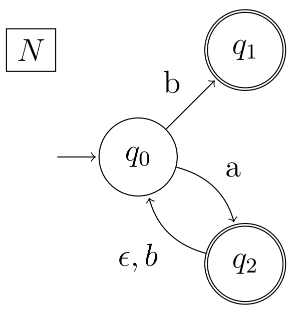
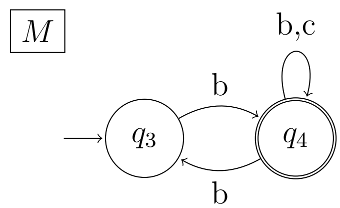
Introduction to the Theory of Computation

Midterm 1 Sample Solutions

Problem 1 (10 pts). Consider the following NFAs, where $\Sigma_N = \{a, b\}, \Sigma_M = \{b, c\}$.



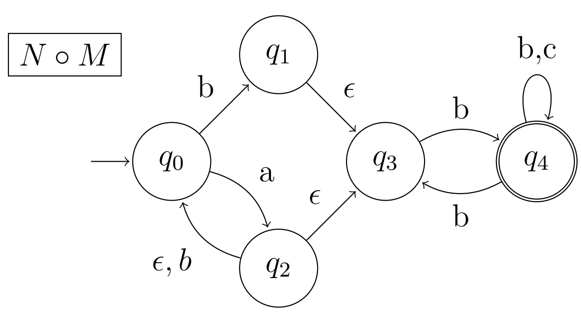


(a) (5 pts) Construct $N \circ M$ using the procedure in Theorem 1.47 of the textbook.

(b) (5 pts) Give the formal definition of the resulting NFA.

Solution.

(a) The resulting NFA is shown below.



(b) $N \circ M = (Q, \Sigma, \delta, q_0, \{q_4\})$, where

- $Q = \{q_i \mid i = 0, 1, \ldots, 4\}$
- $\Sigma = \Sigma_N \cup \Sigma_M = \{a, b, c\}$
- δ is given as

	a	b	c	ϵ
q_0	$\{q_2\}$	$\{q_1\}$	$\emptyset$	$\emptyset$
q_1	$\emptyset$	$\emptyset$	$\emptyset$	$\{q_3\}$
q_2	$\emptyset$	$\{q_0\}$	$\emptyset$	$\{q_0, q_3\}$
q_3	$\emptyset$	$\{q_4\}$	$\emptyset$	$\emptyset$
q_4	$\emptyset$	$\{q_3, q_4\}$	$\{q_4\}$	$\emptyset$

Problem 2 (50 pts). Assume $\Sigma = \{0, 1\}$. Consider the language

$$A = \{w \in \Sigma^* \mid \text{at least one of the last two characters of } w \text{ is } 1\}.$$

(a) (5 pts) Find an NFA that recognizes A with the *smallest* number of states. Give the formal definition and show the tree of running the input string 110.

(b) (10 pts) Explain why your solution in (a) has the smallest number of states. *You must clearly explain all details.*

(c) (5 pts) Convert your NFA in (a) to a DFA by the procedure in Theorem 1.39 of the textbook.

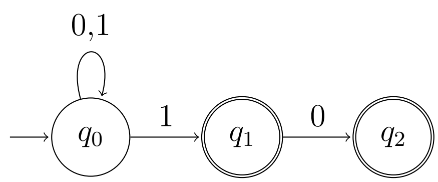
(d) (10 pts) Simplify your DFA in (c) to have the *smallest* number of states. Is there more than one DFA(s) that has the same smallest number of states and recognizes A? If you think so, find out how many are there; otherwise, prove that there can be only one DFA that recognizes A with this number of states. *You must clearly explain all details.*

(e) (10 pts) Give a regular expression that describes A by applying the method in Lemma 1.60 of the textbook to convert your DFA in (d) to a GNFA, and then reducing it to a regular expression. *Please remove the start state of your DFA first.*

(f) (10 pts) In the textbook, Lemma 1.60 shows how to convert a DFA to a GNFA. How about converting an NFA to a GNFA? Try to convert your NFA in (a) to a GNFA, and then reduce it to a regular expression by the same procedure. *Again, please remove the start state of your NFA first.*

Solution.

(a) Below is an NFA N with 3 states:

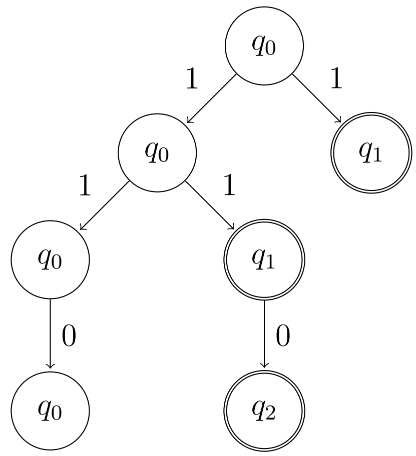


The formal definition of N is $(Q, \Sigma, \delta, q_0, \{q_1, q_2\})$, where

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{0, 1\}$
- δ is given as

	0	1	ϵ
q_0	$\{q_0\}$	$\{q_0, q_1\}$	$\emptyset$
q_1	$\{q_2\}$	$\emptyset$	$\emptyset$
q_2	$\emptyset$	$\emptyset$	$\emptyset$

The tree of running the input string 110 is shown below.



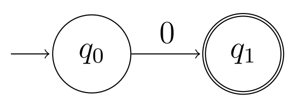
In the end 110 is accepted.

(b) First, we show that any NFA that recognizes A must satisfy the following:

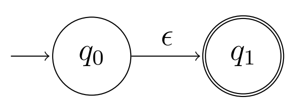
(i) It has at least an accept state since 1 is accepted.

(ii) The start state cannot be an accept state, for otherwise ϵ would be accepted.

(iii) We cannot have the following subgraph, for otherwise 0 would be accepted.

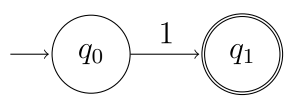


(iv) We cannot have the following subgraph, for otherwise ϵ would be accepted.

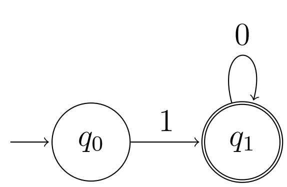


Now we check the number of states as follows:

- 1 state: By (i) and (ii), we have a contradiction.
- 2 states: By (i), (ii), (iii), and (iv), for the accepted state to be reachable, we first have the following subgraph:



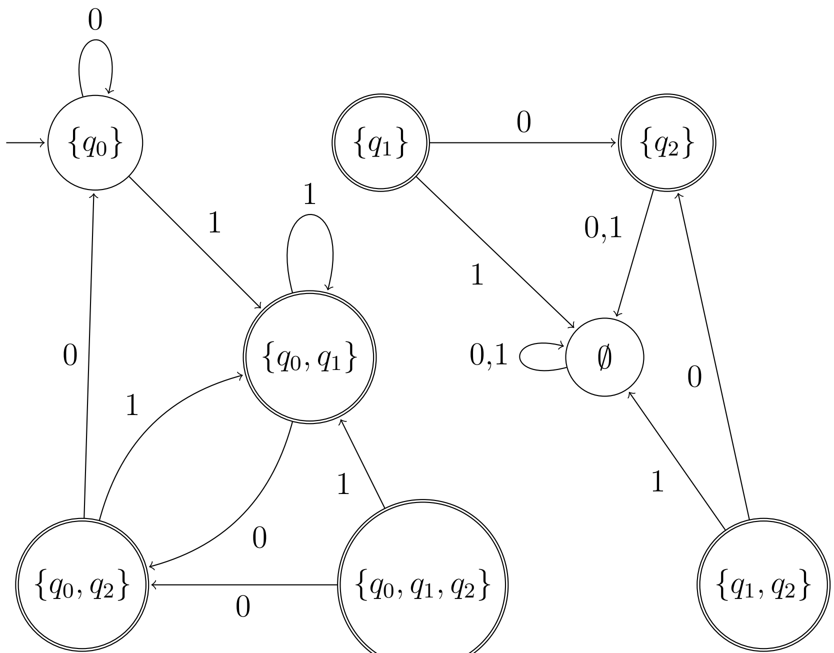
Next, to accept 10, we must add an edge and have the following subgraph:



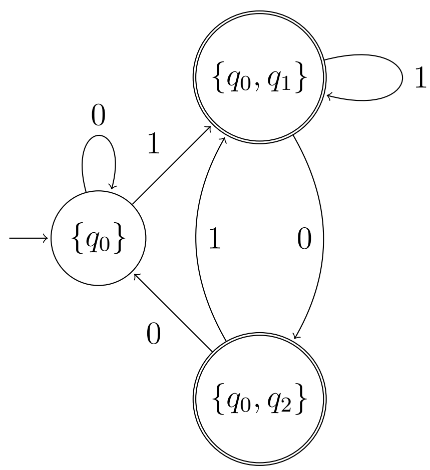
However, this would also accept 100, a contradiction.

So at least 3 states are required.

(c) Below is the converted DFA:

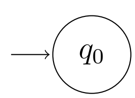


(d) Removing nodes without incoming edges, we have a DFA with 3 states:

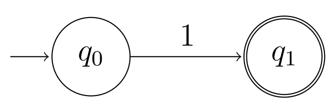


Now we prove step-by-step that there can be only one 3-state DFA recognizing A.

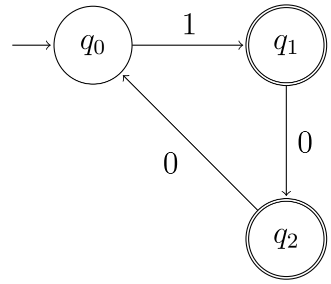
- First, we must have a start state q_0, and it cannot be an accept state because $\epsilon \notin A$. So we have the following subgraph.



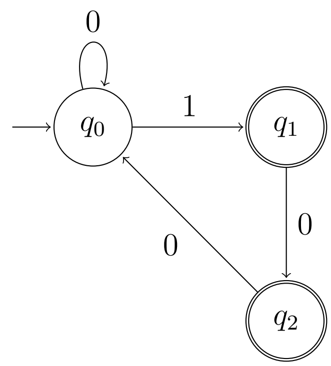
- Because $1 \in A$, there must be an accept state $q_1 = \delta(q_0, 1)$.



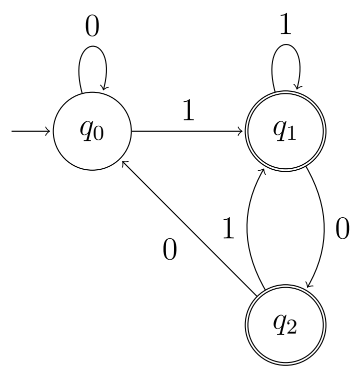
- To accept 10, but reject 100, we must introduce another state $q_2 = \delta(q_1, 0)$ because either $\delta(q_1, 0) = q_0$ or $\delta(q_1, 0) = q_1$ fail to satisfy the requirement. Because $100 \notin A$, we obtain $\delta(q_2, 0) = q_0$.



- Now the number of states has achieved 3, meaning that we will not add a new state anymore. Next, we consider $0 \notin A$, which implies $\delta(q_0, 0) = q_0$ and the figure becomes

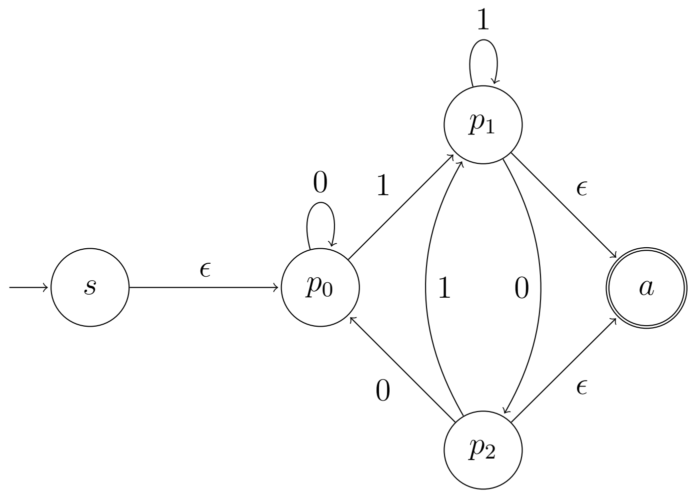


- Next, $110 \in A$ and $1010 \in A$ leads to $\delta(q_1, 1) = q_1$ and $\delta(q_2, 1) = q_1$, respectively.

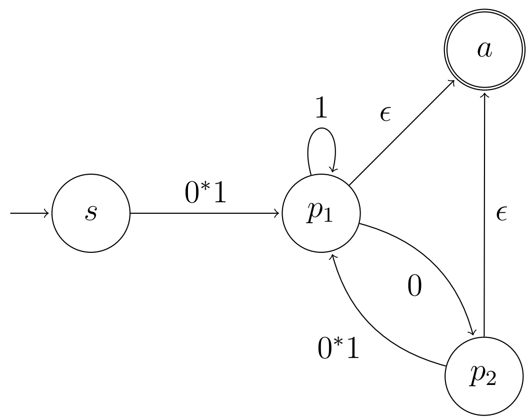


- Now we obtain a complete DFA, which is exactly the same as the one we derived. Hence, there is only one 3-state DFA recognizing A.

(e) First we convert the DFA (after renaming) to a GNFA (in the special form):

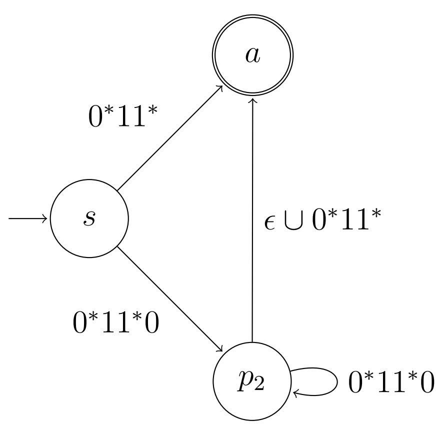


After removing p_0, with $1 \cup 00^*1 = 0^*1$ we have

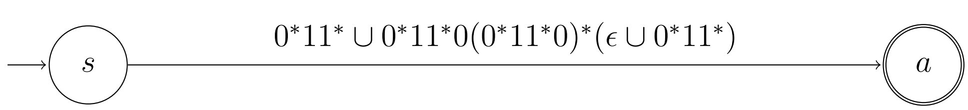


Now there are two cases:

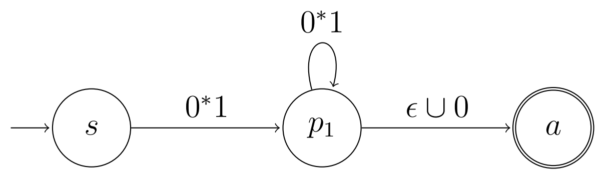
- First p_1 and then p_2.
  After removing p_1, we have



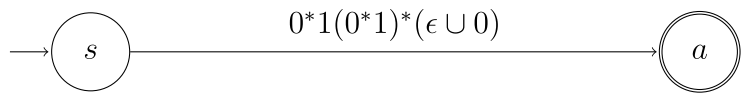
After removing p_2, we have



- First p_2 and then p_1.
  After removing p_2, again with $1 \cup 00^*1 = 0^*1$ we have



  After removing p_1, we have



Hence we may get either

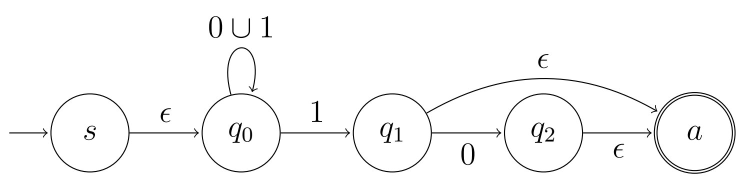
$$0^*11^* \cup 0^*11^*0(0^*11^*0)^*(\epsilon \cup 0^*11^*)$$

or

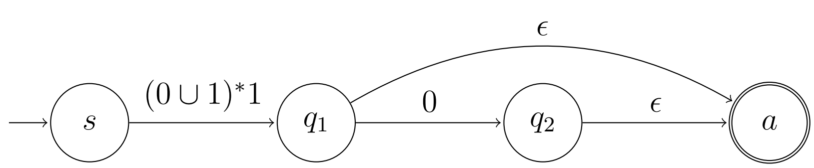
$$0^*1(0^*1)^*(\epsilon \cup 0),$$

both of which are regular expressions that describes A.

(f) First we convert the NFA to a GNFA (in the special form):

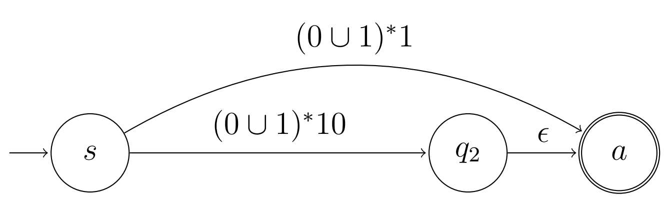


After removing q_0, we have

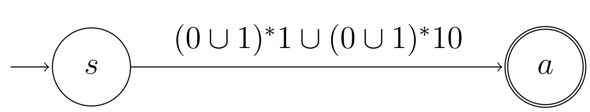


Now there are two cases:

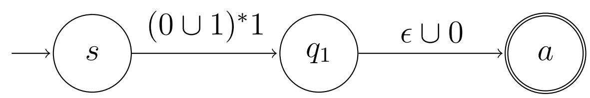
- First q_1 and then q_2.
  After removing q_1, we have



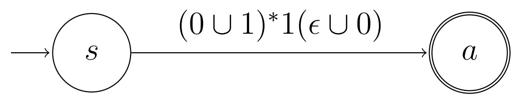
  After removing q_2, we have



- First q_2 and then q_1.
  After removing q_2, we have



  After removing q_1, we have



Hence we may get either

$$(0 \cup 1)^*1 \cup (0 \cup 1)^*10$$

or

$$(0 \cup 1)^*1(\epsilon \cup 0),$$

both of which are regular expressions that describes A.

Note that in (e) and (f) we have found 4 different forms of regular expressions that all describe A. (If not forced to remove the start state first, there are even more.) It seems that the last one is the simplest and most *intuitive* one.

Problem 3 (15 pts). Is the language $B = \{1^n \mid n = 2^k, k \geq 0\}$ regular or not? Either provide an NFA that recognizes B and *explain why it does*, or use pumping lemma to prove it is not regular.

Solution. No, B is not regular.

Proof. Suppose B is regular, and let p be the pumping length. Let $s = 1^{2^p}$. Because $s \in B$ and $|s| = 2^p > p$, by pumping lemma $s = xyz$, where $|y| > 0$ and $|xy| \leq p$. Consider $s' = xy^2z$. Because $0 < |y| \leq p < 2^p$, we have $|s'| = 2^p + |y| \in (2^p, 2^{p+1})$, implying $|s'|$ cannot be in the form 2^q for integer q. So $s' \notin B$. $\square$

Problem 4 (25 pts). Let $\Sigma = \{0, 1\}$. For any $w = w_1 w_2 \cdots w_n \in \Sigma^*$, where $w_i \in \Sigma$ for $1 \leq i \leq n$, the *substrings* of w are $w_{i:j} = w_i w_{i+1} \cdots w_j$, where $i \leq j$. Note that we do not consider ϵ to be a substring here. In particular, the *prefixes* of w are $w_{1:j} = w_1 \cdots w_j$ and the *suffixes* of w are $w_{i:n} = w_i \cdots w_n$, for $1 \leq i, j \leq n$. Also, define the *0/1 difference* of w to be

$$d(w) \equiv |\text{ (number of 0's in } w) - \text{(number of 1's in } w) \text{ }|.$$

For example, if $x = 100111$, then $d(x) = |2 - 4| = 2$.

(a) (10 pts) Consider the language

$$C = \{w \in \Sigma^* \mid d(w) \leq 2\}$$

of all strings whose 0/1 difference is at most 2. For example, $x \in C$.

Is C regular or not? Either provide a DFA/NFA that recognizes C and *explain why it does*, or use pumping lemma to prove it is not regular.

(b) (10 pts) Consider the language

$$D = \{w \in \Sigma^n \mid n \geq 1 \text{ and } d(w_{1:j}) \leq 2 \text{ for } 1 \leq j \leq n\}$$

of all strings whose *every prefix*'s 0/1 difference is at most 2. For example, $x \in D$, since all its prefixes' 0/1 differences are at most 2.

Is D regular or not? Either provide a DFA/NFA that recognizes D and *explain why it does*, or use pumping lemma to prove it is not regular.

(c) (5 pts) Consider the language

$$E = \{w \in \Sigma^n \mid n \geq 1 \text{ and } d(w_{i:n}) \leq 2 \text{ for } 1 \leq i \leq n\}$$

of all strings whose *every suffix*'s 0/1 difference is at most 2. For example, $x \notin E$, since $d(x_{4:6}) = d(111) = 3$.

Is E regular or not? Either provide a DFA/NFA that recognizes E and *explain why it does*, or use pumping lemma to prove it is not regular. *Hint: Consider the relationship between D and E.*

Solution.

(a) No, C is not regular.

Proof. Suppose C is regular, and let p be the pumping length. Pick $s = 0^{p+2}1^p$. Clearly $|s| = 2p + 2 \geq p$, and $s \in C$ since $d(s) = 2$, so by pumping lemma $s = xyz$, where $|y| > 0$ and $|xy| \leq p$.

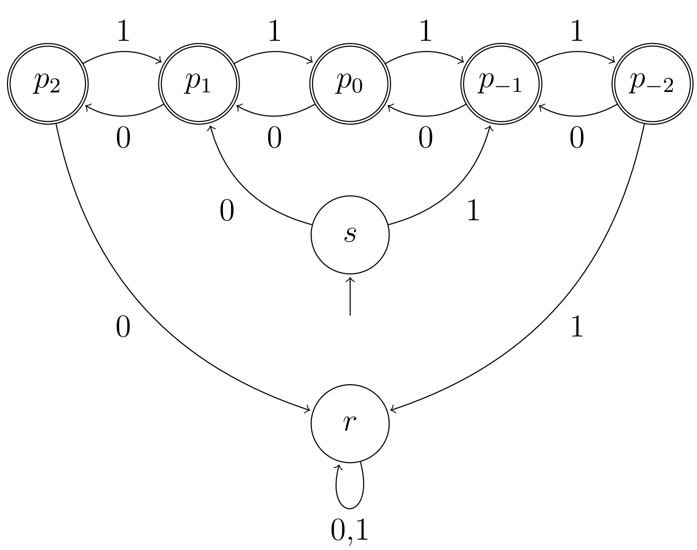
Consider $s' = xy^2z$. Because $0 < |y| \leq |xy| \leq p$ and the first p characters of s are all 0, y must consist of purely 0's. Hence $d(s') = (p + 2 + |y|) - p = |y| + 2 > 2$, so $s' \notin C$, a contradiction. □

(b) Yes, D is regular.

Proof. We prove D is regular by giving a DFA that recognizes it. First, observe that $w \in D$ if and only if for any $1 \leq j \leq n$,

$$d'(w_{1:j}) \equiv (\text{number of 0's in } w_{1:j}) - (\text{number of 1's in } w_{1:j}) \in \{2, 1, 0, -1, -2\}.$$

So we use 5 accept states to keep track of the current $d'(w_{1:j})$ after reading the first j characters, and go to another reject state immediately if $d'(w_{1:j})$ becomes ± 3. See the following DFA.



The DFA starts with the state s. At any time, if 1 is read, the state would move rightwards, meaning that the number of 1 is one larger than before. Conversely, if 0 is read, it moves leftwards, meaning that the number of 0 is one larger than before. Note that each character read changes 0/1 difference exactly by 1.

If the current 0/1 difference becomes ± 3, it must go from p_2 or p_{-2} to the reject state r, and stay at r regardless of the rest of the string. On the other hand, if all the prefixes never violate the requirement, the string would end up at one of the 5 accept states and be accepted.

By the above argument, the DFA recognizes E. $\square$

(c) Yes, E is regular.

Proof. By (b), we know that D is regular.

We then define the *reverse* w^R of a string $w = w_1 w_2 \cdots w_n$ to be w in reverse order, i.e. $w^R = w_n \cdots w_2 w_1$. For example, $(00101)^R = 10100$. We then prove that for any regular language L, its *reverse* $L^R \equiv \{w^R \mid w \in L\}$ is also regular (See problem 1.31 of the textbook). Since $E = D^R$ (Details omitted. Verify it!), E is also regular.

- Given any regular language L, there is a DFA $A = (Q, \Sigma, \delta, q_0, F)$ that recognizes L. Now we construct an NFA $A' = (Q \cup \{q_0'\}, \Sigma, \delta', q_0', \{q_0\})$, assuming $q_0' \notin Q$, where for $q \in Q \cup \{q_0'\}, a \in \Sigma_\epsilon$,

$$\delta'(q, a) = \begin{cases} F & q = q_0', a = \epsilon \\ \emptyset & q = q_0', a \in \Sigma \\ \{p \in Q \mid \delta(p, a) = q\} & q \in Q. \end{cases}$$

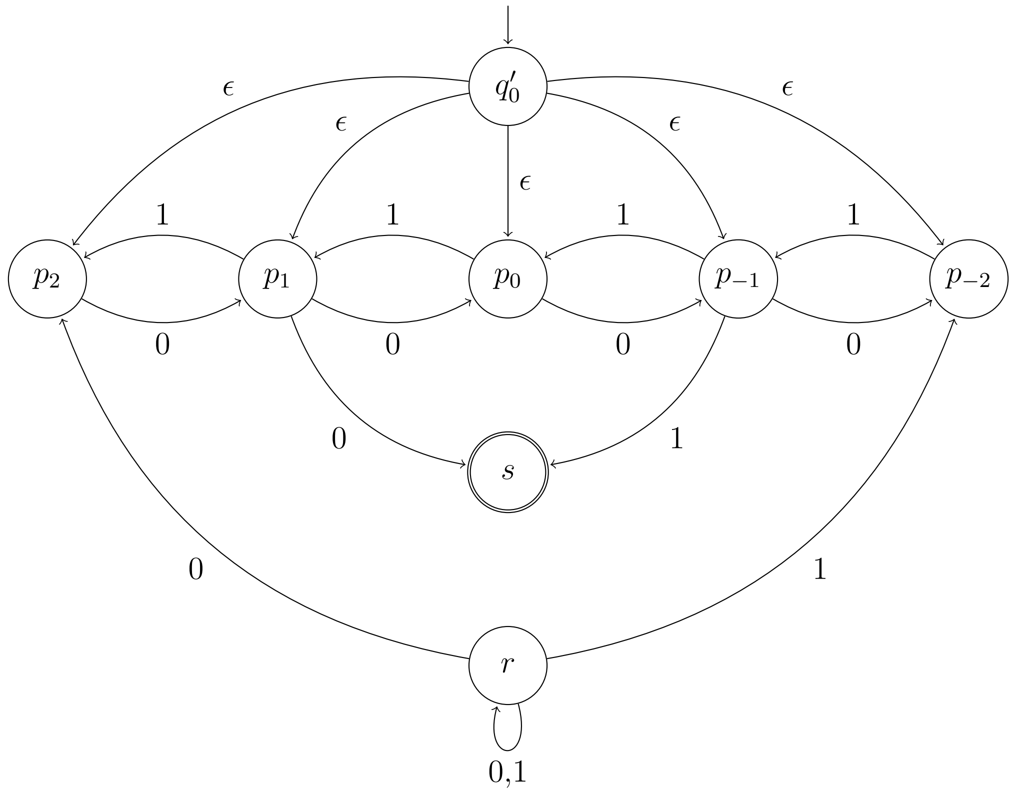
  Intuitively, we get A' by applying the following to A :

  - Reverse the directions of all transitions.
  - Make a new start state, and make ϵ-transitions from it to all the original final states.
  - The original start state becomes the new final state.

  By definition, it can be verified that A' recognizes L^R.

Therefore, E is regular. $\square$

The NFA that recognizes E is provided below.



Note that r has no incoming edge from other nodes, so we can remove it and obtain a simpler NFA as follows.