CS 181 - Homework 3

Kuruvilla Cherian, Jahan
UID: 104436427
Section 1B

October 20, 2017

Problem 1.16

The key insight to this problem is to use the methodology to convert an NFA to a DFA. This can be done by first finding all possible sets of states one can reach, then creating the transitions from each state to finally deleting any states that are unreachable. It is also important to note that the start state remains the same and acceptable states are now the set of all states that contain the original accept states. Using this algorithm we get the following DFA's for their respective NFA's:

a

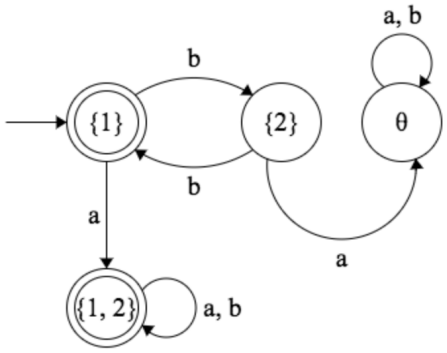

Figure 1: DFA for NFA (a) with $\Sigma = \{a, b\}$

b

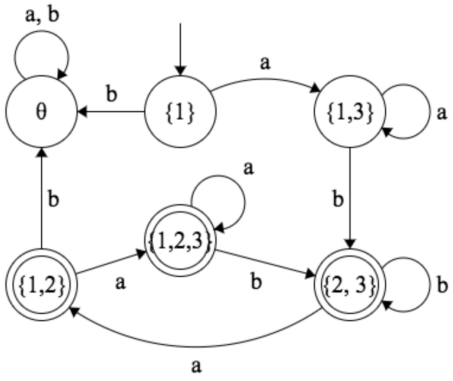

Figure 2: DFA for NFA (b) with $\Sigma = \{a, b\}$

Problem 1.32

If we can prove that an NFA exists for B^R then we can conclude that B is regular by reversal. To construct an NFA for B^R we must enumerate all combinations of 3 bit vectors. For these sets we can have the following:

$$A = \{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \}$$

$$B = \{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \}$$

$$C = \{ \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \}$$

$$D = \{ \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \}$$

The reason for this segregation of vectors into their respective sets is based on the result from addition of the vectors from **right to left** (hence the reversal) with their respective generation of carry bits. What this means is that for example, in A, an addition of a carry bit $= 0$ with the first two rows of any of the vectors results in a carry of 0. Using this administration, we can construct the following NFA:

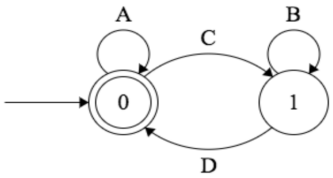

Figure 3: NFA for B^R

Problem 1.40

a

The key insight to showing *NOPREFIX(A)* is regular is to construct a DFA in a way such that we remove all transitions from the accept states. This is important because it means that after an acceptance we can no longer append any symbols to the word to make it such that $x \neq y$. If we assume that some DFA M exists for the language A as $(Q, \Sigma, \delta, q_0, F)$, then let the DFA recognizing *NOPREFIX(A)* be $(Q, \Sigma, \delta', q_0, F)$, wherein

$$\delta'(q, \sigma) \text{ for some q} \in Q \text{ and } \sigma \in \Sigma = \begin{cases} \delta(q, \sigma), \text{ if q} \notin F \\ \emptyset, \text{ if q} \in F \end{cases}$$

Thus since we have a DFA for *NOPREFIX(A)*, we have shown the language is regular.

b

To recognize *NOEXTEND(A)* with some NFA, to showcase its regularity, we have to understand that we are picky about the final states, wherein we want all states to be final such that there exists no path to any further accept states, because you have already found the proper prefix. Essentially we are saying that for some $wz = y$, where z $= \varepsilon$, which means you don't want any further path from the original final states. Thus prune the rest of the paths. This can be more formally stated as, if an NFA M exists for the language A as $(Q, \Sigma, \delta, q_0, F)$, then let the NFA recognizing *NOEXTEND(A)* be $(Q, \Sigma, \delta, q_0, F')$, wherein

$$F' = \{q | q \in F \text{and no path of length } \geq 1 \text{ exists from q to an accept state}\}$$

Problem 1.41

To create a perfect shuffle, we need to keep some memory of where we are coming from and where we want to go to. This means we want to create a tuple of three states per state, wherein we can use {1,2} to shuffle between a and b respectively. Thus if we have a DFA D_A and D_B for the languages A and B, as $(Q_A, \Sigma, \delta_A, s_A, F_A)$ and $(Q_B, \Sigma, \delta_B, s_B, F_B)$, then we can construct a DFA D to recognize the perfect shuffle between A and B as $(Q, \Sigma, \delta, s, F)$ where

$$Q = Q_A \times Q_B \times \{1, 2\}$$

$$\text{For } q_A \in Q_A,\ q_B \in Q_B,\ x \in \{1,2\} \text{ and } \sigma \in \Sigma,\ \delta\left(\left(q_A, q_B, x\right), \sigma\right) = \begin{cases} \left(\left(\delta_A\left(q_A, \sigma\right), q_B, 2\right)\right), & \text{if } x = 1 \\ \left(\left(q_A, \delta_A\left(q_B, \sigma\right), 1\right)\right), & \text{if } x = 2 \end{cases}$$

$$s = (s_A, s_B, 1)$$

$$F = \{(q_A, q_B, 1) \,|q_A \in F_A \text{ and } q_B \in F_B\}$$

Problem 1.62

A DFA M that recognizes the language D_k needs to contains *k+1* states, such that we begin with a reject state that will loop on itself until it arrives at least one a such that it can enter the accept states for any alphabet thereof until it reaches the last state. Formally we can say that $M = (Q, \Sigma, \delta, s, F)$, wherein

$$Q = q_1 q2 ... qk + 1$$

$$\Sigma = \{a, b\}$$

$$\delta\left(q_i, \sigma\right) \text{ for } q_i \in Q \text{ and } \sigma \in \Sigma = \begin{cases} q_i, & \text{if } i = 1 \text{ or } i = k+1 \text{ and } \sigma = b \\ q_{i+1}, & \text{if } i \neq 1 \text{ and } i \neq k+1 \\ q_2, & \text{if } i = k+1 \text{ and } \sigma = a \end{cases}$$

$$s = q_1$$

$$F = Q - q_1$$

This can be visually represented as the following DFA.

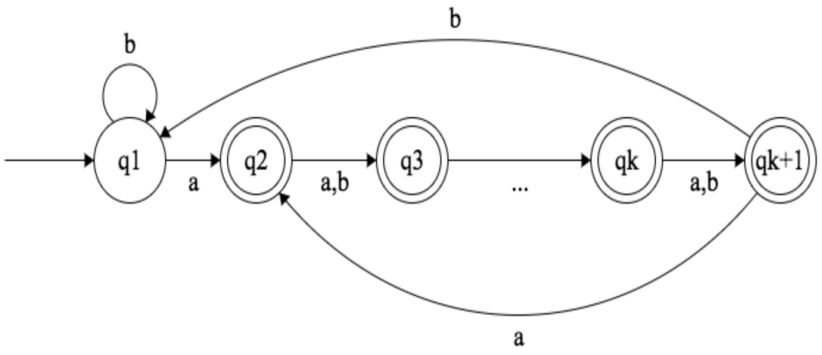

Figure 4: DFA for aforementioned language

Problem 1.66

a

To show that the class of regular languages are closed under homomorphism, we basically have to construct a finite automata where each transition on some $\sigma \in \Sigma$ is converted to a series of transitions made while reading $f(a) \in \Gamma^*$. Formally this can be said as the original FA M

$= (Q, \Sigma, \delta, q_0, F)$ and $M' = (Q', \Gamma, \delta', q_0, F)$, noting that our language for the homomorphism is $h = f(a) = h_1 h_2 ... hk$, where $k = |h|$, and $h_i \in \Gamma$. Here we have:

$$Q' = Q \cup \{q_i | q \in Q, 1 \leq i \leq k, \text{ over the symbols in } \Sigma\}$$

For every $q \in Q$,

$$\delta'(q, \gamma) = \begin{cases} \{r | \delta(q, \sigma) = r \text{ and } h = \varepsilon\}, \gamma = \varepsilon \\ \{q_1 | \gamma = h_1\}, \gamma \neq \varepsilon \end{cases}$$

$$\delta'(q_i, \gamma) = \begin{cases} \{q_{i+1} | \gamma = h_{i+1}\}, 1 \leq i < k \text{ and } \gamma \neq \varepsilon \\ \{r | \delta(q, \sigma) = r\}, i = k \text{ and } \gamma = \varepsilon \end{cases}$$

Problem 1.69

a

To prove that no DFA exists, we approach this problem from a contradictory standpoint. Let us assume that a DFA $M = (Q, \Sigma, \delta, q_0, F)$ exits, wherein M contains m states. Let us state that k is some fixed value such that $W = \{w | w \in \Sigma^k\}$. For each of the 2^k strings, let us say that M enters state q_w after entering from q_0 and reading word w. If $m < 2^k$, then there must exist two separate states a and b in W such that M enters the same state. This means that $q_a = q_b$. WW contains the string aa (enters accept state) but does not contain the string ab (enters reject state), but seeing as $q_a = q_b$, M should enter the same state (that is an accept state) on both aa and ab, which shows a contradiction, thereby proving that $m \geq 2^k$.

b

The complement of WW_k can be thought of as having two conditions, namely the length of the input is not $2k$ and the input contains two unequal symbols separated by k symbols, thus creating an NFA with a total of $4k + 4$ states.