
Problem Set 1 Solutions

Problem 1 Give the formal description of the machine M_1 pictured in Exercise 1.1, page 83 of the text.

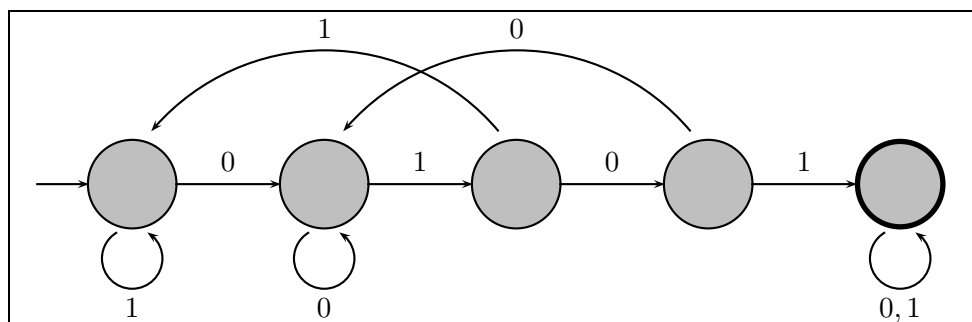
Before trying to do this exercise, refer to Definition 1.1, page 35 of the text and make sure you understand it. Now, the formal description is $M_1 = (Q, \Sigma, \delta, q_0, F)$ where

- $Q = \{q_1, q_2, q_3\}$
- $\Sigma = \{a, b\}$
- $\delta: Q \times \Sigma \rightarrow Q$ is defined via

$$\begin{array}{ll} \delta(q_1, a) = q_2 & \delta(q_1, b) = q_1 \\ \delta(q_2, a) = q_3 & \delta(q_2, b) = q_3 \\ \delta(q_3, a) = q_2 & \delta(q_3, b) = q_1 \end{array}$$

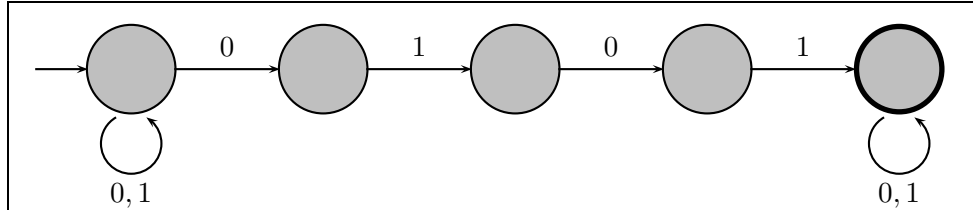
- $q_0 = q_1$
- $F = \{q_2\}$

Problem 2 Exercise 1.4 part (c), page 84 of the text.



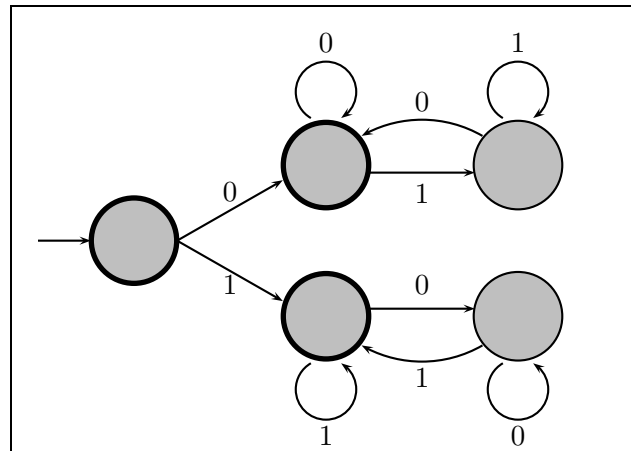
Problem 3 Exercise 1.5 part (b), page 84 of the text.

The DFA of Problem 2 above has five states and any DFA is an NFA, so that is already a solution. However, the following is a simpler and easier to understand NFA:



Problem 4 Problem 1.41, page 90 of the text.

Below is a five state NFA for the language in question.



Problem 5 Problem 1.31, page 88 of the text.

It is important to first understand exactly what it means for an all-paths NFA to accept or reject. If the input is in the language, it is required that *all* possible paths that may be followed on that input end in accepting states. What about rejection? Reject is defined as “not accept.” This means an input is rejected if there exists some path, on this input, that leads to a reject state. Note rejection means there exists a rejecting path, not that all paths are rejecting!

(a) Let L be an arbitrary regular language. Show that there is an all-paths-NFA that accepts L .

Given: L is regular. So there is a DFA M that accepts L .

Want: To construct an all-paths-NFA M' that accepts L .

Construction: Just let $M' = M$. That is, I claim the given DFA does the job.

Correctness of construction: This is pretty clear because since M is a DFA, there is, on any input, only one path that can be followed. When the input x is in L this one path is accepting, so all paths are accepting. When x is not in L this one path is rejecting, so there exists a rejecting path.

(b) Let L be a language accepted by some all-paths-NFA M . Show that L is regular.

We are given that L is accepted by some all-paths NFA. Let M' be this NFA. We want to show that L is regular.

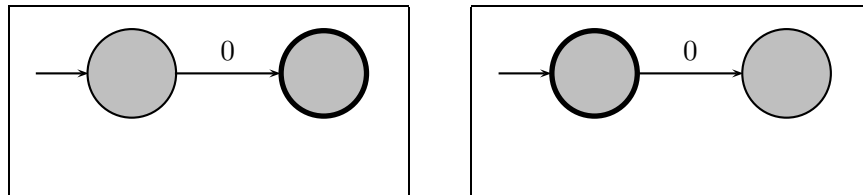
The trick is to consider the machine M formed by reversing accept and reject states in M' . That is, M is the same as M' except that all states of M' that were accepting are now rejecting, and all states that were rejecting are now accepting. What kind of machine is this, and what language does it accept?

A little thought shows that M is an NFA that accepts the complement \bar{L} of L . Why? If $x \in \bar{L}$ then $x \notin L$ so some path of M' ends in a reject state. So some path of M ends in an accept state, meaning M accepts x . On the other hand if $x \notin \bar{L}$, meaning $x \in L$, then all paths of M' end in accept. So all paths of M end in reject, meaning M rejects x .

But this means \bar{L} is regular, since it is accepted by an NFA. But we know that the class of regular languages is closed under complement. So L is also regular.

Problem 6 Exercise 1.10 part (b), page 85 of the text.

Consider the following NFAs. The one on the left recognizes the language $\{0\}$. Swapping the accept and non-accept states produces the NFA on the right, which recognizes the language $\{\varepsilon\} \neq \overline{\{0\}}$.



The class of languages recognized by NFAs is the class of regular languages, which is closed under complement, as was proved in class.

Note that this is not a contradiction. The example above shows that the construction used to prove that regular languages are closed under complement does not work for NFAs. However, given an NFA that recognizes a language C it is possible to construct an NFA that recognizes \bar{C} (for example, by constructing a DFA equivalent to the given NFA and then swapping its accept and non-accept states).

Problem 7 Problem 1.24, page 88 of the text.

In class we proved the closure of regular languages under several operations, namely complement, union, intersection and Kleene Star. Here we have to prove closure under reversal. Besides proving the closure property in question, this solution, and the ones in class, are templates for writing proofs of other closure properties on quizzes. You should remember the template and use it. In particular your proof of a closure property on a quiz is expected to have the four parts indicated

below, namely **Given, Want, Construction, Correctness of construction**. Of course, what you write for each part varies from problem to problem.

Given: A is regular. So there is a DFA $M = (Q, \Sigma, \delta, q_0, F)$ that accepts A .

Want: To show that $A^{\mathcal{R}}$ is regular. We will do this by constructing an NFA N that recognizes $A^{\mathcal{R}}$. Note that we will construct an NFA to accept $A^{\mathcal{R}}$, not a DFA. The non-determinism will be very useful.

Construction: Let us first describe the idea. What we want is to “run M backwards.” We would like to start in a final state of M and, if the input is in $A^{\mathcal{R}}$, end in the start state of M . To do this, we will reverse the directions of all arrows in M . (Notice the resulting machine may not be a DFA because after reversal we may have more than one arrow with a particular label out of some state. That is why we are looking to NFAs.) We also make the start state of M a final state. What now is the new start state? Since M has many final states, we introduce a new start state and put in ε jumps to all the old final states of M .

Now let us describe the construction formally. We let $N = (Q', \Sigma, \delta', s, F')$ where

- $Q' = Q \cup \{s\}$ where $s \notin Q$ is a new state we have introduced
- δ' is defined as follows for any $q \in Q'$ and any $\sigma \in \Sigma \cup \{\varepsilon\}$:

$$\delta'(q, \sigma) = \begin{cases} \{r \in Q : \delta(r, \sigma) = q\} & \text{if } q \in Q \text{ and } \sigma \in \Sigma \\ F & \text{if } q = s \text{ and } \sigma = \varepsilon \\ \emptyset & \text{otherwise.} \end{cases}$$

- $F' = \{q_0\}$

This completes the description of the construction. The claim is that N accepts $A^{\mathcal{R}}$. This must be proved.

Correctness of construction: Consider the operation of N on an input x . First assume $x = x_1 \dots x_n \in A^{\mathcal{R}}$. We want to show N accepts x . Well, we know that $x^{\mathcal{R}} = x_n \dots x_1 \in A$, so M accepts $x^{\mathcal{R}}$. Let us look at the path followed in M on this input. It has the form

$$q_0 \xrightarrow{x_n} q_1 \xrightarrow{x_{n-1}} q_2 \xrightarrow{x_{n-2}} \dots q_{n-1} \xrightarrow{x_1} q_n .$$

where q_0, \dots, q_n are some states in Q , namely the ones followed on this path. Now consider the path

$$s \xrightarrow{\varepsilon} q_n \xrightarrow{x_1} q_{n-1} \xrightarrow{x_2} \dots q_1 \xrightarrow{x_n} q_0 .$$

This is a valid path on input x in the NFA N , and it ends in an accept state of N . So N accepts x . Notice that we used the non-determinism. We guess the right path back, out of many possibilities.

Now assume $x \notin A^{\mathcal{R}}$. This means $x^{\mathcal{R}} \notin A$. We must show that there is *no* accepting path in N on input x . We argue by noting that if there were some accepting path in N on input x then reversing it would yield an accepting path in M on input $x^{\mathcal{R}}$. (This is not hard to see, in the same way as above.) But since M does not accept $x^{\mathcal{R}}$ we know that the path followed in M on input $x^{\mathcal{R}}$ is in fact not accepting. So there can be no accepting path in N on input x .

Problem 8 Problem 1.32, page 89 of the text.

(a) **Given:** A is regular. So there is a DFA $M = (Q, \Sigma, \delta, q_0, F)$ that accepts A .

Want: To show that

$$\text{NOPREFIX}(A) = \{ w \in A : \text{No proper prefix of } w \text{ is a member of } A \}$$

is also regular.

Construction and correctness: Let us first consider the following language:

$$L_1 = \{ w : \text{some string } y \text{ in } A \text{ is a proper prefix of } w \}.$$

We construct an NFA $N = (Q_1, \Sigma, \delta_1, q_1, F_1)$ that recognizes L_1 as follows:

- $Q_1 = Q \cup \{q_f\}$, where $q_f \notin Q$ is a new state we have introduced
- δ_1 is defined as follows for any $q \in Q_1$ and any $\sigma \in \Sigma \cup \{\varepsilon\}$:

$$\delta_1(q, \sigma) = \begin{cases} \{\delta(q, \sigma)\} & \text{if } q \in Q - F \text{ and } \sigma \in \Sigma \\ \{\delta(q, \sigma)\} \cup \{q_f\} & \text{if } q \in F \text{ and } \sigma \in \Sigma \\ \{q_f\} & \text{if } q = q_f \text{ and } \sigma \in \Sigma \\ \emptyset & \text{otherwise.} \end{cases}$$

- $q_1 = q_0$
- $F_1 = \{q_f\}$

N recognizes L_1 because:

- If w is a string in L_1 , there is a string y in A which is a proper prefix of w , or $w = yx$, where x is not empty. If w is taken as the input of N , some computation on y will end at an accepting state in M , and then some computation on x part can end at the state q_f , which means w is accepted by N .
- If w is a string accepted by N , there is some computation that ends at q_f if w is taken as the input of N . That computation must arrive at one of the accepting states in M before it ends at q_f . Say the proper prefix of w which used for that part of computation is y , we can see that if y is the input of M , it will end at one of its accepting state, which means y is in A and w is a string in L_1 .

It is obvious that $\text{NOPREFIX}(A) = A \cap \overline{L_1}$. The class of regular languages is closed under intersection and complement, so $\text{NOPREFIX}(A)$ is regular.

(b) **Given:** A is regular. So there is a DFA $M = (Q, \Sigma, \delta, q_0, F)$ that accepts A .

Want: To show that

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

is also regular.

Construction and correctness: Let us first consider the following language:

$$L_1 = \{ w : w \text{ is a proper prefix of some string } y \text{ in } A \}.$$

We construct a DFA $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ that recognizes L_1 as follows:

- $Q_1 = Q$
- $\delta_1 = \delta$
- $q_1 = q_0$
- $F_1 = \{ q \in Q : \text{there exists } p \in F - \{q\} \text{ such that } q \text{ is on a path from } q_0 \text{ to } p \}$

M_1 recognizes L_1 because:

- If w is a string in L_1 , it is a proper prefix of some string y in A . If M take y as the input, it will end at one of its accepting states p , thus if M_1 take w as its input, it will end at a state on the path from q_0 to p , which means w will end at an accepting state if taken as the input of M_1 .
- If w is accepted by M_1 , it ends at an accepting state in M_1 , which means it ends a state in M which is on a path from q_0 to an accepting state in M . Thus w is a proper prefix of some string y in A and is in the language L_1 .

It is obvious that $\text{NOEXTEND}(A) = A \cap \overline{L_1}$. The class of regular languages is closed under intersection and complement, so $\text{NOEXTEND}(A)$ is regular.

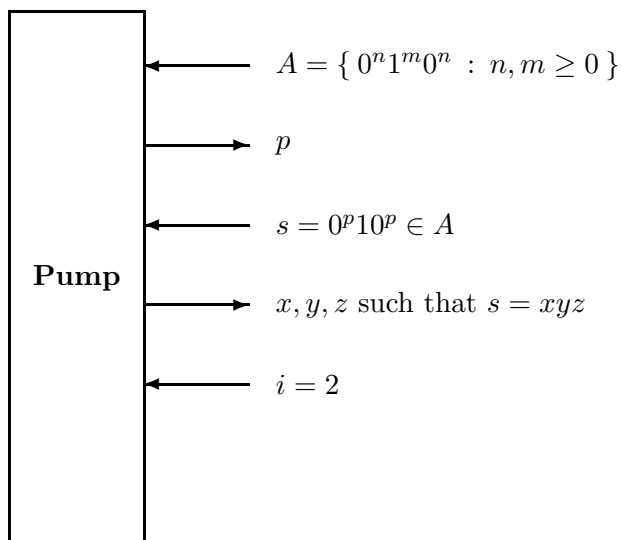
Problem 9 Problem 1.23 part (a), page 88 of the text.

Use this solution as a template for proving non-regularity via the pumping lemma.

The proof is by contradiction.

Assume: $A = \{ 0^n 1^m 0^n : n, m \geq 0 \}$ is regular.

The assumption means that the pumping lemma (Theorem 1.37, page 78 of the text) applies to A . We imagine ourselves “interacting” with the lemma as follows:



We give it A , and it returns a pumping length p . Now, we choose a string $s \in A$ of length greater than p , and return it to the lemma. The choice of string is important for the rest of the argument, and we set it to $s = 0^p 10^p$. Because s is in A and has a length greater than p , the pumping lemma says that s can be split into xyz which obey the three conditions of the pumping lemma. The lemma returns x, y, z to us. We then choose $i = 2$ and return it to the lemma. At this point, the lemma guarantees that

- (1) $xy^2z \in A$
- (2) $|y| > 0$
- (3) $|xy| \leq p$

The interaction emphasizes what we can choose and what we cannot choose. We are allowed to choose s and i , but, having chosen s , we have no control over x, y, z . But we know they satisfy the three conditions.

Because the first p symbols of s are all 0, we know by condition (3) above that x and y must contain only zeros. Condition (2) states that y must have length greater than zero, so we know y contains at least one zero, say $y = 0^j$ where $j \geq 1$. So $xy^2z = x0^{2j}z$. We know that $xyz = 0^p 10^p$, so this means that the string xy^2z has the form $0^a 10^p$ where $a = p + j$. Since $a \neq p$, the definition of A tells us that $xy^2z \notin A$. But condition (1) says $xy^2z \in A$. They can't both be true, so we have a contradiction. This means our assumption that A was regular is false.

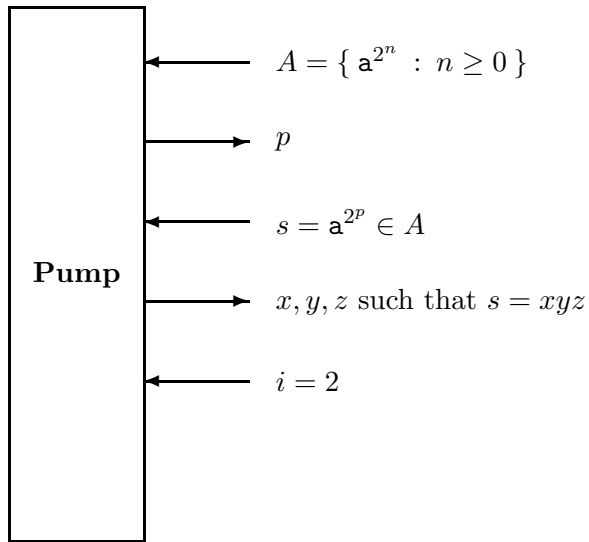
Problem 10 Exercise 1.17 part (c), page 86 of the text.

Use this solution as a template for proving non-regularity via the pumping lemma.

The proof is by contradiction.

Assume: $A = \{ a^{2^n} : n \geq 0 \}$ is regular.

The assumption means that the pumping lemma (Theorem 1.37, page 78 of the text) applies to A . We imagine ourselves “interacting” with the lemma as follows:



We give it A , and it returns a pumping length p . Now, we choose a string $s \in A$ of length greater than p , and return it to the lemma. The choice of string is important for the rest of the argument, and we set it to $s = \mathbf{a}^{2^p}$. Because s is in A and has a length greater than p , the pumping lemma says that s can be split into xyz which obey the three conditions of the pumping lemma. The lemma returns x, y, z to us. We then choose $i = 2$ and return it to the lemma. At this point, the lemma guarantees that

- (1) $xy^2z \in A$
- (2) $|y| > 0$
- (3) $|xy| \leq p$

Conditions (2) and (3) tell us that $y = \mathbf{a}^j$ for some j satisfying $1 \leq j \leq p$. The length of xy^2z is $|s| + |y|$ since $s = xyz$, and s has length 2^p . So the length of xy^2z satisfies $2^p + 1 \leq |xy^2z| \leq 2^p + p$. But $2^p + p < 2^{p+1}$ for any number $p \geq 1$, so the length of xy^2z is not a power of two. So by definition of A the string xy^2z is not in A . But condition (1) says $xy^2z \in A$. They can't both be true, so we have a contradiction. This means our assumption that A was regular is false.

Problem 11 Problem 1.42, page 90 of the text.

Given: A is regular, so there is a DFA $M = (Q, \Sigma, \delta, q_0, F)$ that accepts A .

Want: To show that

$$A_{\frac{1}{2}-} = \{ x : \text{For some } y, |x| = |y| \text{ and } xy \in A \}$$

is regular. We will do this by constructing an NFA N that recognizes $A_{\frac{1}{2}-}$.

Construction: Let $N = (Q', \Sigma, \delta', s, F')$ where

- $Q' = (Q \times Q) \cup \{s\}$

- δ' is defined as follows for any $q = (r, t) \in Q \times Q$ and any $\sigma \in \Sigma \cup \{\varepsilon\}$:

$$\delta'((r, t), \sigma) = \begin{cases} \{ (\delta(r, \sigma), q) : q \in Q \text{ and } \exists \alpha \in \Sigma \text{ such that } \delta(q, \alpha) = t \} & \text{if } \sigma \in \Sigma \\ \emptyset & \text{if } \sigma = \varepsilon \end{cases}$$

Also set $\delta'(s, \varepsilon) = \{ (q_0, q) : q \in F \}$ and $\delta'(s, \sigma) = \emptyset$ if $\sigma \in \Sigma$.

- $F' = \{ (q, q) : q \in Q \}$.

This completes the description of the construction. The claim is that N accepts $A_{\frac{1}{2}-}$. This must be proved.

Correctness of construction: Consider the operation of N on an input x . First assume $x = x_1 \cdots x_n \in A_{\frac{1}{2}-}$. We want to show that N accepts x . Since $x \in A_{\frac{1}{2}-}$, there exists some string $y = y_1 \cdots y_n$ such that $xy \in A$. Since M recognizes A , M accepts xy . Consider the path followed in M on this input. It has the form

$$q_0 \xrightarrow{x_1} q_1 \xrightarrow{x_2} \cdots q_{n-1} \xrightarrow{x_n} q_n \xrightarrow{y_1} q_{n+1} \xrightarrow{y_2} \cdots q_{2n-1} \xrightarrow{y_n} q_{2n},$$

where q_0, \dots, q_{2n} are some states in Q , namely the ones followed on this path, and q_{2n} is an accept state. Now consider the path

$$s \xrightarrow{\varepsilon} (q_0, q_{2n}) \xrightarrow{x_1} (q_1, q_{2n-1}) \xrightarrow{x_2} \cdots (q_{n-1}, q_{n+1}) \xrightarrow{x_n} (q_n, q_n).$$

This is a valid path on input x in the NFA N , and it ends in an accept state of N . Therefore, N accepts x .

Now assume $x \notin A_{\frac{1}{2}-}$. We must show that there is *no* accepting path in N on input x . We provide a proof by contradiction. Assume that there is an accepting path in N on input x . This path has the form

$$s \xrightarrow{\varepsilon} (q_0, q_{2n}) \xrightarrow{x_1} (q_1, q_{2n-1}) \xrightarrow{x_2} \cdots (q_{n-1}, q_{n+1}) \xrightarrow{x_n} (q_n, q_n),$$

where q_0, \dots, q_{2n} are some states in Q and q_{2n} is an accept state of DFA M . By the definition of the transition function δ' , for $i = 1, \dots, n$, $\delta(q_{i-1}, x_i) = q_i$ and there exists $\alpha_i \in \Sigma$ such that $\delta(q_{n+i-1}, \alpha_i) = q_{n+i}$. Now consider the path

$$q_0 \xrightarrow{x_1} q_1 \xrightarrow{x_2} \cdots q_{n-1} \xrightarrow{x_n} q_n \xrightarrow{\alpha_1} q_{n+1} \xrightarrow{\alpha_2} \cdots q_{2n-1} \xrightarrow{\alpha_n} q_{2n}.$$

This is a valid path on input $x\alpha_1 \cdots \alpha_n$ in the DFA M , and it ends in an accept state of M . Therefore, M accepts $x\alpha_1 \cdots \alpha_n$. But, $|\alpha_1 \cdots \alpha_n| = |x|$ and since $x \notin A_{\frac{1}{2}-}$ we know that for all y such that $|y| = |x|$, $xy \notin A$, so $x\alpha_1 \cdots \alpha_n \notin A$. Thus M accepts a string that is not in A . This contradicts the assumption that M recognizes A .
