COL352: Assignment 1

Sachin 2019CS10722

January, 2022

1 Question 1

Given an alphabet $\Gamma = \{l_1, ..., l_k\}$, construct an NFA that accepts strings that don't have all the characters from Γ. Can you give an NFA with k states?

1. **NFA Construction**
   Consider the NFA N $= (Q, \Sigma, \delta, q_0, F)$ where,
   $Q = 2^{\Gamma} = \{\phi, \{l_1\}, \{l_2\}, ...\Gamma\}$
   $\Sigma = \Gamma$
   $q_0 = \phi$
   $F = Q \setminus \Gamma$
   $\delta \subseteq Q \times \Sigma \times Q$ defined as follows:

$$\delta(q, a) = \begin{cases} q & \text{if } a \in q \\ q \bigcup \{a\} & \text{otherwise} \end{cases} \tag{1}$$

   Claim: N accepts only those strings that don't have all the characters from Γ.

   Proof:

   Let S $= x_1x_2.....x_k$ be any arbitrary string, let's consider the run of s on N.
   $\phi \xrightarrow{x_1} \{x_1\} \xrightarrow{x_1} \{x_1, x_2\}..... \xrightarrow{x_k} \bigcup\{x_i\}$
   NOTE: {} is a set in above expression, and $\bigcup\{x_i\}$ will contain single copy of each alphabet.

   (a) Case 1: $\bigcup\{x_i\} = \Gamma$
   This means that string s contains all the alphabets from Γ.
   Also, final state of N $= \bigcup\{x_i\} = \Gamma \notin$ F.
   Hence, N does not accepts the string s.

(b) Case 2: $\bigcup\{x_i\} \subset \Gamma$
This means that string s does not contains all the alphabets from Γ.
Also, final state of N (let p_k) = $\bigcup\{x_i\} \neq \Gamma => p_k \in$ F.
Hence, N accepts the string s.

Hence proved that N accepts only those strings that don't have all the characters from Γ.

2. **Can you give an NFA with k states?**

Since the k is arbitrary and its range is not provided in question, I'm taking it to be $k \geq 0$.
Proving that NFA with k states is not possible by contradiction.
Suppose for every such Language there exists an NFA with k states, then it would also be for k = 2.
So, let's try to construct one such NFA = $(Q, \Sigma, \delta, q_0, F)$ where,
$Q = \{s, q\}$
$\Sigma = \{l_1, l_2\}$
$q_0 = s$
Now, $s \in F$ (since, ϵ has no characters from Σ).
Also, strings $l_1, l_2 \in$ L, $=> \delta(s, l_1) \bigcup F \neq \phi$ and $\delta(s, l_2) \bigcup F \neq \phi$.
And possible values of $\delta(s, l_1)$ and $\delta(s, l_2)$ are $\phi, \{s\}, \{q\}, \{s, q\}$
Clearly, they can't be ϕ, also, $s \notin \delta(s, l_1) \bigcup \delta(s, l_2)$, as if that were the case then all the strings will get accepted.
So, either $\delta(s, l_1) = \{q\}$ or $\delta(s, l_2) = \{q\}$ making q an accepting state, that would make string $l_1 l_2$ to get accepted but $l_1 l_2 \notin L$
That is a contradiction.

k+1 state NFA:
In fact, at best we can create NFA with k+1 states, NFA N $(Q, \Gamma, \delta, Q_0, F)$ of above language, as follows:-
$Q = \{s, q_{l_1}, q_{l_2}, ...q_{l_k}\}$
$Q_0 = s$
$F = Q$ (all states are accepting states)
$\delta \subseteq Q \times \Sigma \times Q$ defined as follows:

$$\delta(q, a) = \begin{cases} \{q_{l_i}\} & \text{if } q = q_{l_i} \ \& \ a \neq l_i \\ \{q_{l_1}, q_{l_2}....q_{l_k}\} & \text{if } q = s \ \& \ a = \epsilon \\ \phi & \text{O.W.} \end{cases} \tag{2}$$

Proof of correcness:
From construction it is clear that NFA N is union of k NFA's (each state

q_{l_i} is considering one individual NFA N_i).
Claim: Each N_i accepts only those strings that don't have the characters l_i.
It is easy to prove since there is only state in NFA and that too is accepting, so evey string will be accepting unless it has any character c such that $\delta(q_{l_i}, c) = \phi$ and for every i it is only one character l_i
Hence proved that $\forall 1 \leq i \leq k : L(N_i) = (\Gamma \setminus l_i)^*$
$=> L(N) = \bigcup L(N_i) = \bigcup (\Gamma \setminus l_i)^*$ = required language.

If multiple start states are allowed then one can create an k-state NFA by removing start state s from above NFA and making all remainining k states start state.

2 Question 2

An all-NFA M is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ that accepts $x \in \Sigma^*$ 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.

To prove that all-NFA's recognise the class of regular languages we need to show two things, firstly that the language accepted by all-NFA's is regular, and secondly given any regular language there exists an all-NFA which accepts it. Following are the proofs of these parts,

To Prove: Language accepted by all-NFA is regular.
Proof: Now by the definition, all-NFA M is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ that accepts $x \in \Sigma^*$ if every possible state that M could be in after reading input x is a state from F . This would mean the all-NFA's are NFA because NFA a ccepts the string even if some of the states reached after reading an input x is in accept state F. NOw we know that the language accepted by NFA is regular. Therefore the language accepted by all-NFA is also regular. Hence proved.

To Prove: For every regular language there exists an all-NFA that accepts it.
Proof: We know that for every regular language there exists a DFA which accepts it. Now the definition of a DFA M is that it is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ that accepts $x \in \Sigma^*$ if the state that M could be in after reading input x is a state from F. Now we also know that the set of states DFA M would be in after reading the input x is a singleton set (Deterministic nature) and the state belongs to F if x is accepted by DFA. So every DFA is an all-NFA. Therefore for every regular language, there exists an all-NFA that accepts it. Hence proved.

Now above two facts would imply that the all-NFA's recognize the class of regular languages.

3 Question 3

4 Question 4

Design an algorithm that takes as input the descriptions of two DFAs, D1 and D2, and determines whether they recognize the same language

Lets take two DFAs D_1=(Q_1,Σ,q_0,δ_1,F_1) and D_2=(Q_2,Σ,r_0,δ_2,F_2).

Idea:

Let L(d) is language recognized by dfa d. If two DFAs will recognise same DFAs implies L(D_1) - L(D_2) and L(D_2) - L(D_1) should be a null set else language recognized by two DFAs are different.

Algorithm:

For computing L(D_2)-L(D_1):

Step1(complementation):Generate another DFA D_3 which is complement of D2 i.e by reversing final states as non final states and vice-versa.Then D_3=(Q_2,Σ,r_0,δ_2, Q_2-F_2).

Step2(intersection):Generate another DFA D_4 by intersecting D_1 and D_3 i.e by first complementing both D_1 and D_3 then take union of this both DFAs and at last take complement of whole.

Step3(BFS):Perform BFS on start state of D_4 if it find any final state then return false else

Step4:Repeat steps 1,2,3 for computing L(D_1)-L(D_2) if bfs in this also finds nothing then return true.

Proof of correctness:

Claim1: Let A,B be two DFAs then if both L(A)-L(B) and L(B)-L(A) are null then both DFAs can recognise same grammer

Proof by contradiction: Lets atleast L(A)-L(B) or L(B)-L(A) is not null then there there exist a string in which will be in L(A) or L(B) implies which will only be detetected by their respective DFAs.

Time complexity: Worst case Time complexity of this algorithm will be O($2^{|Q_1|+|Q_2|}$+no of transistions in D_4)

5 Question 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.

As every regular language has a DFA which accepts it, let D be $(Q, \Sigma, \delta, q_0, F)$ that accepts the language A. Now we will construct an NFA N from this DFA. The steps of the construction are given below. We will then show that the language accepted by NFA is indeed A^R.

Construction: To construct this NFA N we will have to reverse all the edges of the DFA D. Also make the start state of the D as the accepting state of the N. Add ϵ transitions to the accepting states of D from a new state s in NFA. Make s the start state of the NFA.
So N is $(Q_1, \Sigma, \delta^{'}, q_0^{'}, F^{'})$ such that

$$
\begin{aligned}
&Q_1 = Q \cup \{s\} \\
&F^{'} = q_0 \\
&\delta^{'}(q_1, a) = \begin{cases} q_2 & \text{if } q_1 \in Q \text{ and } \delta(q_2, a) = q_1 \\ F & \text{if } q_1 = s \text{ and } a = \epsilon \\ \phi & \text{O.W.} \end{cases} \\
&q_0^{'} = s
\end{aligned} \tag{3}
$$

To Prove: The language accepted by N is A^R.
Proof: Now take any string $w_1w_2....w_n$ from the language A. The path(path is state then alphabet taken then next state reached) taken by this string to accept state in D would be $q_0, w_1, q_1, w_2, q_2.....q_n, w_n, f$ where $q_i \in Q$ and $f \in F$. Now take the reverse of the string $w_nw_{n-1}....w_1$. There exist a path from start to accept state in N as which is as follows $s, \epsilon, f, w_n, q_n, w_{n-1}.....q_1, w_1, q_0$. We know that q_0 is an accept state of N. Thus we have got an NFA that has an accepting path for any string w in A^R.
Now we have proved that for every string in A we have a accepting path in N. Now we can also similarly prove the reverse direction too i.e. for every string w in A^R we have accepting path in D for reverse of w. The proof of this goes as follows:
Take any string $w_1w_2....w_n$ from the language A^R. The pathtaken by this string to accept state in N would be $s, \epsilon, f, w_1, q_n, w_2.....q_1, w_n, q_)$ where $q_i \in Q$ and $f \in F$. Now take the reverse of the string $w_nw_{n-1}....w_1$. There exist a path from start to accept state in D as which is as follows $q_0, w_n, q_1, w_{n-1}, q_2.....q_n, w_1, f$. We know that f is an accept state of D.Thus we have got an DFA that has an accepting path for any string w in A.
Hence we have proved that regular languages are closed under the reverse operation.

6 Question 6

Let Σ and Γ be two finite alphabets. A function $f : \Sigma^* \rightarrow \Gamma^*$is called a homomorphism if for all 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 $f(a)$ for all a $\in \Sigma$ completely determines 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 $f(L) = \{f(x) \in \Sigma^* | x \in L\}$ is regular. Try to informally describe how you will start with a DFA for L and get an NFA for f(L).

1. **$f(\epsilon) = \epsilon$**

   $\forall x, y \in \Sigma^* f(x.y) = f(x).f(y)$ (by defination of homomorphism)
   take y = ϵ
   => $\forall x \in \Sigma^* f(x.\epsilon) = f(x).f(\epsilon)$
   => $\forall x \in \Sigma^* f(x) = f(x).f(\epsilon)$
   but, only $f(\epsilon) \in \Gamma^*$ that upon concatenation with any string generate same string is ϵ only
   => $f(\epsilon) = \epsilon$

   This, is a property of homomorphism that it maps identity elements (of the operation upon which homomorphism property is satisfyied). Here identity element with respect to an operation (say oper) is any element of the set that maps every element to itself upon oper. In our case, since $a.\epsilon = a \forall a \in \Sigma^*$ as well as $a.\epsilon = a \forall a \in \Gamma^*$, Hence ϵ is the identity element of both Σ and Γ

2. **values of $f(a)$ for all a $\in \Sigma$ completely determines f**

   Given: $f(a) \; \forall \; a \in \Sigma$
   To find: $f(x) \; \forall \; x \in \Sigma^*$

   Let x $\in \Sigma^*$
   => $x = a_1 a_2 ..... a_k$ where $a_i \in \Sigma$ and $k \geq 0$
   => $f(x) = f(a_1 a_2 ..... a_k)$
   => $f(x) = f(a_1) f(a_2) .... f(a_k)$
   since, each $a_i \in \Sigma$ hence, they are known.
   => $f(a_1) f(a_2) .... f(a_k)$ is known.
   => $f(x)$ is known.
   Hence, we can determine $f(x) \forall x \in \Sigma^*$

3. **Prove that the class of regular languages is closed under homomorphisms**

Let, L be a regular language L, to prove that regular language is closed under homomorphism it is sufficient to show that for any homomorphism f, f(L) is also regular.
Given: A regular language L, A homomorphism f
To Prove: f(L) is also regular
Proof:

Since, L is a regular language, we can construct a DFA (say D) for L.
So, D = $(Q, \Sigma, \delta, q_0, F)$ such that, L(D) = L
First let me define a function $g : \Gamma \rightarrow \Sigma$ as follows:
$g(a) = \{x | f(x) = a\}$

Now, Consider the NFA N = $(Q, \Gamma, \delta', q_0, F)$ where,
$\delta' \subseteq Q \times \Gamma \times Q$ defined as follows:
$\delta'(q, a) = \{\delta(q, x) | x \in g(a)\}$

Proving that L(N) = f(L)
Let, $x \in f(L) => x = f(a)$ where, $a \in L$
Let, $a = a_1 a_2 .....a_k$
$=> x = f(a_1)f(a_2).....f(a_k)$
Since, $a \in L = L(D) => \exists\{q_0, q_1, ..., q_k\}$ where $d_i \in Q$ such that
$\delta(q_i, a_{i+1}) = q_{i+1} \forall i = 0, ..., k-1$ and $q_k \in F$
by defination of δ', $\delta'(q_i, f(a_{i+1})) = q_{i+1} \forall i = 0, ..., k-1$
also, $F' = F => q_k \in F'$
hence, N accepts x.
=> N accepts every string x $\in f(L)$
similarly it can be shown that N rejects every string x' $\notin f(L)$ (as run of N is parallel to D).
Hence L(N) = f(L).