After drawing the paths, determine whether this DFA **accepts** or **rejects** each input string. Illustrate the computation paths, similar to the example provided (adapted from Figure 1.28 in the textbook).



Please follow the rules in the textbook.

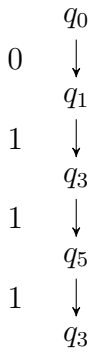
Solution.

For the input string **00110**



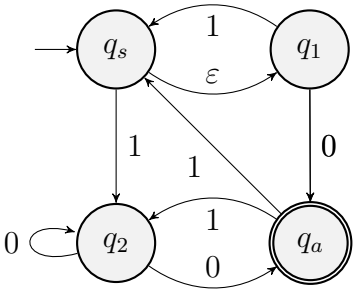
This DFA **rejects** the input string **00110**.

For the input string **0111**

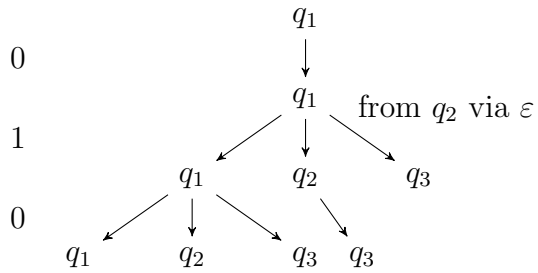


This DFA **accepts** the input string **0111**.

Problem 2 (25 pts). Let $\Sigma = \{0, 1\}$. Consider the following NFA.

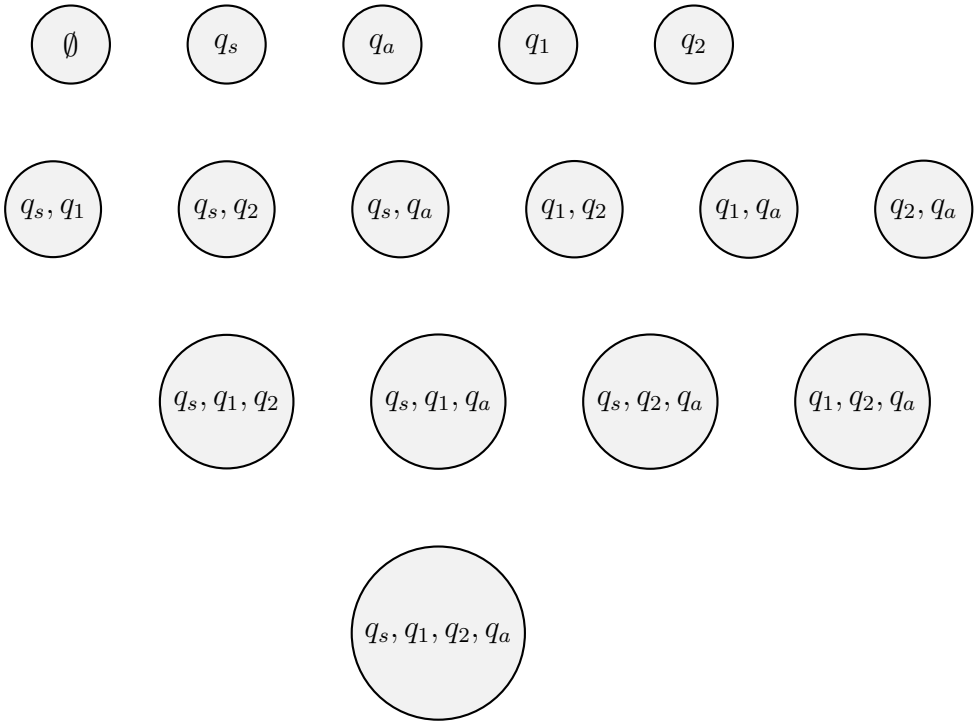


(a) (10 pts) Please draw the computation of this NFA on the input strings **101** and **010** and conclude whether they are **accepted** or **rejected**. A computation is like the following figure (copied from Figure 1.29 in the textbook, as an illustration and not related to the NFA in this subproblem.)



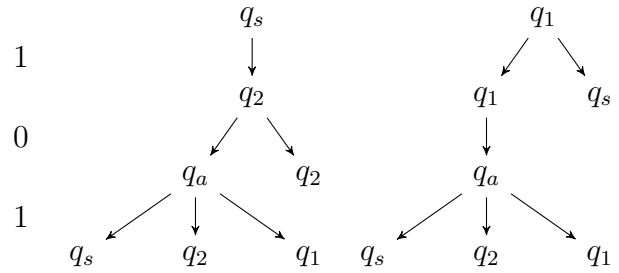
Please follow the rules in the textbook. Note that we handle the ε edge **immediately**, and you have to list all possible states that can be reached once processing each input character.

(b) (15 pts) Convert this NFA to a DFA by the procedure in our slides "chap1_NFA3.pdf" from page 3 to page 9 (Theorem 1.39 in the textbook.). We have provided the following diagram illustrating all possible state combinations. Please provide the detail of the procedure. Use this diagram for your conversion, then remove useless states and show ~~only~~ the resulting diagram on your answer sheet.



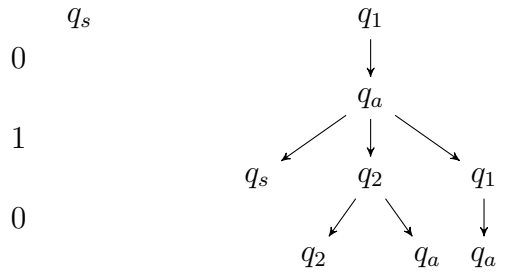
Solution.

(a) For the input string **101**:



This NFA **rejects** the input string **101**.

For the input string **010**:



This NFA **accepts** the input string **010**.

Common mistake: missing states from ε path.

(b) Step 1: We have the conbination of the states as

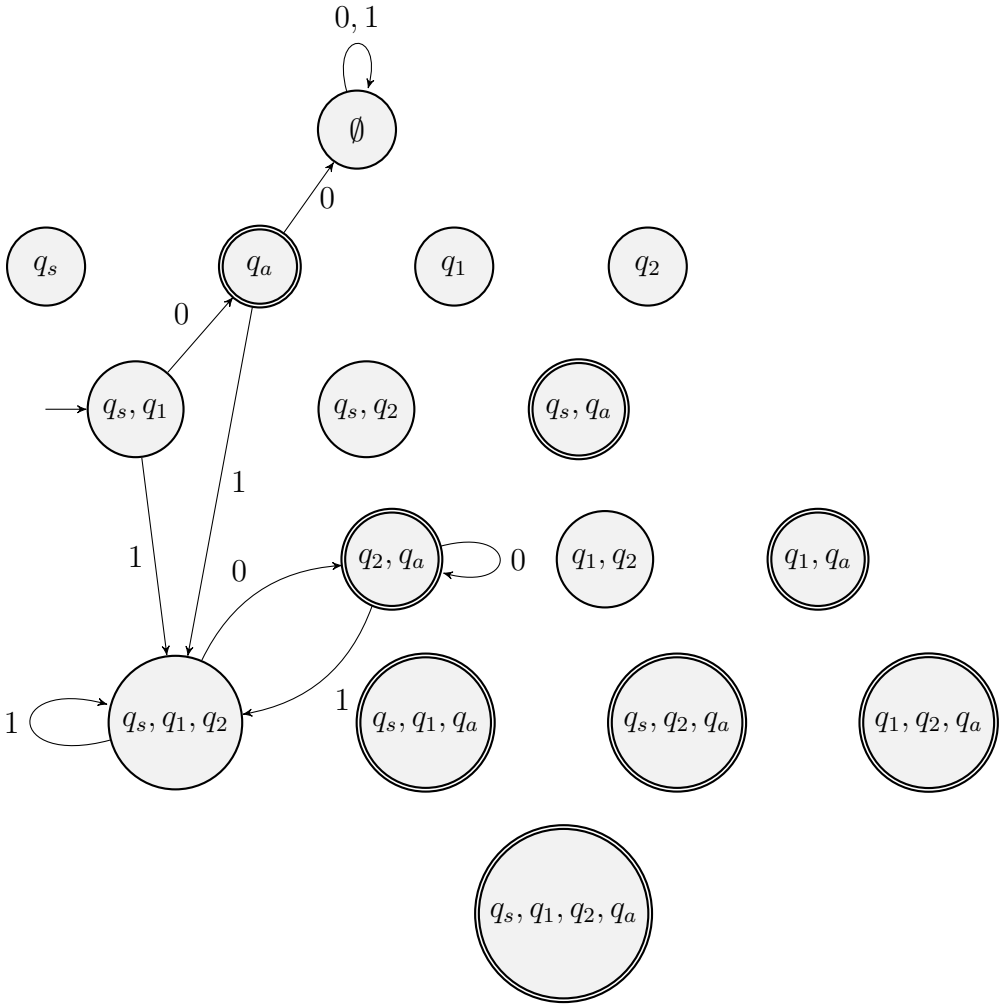
$$\emptyset, \{q_s\}, \{q_1\}, \{q_2\}, \{q_a\}, \{q_s, q_1\}, \{q_s, q_2\}, \{q_s, q_a\}, \{q_1, q_2\}, \{q_1, q_a\}, \{q_2, q_a\},$$
$$\{q_s, q_1, q_2\}, \{q_s, q_1, q_a\}, \{q_s, q_2, q_a\}, \{q_1, q_2, q_a\}, \{q_s, q_1, q_2, q_a\}$$

Step 2: We now follow the procedure outlined in our slides "chap1_NFA3.pdf":

i. Start state: $\{q_s, q_1\}$.

ii. Accept state: Any state that includes q_a is considered an accept state.

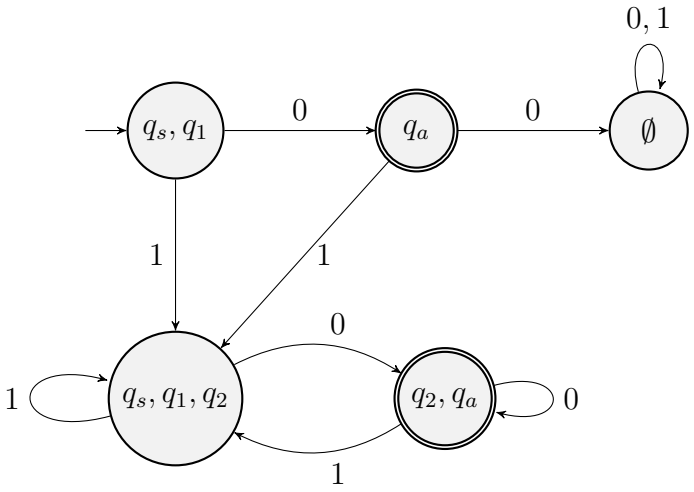
iii. Below, we first demonstrate a few examples of the DFA's transition function δ, followed by the complete transition function:



Its transition function δ is

	0	1
$\emptyset$	$\emptyset$	$\emptyset$
$\{q_s\}$	$\emptyset$	$\{q_2\}$
$\{q_1\}$	$\{q_a\}$	$\{q_s, q_1\}$
$\{q_2\}$	$\{q_2, q_a\}$	$\emptyset$
$\{q_a\}$	$\emptyset$	$\{q_s, q_1, q_2\}$
$\{q_s, q_1\}$	$\{q_a\}$	$\{q_s, q_1, q_2\}$
$\{q_s, q_2\}$	$\{q_2, q_a\}$	$\{q_2\}$
$\{q_s, q_a\}$	$\emptyset$	$\{q_s, q_1, q_2\}$
$\{q_1, q_2\}$	$\{q_2, q_a\}$	$\{q_1, q_s\}$
$\{q_1, q_a\}$	$\{q_a\}$	$\{q_s, q_1, q_2\}$
$\{q_2, q_a\}$	$\{q_2, q_a\}$	$\{q_s, q_1, q_2\}$
$\{q_s, q_1, q_2\}$	$\{q_2, q_a\}$	$\{q_s, q_1, q_2\}$
$\{q_s, q_1, q_a\}$	$\{q_a\}$	$\{q_s, q_1, q_2\}$
$\{q_s, q_2, q_a\}$	$\{q_2, q_a\}$	$\{q_s, q_1, q_2\}$
$\{q_1, q_2, q_a\}$	$\{q_2, q_a\}$	$\{q_s, q_1, q_2\}$
$\{q_s, q_2, q_2, q_a\}$	$\{q_2, q_a\}$	$\{q_s, q_1, q_2\}$

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



Common mistakes:

- Missing accept states or the start state.
- Wrong start state leads to wrong answer.
- Wrong path resulting in adding unreachable states.
- Empty states didn't have 0, 1 path.
- Show the result is NFA, even epsilon.
- Lack of details of procedure.
- Didn't remove unreachable states in the resulting diagram

Problem 3 (25 pts).

(a) (15 pts) Consider the alphabet $\Sigma = \{\texttt{H},\ \texttt{D},\ \texttt{C}\}$.

Construct an NFA for the regular expression

$$\texttt{H}\ (\texttt{C} \cup \texttt{D})\ (\texttt{C} \cup \texttt{D})^*$$

by the following steps:

(i) Give the NFA diagram for the following three regular expressions, each with a single alphabet.

$$R_1 = \texttt{H},\ R_2 = \texttt{C},\ \text{and}\ R_3 = \texttt{D}.$$

(ii) Perform the union operation using the method from the slides "chap1_NFA4.pdf" on the NFAs of R_2 and R_3 to construct the NFA diagram for the regular expression

$$R_4 = R_2 \cup R_3 = \texttt{C} \cup \texttt{D}.$$

(iii) Perform the star operation using the method from the slides "chap1_NFA4.pdf" on the NFA of R_4 to construct the NFA diagram for the regular expression

$$R_5 = R_4^* = (\texttt{C} \cup \texttt{D})^*.$$

(iv) Perform the concatenation operation using the method from the slides "chap1_NFA4.pdf" on the NFAs of R_1 and R_4 to construct the NFA diagram for the regular expression

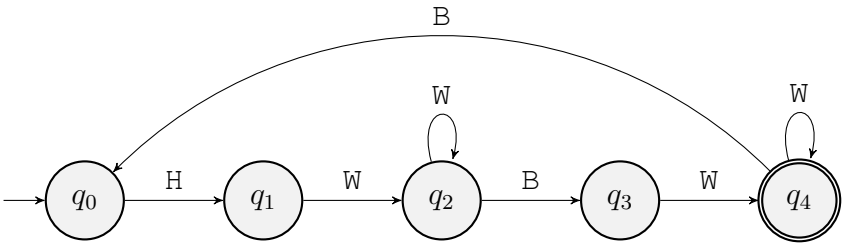
$$R_6 = R_1 \circ R_4.$$

Then, do the concatenation operation again on the NFAs of R_5 and R_6 to construct the NFA diagram for the regular expression

$$R_7 = R_6 \circ R_5.$$

Please show all the NFAs of $R_1, R_2, \ldots, R_7$, and do **NOT** simplify the ε paths.

(b) (10 pts) Give the alphabet $\Sigma = \{\texttt{H},\ \texttt{W},\ \texttt{B}\}$. Convert the NFA



to an equivalent GNFA by the following steps:

(i) Add a new started state q_s and an accepted state q_a to create the initial state diagram S_{initial}.

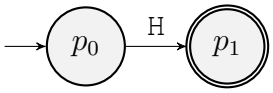
(ii) Remove the states $q_0, q_1, \ldots, q_4$ in **numerical order** during the conversion process to get the final state diagram S_{final}.

Please show all the details of the conversion process from S_{initial} to S_{final}.

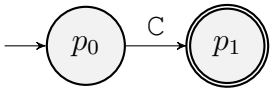
Solution.

(a) Please see the following steps:

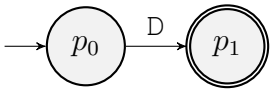
Step 1. NFA for R_1



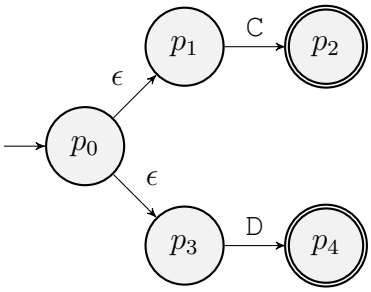
NFA for R_2



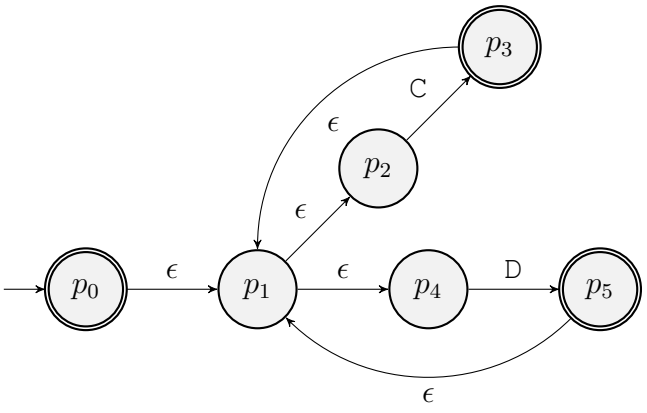
NFA for R_3



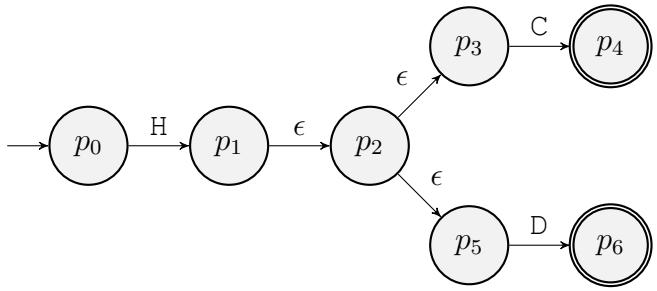
Step 2. NFA for R_4



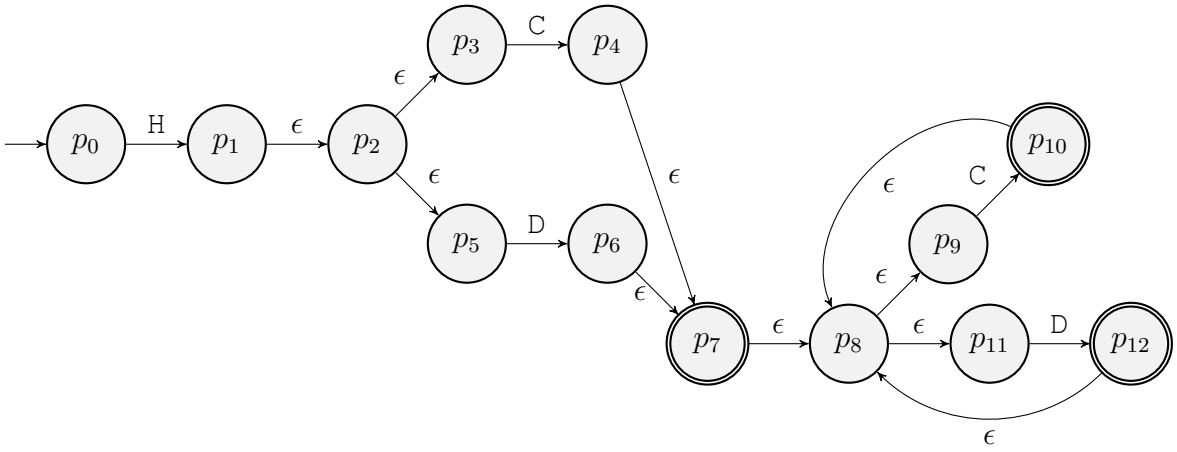
Step 3. NFA for R_5



Step 4. NFA for R_6



Step 5. NFA for R_7

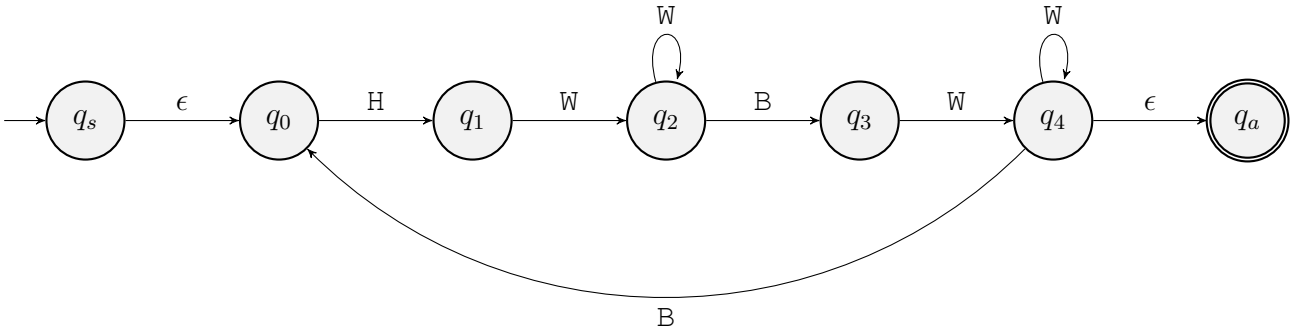


Common mistakes:

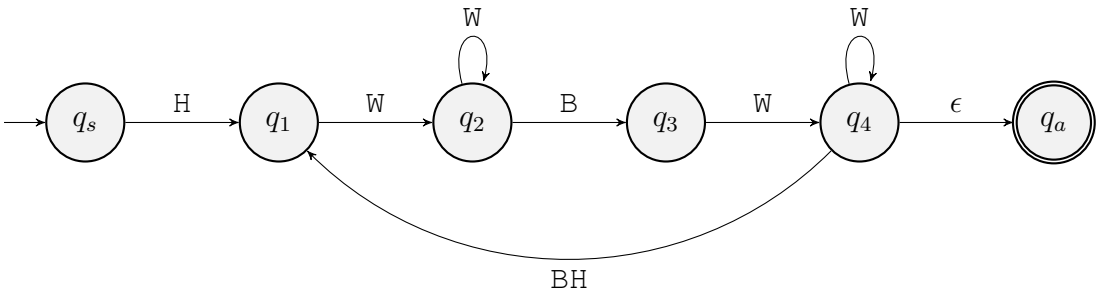
- R5: Do not start with terminal state, epsilon edge.
- R6: (i) Do not have terminal state in the middle. (ii) Accidentally put R5 instead of R4. (iii) Do not have a state and an epsilon edge in the middle.
- R7: (i) Draw two R6. (ii) Do not have a state and an epsilon edge in the middle.

(b) We have the following steps:

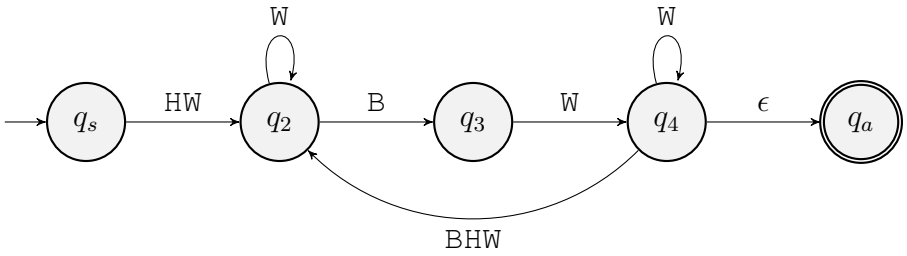
Step 1. Add new start q_s and accept q_a states.



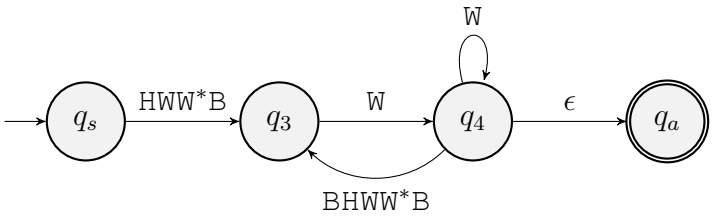
Step 2. Remove q_0.



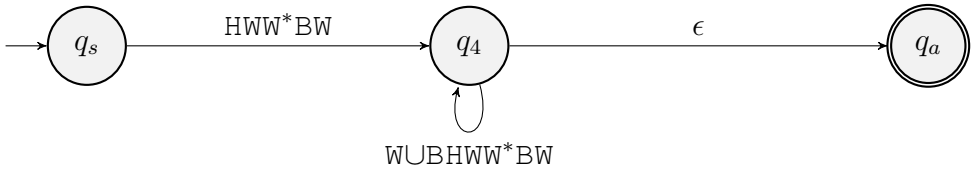
Step 3. Remove q_1.



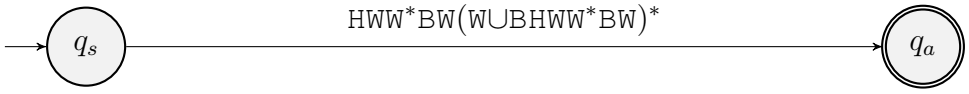
Step 4. Remove q_2.



Step 5. Remove q_3.



Step 6. Remove q_4.



Problem 4 (10 pts). Error-detecting code is a useful technique to check whether an error occurs during the bits transmission. For example, if we have to transmit the bits

011,

we can add an even parity bit behind it

"transmitted target" $\circ$ **"even parity bit"** $= 011\mathbf{0}$.

The even parity bit checks whether the number of 1 in the transmitted target is even or not. Here are other examples to let you understand the even parity bit.

$$011\mathbf{0}, 010\mathbf{1}, 000\mathbf{0}, 111\mathbf{1}.$$

Therefore, if we get the bits

$$0001$$

after the transmission, we can quickly know that an error has occurred. Moreover, we say that 0001 does not pass the even parity check.

Now, let us add even parity bits in every 3 transmitted bits. For example, the transmitted target

$$000\ 010\ 111,$$

becomes to

$$000\mathbf{0}010\mathbf{1}111\mathbf{1}$$

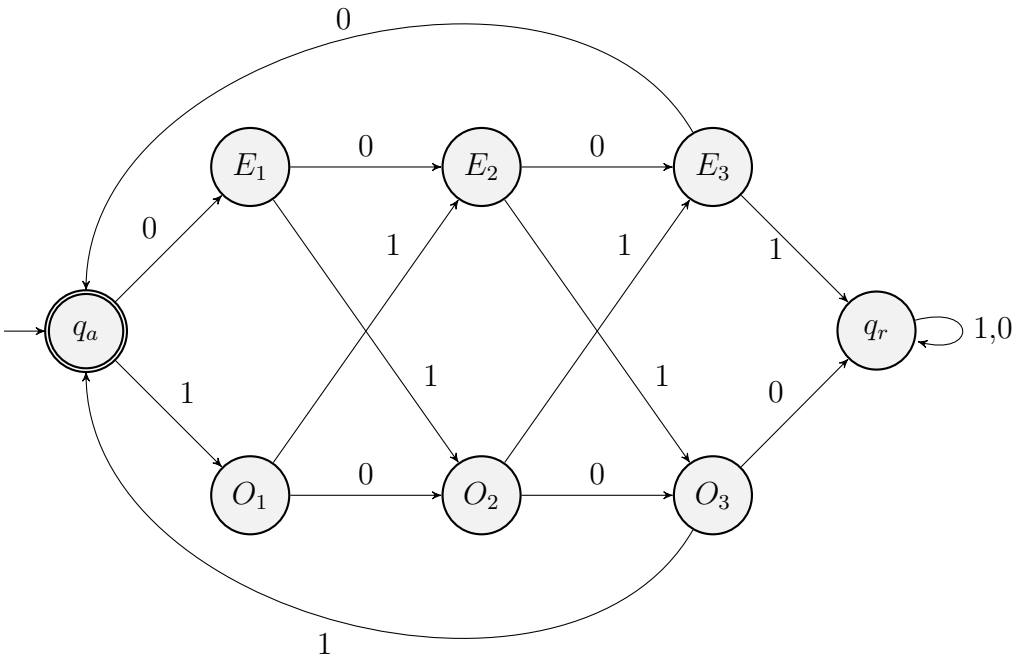
after adding even parity bits, where the bold numbers are the even parity bits. Please give a DFA with less than or equal to 8 nodes to recognize the language

$$\{\boldsymbol{w} = \{0,1\}^* \mid \boldsymbol{w} \text{ can pass the even parity check, and } |\boldsymbol{w}| \bmod 4 = 0.\}$$

Please further explain the concept of your DFA.

Solution.

Please see the following diagram.



We check the odd-even status in previous k bits in the states

$$E_k \text{ and } O_k,$$

for $k = 1, 2,$ and 3. After that, if the even parity bit is correct, we return to the initial state for the next round of checking. Otherwise, we go to the rejected state and stay there for the rest of the string.

Common mistakes:

- Accept wrong strings such as 0001 0000.
- The diagram is not DFA.

Problem 5 (35 pts). Are the following languages regular? If the language is regular, please give a DFA to recognize it. If not, please use the pumping lemma to prove it is not regular. In using the pumping lemma, your s and i must be clearly given. You cannot just roughly say the existence of them. Moreover, you must give clear explanation instead of just giving the answer.

(a) (10 pts) Given the alphabet

$$\Sigma = \{A, B\},$$

we consider the languages

$$L_1 = \{\boldsymbol{w} = \{A, B\}^* \mid \text{score}(\boldsymbol{w}) \geq 0, \}$$

where

$$\text{score}(\boldsymbol{w}) = (\text{number of } A \text{ in } \boldsymbol{w}) - (\text{number of } B \text{ in } \boldsymbol{w}).$$

For example,

$$\text{score}(ABAAB) = 3 - 2 = 1,$$

so

$$ABAAB \in L_1.$$

(b) (10 pts) Given the alphabet

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

we define the language

$$L_2 = \{\text{poly}(n) \mid n = 1, 2, 3, \ldots\},$$

where

$$\text{poly}(n) = 0^n \circ 10^{n-1} \circ 10^{n-2} \circ \cdots \circ 10^1.$$

For example, when $n = 3$,

$$\text{poly}(3) = 0^3 \circ 10^2 \circ 10^1 = 00010010.$$

(c) (15 pts) The language

$$L_3 = \{0^m 1^n \mid m \text{ and } n \text{ are positive integers, and share a common factor greater than 1.}\}$$

For example, when

$$m = 4 \text{ and } n = 2,$$

4 and 2 share a common factor greater than 1. Therefore, the string

$$0^4 1^2 = 000011 \in L_3.$$

Solution.

(a) L_1 is not regular. Let the pumping length p be given. We can take

$$s = B^p A^p \in L_1$$

since

$$\text{score}(s) = 0.$$

Let $s = xyz$ and

$$|xy| \leq p,\ |y| > 0, \tag{1}$$

for every x, y, z. Because (1), we know that

$$y = B^k,$$

where k is a positive integer. Hence, when we take any $i \geq 2$,

$$\text{score}(xy^i z) = \text{score}(xyz) + \text{score}(y^{i-1}) = 0 + (-k)(i-1) < 0.$$

Therefore,

$$xy^i z \notin L_1.$$

By the pumping lemma, we show that L_1 is not regular.

(b) L_2 is not regular. Let the pumping length $p > 0$ be given. We can take

$$s = \text{poly}(p) = 0^p \circ 10^{p-1} \circ 10^{p-2} \circ \cdots \circ 10^1 \in L_2,$$

and

$$|s| = \underbrace{p + (p-1) + \cdots + 1}_{\text{the number of 0s}} + \underbrace{(1 + 1 + \cdots + 1)}_{\text{the number of 1s}} = \frac{p(p+1)}{2} + p - 1 \geq p.$$

Let $s = xyz$ and satisfy the condition (1) for every x, y, z. Because the first p characters of s can only be 0,

$$y = 0^k,$$

for some k such that

$$p \geq k > 0.$$

Hence, we have

$$x = 0^m, y = 0^k, z = 0^{p-m-k} \circ 10^{p-1} \circ 10^{p-2} \circ \cdots \circ 10^1,$$

for some $m \geq 0$, $k > 0$, and $p - m - k \geq 0$. When we take $i = 2$,

$$\begin{aligned} xy^2 z =& 0^m 0^{2k} 0^{p-m-k} \circ 10^{p-1} \circ 10^{p-2} \circ \cdots \circ 10^1 \\ =& 0^{p+k} \circ 10^{p-1} \circ 10^{p-2} \circ \cdots \circ 10^1, \end{aligned}$$

which is not in L_2. By the pumping lemma, we show that L_2 is not regular.

(c) L_3 is not regular. Given a pumping length p, we take a prime number $\hat{p}$ that is strictly greater than p (i.e., $\hat{p} > p \geq 1$). Because $\hat{p}$ and $\hat{p}$ have a common factor $\hat{p} > 1$, we can pick

$$s = 0^{\hat{p}}1^{\hat{p}} \in L_3.$$

Let $s = xyz$ and satisfy the condition (1) for every x, y, z. Because $\hat{p} > p$, we have

$$y = 0^k,$$

for some k such that

$$p \geq k > 0.$$

When $i = 0$, we have

$$xy^0z = 0^{\hat{p}-k}1^{\hat{p}}.$$

Because $\hat{p}$ is a prime number, the common factor of

$$\hat{p} - k \text{ and } \hat{p}$$

must be 1. Hence, $xy^0z \notin L_3$. By the pumping lemma, we show that L_3 is not regular.