COL-352 ASSIGNMENT

DEEPANSHU YADAV(2019CS10343)
SOURAV(2019CS10404) HARIKESH(2019CS10355)

January 2022

1 Problem

To Prove: $L_1 = \{bin(p) : p$ is a prime number$\}$ is not a regular language.
Proof:
Theorem used: Here we will use the refined pumping lemma which states that any language L is not regular if $\forall n\ \exists$ words uwv such that $uwv \in L$ and $|w| \geq n$ and for each breakup of w into three words $xyz = w$ such that $y \neq \epsilon$ and $|xy| \leq n$ then $\exists k \geq 0$ such that $uxy^kzv \in L$.
Assumption: Assume that $L_1 = \{bin(p)|p \in$ prime number $\}$ is regular. Let's assume than its accepted by a DFA D with k states.
Now let's take any $n \geq 0$,
By Dirichlet's theorem we know that 10,1 are co-prime and hence there are infinitely many prime number of form

$$w = \underbrace{1..............}_{\text{n(binary form)}}\underbrace{00........}_{\text{k 0's}}1$$

as for some $s > 0$ it is in the series $10s + 1$.
So,for any $n \geq 0$ our chosen word is

$$w = \underbrace{1............}_{\text{n(binary form)}}\underbrace{00........}_{\text{k 0's}}1$$

$$= 2^{k+1}n + 1$$

Now by our extended pumping lemma,
we will just break this w in parts xyz such that

$$\underbrace{\qquad\qquad}_{\text{n(binary form)}}\underbrace{00........}_{\text{m 0's}}1$$

where

$$y = \underbrace{00........}_{\text{m 0's}}$$

and

$$|y| \geq n\{\text{ because we can choose any number of zeros i.e } m \text{ can have be } n\}$$

Now just split y in abc such that for each breakup we have $|b| \geq 1$ and $|ab| \leq m$.
Let's say $a = 0^i, b = 0^j$ where $j \neq 0$ and $i + j \leq m$
which on pumping gives

$$w' = \underbrace{\qquad\qquad}_{\text{n(binary)}}\underbrace{0..(0^j)^k......}_{\text{m+(k-1) 0's}}1$$

is prime for all k

$$w' = (w-1)2^{(k-1)j} + 1$$
$$\Rightarrow 2^{(k-1)j}w - (2^{(k-1)j} - 1)$$

has to be prime $\forall k \geq 1$
Now let's say $k = w$
$\Rightarrow$ By fermat's theorem we know that for any prime p, $2^{p-1} \equiv 1(modp)$
$\Rightarrow$ Our 2^{nd} part of w' becomes
$\Rightarrow$ $2^{(w-1)j} - 1$ and this is divisible by w {By fermat's little theorem}
Hence in w' we have

$$w|2^{(k-1)j}w$$

also

$$w|2^{(w-1)j} - 1$$

Hence $w|w'$ and contradiction that w' is prime .
Ore assumption was wrong.
Which proves that L_1 is not a regular language.

2 Problem

Given:
$F_n = F_{n-1} + F_{n-2}$ $\forall$ n ≥ 3
$F_1 = 1, F_2 = 1$
We have to comment on the regularity of the language $L_2 = \{a^m | m = F_n\}$ over $\Sigma = \{a\}$
Assumption: Assume that the language $L_2 = \{a^m | m = F_n\}$ is regular.Then by pumping lemma, for each x, $x > 0$ let's take a word w such that $|w| \geq x$ & $w \in L_2$.
Now we have to break w in pqr where $|pq| \leq x$ and $|q| \geq 1$.
Let $pqr = w$ such that $|pq| \leq x$ & $|q| \geq 1$.
Now we can always take two consecutive fibonacci number such that both of them are $> x$ i.e. $F_{r-1} > x$ & $F_r > x$.
Now, let's the word $w = a^{F_r}$ and $a^{F_r} \in L_2$ and also $|a^{F_r}| \geq x$.
Now for any $i \geq 0$, pq^ir on taking $i = 2$ gives pq^2r
$\Rightarrow |pq^2r| = |pqr| + |q| = F_r + |q| \leq F_r + x$ (2.1)
as $|pq| \leq x \Rightarrow |q| \leq x$
Also, $|pq^2r| > |pqr|$ which $\Rightarrow |pq^2r| > F_r$ as $q \notin \epsilon$ (2.2)
We know that the next smallest string having length $> F_r$ which can belong to L_2 is a $a^{F_{r+1}}$ which has length F_{r+1}
From (2.1) and (2.2)

$$F_r < |pq^2r| \leq F_r + x < F_{r+1} \quad (2.3)$$

because,

$$F_{r-1} > x$$

$$\Rightarrow F_r + F_{r-1} > F_r + x$$

$$\Rightarrow F_{r+1} > F_r + x$$

Hence, $pq^2r \notin L_2$ (from 2.3)
Hence contradiction
So,our initial assumption was wrong L_2 is not regular.

3 Problem

Given: $A_{\frac{1}{2}-} = \{x|$ for some y s.t $|x| = |y|$ xy is in $A\}$
To Prove: If A is regular then so is $A_{\frac{1}{2}-}$
Proof: Let A is regular language $\Rightarrow$
There is a DFA $D(Q, \Sigma, q_0, \delta, F)$ accepts A where q_0 is the start state, F are the accepting states,Q is the set of all states,δ is transition function.
Now let's we have a string x.Now for string $x \in A_{\frac{1}{2}-}$ we have to check if there is a string y of same length as x such that xy reaches the accepting states of D.
Let's say D reaches to state q_i on taking string x as input i.e. $\delta(q_0, x) = q_i$ then basically we will be checking of a string y such that
$|x| = |y|$ and $\delta(q_i, y) \in F$ (3.1)
Let's define some notations:-
Let's say $|x| = n$ and S_n is the set of all states that lead to final state on taking some input of length n (3.2)
Claim: We can compute S_n inductively.
Base-Case: $S_0 = F$
Now, let's say we have S_n for some n then we will try to obtain S_{n+1} from S_n and S.
Let T be set of states having transition to a state in S_n i.e $T = \{q|((q, a) \in S_n$ for some $a \in \Sigma\}$
Now this $T \equiv S_{n+1}$ because for each state $q \in S_n$ there is some input w of length n such that $\delta(a, w) \in F \Rightarrow$ there is some input let's say w' of length $n+1$ foro each state $q' \in T$ such that $\delta(q', w') \in F$ thus $T = S_{n+1}$
Now, we can say that from (3.1) & (3.2) that if $q_i \in S_n$ then $x \in A_{\frac{1}{2}-}$ and vice versa. (3.3)
Let's construct a DFA D' that stores both q_i and S_n and then will accept $A_{\frac{1}{2}-}$ i.e DFA $D' = (Q', \Sigma', \delta', q_0', F')$ where $Q' = Q * 2^Q$ where Q is srt of states of D.
Where each state contains a pair of single state and set of states of D.

$$\Sigma' = \Sigma$$

$$q_0' = (q_0, s_0) \text{ i.e } (q_0, F)$$

$$F' = \{(q, s)|q \in Q, S \in 2^Q, q \in S\}$$

Now, formally we have
$\forall S \in S^Q$,previous(S) = $\{q \in Q|\exists a \in \Sigma, q' \in S, \delta(q, a) = q'\}$
Then our transition function is defined as :
$\delta'((q, s), a) = (\delta(q, a), previous(S))$
Claim: $\delta'(q_0', x) = (\delta(q_0, x), S_n)$ where $|x| = n$ we will try to prove it inductively.
Base Case: $x =$ some $a \in \Sigma$
$\delta'((q_0, s_0), a) = \delta((q_0, a), S_1)$ where let's say $\delta(q_0, a)$ be $q_1 \Rightarrow (q_1, S_1)$
Now let's say after reading n-1 symbols we reach at $(\delta(q_0, x'), S_{n-1})$
Now on reading the n^{th} symbol it gives $\delta'((\delta(q_0, x'), S_{n-1}), a_n) = (\delta(q_0, x'a_n), S_n) =$

$(\delta(q_0, x), S_n)$
And from (3.3) we know that x is accepted iff $q_i = \delta(q_0, x) \in S_n$.Thus D' accepting $A_{\frac{1}{2}-}$.Hence we have a DFA accepting $A_{\frac{1}{2}-}$.Thus $A_{\frac{1}{2}-}$ is regular.

Proof of Correctness: We have to prove that $L(D') = A_{\frac{1}{2}-}$
Let's start by proving $L(D') \subseteq A_{\frac{1}{2}-}$
Consider $x \in \Sigma^*$ such that $\delta'(q'_0, x) = (\delta(q_0, x), S_n)$ where $|x| = n$.
Then we have to show that $q'_i \in S_n \Rightarrow x \in A_{\frac{1}{2}-}$
where q_i is the state reached by D after reading x i.e $q_i = \delta(q_0, x)$
As, $q_i \in S_n$ implies that $\exists$ a string y such that $|y| = n$ and $\delta(q', y) \in F \Rightarrow \exists$ string xy such that $|x| = |y| \Rightarrow x \in A_{\frac{1}{2}-}$.
Also $L(D')$ accepts only in a state (q, s) if $q \in S \Rightarrow$ in state (q_i, s) it will accept
So, $L(D') \subseteq A_{\frac{1}{2}-}$
Now on proving $A_{\frac{1}{2}-} \subseteq L(D')$
We will start by saying that any word in language of $A_{\frac{1}{2}-}$ is x such that $\exists$ another word xy where $|x| = |y|$ and $xy \in A$ we will now show that $x \in A_{\frac{1}{2}-} \Rightarrow q_i \in S_n$.
As $x \in A_{\frac{1}{2}-}$ means that on reading x there must be another string xy such that $|x| = |y|$ and $|y| = x$.Therefore $\delta(q_i, y) \in F$ and hence q_i must belong to S_n
Also, language of D' i.e $L(D')$ involve only words in a state $(\delta(q_0, x), S_n)$ when $\delta(q_0, x))$ i.e $q_i \in S_n$.So, $A_{\frac{1}{2}-} \subseteq L(D')$.
Hence our proof of correctness conclude $A_{\frac{1}{2}-} = L(D')$

4 Problem

Given: For any languauge A,

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

Prove: A is regular then $A_{\frac{1}{3}-\frac{1}{3}}$ is not necessarily regular.
Proof: Let $A = \{0^*(2)1^*\}$ and it is regular because there is a DFA of finite states accepting it.
Claim 1:

$$A_{\frac{1}{3}-\frac{1}{3}} \cap \{0^*1^*\} = \{0^n1^n|n \geq 0\} \qquad (4.1)$$

Proof: (i)

$$A_{\frac{1}{3}-\frac{1}{3}} \cap \{0^*1^*\} \subseteq \{0^n1^n|n \geq 0\}$$

We know that $A_{\frac{1}{3}-\frac{1}{3}}$ is only possible in words of A which has length divisible by 3 i.e Odd length.
The case when 2 lies in a word of $A_{\frac{1}{3}-\frac{1}{3}}$ i.e string of type $0^p21^c \cap \{0^*1^*\} = \epsilon$ for some $p, c \geq 0$ which is $\subseteq \{0^n1^n|n = 0\} \subseteq \{0^n1^n|n \geq 0\}$
The case when 2 doesn't lie in the words of $A_{\frac{1}{3}-\frac{1}{3}}$ i.e. 2 is the middle $\frac{1}{3}^{rd}$ part then we know that in $\{0^*21^*\}$ strings of odd length with 2 in middle $\frac{1}{3}^{rd}$ part i.e $0^x2^11^{3n-x-1}$ for some n and x
if 2 lies in middle part then
$n+1 \leq x+1 \leq 2n$
$\Rightarrow n \leq x \leq 2n-1$
And hence on removing the middle part i.e $n+1$ to $2n$ region we can say that when $x = n$
$0^n21^{2n-1} = 0^n21^{n-1}1^n = 0^n1^n$ for some n
and when x = 2n-1
$0^{2n-1}21^n = 0^n0^{n-1}21^n = 0^n1^n$ for some n
Hence any immediate middle value just pushes the 2 by right 1 step and 2 on eliminating only gives 0^n1^n for some n.
Hence when 2 doesn't lie in words of $A_{\frac{1}{3}-\frac{1}{3}}$

$$A_{\frac{1}{3}-\frac{1}{3}} \cap \{0^*1^*\} \subseteq \{0^n1^n|n \geq 0\} \qquad (4.2)$$

(ii)

$$\{0^n1^n|n \geq 0\} \subseteq A_{\frac{1}{3}-\frac{1}{3}} \cap \{0^*1^*\}$$

Now taking any words form L.H.S. language let's say 0^x1^x for some $x \geq 0$
Now we will form a word $w = 0^x21^{x-1}1^x$
Now $A_{\frac{1}{3}-\frac{1}{3}}$ will contain 0^x1^x from A as w is in A i.e $\{0^*21^*\}$ and $0^x1^x \cap \{0^*1^*\} = 0^x1^x$
Hence, 0^x1^x is in $A_{\frac{1}{3}-\frac{1}{3}} \cap \{0^*1*\}$ for any $x \geq 0$
Hence $\{0^n1^n|n \geq 0\} \subseteq A_{\frac{1}{3}-\frac{1}{3}} \cap \{0^*1^*\}$ (4.3)
Hence our claim 1 holds form (4.2) & (4.3)
Claim2: $L = \{0^n1^n|n \geq 0\}$ is not regular (4.4)

Proof: Let L is regular
Then for any $n \geq 0$, $w = a^n b^n, |w| \geq n$
Now we will form the breaks $xyz = w$ such that $y \neq \epsilon$ and $|xy| \leq n$
So, breaks formed from 0^n part.
Let $x = 0^i, y = 0^j$ such that $i + j \leq n$ and $j \neq 0$
$w = 0^i 0^j 0^{n-i-j} 1^n$
On pumping the y we get $w = 0^i (0^j)^k 0^{n-i-j} 1^n$
let's take $k = 0$
$w = 0^{n-j} 1^n \notin L$ as $j \neq 0$
Hence our assumption is wrong $\Rightarrow L$ is not regular.
Now we know that regular languages are closed under intersection and from (4.1) 0^*1^* is regular and $\{0^n 1^n | n \geq 0\}$ is irregular $\Rightarrow A_{\frac{1}{3}-\frac{1}{3}}$ is irregular than $\{0^n 1^n | n \geq 0\}$ must also have been regular.
Hence if A is regular, $A_{\frac{1}{3}-\frac{1}{3}}$ is not necessarily regular.

5 Problem

Given: 2 NFA $A = (Q, s, t, \Sigma, \Delta)$ where Q is set of states s the set of start states,t is an acceptable states

$$\Delta : Qx(\Sigma \cup \{\#, \$\}) \to 2^{Qx\{L,R\}}$$

when A accepts it moves pointer all the way right end maker \$ and enters accepts state (5.1)
(a) $x = a_1....a_n \in \Sigma^*, a_i \in \Sigma, 1 \le i \le n, a_0 = \#, a_{n+1} = \$$
To prove this iff statement i.e x is not accepted by A iff $\exists$ sets $w_i \subseteq Q$,$0 \le i \le n+1$ such that
(i) $s \subseteq w_0$
(ii) If $u \in w_i, 1 \le i \le n+1$ and $(V, R) \in \Delta(u, a_i)$ then $v \in w_{i+1}$
(iii) If $v \in w_i, 1 \le i \le n+1$and $(V, L) \in \Delta(u, a_i)$ then $v \in w_{i-1}$ and
(iv) $t \in w_{n+1}$
We will first try to prove (i)+(ii)+(iii)+(iv) $\Rightarrow x$ is not accepted by A.
Let the input string we have is $\#x\$$ where $x = a_1...a_n$
From(i) There is only one accepting state t and it accepts only after moving to endmarker \$
Now from (i) $s \subseteq w_0$, So we start from w_0 and we can see that for accepting we have to move either left or right on the input string at any state.
We will now make a claim that on all the possible states after coming back and forth must lie in kind of an equivalence class form (i) and (ii) we have let's say we are having some state p in w_i which gives on reading a_i it gives let's say (V, R) then by (ii) $v \in w_{i+1}$
The other possible way is to get to w_{i-1} then back to w_i and then to w_j only if possible.
In that case we have such k, l, m sch that $k \in w_{i-1}, l \in w_i$ and $m \in w_{i+1}$.
here we can conclude that we have to read the whole string atleast once going right till \$ and even if we move left,right somewhere we will still get the states belonging to same w_i.
Thus form (ii) on reading all words from a_s to a_n and thus by induction we end in a state let's say (X, R). Now, we know that $x \in w_{n+1}$ by induction.
Proof:
Base Case: $i = 0, \Delta(u, a_0) = (V, R)$ where $V \in W_{i+1}$ Now on reading any i we are at

$$\Delta(u, a_i) = (V', R)$$

where $V' \in w_{i+1}$ on transiction from V' gives $\Delta(V\, a_i)$ will definitely going to end in a state of w_{i+1}.
Hence $t \in w_{n+1}$ definitely if it has to accepts But by (5.1) $t \in w_{n+1}$ Hence contradiction
Now we will prove if x is not accepted by A then $k \subseteq Q$ such that (i) (ii) (iii) (iv)
We know that if x is not accepted by A then after reading the whole input i.e

$\#x\$$ we end up in a state which is not final.
Proof by Contradiction: We will prove this using contradiction.
Let's assume that $w_i \subseteq Q$ satisfying (i) (ii) (iii) (iv) If (i) is not there then let's say $s \subseteq w_n$ then we can use the fact that (ii) and (iii) will allow to move left and right on take while changing and let's say after reading whole input we reach state $t \in w_{2n+1}$ then it will be accepted as $w_{2n+1} \neq w_{n+1}$
If (ii) or (iii) are not there then there is no such concept of equivalent state in a particular w_i Then we will be able to jump to w_k form w_i where $|k-1| \geq 2$ on reading a single $a \in \Sigma$.Hence in such a condition $t \in w_x$ for any $x \geq 0$ and hence string can be accepted if $x \neq n+1$
If (iv) is not true i.e doesn't exist for any w then by the first three we finally reach at w_{n+1} and $t \in w_{n+1}$ and will be accepted.
Hence contradiction if x is not accepted i.e. then $\exists w_i \subseteq Q$ such that (i) (ii) (iii) (iv) all hold.
Part B:
We have to show that $L(A)$ is regular.
T prove a language is regular we have to build a BFA/DFA accepting it Here we will build an NFA where states are subsets of Q i.e w_i's and our NFA N will gives the sets w_i. we define N as $N = (Q', s', \delta', F', \Sigma')$ such that where

$$Q' = Qx2^Q$$

$$s' = sx2^Q$$

$$\delta'(Q, 2^Q, (a, R)) \rightarrow (\delta(Q, a), q^Q$$

such that if 2^Q is a subset of w_{i+1}
Similarly

$$\delta'((Q, 2^Q), (b, L)) \rightarrow (\Delta(Q, b), 2^Q)$$

such that id $2^Q \in w_i$ then $2^{Q'}$ is subset of w_{i-1}

$$F' = (Q, 2^Q)$$

such that $Q \in t$ and 2^Q is a subset of w_{i+1}

$$\Sigma' = \Sigma$$

Prove: $L(N) = L(A)$
for this we will first show that $L(N) \subseteq L(A)$ as in $L(N)$ we are accepting states only when $Q \in t$ and $2^Q \subseteq w_{n+1}$ i.e $(Q, 2^Q) \in F$ only have elements of w_{n+1} and t also. Hence $L(N) \subseteq L(A)$
for the other side
$L(A) \subseteq L(N)$ for this the language of A has transition words such that $(V, R) \in \Delta(u, a_i) \Rightarrow V \in w_{i+1}$ if $u \in w_i$
$(V, L) \in \Delta(u, a_i) \Rightarrow V \in w_{i+1}$ if $u \in w_i$
Hence $L(A)$ will accepts only when $t \in w_{n+1}$ and in our $L(N)$ we are having same transitions just that with state we are also having the w_i it belongs to. hence for some j if $\delta'(s', w)$ will reach then w_{i+1} along with that one particular

state and F is state when $Q \in t$ and $2^Q \in w_{n+1}$
Hence $L(A) \subseteq L(N)$
Thus $L(A) = L(N)$ for this case.
Hence we can say that $L(A)$ is regular as we have a finite automata accepting it.

6 Problem

Given: $M = (Q, \Sigma, q_0, \delta, F)$ is a DFA.
To Prove: If M is a k-state synchronizable DFA then it has a synchronizable sequence by length k we have to also improve on this bound.
Proof: Let's say the synchronizing sequencr be $w = \epsilon$(Empty string) and let a set of states P such that every state in Q can reach in P.
Initially, we have taken $P = Q$
Now while the number of states is P are more than 1 each time we remove atleast one of the state by taking a word $w' \in \Sigma^*$ such that the number of states left after reading w' on P are less than number of states in P i.e. $|\delta(P, w')| < |P| \Rightarrow$ that in such a case we have assumed that minimum of 2 states are synchronized.Hence we change our set of states to $\delta(P, w')$ i.e $P = \delta(P, w')$ after one iteration.
If this one iteration we are going to append the word w' to w.This way finally we get our synchronizing sequence.
As our DFA has k states the iterations taken by the above process at max $(k-1)$ (as each time one of the states is removed form P.)
Now to finally find the length of synchronizable sequence we have to find the length of w which is intermediate word.
Now at any time let's say we have $|P_l| = n$ where $n > 1$ and let's say we have our word be w.Now since $|\delta(P_l, w')| < |P_l|$ by the condition which was chosen while chosing w'.
Hence by pigeonhole principle there exist two states let's say $s, t \in P_l$ such that $\delta(s, w') = \delta(t, w')$
Claim: In such two states such that if there exists a word w' such that $\delta(s, w') = \delta(t, w')$ then the length of such word w' is atmost k^2.
Proof: In our case we are having $\delta(s, w') = \delta(t, w') = m$
Also,let's say $|w| = k$ and $w = a_1 a_2 ....a_k$
So,there can be loop while synchronizing two states s, t when $\delta((s_j, tj), a_{j+1}) = (s_i, t_i)$ where $i \leq j$
Now, both of these states s, t will run in parallel and we have assumed the order of running as $ss_1s_2.....s_i$ and $tt_1t_2...t_i$
So,every transaction will give us a pair of synchronized state computation i.e. (s_i, t_i) (s_l, t_l)
Total number of such states pairs is k^2 and by pigeon hole we know that if $|w'| > k^2$ then a pair of states repeat and we can actually shrink w' such that its length is $<k^2$
Hence our claim holds.
Thus it has a synchronizable sequence of length atmost $(k-1)k^2$ which is bounded by k^3
Now we have already found a better bound which is $k^3 - k^2$
We can also try to lower its bound by more by seeing the fact that the length of word w' can be more fined to ${}^kC_2$ instead of k^2 as at every point we have to choose just 2 states out of total k states and here the same state can't be reached before as then it will sync even before the word.
So, the bound can be further lowered to $\frac{k(k-1)^2}{2}$

7 Problem

Case I: θ is rational
As $0 \leq \theta \leq 1$ and θ is rational, hence θ can be written as $\theta = (\frac{a[1]}{2}) + (\frac{a[2]}{2^2}) + (\frac{a[3]}{2^3}) \ldots\ldots\ldots (\frac{a[m]}{2^m})$ for some finite natural number m where $a[i] \in \{0, 1\}$ $\forall$ $0 < i \leq m$.
It can also be written as $\theta = 0.A$ where A is of finite size $m, |A| = m$
To Prove: L_θ is regular
Proof: We need to create a set of all the string $0.X$ such that $X \leq A$. As we know that $\theta = 0.A$
To know whether θ is greater than or less than X, we need to compare at most $min(|X|, |A|)$ digits of θ and X $Min(|X|, |A|) \leq |A| = m$
Hence we need to compare at most m initial values of a string to know whether it is less than $|X|$ or not. The digits afterwards do not matter. Hence we can create a Myhill-Nerode Equivalence class with max $2m$ states (states with same first m digits go to the same final state).
As we can find a finite Myhill-Nerode Equivalence class, Hence the language is regular
. **Case II:** θ is irrational
Then there exist no string a such that $\theta = (\frac{a[1]}{2}) + (\frac{a[2]}{2^2}) + (\frac{a[3]}{2^3}) \ldots\ldots\ldots (\frac{a[m]}{2^m})$ for some finite natural number m where $a[i] \in \{0, 1\}$ $\forall$ $0 < i \leq m$.
To Prove: L_θ is non regular language
Proof:
We can prove this if we can prove that there is no Myhill Nerode Equivalence class with finite number of elements Suppose $\theta = (\frac{a[1]}{2}) + (\frac{a[2]}{2^2}) + (\frac{a[3]}{2^3}) \ldots\ldots\ldots$ till infinite
Choose B and C to be equivalent
Let $X = (B[1]/2) + (B[2]/2^2) + (B[3]/2^3) \ldots\ldots\ldots (B[p]/2^p) = 0.B$
$Y = (C[1]/2) + (C[2]/2^2) + (C[3]/2^3) \ldots\ldots\ldots (C[q]/2^q) = 0.C$
Where p and q are finite values and $p < q$
Here we can see that $X \leq \theta$ as θ is X + something non-negative And $Y \leq \theta$
Hence $X \in L_\theta$ and $Y \in L_\theta$
As B and C are equivalent
$[B] = [C]$
$[B][A[q+1] \ldots\ldots A[r]] = [C][A[q+1] \ldots\ldots\ldots\ldots A[r]] = [D]$
Now it can be observed that $0.D$ is less than θ because D consists of r starting values of A.
Hence $0.D \in L_\theta$
But the same thing may not be true for $0.[B][A[q+1] \ldots\ldots A[r]]$ as its value may be higher than θ for some particular case as the length of a is infinite.
So, $0.[B][A[q+1] \ldots\ldots A[r]]$ may not be present in L_θ.
Hence our assumption that B and C are equivalent is not true. Hence there would be infinite number of equivalence classes. In no way there can be finite number of equivalence classes. Hence L_θ is not a regular language.