COL352: Assignment 2

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

August 14, 2023

1.

To show that bin(p) is not a regular language, we will use the pumping lemma. We will show that pumping lemma is violated and hence this cannot be a regular language. We have n, and let's take a word bin(p) such that p is prime number and $|bin(p)| \geq n$. Here || represents the length of the word.

Now consider $bin(p) = xyz$, such that $|xy| \leq n$. consider the individual binary strings x, y, z. The numbers represented by them be p_x, p_y, p_z. Take the following values $|z| = i$, $|yz| = j$, $|xyz| = k$. From the above, we have the following: (assuming LSB is rightmost bit)

$$p = p_z + 2^i p_y + 2^j p_x \tag{1}$$

Now we will pump y 't' times. The updated expression will become:

$$q_t = p_z + 2^i p_y + 2^j p_y \cdots + 2^{i+(j-i)(t-1)} p_y + 2^{i+(j-i)t} p_x \tag{2}$$

The sum of $2^i p_y + 2^{i+(j-i)} p_y \cdots + 2^{i+(j-i)(t-1)} p_y$ can be calculated using GP. The first term of the GP is $2^i p_y$ and the common ratio is 2^{j-i}. The total number of terms in the GP is t. Hence the sum will be:

$$2^i p_y + 2^{i+(j-i)} p_y \cdots + 2^{i+(j-i)(t-1)} p_y = \frac{2^i p_y (2^{(j-i)t} - 1)}{2^{j-i} - 1} \tag{3}$$

Substituting the above in the above equation 2, we get:

$$q_t = p_z + \frac{2^i p_y (2^{(j-i)t} - 1)}{2^{j-i} - 1} + 2^{i+(j-i)t} p_x \tag{4}$$

We need to show that exists some t for which q_t is not a prime number. We will take t = p and show that q_p is divisible by p. For this we will be using the Fermat's theorem which states that $x^p \equiv x \bmod p$.

$$q_p \bmod p = (p_z \bmod p) + \frac{(2^i p_y (2^{(j-i)p} - 1) \bmod p)}{2^{j-i} - 1} + (2^{i+(j-i)p} p_x \bmod p) \tag{5}$$

using the properties of modulo, this can be re-written as:

$$q_p \bmod p = (p_z \bmod p) + \frac{(2^i p_y ((2^{(j-i)p} \bmod p) - 1) \bmod p)}{2^{j-i} - 1} + ((2^{(j-i)p} \bmod p) 2^i p_x \bmod p) \tag{6}$$

Using fermat's theorem we can write $2^{(j-i)p} \equiv 2^{(j-i)} \bmod p$, putting this in (6) we get,

$$q_p \bmod p = (p_z \bmod p) + (\frac{2^i p_y (2^{(j-i)} - 1)}{2^{j-i} - 1} \bmod p) + (2^{(j-i)}) 2^i p_x \bmod p \tag{7}$$

$$q_p \bmod p = (p_z \bmod p) + (2^i p_y \bmod p) + (2^j p_x \bmod p) \tag{8}$$

$$q_p \bmod p = (p_z + 2^i p_y + 2^j p_x) \bmod p \tag{9}$$

$$q_p \bmod p = p \bmod p = 0 \tag{10}$$

Hence we can see that q_p is not a prime number since it is divisible by p. This means that for all n, there exists a word such that for all the possible breakouts of it, there exists a pumping length such that word formed will not be the binary representation of some prime number. This means that pumping lemma is violated and hence we can say that the language is not a regular language.

2.

L2 is not a regular language.

To show that this is not a regular language, we will show that pumping lemma is violated. For a given n, consider a word $a^{fib(i)}$, here $fib(i)$ is i^{th} fibonacci number just greater than n. Now we can break the string into the form xyz such that $|xy| \leq n$ and $|xyz| = fib(i)$. On pumping y 2 times the length becomes, $|xyz| + |y|$ and let this be represented by m. If m is not a fibonacci number then we are done, we have shown that the string formed on pumping does not belong to language. But if m is a fibonacci number then let's denote it with fib(j), where $j > i$. Also we know that $|y| \leq n < fib(i)$. The possible values of j can be i+1, i+2, ...

Take the case of $j = i + 1$. In this case $|y|$ must be $fib(i - 1)$ which is acceptable and possible. But if we take $j = i + 2$ then $|y|$ will be $fib(i + 1)$, this is not acceptable since we know that $|y| < fib(i)$ and $fib(i) \leq fib(i + 1)$. Thus we can say that $|y| = fib(i - 1), |xyz| = fib(i), |xyyz| = fib(i + 1)$

Now pump y 3 times, then the resulting string will be xyyyz. If $|xyyyz|$ is not a fibonacci number then we are done, we have shown that the string formed on pumping does not belong to the language. But if $|xyyyz|$ is a fibonacci number then let's denote it with fib(k), where $k > i + 1$. Let's see if it is possible to form fib(i+2). $fib(k) = fib(i + 1) + |y|$, then $|y|$ should be at least $fib(i)$, but we know that $|y| < fib(i)$. Hence it is not possible to form fib(i+2), and thus any bigger fibonacci number can also not be formed. Thus our assumption was wrong. $|xyyyz|$ cannot be a fibonacci number. Thus the string formed on pumping y 3 times is not accepted in the language. So pumping lemma is violated. Hence L2 is not a regular language.

3.

We need to show that $A_{\frac{1}{2}-}$ is a Regular language given that A is a regular language. For A let's say we have a DFA for language A, given by M = $\{Q, \Sigma, q_0, F, \delta\}$ where the symbols are according to the usual notation. Now we will construct a DFA for $A_{\frac{1}{2}-}$. If we are able to show that there exists a DFA which recognizes the language, then we have shown that the $A_{\frac{1}{2}-}$ is regular.

The definition of the DFA for $A_{\frac{1}{2}-}$, N, is as follows:

- $Q^{'} = Q \times 2^Q$
- $q_0^{'} = (q_0, F)$
- $\Sigma^{'} = \Sigma$
- $F^{'} = \{(q, \hat{q}) | q \in \hat{q}\}$, here $\hat{q}$ is one of the elements from power set of Q
- $\Delta((q, \hat{q}), a) = (\delta(q, a), \{q' | \delta(q', c) \in \hat{q}, c \in \Sigma\})$, the meaning of this will be as we read one symbol and come to $\delta(q, a)$ from q, there exists a symbol which lead to q' from some state in $\hat{q}$.

Now we will show that $N = \{Q^{'}, q_0^{'}, \Sigma, F^{'}, \Delta\}$ accepts the language $A_{\frac{1}{2}-}$ and for any string x accepted by N, we can show there exists y, such that $|y| = |x|$ and $xy \in A$.

Claim: If the automata N (defined above) is in some state $(q, \hat{q})$ after reading some input x ($|x| = k$), then all the state from which there exists a path of length k to some final state in automata M belongs to $\hat{q}$.

Proof: We prove this by induction on length of string read.
Base Case: On reading an empty string, the automata N will be in state (q_0, F). Length of string read so far is k=0. Now, from every state in F, on reading an empty string, automata N will still be in final state. Thus, base case is true.
Induction Step: Let us assume that the claim is true for all strings x such that |x|=k. We need to show that this is also true for all strings of length k+1. Consider any string $y = xa$ where $|y| = k + 1$, $x \in \Sigma^*$ and $a \in \Sigma$. Suppose that automata is in some state $(q, \hat{q})$ after reading string x. Then, after reading character a in state q, N will move to state $(\delta(q, a), \hat{q}')$ where $\hat{q}' = \{q' | \delta(q', c) \in \hat{q}, c \in \Sigma\}$. Note that $\hat{q}'$ contains all the states from which there exist a path to reach some state in $\hat{q}$ by reading a single character. Also, by inductive assumption, all the state from which there exists a path of length k to some final state in automata M belongs to $\hat{q}$. Thus, all the state from which there exists a path of length k+1 to some final state in automata M belongs to $\hat{q}'$.
Thus, we proved the claim by induction.

Now, we prove that N recognizes the language $A_{\frac{1}{2}-}$.
Show that N accepts $A_{\frac{1}{2}-}$
We have a string x such that for it there exists y such that $|x| = |y|$ and $xy \in A$. We need to show that N accepts x. After reading the $|x|$ symbols in M, we end up in the state q_x, after this on reading $|y|$ symbols more, we end up in one of the final states say q_f. Hence the sequence of states followed when we read x will be $q_0, \ldots, q_x$ and then $q_x, \ldots, q_f$ on reading y. Now we can see that, there exists $|y|$ alphabets such that we can come to state q_f starting from q_x after reading them. Thus when we parse x in N, let's say we move along the states $((q_0, F), \ldots, (q_x, \hat{q_x}))$. By the above claim, all the states which contain a path of length |x| to some final state belongs to $\hat{q_x}$. Since, on reading y from state q_x, we reach in some final state, there exists a path of length |x|=|y| from q_x to some final state. Thus, $q_x \in \hat{q_x}$. By definition of accept state for N, $(q_x, \hat{q_x}) \in F'$. Thus, x is accepted by N.

Show that for any string x accepted by N, there exists y, such that $|y| = |x|$ and $xy \in A$
Now we are given that string x is accepted in the N. We need to show that we can find a string such that $|x| = |y|$ and $xy \in A$. Let's say we go through the following states during the run of DFA N, $((q_0, F), \ldots, (q_x, \hat{q_x}))$ and $q_x \in \hat{q_x}$. Note that the transition functions are made such that there exists a |x| length string such that we can go from q_x to F (see the claim proved above). Thus, there exist some string y such that |x|=|y| and xy is accepted by M. Therefore, by definition, $x \in A_{\frac{1}{2}-}$.

Hence the DFA, N constructed by recognizes the language $A_{\frac{1}{2}-}$. Thus $A_{\frac{1}{2}-}$ is a Regular language.

4.

We will give an example of regular language in which the language formed after removing the middle third of it is not a regular language. Consider the following language:

$$\mathbf{A} = \{a^*D^+b^*\} \tag{11}$$

D is a dummy alphabet. The above is a regular expression since a^*, b^* and D^+ are regular expressions and we know that regular expressions are closed under concatenation. Hence, we can conclude that $a^*D^+b^*$ is a regular expression and thus a regular language.

$$A_{\frac{1}{3}-\frac{1}{3}} = \{xz| \text{ for some } y, |x| = |y| = |z| \text{ and } xyz \in A\}$$

Also note that $|xyz| \equiv 0 \bmod 3$. Let's assume that $A_{\frac{1}{3}-\frac{1}{3}}$ is regular. Now consider another regular language B which is represented by the regular expression $B = a^*b^*$. We know that the regular languages are enclosed under intersection. Hence $A_{\frac{1}{3}-\frac{1}{3}} \cap a^*b^*$ must be a regular language. Now, we prove the following claim:

Claim 4.1: $A_{\frac{1}{3}-\frac{1}{3}} \cap B = \{a^nb^n|n \in N\}$

Proof of Claim:
First of all, we are only interested in those words of A whose length is a multiple of 3 i.e. words of the form $a^iD^jb^k$ where $i + j + k \equiv 0 \bmod 3$ and $j > 0$. Further, we are interested in only those words of $A_{\frac{1}{3}-\frac{1}{3}}$ which have no D (words of the form a^xb^y), since only those may appear in the intersection, the rest of them are not present in B, since B has no word with letter D. Now, we prove that for every word of the form $a^xb^y \in A_{\frac{1}{3}-\frac{1}{3}}$, x = y.

Let's say that such word $a^xb^y \in A_{\frac{1}{3}-\frac{1}{3}}$ is derived from some word $a^iD^jb^k \in$ A. Since there is no D in a^xb^y, then this means that it should be contained entirely in the middle one-third section of $a^iD^jb^k$. In other words, no D should be present in the first one-third part or the last one-third part of the $a^iD^jb^k$. So, the first one-third part consists only of a and the last third part contains only b. So, after removing the middle one-third, all that is left is $a^{\frac{i+j+k}{3}}b^{\frac{i+j+k}{3}} = a^xb^y$. Clearly, x = y.

From the above paragraph, we proved that all strings in $A_{\frac{1}{3}-\frac{1}{3}}$ which don't contain any D are of the form a^nb^n. Thus, $A_{\frac{1}{3}-\frac{1}{3}} \cap B = \{a^nb^n|n \in N\}$.

But we have already proved that the language $\{a^nb^n|n \in N\}$ is not a regular language. So, we have a contradiction. Thus, by contrapositive, at least one of $A_{\frac{1}{3}-\frac{1}{3}}$ and B should be non-regular. But we already know that B is regular. So, $A_{\frac{1}{3}-\frac{1}{3}}$ is non-regular for the given A. Thus, $A_{\frac{1}{3}-\frac{1}{3}}$ may not always be a regular language even if A is a regular language.

5.

A 2-NFA A is a 5-tuple

$$A = (Q, S, t, F,)$$

where Q is the set of states, S the set of start states, t is an accept state, the transition function

$$\Delta : Q \times (\Sigma \cup \{\#, \$\}) \rightarrow 2^{Q\times(L,R)}$$

Assume that whenever M accepts, it does so my moving the head (pointer) all the way to the right endmarker \$ and entering accept state t. In the subsequent two questions, we will try to prove that 2-NFAs accept only regular languages.

(a): Let $x = a_1 ... a_n \in \Sigma,\ 1 \leq i \leq n$. Let $a_0 = \#, a_{n+1} = \$$. Argue that x is not accepted by A if and only if there exist sets $W_i \subseteq Q, 0 \leq i \leq n+1$ such that the following hold:

- $S \subseteq W_0$
- If $u \in W_i, 0 \leq i \leq n$, and $(v, R) \in \Delta(u, a_i)$, then $v \in W_{i+1}$
- If $u \in W_i, 1 \leq i \leq n+1$, and $(v, L) \in \Delta(u, a_i)$, then $v \in W_{i-1}$ and
- $t \in W_{n+1}$

Solution:
To prove the given statement, we need to show the implications in both the directions.

Proof of $\Rightarrow$:
Let us assume that x is not accepted by A. Then we need to show that $\exists\ W_i, 0 \leq i \leq n+1$ that satisfy the given conditions. We construct such W_i's by using the construction algorithm as shown below:

Algorithm to construct W_i:

```
Initialise $W_0 \leftarrow \{S\}$ and $W_i \leftarrow \{\}, 1 \leq i \leq n+1$
While(there is any change in any $W_i$){
    for $i \leftarrow$1 to n+1 do
        $W_i \leftarrow W_i \cup \{v \mid (v, R) \in \Delta(u, a_{i-1}),\ u \in W_{i-1}\}$
    for $i \leftarrow$n downto 0 do
        $W_i \leftarrow W_i \cup \{v \mid (v, L) \in \Delta(u, a_{i+1}),\ u \in W_{i+1}\}$
}
```

Now we will show that this algorithm will finally stop and at the end of this algorithm, we will get our W_i's which satisfy the given conditions.

Proof that the algorithm will finally stop:
Note that the maximum number of states in each W_i will always be atmost |Q|, the number of states in the automata. Also, $0 \leq i \leq n+1$. Thus, the sum of number of states in all the W_is will be atmost $(n+1) \times |Q|$. In each complete iteration of the outer while loop, atleast one of W_is will increase in size. Since the sum of size of all W_is is limited by $(n+1) \times |Q|$, the number of iterations of the outer while loop can also be atmost $(n+1) \times |Q|$. Thus, the algorithm will stop.

Proof that the W_is obtained at the end of the algorithm will satisfy all the 4 given conditions

- $S \subseteq W_0$ since W_0 is initialised to {S}
- If $u \in W_i, 0 \le i \le n$, and $(v, R) \in \Delta(u, a_i)$, then $v \in W_{i+1}$. This will be true at the end of the algorithm because if it is not so, then the first for loop will detect that and make change to any such state and the algorithm would not have stopped at these values of W_is.
- If $u \in W_i, 1 \le i \le n+1$, and $(v, L) \in \Delta(u, a_i)$, then $v \in W_{i-1}$. This will be true at the end of the algorithm because if it is not so, then the first for loop will detect that and make change to any such state and the algorithm would not have stopped at these values of W_is.
- $t \notin W_{n+1}$. This will be true because if t lies in W_{n+1}, then there will be a run of states such that it will land in accept state t and the head(pointer) will point to the right endmarker \$. In that case, it will accept the string x but this is not possible according to the assumption. Thus, t cannot lie in W_{n+1}.

Thus, we proved that if there is a string x that is not accepted by the automata, then there exist W_is (constructed using the above algorithm) that satisfy the given conditions.

Proof of $\Leftarrow$:
Now we prove that if, for a given string x, there exist W_is that satisfy the four given points mentioned in the question, then x is not accepted by A.
We will prove this by contradiction. Let us assume that there is a x, such that there exist W_is that satisfy the four given points mentioned in the question and x is accepted by A. Then, there will be a sequence of configurations

$$(q_0, 0) \rightarrow (q_1, i_1) \rightarrow (q_2, i_2) \rightarrow (q_3, i_3) \rightarrow ...(q_{k-1}, i_{k-1}) \rightarrow (q_k, i_k)$$

such that $q_k = t$ and $i = n+1$. Note that this sequence shows the change of configurations as the input is read in the automata. Here, the automata moves from state q_j to q_{j+1} on reading the i_j^{th} alphabet in the input and its pointer move from i_j to i_{j+1}.

Claim: While reading the input, if configuration of the automata is (q_j, i_j) after taking j steps, then $q_j \in W_{i_j}$.
Proof: We prove the above claim by induction.
Base Case: After taking 0 steps, automata is in configuration $(q_0, 0)$ where $q_0 \in S$. Now, by definition, $S \subseteq W_0$, thus $q_0 \in W_0$.
Induction step: Assume that the claim is true after taking k steps and the configuration is (q_k, i_k) such that $q_k \in W_{i_k}$. Now on reading the i_k^{th} alphabet in the input, the automata moves to state q_{k+1} and the head pointer moves to i_{k+1}. There are two cases possible (i). $i_{k+1} = i_k + 1$ and (ii). $i_{k+1} = i_k - 1$ based on the direction in which the pointer moves.
If the pointer moves right $(i_{k+1} = i_k + 1)$, then $(q_{k+1}, R) \in \Delta(q_k, a_{i_k})$ and thus $q_{k+1} \in W_{i_k+1} \Rightarrow q_{k+1} \in W_{i_{k+1}}$ by definition of W.
Similarly, if the pointer moves to the left $(i_{k+1} = i_k - 1)$, then $(q_{k+1}, L) \in \Delta(q_k, a_{i_k})$ and thus $q_{k+1} \in W_{i_k-1} \Rightarrow q_{k+1} \in W_{i_{k+1}}$ by definition of W.
Thus, by induction we proved that while reading the input, if configuration of the automata is (q_j, i_j) after taking j steps, then $q_j \in W_{i_j}$.

Now, if x is accepted by this language, then the pointer will move all the way to right endmarker and will be in accepting state t. Thus, it will finally reach configuration $(t, n+1)$. By the above claim, $t \in W_{n+1}$.But this contradicts the definition of W_is that $t \notin W_{n+1}$. Thus, there cannot exist any x that is accepted by this automata. Hence proved.

(b). Using the previous part, show that $L(A)$ is regular.
To prove that $L(A)$ is regular, we will prove that its complement ($\overline{L(A)}$) is regular. Then, by closure property of complements, we can say that $L(A)$ is also regular.
We will create a NFA N which recognizes the complement of language $L(A)$. To define the NFA N, we first define the following terms.
Assume that $U, V, W \in 2^Q$ and $c \in \Sigma$. Then

- **#-closed**: Ordered pair (U,V) is called #-closed, if $(q, R) \in \Delta(u, \#)$ where $u \in U$, then $q \in V$.
- **\$-closed**: Ordered pair (U,V) is called \$-closed, if $(q, L) \in \Delta(v, \$)$ where $v \in V$, then $q \in U$.
- **c-closed**: For any $c \in \Sigma$, (U,V,W) is called c-closed, if $\forall v \in V$, $(q, L) \in \Delta(v, c) \Rightarrow q \in U$ and $(q, R) \in \Delta(v, c) \Rightarrow q \in W$.

Now, we will define a NFA N using the above definitions. We define the NFA $N = (Q', S', F', \Delta')$ where

- $Q' = 2^Q \times 2^Q$
- S' = {(X,Y) | (X,Y) is #-closed and $s \in X$}
- F' = {(X,Y) | $F \cap Y = \Phi$ and (X,Y) is \$-closed}
- $\Delta'((X, Y), c)$ = {(Y,Z)| (X,Y,Z) is c-closed}

Now, we will prove that the above NFA N will accept $\overline{L(A)}$.

Any string accepted by N will lie in $\overline{L(A)}$.
Let us assume that string x is accepted by NFA N where

$$x = x_1x_2x_3...x_n$$

. Let the accepting run for this string x be

$$(Q_0, Q_1) \rightarrow (Q_1, Q_2) \rightarrow ... \rightarrow (Q_n, Q_{n+1})$$

where $s \in Q_0$, $(Q_0, Q_1) \in S'$ and $(Q_n, Q_{n+1}) \in F'$.
Claim 1: $Q_0, Q_1, Q_2, ..., Q_{n+1}$ satisfy all the 4 properties mentioned in part 5(a).
Proof for the claim: We will show that all the four properties in part 5(a) are satisfied by sequence of Q_is.

- Note that $(Q_0, Q_1) \in S'$. Thus, $s \in Q_0$ by definition of S'.
- For any $u \in Q_i$, $0 \leq i \leq n$, if $(v, R) \in \Delta(u, a_i)$, then $v \in Q_{i+1}$. This is because $\Delta'((Q_i, Q_{i+1}), c)$ is defined only when (Q_i, Q_{i+1}, Q_{i+2}) is c-closed for $0 \leq i < n$.
- For any $u \in Q_i$, $1 \leq i \leq n+1$, if $(v, L) \in \Delta(u, a_i)$, then $v \in Q_{i-1}$. This is based on the same reason which is used in point two above.

- Note that (Q_n, Q_{n+1}) is an accepting state of the NFA. By definition, $F \cap Q_{n+1} = \Phi$.

Thus, all the four conditions mentioned in 5(a) are satisfied by the sequence of $Q_i s$. Using the result obtained in 5(a), we can say that $x \notin L(A)$. Thus $x \in \overline{L(A)}$.

For any string $x \in \overline{L(A)}$, there is an accepting run the NFA N.
By result of problem 5(a), we can say that if $x \in \overline{L(A)}$, then there exists a sequence of $W_i s$ that satisfy the 4 given conditions in 5(a). For such a sequence of $W_i s$, we can see that

$$(W_0, W_1) \rightarrow (W_1, W_2) \rightarrow ... \rightarrow (W_n, W_{n+1})$$

is an accepting run for x in NFA N. This is because, (W_0, W_1) is $\#$-closed and $S \in W_0$ according to part 5(a) and thus this is an start state in the NFA. Also, note that (W_n, W_{n+1}) is an accepting state according to the definition of F' and properties mentioned in part 5(a). Similarly, we can see that (W_{i-1}, W_i, W_{i+1}) is x_i-closed $(1 \leq i \leq n)$ according to properties 2 and 3 mentioned in part 5(a). Thus, there exist an accepting run for x in NFA N.

Thus, we successfully proved that there is a NFA N for $\overline{L(A)}$. Since, regular languages are closed under complement, L(A) is a regular language.

6.

Let M = (Q, Σ, q_0, δ, F) be a DFA and let h be a state of M called its "home". A synchronizing sequence for M and h is a string s ∈ Σ^* where $\hat{\delta}(q, s)$ = h for every q ∈ Q. Say that M is synchronizable if it has a synchronizing sequence for some state h. Prove that if M is a k-state synchronizable DFA, then it has a synchronizing sequence of length at most k^3. Can you improve upon this bound?

Solution:
We are given the DFA M(Q,Σ,q_0,δ,F) with k states. We assume that such a state as "home" exists, which means that ∃ a string w such that ∀ q ∈ Q, $\hat{\delta}(q, w) = h$. If not, then there is nothing to prove. We need to obtain an upper bound on the length of this string w. We first try to derive a possible candidate of w.

For any set of states S and a string a, define $\Delta(S, a) = \{\hat{\delta}(s, a) | s \in S\}$. It is obvious that $\Delta(S, \epsilon)$ = S for any set S. We want a string w such that $\Delta(Q, w) = \{h\}$, i.e. $|\Delta(Q, w)| = 1$. We construct w as follows:

- Define a sequence of set of states Q_0, Q_1, ... Q_{k-1} such that Q_0 = Q and ∀ 1 ≤ i < k, we find a shortest string w_i such that $\Delta(Q_{i-1}, w_i) = Q_i$, and $|Q_{i-1}|$ - $|Q_i|$ ≥ 1.
- w = $w_1.w_2.w_3......w_{k-1}$

This means that, starting from all the k states of the automata (i.e. Q_0), we iteratively refine the current set of states(say Q_{i-1}) by applying the substring w_i so that the new set of states(i.e. Q_i) has atleast one element less than Q_{i-1}. Then since $|Q_0|$ = k, the size of $|Q_i|$ ≤ k-i. Clearly, $|Q_{k-1}|$ = 1, which is the set containing the home state, since starting from all set of states, we arrived at a single state after processing the string $w_1.w_2.w_3......w_{k-1}$. This state is the home state defined in the question.

Note that we have assumed that Q_i decreases by one after reading string w_i, since we are trying to find the maximum length synchronising string w. And intuitively, such a string would be obtained if we decrease the size of the set of active sets(i.e. Q_i's at the slowest rate) i.e. one state at a time.

Also observe that if Q = Q_0 has a synchronising string 's', then ∃ a synchronising string for its subsets as well. One trivial example is 's' itself. Consequently, all of Q_i's which are subsets of Q will have a synchronising string and thus a "home" state. So, in no point in the subsequent proof do we need to prove the existence of a synchronising string.

We now try to come up with a bound on the length of sub-strings w_i. Then, since w is just the string obtained by concatenating all w_i's, length of w will simply be the sum of their lengths. Consider a particular set of states Q_i and the string $w_{i+1} = a_1a_2....a_n$, where a_i's are letters of the alphabet Σ. Also, consider the sequence of states $P_1.....P_n$ so that

- $P_0 = Q_i$
- ∀ i, 0 ≤ i < n, $P_{i+1} = \Delta(P_i, a_i)$
- $\Delta(P_n, a_n) = Q_{i+1}$

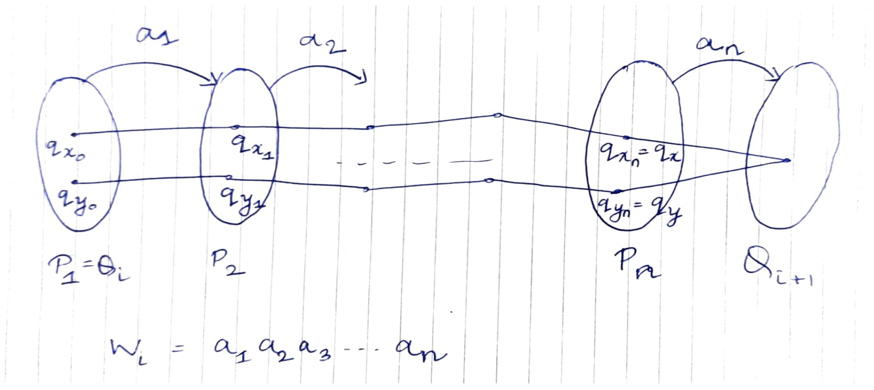
Claim 6.1: $|P_j| = |Q_i|$ ∀ 1 ≤ j ≤ n
This can be proved by contradiction. Suppose not, i.e. ∃ j such that $|P_j| \neq |Q_i|$. Particularly, $|P_j| < |Q_i|$ since we cannot get more states than what we started from, just by applying transitions. But if $|P_j| < |Q_i|$,

then the condition is violated that w_{i+1} is the shortest string such that the set of active states decreases by one after processing it, since we achieved the decrease in number of states after reading just the string $a_1a_2....a_{j-1}$ which is shorter than n. So, we have a contradiction.

From claim 6.1, $|P_n| = |Q_i| = |Q_{i+1}| + 1$. Also, $\Delta(P_n, a_n) = Q_{i+1}$. Then, from Pigeonhole principle (pigeons being the elements of P_n and holes being the elements of Q_{i+1}), $\exists\ q_x,\ q_y \in P_n$ such that $\hat{\delta}(q_x, a_n) = \hat{\delta}(q_y, a_n)$. Now, define two state sequences $Q_x = q_{x_1}, q_{x_2}, ...q_{x_n}$ and $Q_y = q_{y_1}, q_{y_2}, ...q_{y_n}$ such that:

- $\forall\ 1 \leq$ j $\leq$ n, $q_{x_j},\ q_{y_j}\ \in P_j$.
- $\forall\ 1 \leq$ j $<$ n, $q_{x_{j+1}} = \delta(q_{x_j}, a_j)$ and $q_{y_{j+1}} = \delta(q_{y_j}, a_j)$
- $q_{x_n} = q_x$ and $q_{y_n} = q_y$

$q_{x_j} \neq q_{y_j}\ \forall\ 1 \leq$ j $\leq$ n, otherwise $|P_j| < |Q_i|$ which contradicts claim 6.1. All the above arguments can be summarised in the following figure.



Now, using all the above claims, definitions and properties, we obtain a bound on the size of w_{i+1}.

Claim 6.2 The length of w_{i+1} is at most ${}^kC_2$.
This can be proved with the help of Pigeonhole principle. The maximum value of w_{i+1} is decided by the maximum number of sets P_j's that can be derived while following the above conditions. Since, we already proved that $q_{x_j} \neq q_{y_j}\ \forall\ 1 \leq$ j $\leq$ n, they cannot take same values. So, the maximum value of distinct unordered pairs of values that they can take is ${}^kC_2$.
Then, from Pigeonhole principle (with pigeons being the sets P_j and the holes being the unordered values of $(q_x$,$q_y)$), if there are more than ${}^kC_2$ sets P_j's, then $\exists$ two indices p,q such that unordered pairs $(q_{x_p}$,$q_{y_p})$ = $(q_{x_q}$,$q_{y_q})$. In such a case, we can remove the portion between P_p and P_q without changing the result. In other words, the string $w'_{i+1} = a_1a_2....a_{p-1}a_qa_{q+1}...a_n$ would also be such that $\Delta(Q_i, w'_{i+1}) = Q_{i+1}$ and $w'_{i+1} < w_{i+1}$, which contradicts the fact that w_{i+1} is the shortest string obeying the above constraint. Thus, the number of sets P_j and thus the length of the string w_{i+1} is at most ${}^kC_2 = \left(\frac{k^2-k}{2}\right)$.

Now, from claim 3.2, and using w = $w_1.w_2.w_3......w_{k-1}$,
|w| $\leq (k-1) * ({}^kC_2) = \left(\frac{k^3-2k^2+k}{2}\right)$.

$k \geq 1 \implies$
$k^3 + 2k^2 - k > 0 \implies$
$k^3 - 2k^2 + k < 2 * k^3 \implies$

$|\text{w}| \leq \left(\frac{k^3-2k^2+k}{2}\right) < k^3$

Hence Proved. Note that $\left(\frac{k^3-2k^2+k}{2}\right)$ is a stricter bound.

7.

For every string x $\in \{0,1\}^+$ consider the number

$$0.\text{x} = \text{x}[1].\left(\tfrac{1}{2}\right) + \text{x}[2].\left(\tfrac{1}{2^2}\right) + ... + \text{x}[|\text{x}|].\left(\tfrac{1}{2^{|x|}}\right)$$

where |x| is the length of x. For a real number $\theta \in$ [0, 1] let

$$L = \{x : 0.x \leq \theta\}$$

Prove that L_θ is regular if and only if θ is rational.

Solution:
($\Rightarrow$)
L_θ is regular $\implies$ θ is rational
We will prove this by contradiction. Let us assume that L_θ is regular but θ is irrational for some theta. Let the binary representation of theta be

$$\theta = 0.\theta_1\theta_2\theta_3\theta_4....$$

Since theta is irrational, the binary representation of theta is non-repeating infinite. If L_θ is regular, then there is myhill-nerode relation with finite number of equivalent classes. Since the no. of classes are finite, there will exist atleast two prefixes of theta $\theta_1\theta_2$ ...θ_i and $\theta_1\theta_2$ ...θ_k, i<k, which belong to the same equivalent classes. Now, start comparing θ_{i+j} with θ_{k+j} (j>0) until you find the smallest j (assume that smallest such j is m) such that $\theta_{i+m} \neq \theta_{k+m}$ (such m will always exist since theta is irrational).
Now, if $\theta_{i+m} = 0$ and $\theta_{k+m} = 1$, then consider $x = \theta_1\theta_2 ...\theta_i\theta_{k+1}\theta_{k+2}...\theta_{k+m}$ and $y = \theta_1\theta_2 ...\theta_k\theta_{k+1}\theta_{k+2}...\theta_{k+m}$. Now, we can see that $x > \theta$ and $y \leq \theta$. Thus, $x \notin \theta$ and $y \in \theta$, but this violates the right congruence and refinement property of myhill-nerode relation since $\theta_1\theta_2$ ...θ_i and $\theta_1\theta_2$ ...θ_k belonged to the same equivalent class.
Similarly, if $\theta_{i+m} = 1$ and $\theta_{k+m} = 0$, then consider $x = \theta_1\theta_2 ...\theta_i\theta_{i+1}\theta_{i+2}...\theta_{i+m}$ and $y = \theta_1\theta_2 ...\theta_k\theta_{i+1}\theta_{i+2}...\theta_{i+m}$. In this case, $y \notin \theta$ and $x \in \theta$.
Thus, in both the cases, we proved that it is not possible to have finite number of equivalent classes. Thus, the language cannot be regular.

($\Leftarrow$)
θ is rational $\implies$ L_θ is regular
Assume that θ is rational, then its binary form can be represented in either terminating or non-terminating but recurring form. Let the general representation of θ be $0.\alpha_1\alpha_2....\alpha_x\overline{\beta_1\beta_2....\beta_y}$. If the representation is terminating, then y = 0, otherwise the beta part is recurring. So, this can be a representation of a rational number.

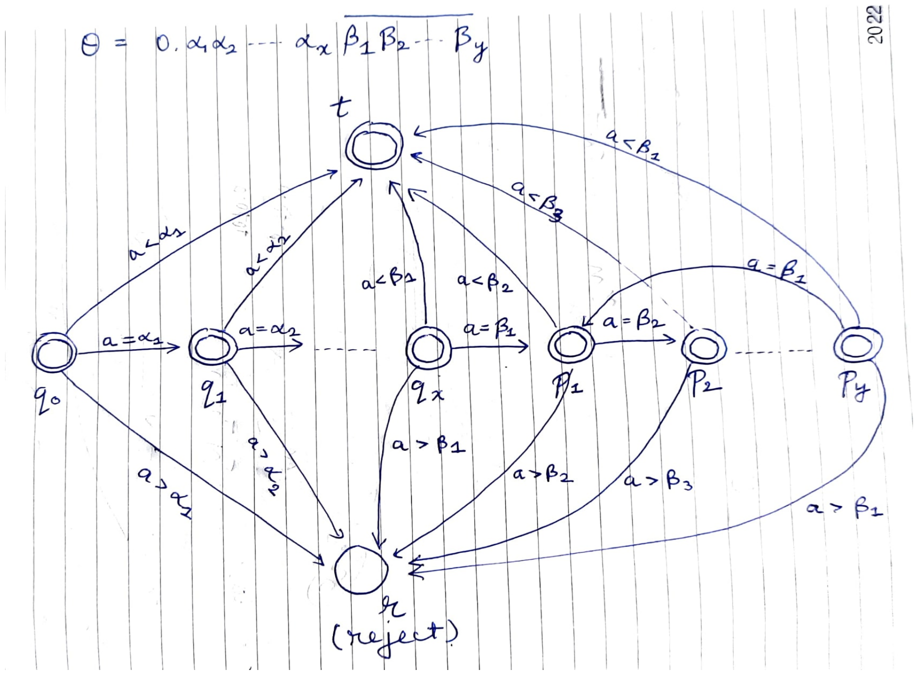
We show that L_θ is regular by constructing an DFA D(Q,Σ,q_0,δ,F) for it as follows:

- Q = $\{q_0, q_1, ..., q_x, p_1, p_2, ...p_y, r, t\}$ where q_i's are states corresponding to the α part while p_i's are states corresponding to the β part. r is the reject state while t is an additional accept state.
- $\Sigma = \{0, 1\}$
- q_0 is the start state.

- F = Q $\setminus \{r\}$
- Define δ as:

  1. $\forall\ 0 \leq$ i $<$ x
     $$\delta(q_i, a) = \begin{cases} r & a > \alpha_{i+1} \\ q_{i+1} & a = \alpha_{i+1} \\ t & a < \alpha_{i+1} \end{cases}$$
  2. $$\delta(q_x, a) = \begin{cases} r & a > \beta_1 \\ p_1 & a = \beta_1 \\ t & a < \beta_1 \end{cases}$$
  3. $\forall\ 1 \leq$ i $<$ y
     $$\delta(p_i, a) = \begin{cases} r & a > \beta_{i+1} \\ p_{i+1} & a = \beta_{i+1} \\ t & a < \beta_{i+1} \end{cases}$$
  4. $$\delta(p_y, a) = \begin{cases} r & a > \beta_y \\ p_1 & a = \beta_y \\ t & a < \beta_y \end{cases}$$
  5. $\delta(t, a) =$ t $\forall$ a $\in \Sigma$
  6. $\delta(r, a) =$ r $\forall$ a $\in \Sigma$

This can be summarized in the following figure



Now, we prove that D recognizes the language L_θ. For that we need to prove both directions of:

$$w \in L_\theta \iff \text{D accepts w}$$

(⇒)
Assume that $w \in L_\theta$. Take $\theta = 0.\alpha_1\alpha_2....\alpha_x\overline{\beta_1\beta_2....\beta_y}$. Then, if $w = w_1w_2.....w_n$, $w_i \leq \alpha_i\ \forall\ 1 \leq i \leq x$ and $w_i \leq \beta_{(i-x-1)\ mod\ y+1}$ otherwise, from the definition of L_θ. Given this, we need to prove that w is accepted by the DFA. It is sufficient to prove that the DFA never reaches the reject state. For that, we prove a stronger claim using induction on the length of the input string using the following induction hypothesis:

Claim 7.1: H(i): After reading the string $w_1w_2...w_i$, the DFA is either in t, or in state q_i if $0 \leq i \leq x$ and in state $p_{(i-x-1)\ mod\ y+1}$ otherwise.

Proof of claim:

- **Base Case:** i=0 i.e. when nothing is read. Then our claim is vacuously true, since our DFA is in the start state i.e. q_0.
- **Induction Step:** Assume that H(i) holds. Then we prove H(i+1). Since H(i) holds, there can be various cases after reading the string $w_1w_2....w_i$:
  1. DFA is in state t. Then, after reading w_{i+1} too, DFA will be in state t. This is because $\delta(t, a) =$ t ∀ a ∈ Σ. So, H(i+1) holds.
  2. DFA is in state q_i. Then i ≤ x. Then, if i < x, then after reading w_{i+1}, DFA is in state q_{i+1} if $w_{i+1} < \alpha_{i+1}$ and in state t if $w_{i+1} = \alpha_{i+1}$. This is consistent with the definition of δ. Note that $w_{i+1} > \alpha_{i+1}$ is not possible since $w \in L_\theta$. On the other hand, if i=x, then i+1 = x+1 > x and $(i-x-1)\ mod\ y+1 = 1$. This means that if $w_{x+1} < \beta_1$, then the DFA will move to the state p_1 or to state t if $w_{x+1} = \beta_1$. T(i+1) holds.
  3. DFA is in state $p_{(i-x-1)\ mod\ y+1}$. Then i > x. So, i+1 > x. Then from the defintion of δ, the DFA moves to state t if $w_{(i-x)\ mod\ y+1} = \beta_{(i-x)\ mod\ y+1}$ or it moves to $p_{(i-x)\ mod\ y+1}$ if $w_{(i-x)\ mod\ y+1}$ $< \beta_{(i-x)\ mod\ y+1}$. Again, only two cases are considered since $w \in L_\theta$. And from the above cases, H(i+1) holds.

Thus, H(i) holds for all i ≥ 0. This means that after reading any string $w \in L_\theta$, our DFA never reaches the reject state. So, w is accepted by the DFA.

(⇐)
Assume that the DFA accepts a string w = $w_1w_2...w_n$. Then we need to prove that $w \in L_\theta$. Since D accepts w, then D can not be in reject state after reading any prefix of the string w. From this, there are two possible cases:

1. The final state is t. In this case, ∃ i such that $w_i < \alpha_i$ if i ≤ x or $w_i < \beta_{(i-x-1)\ mod\ y+1}$ otherwise and ∀ j < i, $w_j = \alpha_j$ if j ≤ x or $w_j = \beta_{(j-x-1)\ mod\ y+1}$ otherwise. From this case, the number $0.w_1w_2....w_n < 0.\alpha_1\alpha_2....\alpha_x\overline{\beta_1\beta_2....\beta_y}$. So, 0.w < θ. So, w ∈ L_θ.
2. The final state is not t. In this case, ∀ 1 ≤ j ≤ n, $w_j = \alpha_j$ if j ≤ x or $w_j = \beta_{(j-x-1)\ mod\ y+1}$ otherwise. In this case, the number $0.w_1w_2....w_n \leq 0.\alpha_1\alpha_2....\alpha_x\overline{\beta_1\beta_2....\beta_y}$ (equality is possible when θ has a terminating binary representation. So, 0.w ≤ θ. So, w ∈ L_θ.

So, from both the cases in which DFA D can accept w, we obtained that w ∈ L_θ.

Hence, we proved both the directions of the proof. This proves that the DFA D recognizes the language L_θ.