CISS362: Introduction to Automata Theory, Languages, and Computation
Assignment a07

Name: ____

Objectives

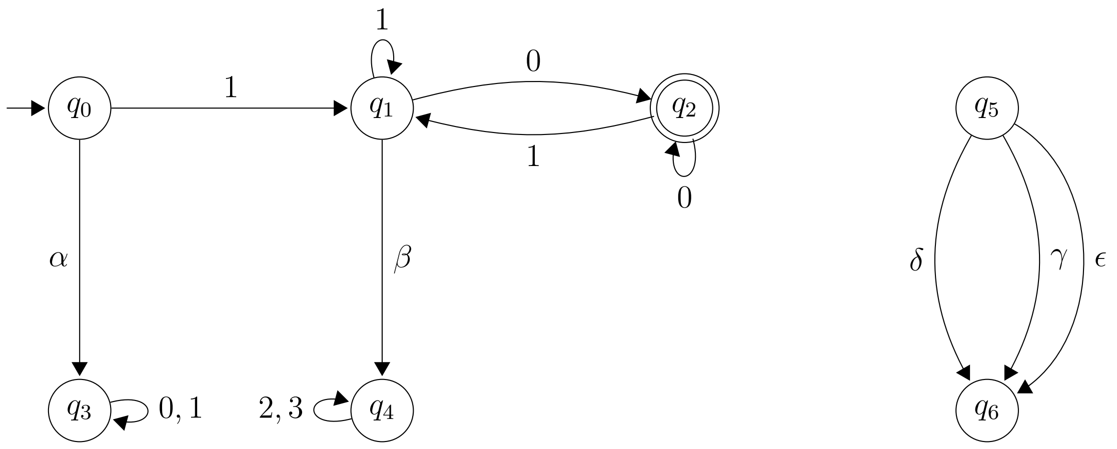
- Design DFAs.
- Design NFAs.
- Design regexes.
- Show a language is regular by designing a DFA, NFA, or regex.
- Show a language construction (operator) is a closed operator on regular languages.

For questions asking you show an operation on regular language is closed, you need not prove your construction is correct. (But you are welcome to do so. Most of such proofs, if not immediate, is by induction.) Drawing a diagrma usually helps. For solutions provided, make sure you study it carefully as a guide for answering such questions.

- Sipser 1.31. Solution provided. Study it carefully. This is from a06.
- Sipser 1.32. Q1
- Sipser 1.33. Q2
- Sipser 1.34. Q3
- Sipser 1.35. Q4
- Sipser 1.36. Solution provided. Study it carefully.
- Sipser 1.37. Q5
- Sipser 1.38. Q6
- Sipser 1.40. Q7. Solution to 1.40(a) is in the textbook. Study it carefully.
- Sipser 1.41. Q8
- Sipser 1.42. Q9

How to draw a state diagram

Here's an example showing you how to draw the elements of a state diagram. Also, look at the solution to 1.3 below.



For more information on drawing state diagrams go to my tutorials and look for `latex-automata.pdf`:

https://drive.google.com/file/d/1AeE-POWNvQlitzPDxQpGE8bMR9Yc9gMW

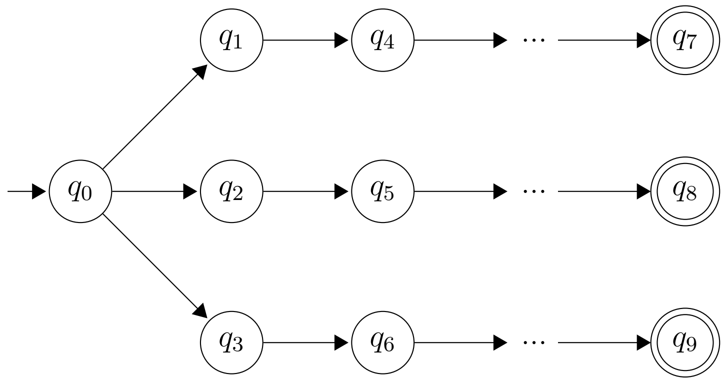
Let me know if you have any questions about drawing state diagram.

Sipser 1.31. (This is from a06. Solution provided.)

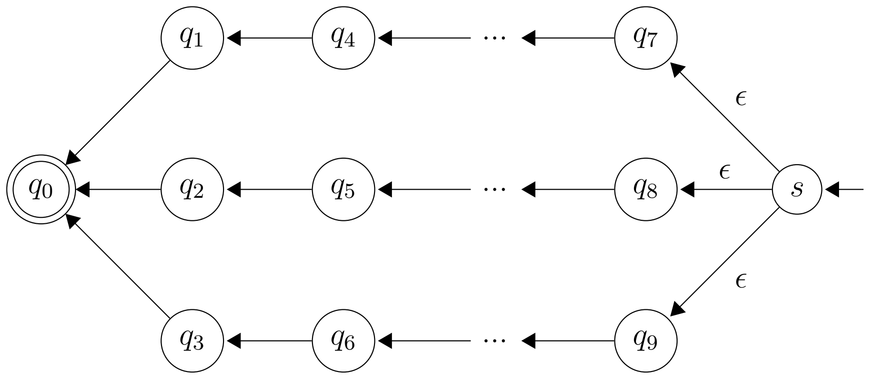
Solution.

We want to show that if A is regular, then A^R is also regular.

Let M be a DFA for A:



Suppose M accepts the string $abaab$. That means there's a path from the initial state of M to an accept state that travels along a sequence of transitions labeled a, b, a, a, b. Suppose the accept state is q_7. Then traveling in the reverse direction of this path, we will see the symbols b, a, a, b, a, going from q_7 to q_0. Therefore we will need to construction an automata from M where the direction of the transitions are reversed. For this to be a valid automata, we can have only one start state. M might have multiple accept states. We cannot simply choose anyone of the accept states of M. That's not a problem: we will use nondeterminism to allow us to try all the accept states:



Note that in the new automata, the accept state is q_0. That's the general idea. We are now ready to construction our automata.

Let $M = (\Sigma, Q, q_0, F, \delta)$ be a DFA accepting A. Define an NFA $N = (\Sigma, Q^R, s, F^R, \delta^R)$

where s is a new state (i.e., $s \notin Q$), $Q^R = Q \cup \{s\}$, $F^R = \{q_0\}$, and

$$\delta^R : Q \cup \{s\} \times \Sigma_\epsilon \to P(Q \cup \{s\})$$

is the transition function that behaves as stated above, i.e., they are basically transitions from M but with their directions reversed. Furthermore there are new transitions from s to all states in F.

First of all, in N, at state s, there are ϵ–transitions to all the states in F. Therefore

$$\delta^R(s, \epsilon) = F$$

Next, To describe the transitions of N which are the reverse of transitions of M, if in the DFA M, we have

$$\begin{tikzcd} q' \arrow[r, "c"] & q \end{tikzcd}$$

the new automata N will have

$$\begin{tikzcd} q' & q \arrow[l, "c"'] \end{tikzcd}$$

However, the definition of δ^R is not just

$$\delta^R : Q \cup \{s\} \times \Sigma_\epsilon \to P(Q \cup \{s\})$$
$$\delta^R(q, c) = \begin{cases} \{q'\} & \text{if } \delta(q', c) = q \\ F & \text{if } q = s \text{ and } c = \epsilon \end{cases}$$

for two reasons. The first correction is due to the fact that M we might have

$$\begin{tikzcd} q' \arrow[dr, "c"] & \\ & q \\ q'' \arrow[ur, "c"] & \end{tikzcd}$$

In this case

$$\delta^R(q, c) = \{q', q''\}$$

More generally

$$\delta^R(q, c) = \{q' \in Q \mid \delta(q', c) = q\}$$

Furthermore, this behavior of δ^R only applies to the case where $q \in Q$ and $c \in \Sigma$. Therefore the transition function of the new automata should be modified to this:

$$\delta^R : Q \cup \{s\} \times \Sigma_\epsilon \to P(Q \cup \{s\})$$
$$\delta^R(q,c) = \begin{cases} \{q' \in Q \mid \delta(q',c) = q\} & \text{if } q \neq s \text{ and } c \neq \epsilon \\ F & \text{if } q = s \text{ and } c = \epsilon \end{cases}$$

The second correction is that δ^R is not complete since it is not defined when $q = s$ and $c \neq \epsilon$ and when $q \neq s$ and $c = \epsilon$. From the diagram above, you see that in the new automata there are no ϵ–transitions other than from state s. In other word, we will need to fill in two blanks here:

$$\delta^R(q,c) = \begin{cases} \{q' \in Q \mid \delta(q',c) = q\} & \text{if } q \neq s \text{ and } c \neq \epsilon \\ F & \text{if } q = s \text{ and } c = \epsilon \\ ? & \text{if } q \neq s \text{ and } c = \epsilon \\ ? & \text{if } q = s \text{ and } c \neq \epsilon \end{cases}$$

From our diagram above, you see that $\delta^R(q,c) = \{\}$ for the last two cases. The complete definition of δ^R is therefore

$$\delta^R : Q \cup \{s\} \times \Sigma_\epsilon \to P(Q \cup \{s\})$$
$$\delta^R(q,c) = \begin{cases} \{q' \in Q \mid \delta(q',c) = q\} & \text{if } q \neq s \text{ and } c \neq \epsilon \\ F & \text{if } q = s \text{ and } c = \epsilon \\ \emptyset & \text{otherwise} \end{cases}$$

□

NOTES. Here are some DIYs.

1. Prove that if $w \in \Sigma^*$, then

$$\delta^{R*}(s,w) - \{s\} = \{q \in Q \mid \delta^*(q, w^R) \in F\}$$

   (Hint: Induction on $|w|$.)
2. Prove formally that $L(N) = L(M)$.

Q1. Sipser 1.32.

SOLUTION.

Q2. Sipser 1.33.

Solution.

Q3. Sipser 1.34.

Solution.

Q4. Sipser 1.35.

SOLUTION.

Sipser 1.36. Solution provided. Study it carefully.

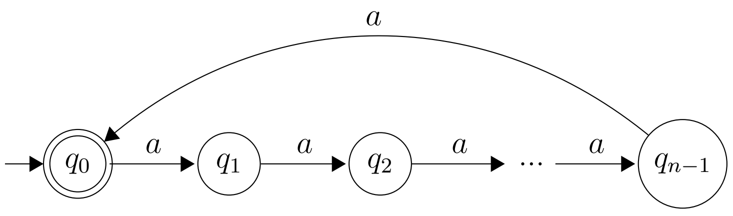
(Hint: In this case, it's easier not to use automata construction. Use a regular expression instead.)

SOLUTION.

SOLUTION 1: Let $n \geq 1$. We want to show

$$B_n = \{a^k \mid k \text{ is a multiple of } n\}$$

is regular. Define the following DFA M (over $\Sigma = \{a\}$) as follows:



Formally the DFA M is defined as $M = (\Sigma, Q, q_0, \delta, F)$ where:

1. $\Sigma = \{a\}$
2. $Q = \{q_0, q_1, \ldots, q_{n-1}\}$
3. $\delta : Q \times \Sigma \to Q$ is defined by

$$\delta(q_i, a) = \begin{cases} q_{i+1} & \text{if } 0 \leq i < n-1 \\ q_0 & \text{if } i = n-1 \end{cases}$$

4. $F = \{q_0\}$

Clearly the strings accepted by M are ϵ, a^n, a^{2n}, $\ldots$, i.e. $L(M) = \{a^k \mid k \text{ is a multiple of } n\} = B_n$. Hence B_n must be regular.

SOLUTION 2: Let $n \geq 1$. We want to show

$$B_n = \{a^k \mid k \text{ is a multiple of } n\}$$

is regular. Note that

$$\begin{aligned} B_n &= \{a^k \,|\, k \text{ is a multiple of } n\} \\ &= \{a^{mn} \,|\, m \geq 0\} \\ &= \{(a^n)^m \,|\, m \geq 0\} \\ &= \{a^n\}^* \\ &= L(a^n)^* \\ &= L((a^n)^*) \end{aligned}$$

i.e. B_n is the language accepted by the regular expression $r = (a^n)^*$. We already know that the language generated by a regular expression is also accepted by a DFA. Hence B_n must be regular.

Note. Note that the second solution is a lot clearer since $L(r) = \{a^k \mid k \text{ is a multiple of } n\}$ is shown completely. However in the first solution the statement

$$L(M) = \{a^m \mid m \text{ is a multiple of } n\}$$

is not so immediate and properly speaking requires some proof. You can formally show that

$$L(M) = \{a^m \mid m \text{ is a multiple of } n\}$$

using mathematical induction. This is the reason why in CS and Math, we frequently have several different ways of looking at the same concept. (In the case of regular languages, a regular language is one that is accepted by or DFA, *or* is accepted by an NFA, *or* is generated by a regular expression.) Sometimes one way of looking at a problem will yield a more natural solution/proof or one that is shorter.

Q5. Sipser 1.37.

SOLUTION.

Q6. Sipser 1.38.

SOLUTION.

Q7. Sipser 1.40. Solution to 1.40(a) is in the textbook – study it carefully.

SOLUTION.

(a)

(b)

(c)

Q8. Sipser 1.41.

SOLUTION.

(a) SOLUTION PROVIDED.

$$\begin{aligned} \epsilon, a &\in L(a^*b^*) \\ ba, baa &\notin L(a^*b^*) \end{aligned}$$

(b) SOLUTION PROVIDED.

$$\begin{aligned} ab, abab &\in L(a(ba)^*b) \\ b, b &\notin L(a(ba)^*b) \end{aligned}$$

(c) SOLUTION PROVIDED.

$$\begin{aligned} \epsilon, a &\in L(a^* \cup b^*) \\ ab, ba &\notin L(a^* \cup b^*) \end{aligned}$$

(d)

(e)

(f)

(g)

(h)

Q9. Sipser 1.42.

Solution.