COL352
Problem Sheet 1

Himanshi Ghai (2019CS50433)
Mallika Prabhakar (2019CS50440)
Sayam Sethi (2019CS10399)

January 2022

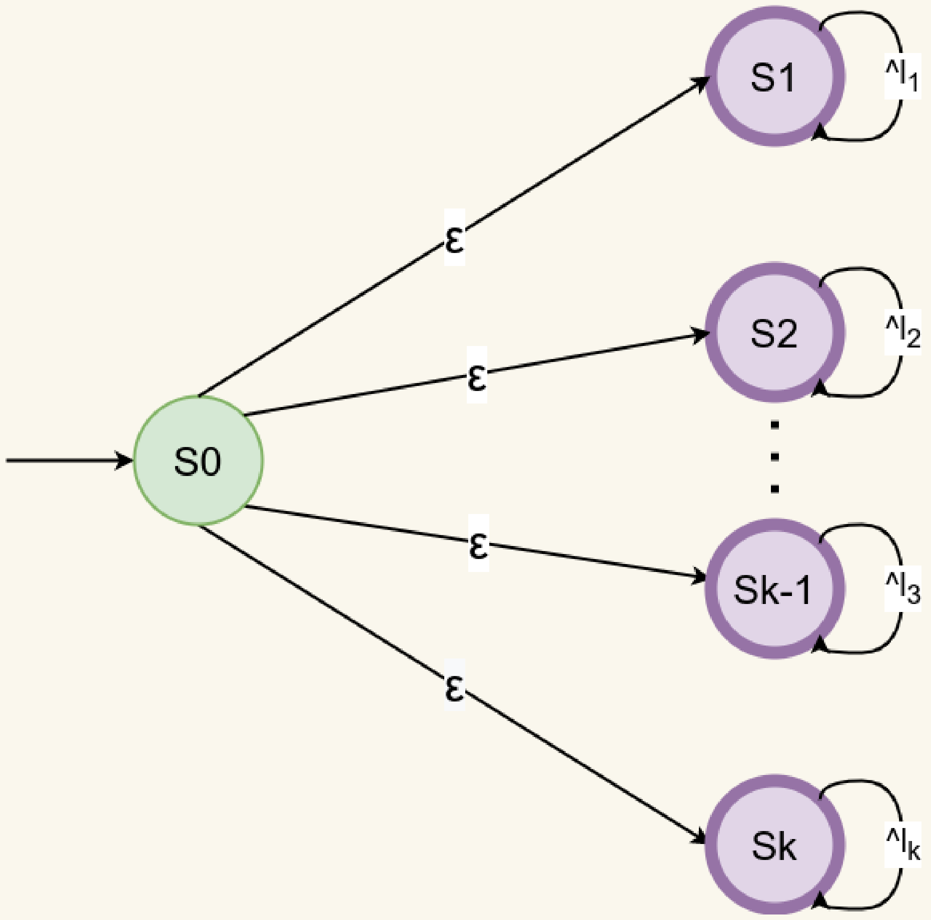
Contents

1 Question 1

Question 1

Question. *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?*

Solution. We have to construct an NFA such that the string (x) to be accepted has atmost $k-1$ unique letters used out of language Γ. The regular expression for the problem can be defined as $x = [\Sigma \setminus l_1]^* + [\Sigma \setminus l_2]^* + \cdots + [\Sigma \setminus l_{k-1}]^* + [\Sigma \setminus l_k -]^*$
(where + is a union and we are following code regex syntax)
For every regular expression, we can define an $\epsilon - NFA$. The $NFA(Q, \Sigma, \delta, q_0, F)$ created by aforementioned regular expression is defined by following diagram:



There are a total of $k+1$ states $(Q = S0, S1, ...SK)$, Σ is the alphabet which can be equal to Γ. The transitions δ are as shown in the figure. $^\wedge l_r$ denotes all characters allowed for transition except l_r in state Sr. q_0 is initial state which is $S0$ here. The final states F are all states except $S0$.
We could not give an NFA with k states as the regex couldn't be reduced further nor the NFA since it considers all the k characters apart from the start state involving ϵ transitions.
□

2 Question 2

Question 2

Question. *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. To prove: All-NFAs recognize the class of regular languages.
To prove this, we first show that for every regular language there is one NFA which accepts it and every NFA recognizes a regular language.

Claim 2.1. *For every regular language, there exists an all-NFA which recognizes the regular language.*

Proof. We have a regular language L which implies there exists a DFA representing it. We have to prove it can be recognized by an all-NFA.
Proof by deduction: Either the string is accepted or the string is rejected based on the run of the DFA which is deterministic. The definition of all-NFAs is that it accepts a string only when the string is accepted by all the possible final states. Since each run of the DFA on a string has only one final state it is a type of all-NFA with single final state (which may be accepting or rejecting). Therefore, a DFA is a subset of all-NFAs.
A regular language is a language which is accepted by a deterministic finite automaton. Since every regular language has a DFA which accepts it, and DFA $\in$ set of all-NFAs; for every regular language there exists an all-NFA which recognizes it. □

Claim 2.2. *Every all-NFA recognizes some regular language.*

Proof. We have to prove that any language L recognized by all-NFA is a regular language. For a language to be regular, it is to be identified by some finite automaton. Therefore we need to show that an all-NFA can be translated into a DFA. To do this, we try to find a DFA equivalent to that all-NFA.
Proof by construction: Let the all-NFA which recognizes language L be $N = (Q, \Sigma, \delta, q_0, F)$. Also, let $D = (Q', \Sigma, \delta', q_0', F')$ be a DFA which recognizes L and will imply that L is regular.
Such DFA can be constructed in the following way:

1. The alphabet Σ of both D and N are same to ensure same input strings.

2. Since all-NFA requires the string to be accepted for all final states and the DFA will have to consider all possible transitions in all-NFA, the total number of states Q' for DFA will be equal to *power set*$(Q) = 2^Q$.

3. Since all-NFA might involve epsilon transitions, to have a DFA version, the transition table δ' for a state S and letter a (set of corresponding states in N) will be:

$$\delta'(S, a) = \bigcup_{s \in S} \epsilon - closure(\delta(s, a)) \tag{1}$$

4. For a string to be accepted, in each run of the all-NFA, the run should end in a state $\in F$. Therefore the possible states F' the string can end up in must be equal to $\textit{power set}(F) = 2^F$.

Following the construction, it can be observed that for any string $x \in \Sigma^*$, let S be the set of all possible states of all-NFA that we can end up in. This should be a subset of 2^F for it to be accepted. Our constructed DFA does exactly this. Conversely, each string accepted by D will be accepted by N since the final state in D exactly corresponds to the possible states that a string can end up in N. □

Both Claim 2.1 and Claim 2.2 imply that for every regular language, there exists an all-NFA which recognizes the regular language L and all-NFAs only recognize regular languages. Hence all-NFAs recognize the class of regular languages. Hence proved.

□

3 Question 3

Question 3

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

$$repeat(L) = \{l_1l_1l_2l_2.....l_kl_k | l_1l_2l_3....l_k \in L\}$$

Solution. To show regular languages are closed under the *repeat* operation we will construct an NFA which recognises $repeat(L)$ language.
Let $D = (Q, \Sigma, \delta, q_0, F)$ be the DFA which recognizes language L.

Construction: Using D lets construct NFA $N = (Q', \Sigma', \delta', q'_0, F')$ which recognize langauge $repeat(L)$.
Consider two states q_i and q_j in D such that $q_i, q_j \in Q$ and $\delta(q_i, l) = q_j$ where $l \in \Sigma$. We introduce here a new state q' in N for every transition in D such that

$$\begin{aligned} \delta'(q_i, l) &= q' \\ \delta'(q', l) &= q_j \\ \delta'(q', l') &= \phi \textit{ where } l' \in \Sigma \textit{and } l' \neq l \end{aligned} \tag{2}$$

So N could be defined as following

1. Q' is set of all states in Q and the new states introduced as described above.
2. $q'_0 = q_0$
3. $F' = F$
4. δ' can be defined from δ. Replace every transition in δ with transitions defined in equation 1.

Now we need to prove that language accepted by N is same as $repeat(L)$. We will prove it using the following claim.

Claim 3.1. *Let $s = l_1l_2l_3,,,l_k$ be a string accepted by N then NFA will be at state which belongs to Q after accepting character l_{2i} $\forall i$.*

Proof. We will prove the following hypothesis using induction to prove the above claim as true

$$P(i) = N \textit{ will be at a state belonging to } Q \textit{ after accepting character } l_{2i} \ \forall i \textit{ in strings}$$

Base case: For $i = 0$, N will be at start state which is q_0 and belongs to Q. Hence $P(0)$ holds true.
Induction Step: Lets assume that $P(i)$ is true i.e N is in state belonging to Q after accepting l_{2i}. Let the state be q. Now on accepting the next character l_{2i+1} it can make transition to only states which were introduced in N from D according to equation 1. Let the state be q' which belong to $Q' \setminus Q$. According to δ' next state which can be reached from q' is a state which belongs to Q. So the next character accepted in string s which is l_{2i+2} will be same as l_{2i+1} and N will make transition to state which belongs to Q.
Thus $P(i + 1)$ is also true. Hence using induction we proves the above claim. □

In the proof above, we also stated that $l_{2i-1} = l_{2i}$ $\forall i$ in a string s accepted by N.

Using the above claims we can state that a string s accepted by N is of the form $l_1 l_1 l_2 l_2 ... l_k l_k$ where $l_1, l_2, ...l_k \in \Sigma$. As repeat is a bijection from to L to language defined by $repeat(L)$ thus on applying the inverse of *repeat* operation on s we get $l_1 l_2 l_3 .. l_k$ which belongs to L. Hence langauge accepted by N is same as $repeat(L)$.

As $repeat(L)$ is accepted by an NFA, we can conclude that the language is regular and regular languages are closed under *repeat* operation.

$\square$

4 Question 4

Question 4

Question. *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 D_1 and D_2 be two DFAs s.t

$$L(D_1) = L_1$$

$$L(D_2) = L_2$$

If languages L_1 and L_2 are same then following must hold true

$$L_1 \cap L_2^C = \{\phi\} \tag{3}$$

and

$$L_1^C \cap L_2 = \{\phi\} \tag{4}$$

As regular language is closed under intersection and complement we can find DFAs for languages in eq (2) and eq (3) using D_1 and D_2. Language accepted by a DFA is ϕ only when any final state is not reachable from the start state. We will use these claims to design the algorithm for checking if given two DFAs has same language.
We will use a DFA class object in following algorithms. It has following attributes.

1. Q is set of states of DFA.
2. q_0 represent start state
3. Σ is alphabet of DFA
4. F is set of final states.
5. δ represent transition table for DFA.

Using the following algorithm we can compute the DFA for complement language.

Algorithm 1 Finding DFA of complement of langauge accepted by given DFA D

```
1: procedure COMPLEMENT($D$)
2:     $DC \leftarrow$ new DFA
3:     $DC.Q \leftarrow D.Q$
4:     $DC.q_0 \leftarrow D.q_0$
5:     $DC.\delta \leftarrow D.\delta$
6:     $DC.final \leftarrow D.Q\setminus D.f$          ▷ Non accepting states of $D$
7:     return DC
8: end procedure
```

Now we need to find DFA for the intersection of two languages. DFA of intersection of two languages, say L_1 and L_2 accepted by D_1 and D_2 respectively is given by product of DFAs D_1 and D_2. We will compute the product DFA using the following algorithm.

Algorithm 2 Finding product DFA of two given DFAs

```
1: procedure PRODUCT($D_1, D_2$)
2:     $D \leftarrow$ new DFA
3:     $D.Q \leftarrow D_1.Q \times D_2.Q$
4:     $D.q_0 \leftarrow (D_1.q_0, D_2.q_0)$
5:     $D.F \leftarrow D_1.F \times D_2.F$
6:     for all $(q_i, q_j) \in D.Q$ do
7:         for all $a \in D.\Sigma$ do
8:             $D.\delta((q_i, q_j), a) \leftarrow (D_1.\delta(q_i, a), D_2.\delta(q_j, a))$
9:         end for
10:    end for
11:    return $D$
12: end procedure
```

Now too check whether the language of a DFA D is $\{\phi\}$, we will do BFS on D to check if the final state is reachable from start state in some transitions. If final state is not reachable then we can say that language of DFA is $\{\phi\}$ as no string would be accepted by D.

Algorithm 3 Checking if any final state is reachable from start state in given DFA D

```
1: procedure ISLANGUAGEEMPTY($D$)
2:     visited $\leftarrow$ init map with every state in $D.Q$ mapping to 0
3:     queue$\leftarrow$ $[D.q_0]$
4:     visited$[D.q_0] \leftarrow 1$
5:
6:     while len(queue) $! = 0$ do
7:         state $\leftarrow$ queue.pop(0)
8:         for all $a$ in $D.\sum$ do
9:             if not visited$[D.\delta(state, a)]$ then
10:                queue.append($D.\delta(state, a)$)
11:                visited$[D.\delta(state, a)] = 1$
12:            end if
13:        end for
14:    end while
15:
16:    emptyLanguage $\leftarrow 1$
17:    for all $q$ in $D.F$ do
18:        if visited$[q]$ then
19:            emptyLanguage $\leftarrow 0$
20:            break
21:        end if
22:    end for
23:
24:    return emptyLanguage
25: end procedure
```

Using the above three procedures and stated claims we can check if two DFAs have same language. Following is the pseudo code for the algorithm.

Algorithm 4 Checking if DFA D_1 and D_2 recognize same language

```
procedure sameLanguage($D_1, D_2$)
    $DFA_1 \leftarrow$ PRODUCT($D_1$, COMPLEMENT($D_2$))
    $DFA_2 \leftarrow$ PRODUCT(COMPLEMENT($D_1$), $D_2$)

    if EMPTYLANGUAGE($DFA_1$) and EMPTYLANGUAGE($DFA_2$) then
        return true
    end if
    return false
end procedure
```

□

5 Question 5

Question 5

Question. *For any string $w = w_1w_2 \dots w_n$ the reverse of w written w^R is the string $w_n \dots w_2w_1$. 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.*

Proof. Consider the DFA for $D_A = (Q, \Sigma, \delta, q_0, F)$. Construct the following NFA $N_A = (Q', \Sigma', \delta', q', F')$:

$$\begin{aligned} Q' &= Q \cup q' \\ \Sigma' &= \Sigma \\ \delta'(q_i, a) &= q_j \iff \delta(q_j, a) = q_i \\ \delta'(q', \epsilon) &= F \\ F' &= \{q_0\} \end{aligned} \tag{5}$$

Claim 5.1. *N_A recognises the language A^R, i.e., every word in A^R is accepted by N_A and every word not in A^R is rejected by N_A*

Proof. Consider any word w in A, it is recognised by D_A (from construction). Consider the sequence of states visited during the run of acceptance of w. Let it be $S = \{q_0 = q_{w_1}, q_{w_2}, \dots, q_{w_n}\}$. Now consider the following run in N_A of w^R:

$$\{q', q_{w_n}, \dots, q_{w_2}, q_0 = q_{w_1}\} \tag{6}$$

The above is a valid run since each transition is a valid one from the construction of N_A (the first one is an ϵ-transition and the remaining are fixed transitions). This proves that every word in A is recognised by N_A.
To prove the converse, assume by contradiction that a word $w^R = w_n \dots w_2w_1 \notin A$ is accepted by N_A. Let the sequence of states be $S' = \{q', q_{w_n}, \dots, q_{w_2}, q_{w_1} = q_0\}$. Now, from the construction of N_A, the following sequences of states should be accepted by D_A:

$$\{q_{w_1} = q_0, q_{w_2}, \dots, q_{w_n}\} \tag{7}$$

This corresponds to accepting the string w. However, D_A accepts only strings in A. This is a contradiction to our assumption. Therefore, N_A rejects every string not in A.
Therefore, A^R is recognised by N_A which is an NFA. Therefore, A^R is a regular language as well. Thus, regular languages are closed under the reverse operation. Hence, proved. □

□

6 Question 6

Question 6

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

Proof. Consider the regular expression, R of the language L. Now, we define $f(R)$ by applying f on each alphabet in R.

Claim 6.1. *Any string $w \in f(L)$ is accepted by $f(R)$ and vice versa. We prove this claim using structural induction on the regular expression R.*

Proof. **Base case:** The claim is trivially true for $R = \epsilon$ and for $R = a$ (for all $a \in \Sigma$)
Inductive Step: Assume it is true for all regular expressions with number of symbols $\leq i$. A regular expression with $i + 1$ symbols will look like one of the following -

1. $R_1 + R_2$
2. $R_1 \cdot R_2$
3. R_1^*

For each of the following cases, induction is true for R_1 and R_2 (if applicable) since number of symbols in each is $< i$. Also, for each of the cases, we know that for any $w \in f(L_1)$ is accepted by $f(R_1)$ (also for L_2 corresponding to R_2). Consider the following languages:

1. $R_1 + R_2$: For a string to be accepted by $f(R)$, it has to be accepted by either $f(R_1)$ or $f(R_2)$. This will be true iff the string is in $f(L_1)$ or $f(L_2)$. This is true from induction. Therefore, any string in $f(L_1) \cup f(L_2) = f(L_1 \cup L_2) = f(L)$ will be recognised by $f(R) = f(R_1 + R_2)$ and vice versa.
2. $R_1 \cdot R_2$: For a string to be accepted by $f(R)$, it has to be of the form $w_1 w_2$ where w_1 is accepted by $f(R_1)$ and w_2 is accepted by $f(R_2)$. Therefore, the language recognised by $f(R)$ is $f(L_1) \cdot f(L_2) = f(L_1 \cdot L_2) = f(L)$. Also, any string $w = w_1 w_2 \in f(L_1 \cdot L_2)$ is recognised by $f(R)$.
3. R_1^*: This is equivalent to the language $\bigcup_{i=0}^{\infty}(L_1 \cdot L_1 \cdots i \textit{ times} \cdots L_1)$ or the regex $\sum_{i=0}(R_1 \cdot R_1 \cdots i \textit{ times} \cdots)$. We have already proved the correctness for concatenation and union. Therefore, we can apply the same for the finite union of the concatenation. Therefore, $f(R_1^*)$ recognises the same language as $f(L_1^*)$.

Therefore, we have proved the inductive hypothesis and hence the claim. □

From the above claim, we have shown that regular languages are closed under homomorphism. We now propose the construction of an NFA for $f(L)$.

Let the DFA of L be $D = (Q, \Sigma, \delta, q_0, F)$. Now construct the following NFA, $N = (Q, \Gamma, \delta', q_0, F)$:

$$\delta'(q_i, f(a)) = q_j \iff \delta(q_i, a) = q_j \tag{8}$$

The above construction doesn't create an NFA in the strictest sense since each transition can have a sequence of alphabets rather than a single alphabet. In such a case, we can add states for each alphabet connecting the two states q_i and q_j. $\square$