COL352: Assignment 1

Aniket Gupta 2019CS10327
Shrey J. Patel 2019CS10400
Aayush Goyal 2019CS10452

August 14, 2023

1.

Given an alphabet $\Gamma = \{1, . . . , k\}$, construct an NFA that accepts strings that don't have all the characters from Γ. Can you give an NFA with k states?

Solution:
The idea behind the NFA that we have created is that there is a different state corresponding to every subset of the alphabet Γ. At any time, we are in any state $q_s \in Q$ (which corresponds to a subset $s \subset \Gamma$) iff the set of distinct characters read so far is exactly s.

A NFA for this language is $A = (Q, \Gamma, q_0, F, \delta)$ where,

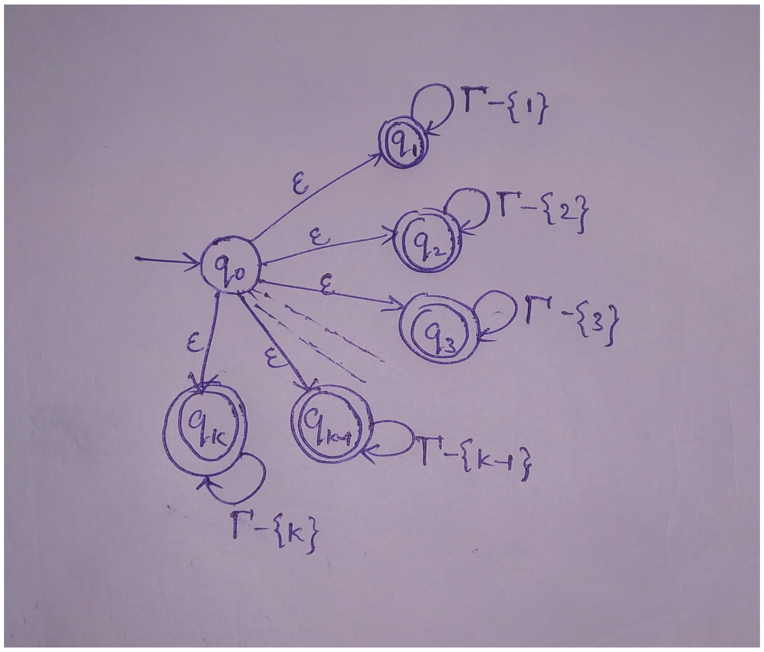
- Q = set of states in the NFA. Consider the NFA with $2^{|\Gamma|}$ states where each state corresponds to a different subset of Γ. For every $S \subset \Gamma$, there is a state $q_S \in Q$. Since there are $2^{|\Gamma|}$ subsets of Γ, so are the number of states in this NFA.
- Γ = The input alphabet $\{1, . . . , k\}$.
- q_0 = Initial state for this NFA is the state corresponding to the empty subset of Γ.
- δ = The set of transitions in the NFA. For any state $q_s \in Q$, which corresponds to subset $s \subset \Gamma$, and an alphabet $x \in \Gamma$, the transition can be defined as follows:

$$delta(q_s, x) = \begin{cases} q_s, & \text{if } x \in s, \\ q_{s \cup \{x\}}, & \text{otherwise} \end{cases}$$

  Note that $q_{s\cup\{x\}}$ is the state in Q corresponding to the subset $s \cup \{x\}$.
- F = The set of accepting states in this NFA contains all the states in Q, except the one corresponding to subset containing all the characters from Γ i.e., q_Γ. Thus, the set of accepting states is F = $2^Q - q_\Gamma$.

The intuition behind the above NFA model is that every state in the NFA represents the set of distinct characters that have already been read. At any state, we change the state only when a new character occurs. Also, the only non-accepting state in this model will be the state that represents the entire alphabet Γ because once we reach q_Γ, all the distinct characters in the alphabet has already been read and the input string cannot be accepted.

The NFA, with minimal number of states, that we could design for this language is as shown below.



The formal description for this NFA is $N = (Q, \Gamma, q_0, F, \delta)$ where,

- Q = There are k+1 states in this NFA, $\{q_0, q_1, q_2, ..., q_k\}$
- Γ = The input alphabet {1, . . . , k}.
- q_0 = Initial state for this NFA is q_0.
- δ = There is an ϵ transition from q_0 to every other state. For a state q_i (i>0), there is a transition to the same state q_i on reading any character from set $\Gamma - \{i\}$. There is no transition from state q_i (i>0) on reading a character i.
- F = The set of accepting states is Q-q_0. A state q_i (i>0) accepts a string only if character i has not occurred in the input string.

The intuition behind the above NFA model is that every string in the language defined in the problem will have atleast one missing character from the alphabet. Thus, we define k accepting states such that state q_i (i>0) accepts a string only if character i has not occurred in the input string. In this way all the strings that have atleast one character missing from the alphabet is accepted in some final state. Also, if a string contains all the characters from the alphabet, then none of the accepting states will accept that string because every state q_i (i>0) accepts a string only if character i has not occurred in the input string.

2.

An all-NFA M is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ that accepts $x \in \Sigma^\star$ if every possible state that M could be in after reading input x is a state from F. Note, in contrast, that an ordinary NFA accepts a string if some state among these possible states is an accept state. Prove that all-NFAs recognize the class of regular languages.

Solution:
To show that all-NFAs recognise the class of regular languages, we need to show the following two things:

1. For every regular language (L), there is an all-NFA (A) such that L(A) = L.
2. For every all-NFA (A), L(A) is a regular language.

For every regular language (L), there is an all-NFA (A) such that L(A) = L.
We know that for every regular language L, there is a DFA, D, such that L(D) = L. Note that in a DFA, there is exactly one run for any string and thus it can be in exactly one state after reading any string. The string is accepted only if this final state is an accepting state. Thus, any DFA can also be run as an all-NFA. Thus, for every regular language (L), there is an all-NFA (A) such that L(A) = L.

For every all-NFA (A), L(A) is a regular language
To show that for every all-NFA (A), L(A) is a regular language, we can equivalently show that for every all-NFA (A), there is an equivalent DFA (D) i.e., L(A) = L(D).
We prove this by constructing the equivalent DFA. The construction is similar to the construction of DFA equivalent to any NFA by using subset construction. Let all-NFA $A = (Q, \Sigma, \delta, q_0, F)$. Assume, for simplicity, that all the ϵ transitions are removed from A by taking proper ϵ closures. We now construct the DFA $D = (Q', \Sigma, \delta', q'_0, F')$ where:

- $Q' = 2^Q$
- $q'_0 = \{q_0\}$
- $\delta(S, a) = \cup_{p \in S} \delta(p, a)$. Note here $S \in Q'$ and $a \in \Sigma$.
- $F' = 2^F - \phi$ i.e., the final accept states in the DFA are only those states that are non-empty subset of the initial set of final states (F) in A.

Note that this construction is similar to DFA construction from an ordinary NFA with the only difference in the selection of final accepting states. In case of DFA construction from an ordinary NFA, the final states in the resulting DFA were those states in Q' that contains atleast one final state of the initial NFA. But, in case of DFA construction from an all-NFA, the final states are those states in Q' that are non-empty subsets of F (set of final states in the initial all-NFA). This is so because in a DFA constructed using subset construction from a NFA or all-NFA, if you are in some state $q \in Q'$ after reading input x, then q is the set of all possible states that you can be in after reading the input for the initial NFA or all-NFA. Thus, the DFA that we have constructed accepts a string x only if every possible state the initial all-NFA could be in after reading x is a state from F.

What we have shown above is an informal way to give the intution about our approach. Now we formally prove that if for a given all-NFA A, we construct a DFA D using the method discussed above then L(A) = L(D).
First we prove the following claim:

Claim: For the DFA D (constructed from all-NFA A using the method described above), after reading input string x, D will be in state s∈Q' such that s⊆Q is the set of all states A can be in after reading input x.

Proof of Claim:
We prove this by induction on the length of x.
Induction Predicate, P(n): For a string x of length n (≥0), D will be in state s∈Q' such that s⊆Q is the set of all states A can be in after reading x.
Base Case: For a string of length 0, this is trivial since $q'_0 = \{q_0\}$
Induction Step: We will the inductive definition $\hat{\delta}'(q'_0, y\text{a}) = \delta(\hat{\delta}'(q'_0, y), \text{a})$ where $x, y \in \Sigma^\star$, a∈ Σ and $y\text{a} = x$. Let us assume that induction predicate is true for some n(≥0), then we show that it is also true for n+1. Let x be string of length n+1 where $y\text{a} = x$ that is input to D. Then D will be in state $\hat{\delta}'(q'_0, x)$ $= \hat{\delta}'(q'_0, y\text{a})$ after reading x. Now by definition, $\hat{\delta}'(q'_0, x) = \delta'(\hat{\delta}'(q'_0, y), \text{a})$. By our assumption, induction predicate is true for string y (of length n). Thus, $\hat{\delta}'(q'_0, y)$ is the set of all states A can be in after reading y. By definition of δ', we get $\hat{\delta}'(q'_0, x) = \delta'(\hat{\delta}'(q'_0, y), \text{a}) = \cup_{p \in \hat{\delta}'(q'_0, y)} \delta(p, a)$, which is exactly the set of all states that A could be in after reading x.

Thus, by principal of induction, we proved that D will be in state s∈Q' such that s⊆Q is the set of all states A can be in after reading input x.

Now we prove that L(A) = L(D) using the above claim.

To Prove: For a all-NFA A, if we construct DFA D using the above method, then L(A) = L(D).
Proof:
To show that L(A) = L(D), we show that L(A)⊆L(D) and L(D)⊆L(A).

- L(A)⊆L(D)
  Let x be a string in L(A). Let $f \subseteq Q$ be the set of states that A could be in after reading x. Then, by definition of all-NFA, f is a non-empty subset of F.
  Now, assume that when we pass the same string x through D, then D is in some state $q' \in Q'$. By using the above claim, we know that q' is the set of all states A can be in after reading input x. But, since x∈L(A), q' is subset of F. Thus, q'∈ 2^F which implies that q'∈F'. Thus, x∈L(D).

- L(D)⊆L(A)
  Let x be a string in L(D). If we give x as input to D, it will be in some state q'F'. By definition of F', q' is a subset of F. Also, by above proved claim, q' is same as the set of states A will be in after reading x. Thus, every possible state that A could be in after reading input x is a state from F. Hence, L(D)⊆L(A).

Thus, we showed that L(A) = L(D).

Thus, we showed that for every all-NFA (A), there is an equivalent DFA D i.e., L(A) = L(D). Equivalently, we showed that for every all-NFA A, L(A) is a regular language.

Thus, we proved that all-NFAs recognise the class of regular languages.

3.

Show that regular languages are closed under the repeat operation, where repeat operation on a language L is given by:

$$\textbf{repeat}(\mathrm{L}) = \{l_1 l_1 l_2 l_2 \ldots. l_k l_k \mid l_1 l_2 l_3 \ldots. l_k \in L\}$$

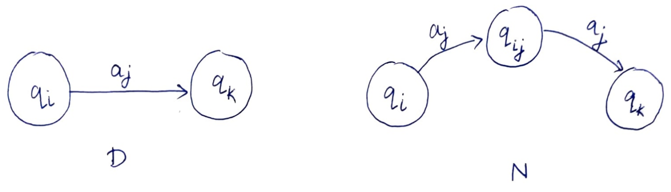
Solution:

To prove: L is regular $\implies$ repeat(L) is regular.

Proof by construction: Since L is regular, there is a DFA D = (Q,Σ,δ,q_0,F) which recognizes L. Assume Q = $\{q_1, q_2, ...q_n\}$ and $\Sigma = \{a_1, a_2, ...a_m\}$. To prove that repeat(L) is also regular, we need to construct an NFA N = (Q',Σ',δ',q_0',F') from D, which recognizes repeat(L).

Define N as:

1. Q' = Q ∪ $\{q_{ij} \mid \delta(q_i, a_j) = q_k$ where $1 \leq i, k \leq n, 1 \leq j \leq m$ for some k$\}$.
   In other words, Q' can be obtained by adding one state to Q for each transition in D (since D is a DFA, it has a transition for every state-action pair).
2. Σ' = Σ = $\{a_1, a_2, ...a_m\}$
3. δ' can be defined as:
   (a) $\delta'(q_i, a_j) = q_{ij}$, ∀ q_i ∈ Q, $a_j \in \Sigma$
   (b) $\delta'(q_{ij}, a_j) = \delta(q_i, a_j)$ ∀ q_{ij} ∈ Q' \Q
4. $q_0' = q_0$
5. F' = F



The above figure shows the extra states and transitions added to the DFA D to obtain NFA N. Note that **none of them represent the entire automaton**.

Claim: D recognizes L $\implies$ N recognizes repeat(L)

Proof of Claim:
Assume that D recognizes L. To prove that N recognizes, we need to prove both the sides of:

$$w \in \text{repeat(L)} \iff w \text{ is accepted by N}$$

($\Rightarrow$)
Suppose D accepts a string $l = l_1 l_2 l_3 .....l_k$, and let the corresponding sequence of steps be $s = s_0 s_1 s_2 ...s_k$ such that:

1. $s_0 = q_0$
2. $s_k \in$ F
3. $\delta(s_{i-1}, l_i) = s_i$, $\forall$ 1 $\leq$ i $\leq$ k

Now we need to find a sequence of states in Q' such that N accepts w = repeat(l). Let w be $w_1 w_2 ....w_{2k} = l_1 l_1 l_2 l_2 .....l_k l_k$. Define a sequence of states $t = t_0 t_1 t_2 .....t_{2k}$ such that:

- $w_i = l_{\lfloor (i+1)/2 \rfloor}$
- $t_{2i} = s_i$, $\forall$ 1 $\leq$ i $\leq$ k
- $t_{2i+1} = \delta'(t_{2i}, w_{2i+1}) = \delta'(t_{2i}, l_{i+1})$, $\forall$ 0 $\leq$ i $\leq$ k-1

From the above two points, $t_i \in$ Q' $\forall$ 1 $\leq$ i $\leq$ 2k. This sequence of states t accepts repeat(l) because:

1. $t_0 = s_0 = q_0 = q'_0$
2. $t_{2k} = s_k \in$ F (which is also equal to F')
3. (a) From point 2 in above definition of sequence t:

   $$t_{2i+1} = \delta'(t_{2i}, w_{2i+1}) = \delta'(t_{2i}, l_{i+1}) \text{ , } \forall \text{ 0} \leq \text{i} \leq \text{k-1}$$

   (b) From the definition of N, putting 3(a) in 3(b):

   $\delta'(\delta'(q_i, a_j), a_j) = \delta(q_i, a_j)$ $\forall$ $q_i \in$ Q, $a_j \in \Sigma$ $\implies$

   Since $t_{2i} = s_i \in$ Q and $l_{i+1} \in \Sigma$, putting $q_i = t_{2i} = s_i$ and $a_j = l_{i+1}$
   $\delta'(\delta'(t_{2i}, l_{i+1}), l_{i+1}) = \delta(s_i, l_{i+1})$ $\implies$

   From point(a) above, $t_{2i+1} = \delta'(t_{2i}, l_{i+1})$ and from point 3 of definition of sequence s,
   $\delta(s_i, l_{i+1}) = s_{i+1} = t_{2i+2}$

   So, $\delta'(t_{2i+1}, l_{i+1}) = t_{2i+2} = \delta'(t_{2i+1}, w_{2i+2})$, $\forall$ 0 $\leq$ i $\leq$ k-1

   Combining (a) and (b), $\delta'(t_i, w_{i+1}) = t_{i+1}$, $\forall$ 0 $\leq$ i $\leq$ 2k-1

So, from 1, the start of sequence t is a start state of NFA. From 2, the end of sequence t is an accept state of NFA, and lastly, from 3, every next state can be obtained by applying a δ' transition on the previous state of the sequence and the input action. So, the NFA N constructed from D accepts the string repeat(l), provided that D accepts the string l.

Hence Proved.

($\Leftarrow$)
Suppose the NFA N accepts a string w = $w_1w_2.....w_m$. Then we need to prove that w $\in$ repeat(L). Let's assume the the sequence of states while passing w through N be $p = p_0p_1....p_m$.

Claim 1: H(i): $p_i \in$ Q if i is even and $p_i \in$ Q' $\setminus$ Q if i is odd.

Proof of Claim 1: Proof by induction on i:

- **Base Case:** $p_0 = q_0$, since it is the start state. So, $p_0 \in$ Q. H(0) holds.
- **Induction Step:** Assume that H(i-1) holds. Then:
  - **Case 1:** i is odd. Then from H(i-1), $p_{i-1} \in$ Q since (i-1) is even. So, from point 3(a) of the definition of N, $\delta'(p_{i-1}, w_i) = p_i \in$ Q' $\setminus$ Q, for any $w_i \in \Sigma$.
  - **Case 2:** i is even. Then from H(i-1), $p_{i-1} \in$ Q' $\setminus$ Q since (i-1) is odd. So, from point 3(b) of the definition of N, $\delta'(p_{i-1}, w_i) = p_i \in$ Q, for any $w_i \in \Sigma$.

  From above two cases, H(i) holds.

Hence Proved.

Note that p_m is the accept state. So, $p_m \in$ F' = F. But, F $\subseteq$ Q. So, $p_m \in$ Q. Hence, **m is even**. Say, m = 2k.

Claim 2: $w_{2i-1} = w_{2i}$, $\forall$ 1 $\leq$ i $\leq$ k.

Proof of Claim 2: For any i in the given range, by definition of sequence p:

1. $\delta'(p_{2i-2}, w_{2i-1}) = p_{2i-1}$
2. $\delta'(p_{2i-1}, w_{2i}) = p_{2i}$

But, from claim 1, p_{2i-2}, $p_{2i} \in$ Q and $p_{2i-1} \in$ Q' $\setminus$ Q.

In 3(a) of the definition of N, putting $q_i = p_{2i-2}$ and $a_j = w_{2i-1}$. Then comparing this with transition 1 above, $q_{ij} = p_{2i-1}$.

In 3(b) of the definition of N, putting $q_{ij} = p_{2i-1}$. Then, the only transition possible is when the input letter is a_j. Here, $a_j = w_{2i}$. So, $\underline{a_j = w_{2i-1} = w_{2i}}$ Comparing this with transition 2 above,
$\underline{p_{2i} = \delta(q_i, a_j) = \delta(p_{2i-2}, w_{2i-1})}$.

From the first result above, $w_{2i-1} = w_{2i}$ for all i. So, $w = w_1w_1w_3w_3.....w_kw_k$.

Additonally, from both claims, we have that $p_{2i} \in$ Q and $p_{2i} = \delta(q_i, a_j) = \delta(p_{2i-2}, w_{2i-1})$ for all i. In other words, the string w' = $w_1w_3w_5....w_k$ is accepted by the DFA D, using the sequence of states p' = $p_0p_2...p_{2k}$.

So, from above two arguments, w' $\in$ L for any string w. But, w = repeat(w'). So, w $\in$ repeat(L).

Hence proved both the sides.

So, the regular languages are closed under the **repeat** operation.

4.

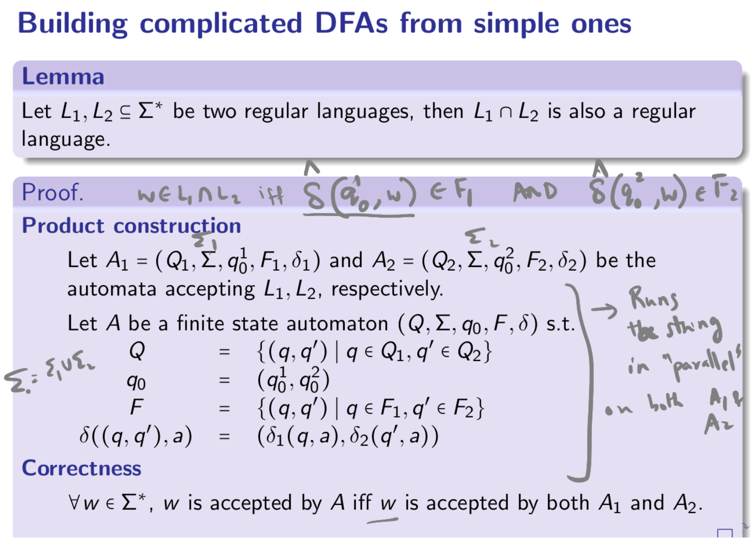
Design an algorithm that takes as input the descriptions of two DFAs, D_1 and D_2 , and determines whether they recognize the same language.

Solution:
Let the language determined by D_1 be L_1 and the language determined by D_2 be L_2. Now we need to determine whether both the languages L_1 and L_2 are the same or not. This can be done by checking if L_1 is contained in L_2 and L_2 is contained in L_1. If the algorithm determines that L_1 is contained in L_2 and L_2 is contained in L_1 that means D_1 and D_2 determine the same languages. Denote the compliment of L by $\overline{L}$. To check whether L_1 is contained in L_2, we will alternatively check whether L_1 intersection with $\overline{L_2}$ is empty. If the intersection is empty that means there is no such elemnent in L_1 which belongs to $\overline{L_2}$, which is same as all elements of L_1 are present in L_2. And then simillarly we have to check whether $\overline{L_1} \cap L_2 = \phi$ or not. If both of them turn out to be ϕ then L_1 and L_2 are the same. Now below is the algorithmic way to check if $L_1 \cap \overline{L_2}$ is ϕ or not.

Algorithm:

1. **Construct DFA $\overline{D_2}$ that accepts $\overline{L_2}$:**
DFA D_1 determines the language L_1. Now we need to construct a DFA which accepts the language $\overline{L_2}$. In the class we have discussed how to make a DFA which accepts the complement of a language. We will turn all the final accepting states into normals states, and all the nornal states into accepting states, call this DFA as $\overline{D_2}$. Any string that is accepted in D_2 ends at some of it's accepting state. But if those states are not accepting states, then the strings are still going to end up on the same state but they will not be accepted. Similarly strings that end up on normal states are now accepted as, those states are turned into accepting states. Hence $\overline{D_2}$ is the DFA which accepts the language $\overline{L_2}$.

2. **Construct DFA which accepts** $L_1 \cap \overline{L_2}$:
We have discussed in class the method to make a DFA which acccepts a language which is formed from intersection of 2 DFAs. The method is:



Building complicated DFAs from simple ones

Lemma
Let $L_1, L_2 \subseteq \Sigma^*$ be two regular languages, then $L_1 \cap L_2$ is also a regular language.

Proof. $w \in L_1 \cap L_2$ iff $\hat{\delta}(q_0^1, w) \in F_1$ AND $\hat{\delta}(q_0^2, w) \in F_2$

Product construction

Let $A_1 = (Q_1, \overset{\Sigma_1}{\Sigma}, q_0^1, F_1, \delta_1)$ and $A_2 = (Q_2, \overset{\Sigma_2}{\Sigma}, q_0^2, F_2, \delta_2)$ be the automata accepting L_1, L_2, respectively.

Let A be a finite state automaton $(Q, \Sigma, q_0, F, \delta)$ s.t.

$\Sigma_0 = \Sigma_1 \cup \Sigma_2$

$$\left.\begin{aligned} Q &= \{(q, q') \mid q \in Q_1, q' \in Q_2\} \\ q_0 &= (q_0^1, q_0^2) \\ F &= \{(q, q') \mid q \in F_1, q' \in F_2\} \\ \delta((q, q'), a) &= (\delta_1(q, a), \delta_2(q', a)) \end{aligned}\right\} \rightarrow$$

Runs the string in "parallel" on both A_1 & A_2

Correctness

$\forall w \in \Sigma^*$, w is accepted by A iff $\underline{w}$ is accepted by both A_1 and A_2.

Using the method above we can construct a DFA which accepts the intersection of languages L_1 and $\overline{L_2}$, since we know the DFAs D_1 and $\overline{D_2}$.

3. **Check if the $D_1 \cap \overline{D_2}$ accepts any string or not:**
   Now we need to check if their is any path from the start state to atleast one of the accepting states. If there is no such path that means the DFA doesn't accept anything, or it accepts the language ϕ. To check if there is any such path or not, we can convert the DFA into a graph, where all the states are nodes of graph and edges are the directed edges of the graph. Once we have converted the DFA into the form of a graph, we can use some algorithm like BFS or DFS to determine to all the states that are reachable from the start state. Once we have the list of all states that can be reached from the start state, we can check if there is any accepting state in the list. if any accepting state is reachable that means there are some strings that are accepted in $L_1 \cap \overline{L_2}$. But if no accepting state is reachable that means $L_1 \cap \overline{L_2} = \phi$.

Thus using the above 3 steps we can determine wheather L1 is contained in L2 or not. Similarly we can check if L2 is contained in L1 or not. If both of them are contained in one another then we can conclude that they are the same languages. This way we can determine if given 2 DFAs, they determine the same language or not.

5.

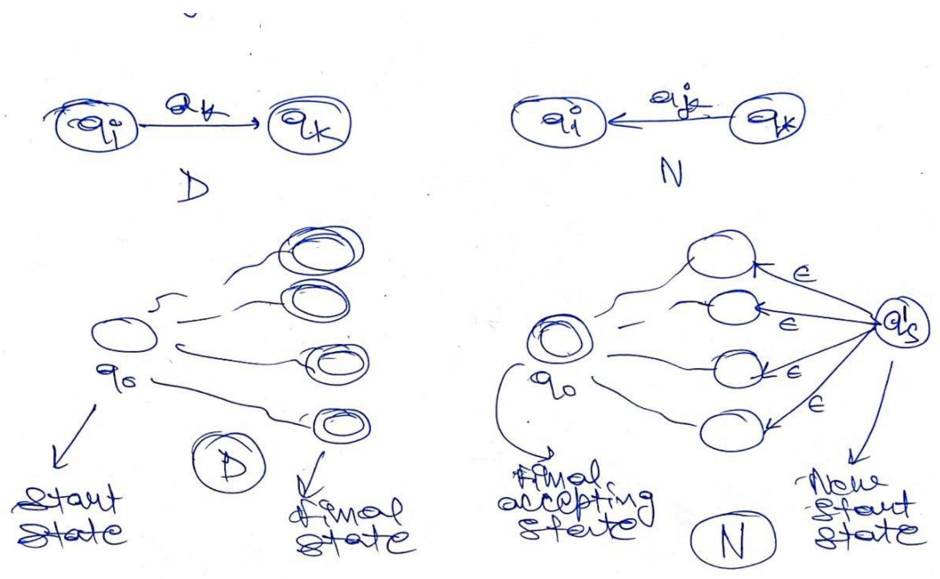
For any string w = $w_1w_2...w_n$ the reverse of w written w^R is the string $w_n...w_2w_1$. For any language A, let $A_R = w_R|w \in A$. Show that if A is regular, then so is A_R . In other words, regular languages are closed under the reverse operation.

Solution:
To Prove: A is regular $\implies$ A_R is regular

Proof by construction: Since A is regular, there is a DFA D = (Q,Σ,δ,q_0,F) which recognizes A.
Assume Q = $\{q_1, q_2, ...q_n\}$ and $\Sigma = \{a_1, a_2, ...a_m\}$. To prove that A_R is also regular, we need to construct an NFA N = (Q',Σ',δ',q_0',F') from D, which recognizes A_R.

Define N as:

1. Q' = Q ∪ $\{q_s^{'}\}$
2. Σ' = Σ ∪ $\{\epsilon\}$
3. δ' can be defined as:
   (a) $\delta'(q_s^{'}, \epsilon)$ = F, that is we have have added an epsilon tranisition from the new state that we defined $q_s^{'}$ to every final state in F of the old DFA D.
   (b) $\delta'(q_k, a_j) = q_i$, such that $\delta(q_i, a_j) = q_k$ $\forall$ q_i, $q_k \in$ Q, $a_j \in \Sigma$
   Effectively we have reversed the direction of every arrow in the old DFA D. The alphabet a_j which takes us from q_i to q_k, now takes us from q_k to q_i.
4. $q_0' = q_s^{'}$, this will new start state, and it has epsilon tranisitions to all final states of D.
5. F' = q_0, start state of D is now the final state



Claim: D recognizes A $\implies$ N recognizes A_R
Suppose D accepts a string $A = a_1a_2a_3.....a_k$, and let the corresponding unique sequence (it will be unique since it is a DFA) of steps be $s = s_0s_1s_2...s_k$ such that:

1. $s_0 = q_0$
2. $s_k \in$ F
3. $\delta(s_{i-1}, a_i) = s_i$, $\forall$ 1 $\leq$ i $\leq$ k

Now when we start reading the reverse string in the NFA, there are only epsilon transitions from the start state $q_s^{'}$ to all final states of D. Using that epsilon transition, we will reach the state s_k. Now once we reach the state s_k, we will encounter the word a_k. In the DFA there is a transition from s_{k-1} to s_k for the letter a_k. And thus the way we defined the state actions in NFA, the transition from s_k on encountering a_k will be to s_{k-1}. Thus we will reach the state s_{k-1} after reading the word s_k. And thus similarly continuing the pattern ahead will lead us to s_0 after reading the word a_1. Now we have read the whole string and we are at s_0. Also $s_0 = q_0$. And q_0 is the final accepting state of the NFA. Hence we can say the string A_R will be accepted in the NFA. So, the NFA N constructed from D accepts the string A_R if D accepts the string A.

Now we also need show that it doesn't accept any string whose reverse is not accepted by the DFA D. Suppose there is a string B which is accepted by N, we will show that B_R is always a string belonging to the regular language accepted by D.
Take B = $b_1b_2...b_k$. Now if this is accepted by the NFA, there must be some sequence of steps. The start state $q_s^{'}$ has only epsilon transitions from it, and after that all the state action pairs are deterministic. So let the sequence of steps be $s = s_0s_1s_2...s_ks_{k+1}$, here the following will be true:

1. $s_0 = q_s^{'}$
2. $s_1 \in$ F
3. $s_{k+1} = q_0$

Now when we start reading the string B_R in DFA D, we have letter b_k, and their is a transition from s_k to s_{k+1} in N, so there will be a transition from s_{k+1} to s_k in D. Hence on reading the letter b_k we reach the state s_k in D. similarly we will reach the state s_1 once we read the letter b_1. Now as stated above $s_1 \in F$ and hence we can see that the string B_R is accepted by D. So if any string is accepted by N that means it is only formed by taking reverse of a string that belongs to language of D.

Hence N is the correct NFA and regular languages are closed under the reverse operation.

6.

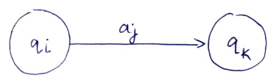
Let Σ and Γ be two finite alphabets. A function $f : \Sigma^* \longrightarrow \Gamma^*$ is called a homomorphism if $\forall\ x, y \in \Sigma^*$, $f(x.y) = f(x).f(y)$. Observe that if f is a string homomorphism, then $f(\epsilon) = \epsilon$, and the values of $f(a)\ \forall\ a \in \Sigma$ completely determine f. Prove that the class of regular languages is closed under homomorphisms. That is, prove that if $L \subseteq \Sigma^*$, is a regular language, then $f(L) = \{f(x) \in \Gamma^* \mid x \in L\}$ is regular. Try to informally describe how you will start with a DFA for L and get an NFA for $f(L)$.

Solution:
To prove: L is regular $\implies$ f(L) is regular.

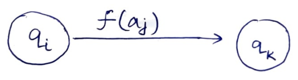
Proof by construction: L is regular, so there is a DFA D = (Q,Σ,δ,q_0,F) which recognizes L.
Assume Q = $\{q_1, q_2, ...q_n\}$ and $\Sigma = \{a_1, a_2, ...a_m\}$. To prove that f(L) is also regular, we need to construct an NFA N = (Q',Σ',δ',q_0',F') from D, which recognizes f(L).

Define N as:

1. Q' = Q
2. $\Sigma' = \{f(a) \mid a \in \Sigma\}$
   Note that $\Sigma' \subseteq \Gamma$ since we are concerned with only those letters in Γ which have a pre-image in Σ
3. $\delta'(q_i, f(a_j)) = \delta(q_i, a_j)$, $\forall\ q_i \in$ Q, $a_j \in \Sigma$, $f(a_j) \in \Sigma$'
4. $q_0' = q_0$
5. F' = F



D



N

The above figure shows the modifications to be done to the DFA D to obtain NFA N. Note that **none of them represent the entire automaton**.

Claim: D recognizes L $\implies$ N recognizes f(L)
Assume that D recognizes L. To prove that N recognizes, we need to prove both the sides of:

$$w \in \text{f(L)} \iff w \text{ is accepted by N}$$

Claim 1: $f(w) = f(w_1).f(w_2).f(w_3)....f(w_n)$ if $w = w_1.w_2.w_3....w_n$,
where $w_i \in \Sigma\ \forall\ 1 \leq$ i $\leq$ n

Proof of Claim 1:
Proof by induction on the length of string w using the following induction hypothesis:
H(i) : $f(w) = f(w_1).f(w_2).f(w_3)....f(w_i)$ if $w = w_1.w_2.w_3....w_i$

- **Basis:** |w| = 0, i.e. w = ϵ, then $f(\epsilon) = \epsilon$, as given in the question. Thus, H(0) holds.
- **Induction Step:** Assume that H(i-1), i.e. the hypothesis holds for all strings of length i-1, say w = $w_1.w_2.w_3....w_{i-1}$. So, $f(w) = f(w_1).f(w_2).f(w_3)....f(w_{i-1})$. Now, for some $w_i \in \Sigma$:

  $f(w.w_i) = f(w).f(w_i)$ from the definition of f since $w, w_i \in \Sigma^* \implies$
  $f(w') = f(w_1).f(w_2).f(w_3)....f(w_{i-1}).f(w_i)$ where w' = $w_1.w_2.w_3....w_{i-1}w_i$.

  Thus, H(i-1) $\implies$ H(i)

So, H(n) holds for all n $\geq$ 1.

Now, we prove the two sides of the original claim:

($\Rightarrow$) Suppose D accepts a string $w = w_1w_2w_3.....w_k$, and let the corresponding sequence of steps be $p = p_0p_1p_2...p_k$ such that:

1. $p_0 = q_0$
2. $p_k \in$ F
3. $\delta(p_{i-1}, w_i) = p_i$, $\forall$ 1 $\leq$ i $\leq$ k

Now, we need to find a sequence of states in Q such that N accepts $f(w)$, $w = w_1w_2w_3.....w_k$ From the above sub-claim, $f(w) = f(w_1)f(w_2)f(w_3)....f(w_k)$. We claim that the same sequence of states $p = p_0p_1p_2...p_k$ in N also accepts the string f(w). The proof is as follows:

1. Since Q = Q', $p_i \in$ Q' $\forall$ 0 $\leq$ i $\leq$ k
2. $p_0 = q_0 = q_0'$
3. $p_k \in$ F' since F = F'
4. From the definition of δ':

   $\delta'(q_i, f(a_j)) = \delta(q_i, a_j) \implies$

   Substituting $q_i = p_{i-1}$ and $a_j = w_i$:
   $\delta'(p_{i-1}, f(w_i)) = \delta(p_{i-1}, w_i) \implies$
   $\delta'(p_{i-1}, f(w_i)) = p_i$

Note that $p_i \in$ Q $\forall$ i. And, from 1, the start of sequence p is a start state of the NFA. From 2, the end of sequence p is an accept state of NFA, and lastly, from 3, every next state can be obtained by applying a δ' transition on the previous state of the sequence and the input action (i.e. $f(w_i)$). So, the NFA N constructed from D accepts the string $f(w)$, provided that D accepts the string w.

Hence Proved.

($\Leftarrow$) Now, we prove the converse. For simplicity, take $\Sigma' = \{b_1, b_2, ...b_l\}$.

Observation:
$\forall$ $b \in \Sigma'$, $\exists$ a $\in \Sigma$ such that $f(a)$ = b. Proof follows from the definition of Σ'.

Now, suppose the NFA N accepts a string w = $w_1 w_2 ....w_k$. Then we need to prove that w $\in f(L)$, where w_i $\in \Sigma'$ $\forall$ i. Let the corresponding sequence of steps through N be $p = p_0 p_1 p_2 ....p_k$. Note that $p_i \in$ Q = Q' $\forall$ i. Also,

1. $p_0 = q_0$
2. $p_k \in$ F = F'
3. $\delta'(p_i, w_{i+1}) = p_{i+1}$ $\forall$ 0 $\leq$ i $\leq$ k-1

From observation proved above, since $w_i \in \Sigma'$, $\exists$ some $u_i \in \Sigma$ $\forall$ i, such that $w_i = f(u_i)$. Then using point 3 from the definition of NFA N:

$\delta'(q_i, f(a_j)) = \delta(q_i, a_j)$, $\forall$ $q_i \in$ Q, $a_j \in \Sigma \implies$

Putting $q_i = p_i$ and $a_j = u_{i+1}$
$\delta'(p_i, f(u_{i+1})) = \delta(p_i, u_{i+1}) \implies$
$\delta(p_i, u_{i+1}) = \delta'(p_i, w_{i+1}) = p_{i+1}$ (from point 3 above and $f(u_{i+1}) = w_{i+1}$)

Now, consider the string $u = u_1 u_2 ....u_k$ and the sequence of states $p_0 p_1 ....p_k$ in D. $p_0 = q_0$ is the start state of D, $p_k \in$ F is an accept state of D. And finally, $\forall$ 0 $\leq$ i $\leq$ k-1, $\delta(p_i, u_{i+1}) = p_{i+1}$. So, DFA D accepts the string u. So, $u \in$ L

Now, $w = w_1 w_2 w_3 ....w_k = f(u_1) f(u_2) f(u_3) ....f(u_k)$. So, from claim 1, $w = f(u_1 u_2 u_3 ....u_k) = f(u)$. So, w $\in f(L)$, since $u \in$ L.

Hence proved both the sides.

So, the regular languages are closed under f.