Abhinav Jain, 2019CS10322
Arpit Chauhan, 2019CS10332
Himanshu Gaurav Singh, 2019CS10358

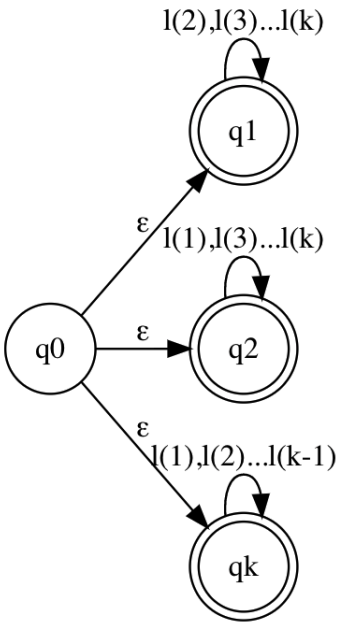
Problem Set - 1
COL352 - Introduction to Automata and ToC
Term: Holi 2022
Due Date: 28^{th} January, 2022

Problem 1
Given an alphabet $\Sigma = \{l_1, l_2, .., l_k\}$, construct an NFA that accepts strings that don't have all the characters from Σ. Can you give an NFA with k states?

Solution: Consider an NFA $N = (Q, \Sigma, \delta, q_0, F)$, where

- $Q = \{q_0, q_1, ...q_k\}$, $k+1$ states
- $\Sigma = \{l_1, l_2, .., l_k\}$, is the alphabet.
- The state q_0 is the start state.
- $F = \{q_1, q_2....q_k\}$, is the set of accept states.
- The start state q_0 has ϵ transitions to each of the other k states which themselves have $k-1$ self-transitions each for $a \in \Sigma - \{l_i\}$ for the i^{th} state,

$$\delta(q_i, a) = \begin{cases} \{q_1, q_2....q_k\} & \text{if i} = 0 \text{ and a} = \epsilon \\ q_i & \text{if i} \in \{1, 2....k\} \text{ and } a \in \Sigma - \{l_i\} \end{cases}$$



Proof of correctness N accepts only those strings that don't have all the characters from Σ.
Proof: Consider any string t such that it does **NOT** have some character l_j. Observe that this implies that $q_i \in \hat{\delta}^*(\{q_0\}, t)$ because as the computations proceeds, from the starting state, the state first ϵ transitions to q_i(along with other states),which is an accepting state and then retain it till termination since l_j does not lie in t. Hence, all strings that do not have all characters from Σ are accepted. $-$**1**
To prove the converse, consider any string s that contains all characters from Σ. As the computation proceeds, the state first ϵ transitions to the set $\{q_0, q_1, ..q_k\}$ but the states $\{q_1, ..q_k\}$ die out as the corresponding character is acted upon by N. Hence the final state is ϕ. Thus, s is not accepted. $-$**2**.
From **1** and **2**, we conclude that N accepts exactly those strings that dont have all characters of Σ.

The above NFA has $k+1$ states. Observe that the regular expression required in this question is basically the union of k disjoint regular expressions Σ_i^* where $\Sigma_i = \Sigma - \{l_i\}$ for $i \in \{1, 2...k\}$ and thus if we consider one additional start state, it might not be possible to give an NFA with only k states which recognizes this language, except for k=1 which is the trivial case.

Problem 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.

Solution: We will prove that all-NFAs are exactly as powerful as DFAs. We will prove that any langauge accepted by a DFA is also accepted by an all-NFA and vice-versa.

Proof

Consider any language L accepted by a DFA D. Consider the all-NFA M which has the transition function(δ), set of states(Q) and the starting state(q_0) same as D and the set F equal to the set of accepting states of D. Observe that M accepts exactly the same language L as accepted by D, since all strings terminating at an accepting state lie within F(since it is same as the set of all accepting states of D), whereas all others do not lie within F(since, they do not end at an accept state). Hence, any language which can be accepted by a DFA can also be accepted by an all-NFA. $-$**1**

To prove the converse, will use a variant of the subset construction. Consider any all-NFA M that accepts a langauge L. Consider the DFA $D(Q_D, \Sigma_D, q_{D_0}, \delta_D, F_D)$, such that

$$Q_D = 2^Q$$
$$\Sigma_D = \Sigma$$
$$q_{D_0} = \{q_0\}$$
$$\delta_D(S, a) = \cup_{q \in S} \delta(q, a), \ where\ S \in Q_D\ and\ d \in \Sigma_D$$
$$F_D = \{X \in 2^Q, X \subseteq F\}$$

Observe that this is similar to subset construction except that the set of accepting states of D is the set of all subsets of the set of accepting states in M.

Consider any string t, t is accepted by the M iff $\hat{\delta}(q_0, t) \subseteq F$. From the subset construction, we know that $\hat{\delta}(q_0, t) = \hat{\Delta}_D(q_0, t)$. This implies that $\hat{\Delta}_D(q_0, t) \subseteq F$ iff $t \in L$. Since F_D is equal to the set of all subsets of F, we can conlude that $t \in L$ iff $t \in L(D)$.

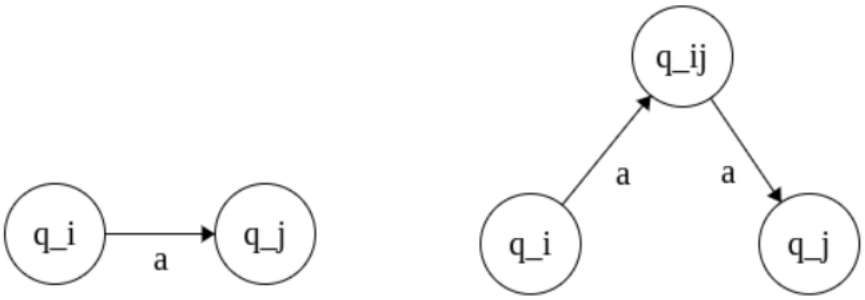
Hence, any language which can be accepted by an all-NFA can also be accepted by a DFA. $-$**2**

From **1** and **2**, we can conclude that all-NFAs recognise the same languages as DFAs which are exactly the regular languages.

Problem 3
Show that regular languages are closed under the **repeat** operation, where repeat operation on a language L is given by

$$\textbf{repeat}(L) = \{l_1 l_1 l_2 l_2 \dots l_k l_k \mid l_1 l_2 \dots l_k \in L\}$$

Solution: Since L is regular, there exists a DFA D such that $L(D) = L$. We show the construction of an NFA that accepts $repeat(L)$ using D. The NFA N is constructed by iterating through every transition arrow of D(say $\delta(q_i, a) = q_j$), and adding an extra state q_{ij} in between q_i and q_j such that $\delta(q_i, a) = q_{ij}$ and $\delta(q_{ij}, a) = q_j$. We will keep the start state and the set of accepting states as it is in D. We prove that this NFA accepts $repeat(L)$.



Proof of correctness Consider any string $s \in L$. If s is accepted by D with the path $q_0(start\ state), q_1, ...q_f$, we will get a path to q_f(also an accepting state in N) of the form $q_0, q_{01}, q_1, q_{12}, q_2, q_{23}, ...q_f$ for the string $repeat(L)$. Thus N accepts all strings in $repeat(L)$. –1
To prove the converse, consider any string t, accepted by N. Consider the path from the start state(q_0) to any accepting state. This will be of the form $q_0, q_{01}, q_1, q_{12}, q_2, ..q_k$. Observe that the corresponding character for the transition $q_0 \to q_{01}$ and $q_{01} \to q_1$ will be the same and similarily for each pair of transitions. Thus, t has the form $l_1 l_1 l_2 l_2, ..l_k l_k$, where $\delta(q_i, l_i) = q_{i+1}$. Thus, the string $l_1 l_2 ..l_k$ is accepted by L. **–2**
From **1** and **2**, we can conclude that N recognizes $repeat(L)$, hence $repeat(L)$ is regular.

ALITER
Consider a function $f : \Sigma* \to \Sigma*$ such that for a string $s = l_1 l_2 \dots l_k$ (where $l_i \in \Sigma$)

$$f(s) = l_1 l_1 l_2 l_2 \dots l_k l_k$$

.

Claim : Function f is a string homomorphism.
Proof : Consider any 2 strings x,y belonging to Σ^*. Let $x = a_1 a_2 \dots a_k$ and $y = b_1 b_2 \dots b_j$. By definition of f we have $f(x) = a_1 a_1 a_2 a_2 \dots a_k a_k$ and $f(y) = b_1 b_1 b_2 b_2 \dots b_j b_j$.
Since $x.y = a_1 a_2 \dots a_k b_1 b_2 \dots b_j$ and by the definition of f we have

$$f(x.y) = a_1 a_1 a_2 a_2 \dots a_k a_k b_1 b_1 b_2 b_2 \dots b_j b_j$$

Observe that $f(x.y) = (a_1 a_1 a_2 a_2 \dots a_k a_k).(b_1 b_1 b_2 b_2 \dots b_j b_j) = f(x).f(y)$. Since for all $x, y \in \Sigma^*$, $f(xy) = f(x)f(y)$ hence the claim holds.

Observe that the **repeat** operation on a language L can now be defined in terms of the function f as

$$\textbf{repeat(L)} = f(L) = \{f(x) \mid x \in L\}$$

Since L is a regular language and function f is a string homomorphism then by using the proof given in problem 6 we can say that **repeat(L)** $= f(L)$ is a regular language.

Problem 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:
Assume that the languages accepted by the automaton D_1, D_2 are L_1, L_2 respectively. We need to prove that $L_1 = L_2$. Observe that the sets L_1, L_2 are equal iff $L_1 \subseteq L_2$ and $L_2 \subseteq L_1$. This is equivalent to saying that $L_1 \cap L_2' = \phi$ and $L_2 \cap L_1' = \phi$.
We have learnt in class how to construct the DFA for the language $L_1 \cap L_2$ given the DFA of L_1 and L_2 and how to construct the DFA for L' given the DFA for L. We will use them in the algorithm.

The algorithm is as folllows:

- *Compute the DFAs that accept langauges L_1' and L_2' say, D_1' and D_2' respectively*
- *Compute the DFAs for the langauge $L_1' \cap L_2$ and $L_2' \cap L_1$, say D_{12} and D_{21}.*
- *If no accepting state is reachable from the start state for both the automata D_{12} and D_{21}, return true else return false.*

Proof of correctness

- The DFA for L_1'/L_2'(say D_1'/D_2') can be computed from D_1/D_2 by changing the set of acceptings to exactly those states that were not accepting states in D_1. This works, since exactly all those strings not belonging to L_1' are the ones that end at a non-accepting state.
- The DFA for $L_1' \cap L_2$(say D_{12}) can be computed using D_1' and D_2 using the product construction as discussed in the lecture. Similarily for D_{21}.
- D_{12} has no accepting states reachable from the start state iff the language $L_1 \cap L_2'$ is empty. To prove this, observe that if there is some reachable state from the starting state, the string formed by the alphabets lying on the path to the state lie in the corresponding language. Conversely, if there is no reachable accepting state from the start state, no string will ever be accepted by the automata. Thus, the claim is true.
  Similarily, D_{21} has no accepting states reachable from the start state iff the language $L_2 \cap L_1'$ is empty. Thus, as discussed above, this would imply that the languages L_1 and L_2 are identical. Observe that reachability in a DFA can be checked using any path-finding graph algorithm like BFS-DFS.

Problem 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: We know that a language is regular if and only if some nondeterministic finite automaton (NFA) recognizes it. It is given that A is a regular language and thus, there must exist some NFA $N = (Q, \Sigma, \delta, q_0, F)$ that recognizes the language A. We define $A^R = \{w^R | w \in A\}$ where $w^R = w_n...w_2w_1$ is the reverse of string $w = w_1w_2...w_n$.
Now we construct an NFA $N' = (Q', \Sigma, \delta', q_0', F')$ with the same alphabet Σ but

- $Q' = Q \cup \{q_0'\}$
  The states of N' are all the states of N with an additional state q_0'.
- The state q_0' is the start state of N'.
- $F' = \{q_0\}$, the start state of N is the only accept state of N'.
- We reverse all the state transition arrows of N and add additional ϵ transitions from the new start state q_0' to all the accept states of N i.e. the states in set F,

$$\delta'(q,a) = \begin{cases} F & \text{if q} = \text{q}_0' \textit{ and } a = \epsilon \\ \{\text{s} \in Q | q \in \delta(s,a)\} & \text{if } q \in Q \end{cases}$$

Proof of correctness If N recognizes A, then N' recognizes A^R.
For any string $w = y_1y_2..y_m \in A$, there will exist a sequence of states $r_0, r_1....r_m$ in Q such that $r_0 = q_0$, $r_{i+1} \in \delta(r_i, y_{i+1})$ for all $i = 0, 1, ..m-1$ and $r_m \in F$.
Now suppose we have the string $w^R = y_my_{m-1}..y_1 \in A^R$, we can obtain the sequence of states $q_0', r_m....r_1, r_0$ in Q' with q_0' as the start state, $r_0 = q_0$ as the accept state and transitions as per δ' in N' which is possible because all the transitions in N have been reversed while constructing N'.
So, we observe that corresponding to every string $w = y_1y_2..y_m \in A$ accepted by N, the string $w^R = y_my_{m-1}..y_1 \in A^R$ is accepted by N' and vice-versa.
Thus, NFA N' recognizes the language A^R. **–1**

To prove the converse, consider any string s accepted by N', since N' has only one accept state q_0, there must be a path from the start state$(_0^{'})$ to q_0 such that the characters along the path form s. Since, the arrows of N' are all reversed, s^R when computed on N, will eventually reach one of the accept states of N, since the state adjacent to $_0^{'}$ in the path in N' will certainly be an accept state of N. Thus, any string accepted by N' will have its reverse accepted by N. **–2**
Using **1** and **2**, we conlude that there exists an NFA that accepts A^R. Thus, A is regular implies A^R is regular.

Problem 6
Prove that the class of regular languages is closed under homomorphisms. That is, prove that if $L \subseteq \Sigma^*$ is a regular language, then $f(L) = \{f(x) \mid x \in L\}$ is regular. Try to informally describe how you will turn a DFA for L into an NFA for $f(L)$.

Solution: We will prove that $f(L)$ is regular by constructing an NFA that recognises it. Observe that since L is a regular language, there exists a DFA $D = (Q, \Sigma, \delta, q_0, A)$ that recognizes it. Consider an NFA $N = (Q', \Gamma, \delta', q_0, A)$ where δ' and $Q'$are defined below. For each character $a \in \Sigma$ there are 2 cases :

1. If $f(a) \in \Gamma \cup \epsilon$. $\{(q, f(a), \delta(q, a)) \mid q \in Q\}$ will be the transitions corresponding to $f(a)$ in δ'.
2. If $f(a) \notin \Gamma \cup \epsilon$. Hence in this case $f(a)$ consists of multiple characters and is of form $f(a) = l_1 l_2 \cdot \cdots l_k$ where $l_i \in \Gamma$. In this case each transition edge in the transitions representing $(q, a, \delta(q, a)) \in \delta$ is broken down into a path of length $k+1$ by creating $k-1$ intermediate states, say $q'_{al_1}, q'_{al_2}, \ldots, q'_{al_{(k-1)}}$, and the transition edges $(q, l_1, q'_{al_1}), (q, l_1, q'_{al_2}), \ldots (q'_{al_{k-1}}, l_k, \delta(q, a))$ belonging to δ'.

Note that Q' is the union of Q and all of the new states formed in the above case. Now we will show that $f(L)$ is the language accepted by N. We will show that in the following 2 claims.

Claim : For each $x \in L$, the string $f(x)$ is accepted by N.
Proof : Observe that $x = l_1 l_2 \ldots l_n$ where $l_i \in \Sigma$. Observe that since f is a string homomorphism we have $f(x) = f(l_1.(l_2 \ldots l_n)) = f(l_1).f(l_2 \ldots l_n)) = \ldots = f(l_1).f(l_2) \ldots f(l_n)$. As x is accepted by D there exists a sequence of states $q_{i_1}(= q_0), q_{i_2}, \ldots, q_{i_n}$ such that $q_{i_{k+1}} = \delta(q_{i_k}, l_k)$ for each $k \in \{1, 2, \ldots, n-1\}$ and $q_{i_n} \in A$. Observe that in the case of N, by the very definition of δ' we have $q_{i_{k+1}} \in \hat{\delta}'(q_{i_k}, f(l_k))$ for each $k \in \{1, 2, \ldots, n-1\}$. Since $f(x)$ is a concatenation of $f(l_1), f(l_2) \ldots f(l_n)$ hence we have $q_{i_n} \in \hat{\delta}'(q_{i_0}, f(x))$. As $q_{i_n} \in A$ therefore the string $f(x)$ is accepted by N. Hence the claim is proved.

Claim : For each string y accepted by N, $y \in f(L)$.
Proof : Let S be the sequence of states observed in an accepting run of N for the string y. Observe that S is starts with $q_{i_1}(= q_0)$ and ends with q_{i_n} such that $q_{i_n} \in A$. Let $Q'' = Q' - Q$ i.e it represents the set of newly added states in N. Now observe that S is of form $q_{i_1}, r_{i_2}, q_{i_2}, r_{i_2}, \ldots, r_{i_n}, q_{i_n}$ where each $q_{i_k} \in Q$ and each r_{i_k} $(k > 1)$ represents some sequence of states where each state is in Q''. Observe that for each $k \in \{1, 2, .., n-1\}$ the sequence of states in S between q_{i_k} and $q_{i_{k+1}}$ i.e r_{k+1} forms a path as each state in Q'' has both indegree and outdegree as 1. Further by the very definition of δ' the string corresponding to the sequence of states q_{i_k},r_{k+1},$q_{i_{k+1}}$ is $f(x_k)$ for some $x_k \in \Sigma$ such that $q_{i_{k+1}} = \delta(q_{i_k}, x_k)$. Hence $y = f(x_1).f(x_2)...(f_{x_{n-1}})$ where each $x_i \in \Sigma$. Observe that since $q_{i_{k+1}} = \delta(q_{i_k}, x_k)$, therefore $q_{i_n} = \hat{\delta}(q_{i_1}, x = x_1.x_2...x_{n-1})$ and since $q_{i_n} \in A$, therefore x is accepted by D. As f is a string homomorphism we have $y = (f(x_1).f(x_2))...f(x_{n-1}) = (f(x_1.x_2).f(x_3)) \ldots f(x_{n-1}) = \ldots = f(x_1.x_2...x_{n-1}) = f(x)$. Since $x \in L$ therefore we have $y \in f(L)$. Hence the claim is proved.

Using the above 2 claims we have proved that $f(L)$ is the language accepted by N.