90/90

UMC 205

ASSIGNMENT - 01

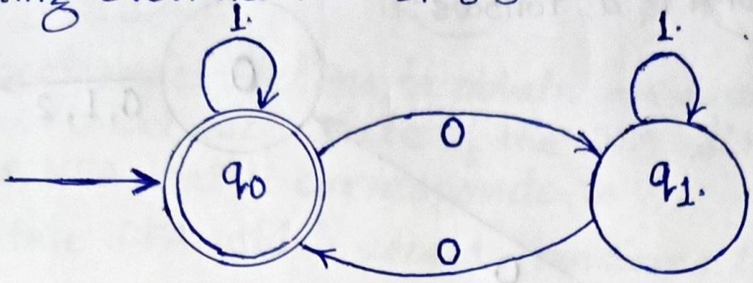
Piyush Kumar
(23801)

February 3, 2025

Q1. Give a DFA for the language of all string over the alphabet {0,1} Which Contain an even number of 0's and least one 1.

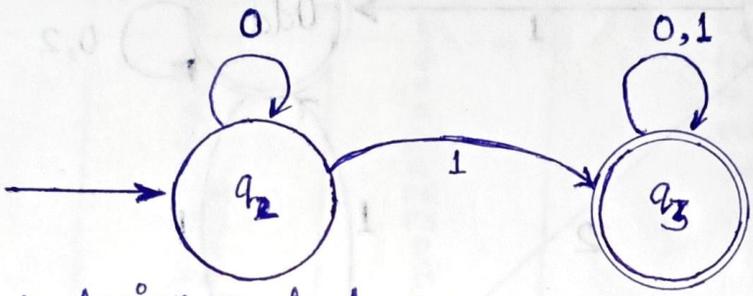
We can use the closure of 2 DFA to Construct this DFA.

DFA 1. Accepting even number of 0's



state q_0 represents even no. of '0s' in the strings Parsed and q_1 represents odd no. of '0s' in the string Parsed. Since ending state is q_0 the DFA only accepts states with even no. of '0s'.

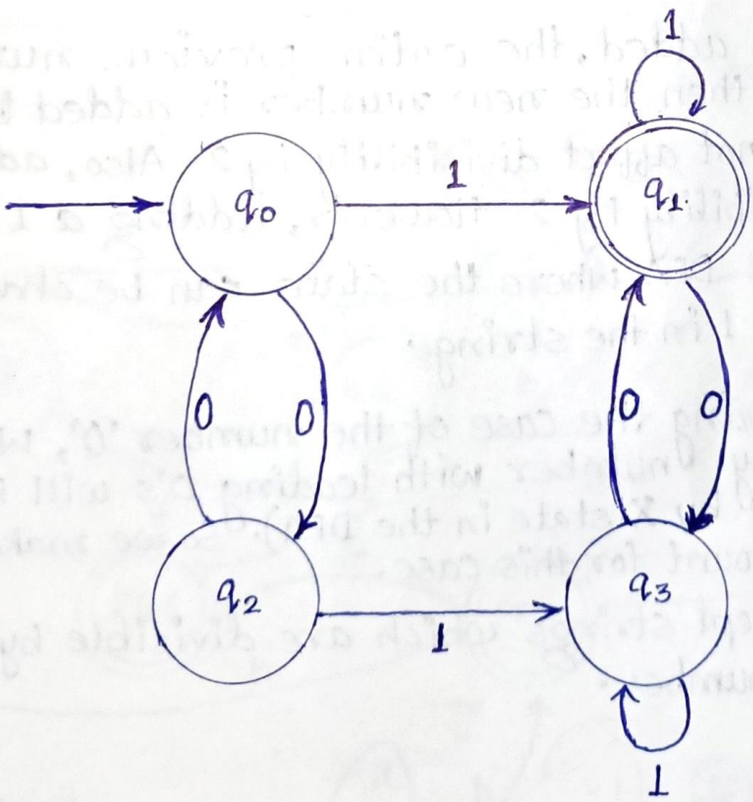
DFA 2. Accepting at least one 1.



In this DFA, one 1 is required to reach the accepting state.

Now, we take the union of these intersection of these 2 DFA's.

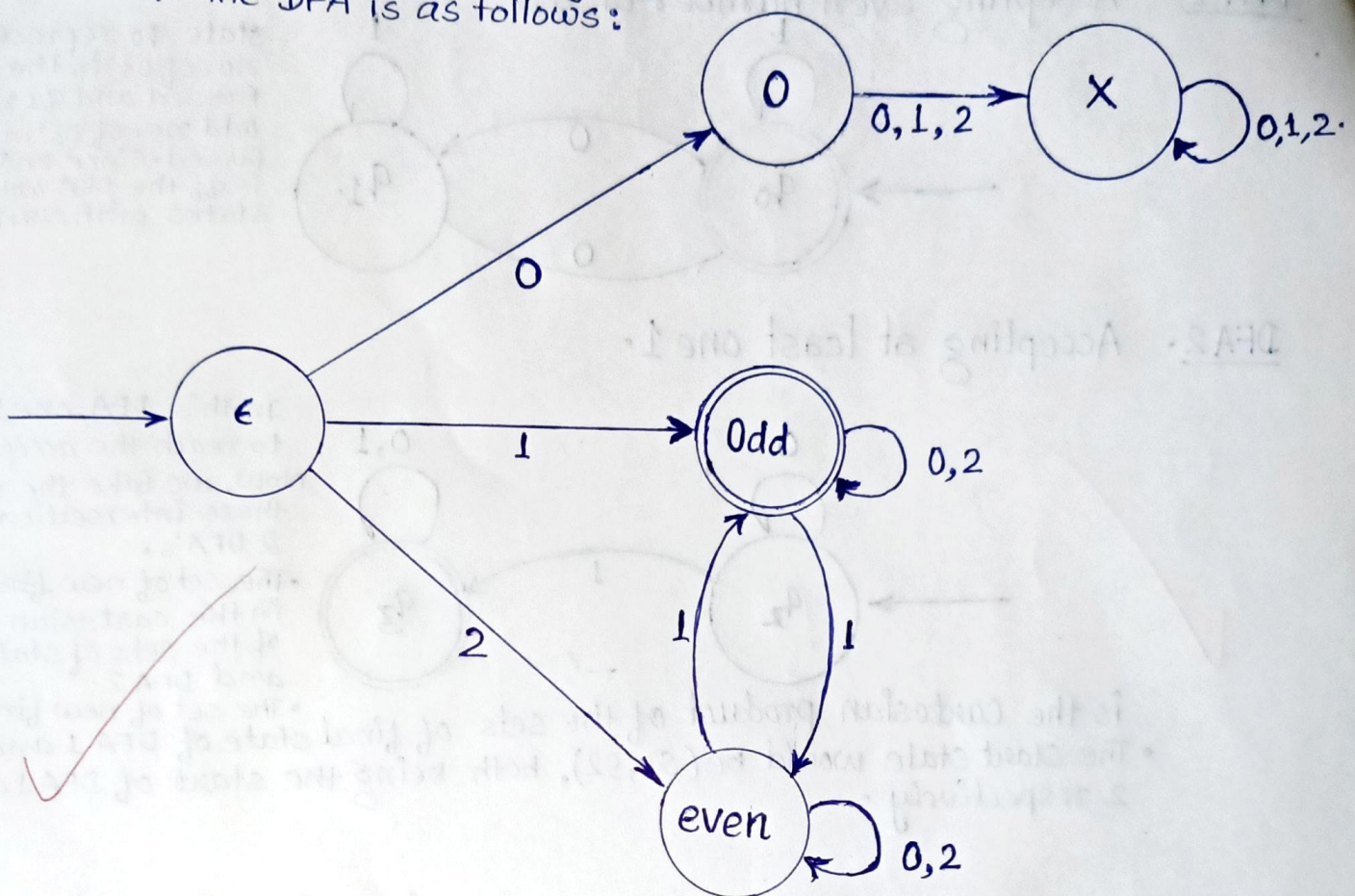
- The set of new ~~final~~ states in the cartesian product of the sets of states of DFA 1 and DFA 2.
- The set of new final states is the cartesian product of the sets of final state of DFA 1 and DFA 2.
- The start state would be (S1, S2), both being the start of DFA 1 and DFA 2 respectively.



5

Q2. Show that the set of strings in $\{0,1,2\}^*$ which are base 3 representations of odd numbers, is regular.

The set of strings in $\{0,1,2\}^*$ which are base 3 representations of odd numbers is a regular language since we can construct a DFA for it. The DFA is as follows:

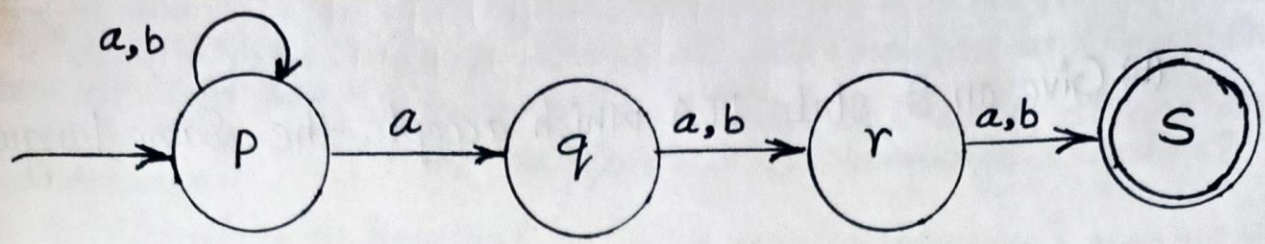


When a new digit is added, the entire previous number gets multiplied by 3 and then the new number is added to it. Multiplying by 3 does not affect divisibility by 2. Also, adding 0 or 2 does not affect divisibility by 2. However, adding a 1 changes it. Hence we obtain this DFA where the state can be changed only when we encounter a 1 in the string.

We are also considering the case of the number '0', which is an even number. However, any number with leading 0's will be considered invalid (representation by X state in the DFA). So we make a separate branch in the DFA to account for this case.

Since we want to accept strings which are divisible by 2, the final states are '0' and odd number.

Q3. Consider the NFA below:



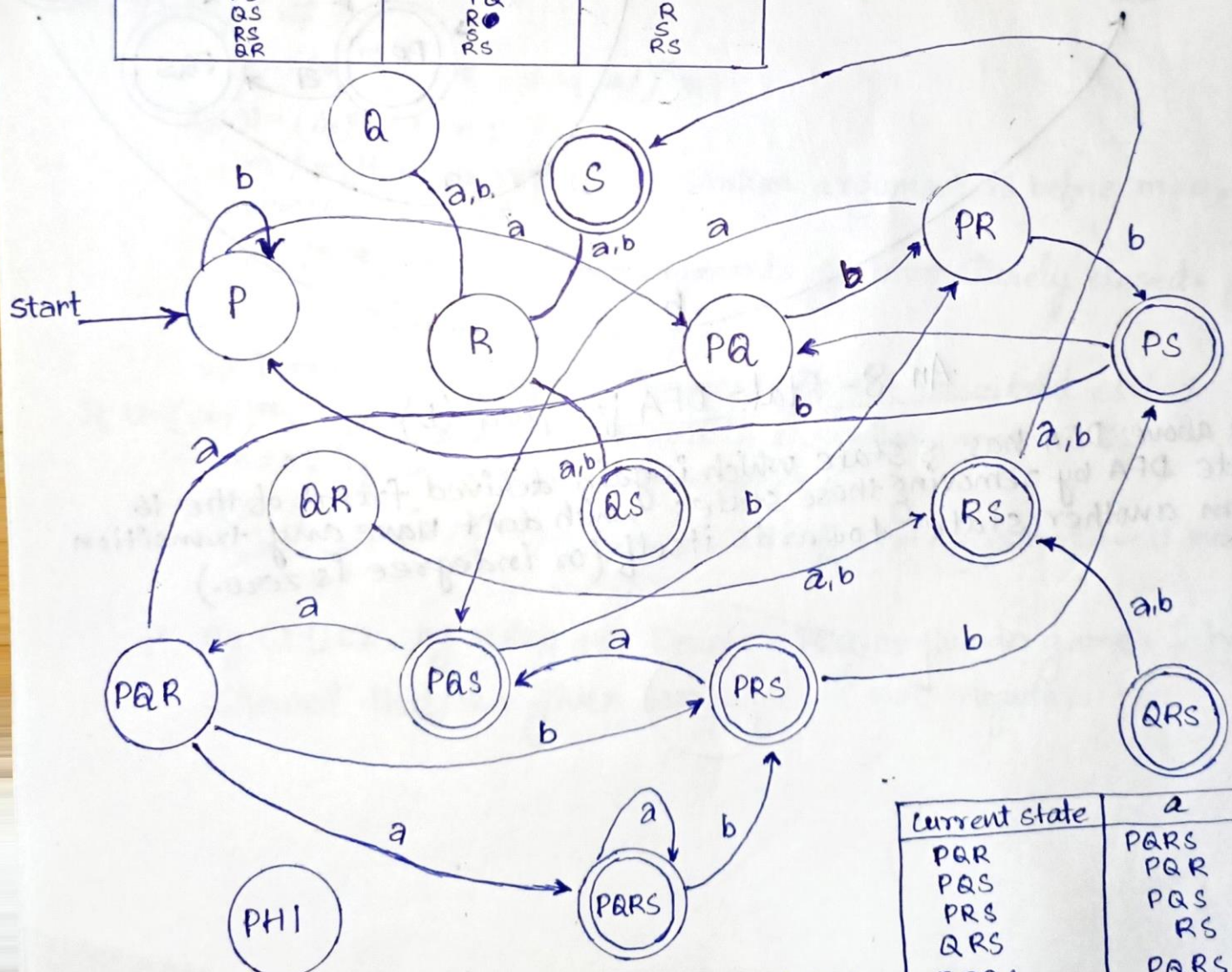
(a) Use the subset construction to obtain an equivalent DFA for the NFA below. Label each state of the DFA with the subset of states of the NFA that it corresponds to.

(b) Give an 8 state DFA which accepts the same language.

(a) This DFA is of 16 state and it is obtained by the subset construction of the given NFA. The subset construction is done by talking the power set of the states of the NFA. The initial state of the DFA is the set of states which are reachable from the initial state of the transition function of 16 states DFA is give below

Current state	a	b
Phi ϕ	ϕ	ϕ
P	PQ	P
Q	R	R
R	QS	QS
S	R	R
PQ	PQR	PR
PR	PQS	PS
PS	PQ	P
QS	R	R
RS	S	S
QR	RS	RS

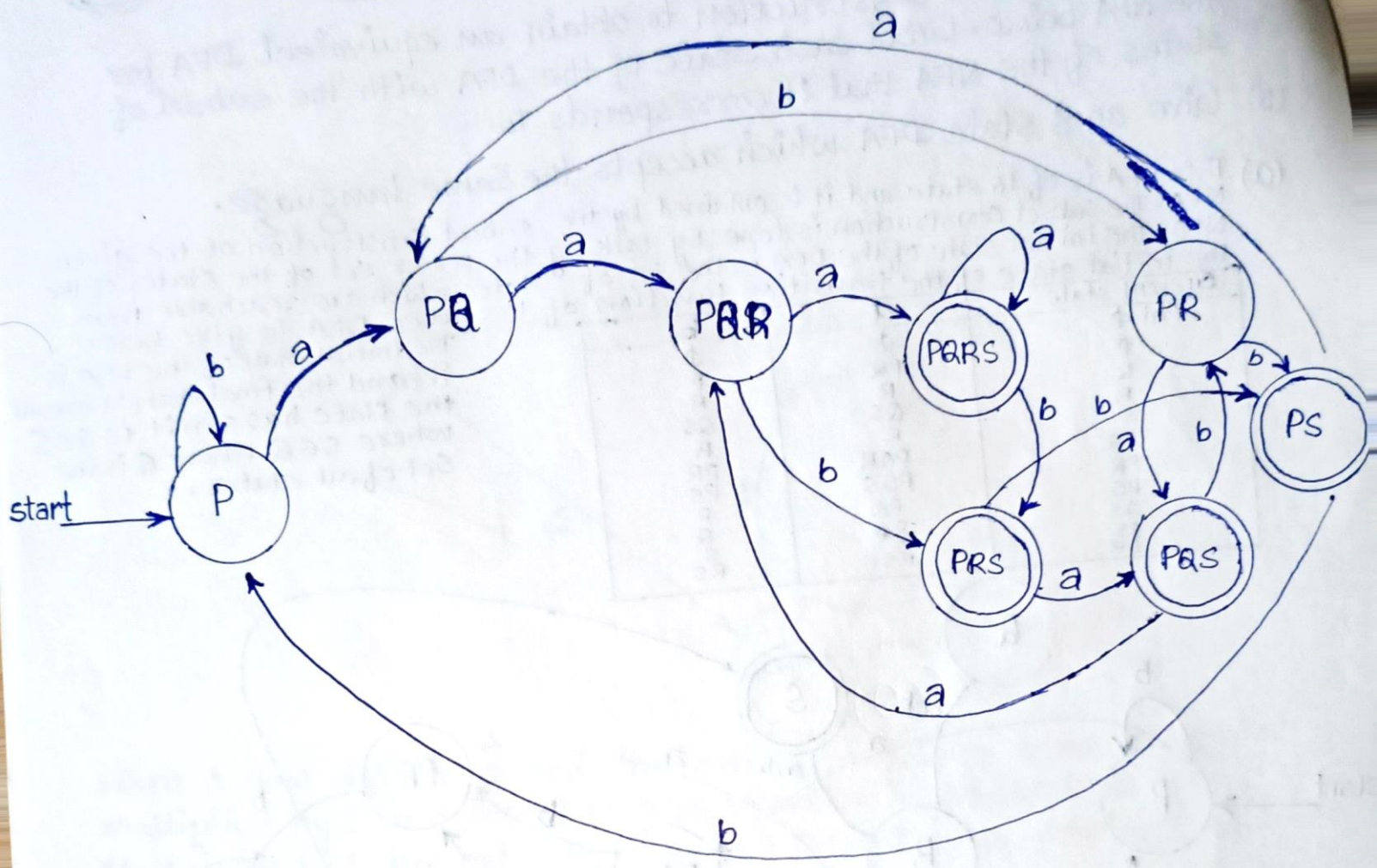
The Initial state of the DFA is {P} and the final state's are all the state has s in it is $S \in S$ where $S \in Q$ where Q is the set of all states.



Current state	a	b
PQR	PQRS	PRS
PQS	PQR	PR
PRS	PQS	PS
QRS	RS	RS
PQRS	PQRS	PRS

16-State DFA for part (a).

(b) Give an 8- state DFA which accepts the same language.



An 8- State DFA for part (b).

This above DFA has 8 state which is again derived from of the 16 state DFA by removing those states which don't have any transition from another state towards itself (or Indegree is zero.)

(23801)

Consider the language of nested C-style Comments. The alphabet Comprises racters "/", "*" and "c" (the latter symbol representing any ASCII character art from "/" and "*"). The language allows all well-nested and Complete mments. Thus strings like "cc/*ccc*/c" and "cc/*cc/*ccc*/c*/ccc" are the language, but not "cc/*c/*cc*/cc". Is this language regular? ustify your Answer.

The language mentioned in the Question is "not a regular language". Now we will show that pumping lemma does not hold for this language.

By Demon-human game for the pumping lemma and its provide an integer $k>0$ to human. The human provide the string: $(/*)^k(*/)^k$ such that $n=\varepsilon$, $g=(/*)^k$ or $z=(*/)^k$ and $|y|=2k$.

Now, the Demon must choice of decomposition can essentially be broken down into cases:

- C1: No. of '/' = No of '*' i.e. $v=(/*)^m$ where $m \leq k$.
  If I choose any $i \neq 1$, then the no. of opening comments $\neq$ no. of closing comments.

∴ The string does not contain closed Comments as $i-1$ comments are not closed.

- C2: #/ ≠ # *

  So $v=(/*)^m/$ or $v=(/*/)^m*$

  If $v=(/*)^m/$ then:

  $m \neq 0$, then we produce a similar argument to before $m=0$, then choose $i=0$. then,

  .../*/*_*/*/... then Comments get prematurely closed.

  ↓ /removed

  ⇒ There are incomplete Comments in the resultant string

If $v=(/*/)^m*$, then $m \neq 0$ will result in a Similar proof as C1.

$m=0$, If we remove * then for Ex:

$(/_{x^*}/\underbrace{*\cdots}_{k-1})(*/)^k$. we see resulting string is not well nested.

∴ By C1 ∪ C2, by using the Demon - ~~Player~~ Human game. I have showed that the given language is not regular.

10

Q5. For a set of natural number 'x', define binary(x) to be the set of binary representations of numbers in x. Similarly define unary(x) to be the set of unary representations of numbers in x: unary(x) = $\{1^n \mid n \in x\}$ Thus for x = {2,3,6} binary(x) = {10, 11, 110} and unary(x) = {11, 111, 111111} consider the two Propositions below:

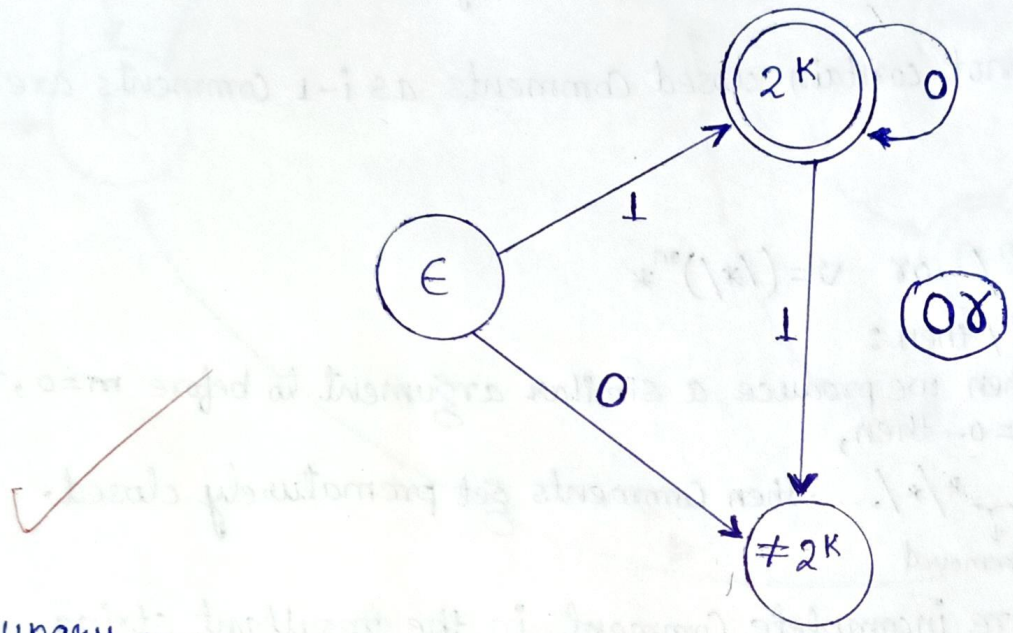
(a) For all x, if binary(x) is regular then so is unary(x).

(b) For all x, if unary(x) is regular then so is binary(x).

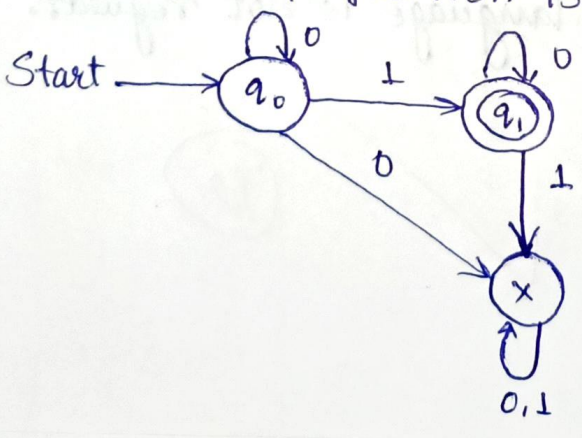
One of the statements above is true and the other is False. which is which? Justify your Answer.

Proposition (a): For all x, if binary(x) is regular then so is unary(x). we can demonstrate this by using a counter-example. To show why (a) is False.

Now Assume 'x' = $\{2^k \mid k \geq 0\}$. Then the binary representation of this set(x) is {10, 100, 1000, 10000, ...} or (x) = $\{1a^k \mid k \geq 0\}$ this is a regular language. Now we can construct a DFA for it.



Now unary representation of set(x) is {11, 1111, 111111, ...} or (x) = $\{1^{2k} \mid k \geq 0\}$ which is not a regular. The reason is that the language does not satisfy the pumping lemma for regular languages (number of 1's in these string grows exponentially (i.e. 2^k for $k \geq 0$) A regular language cannot count the number of 1's in such exponential. Hence, the proposition is False.



Proposition (b): For all x, if unary(x) is regular then so is binary(x)

we can demonstrate this ~~by~~ To show that why (b) is true.

Let us Assume that unary(x) is regular, then it must satisfy the ultimate periodicity criteria. If it satisfies the criteria of ultimate periodicity, then len(unary(x)) should consist of a finite set along with finite no. of arithmetic progressions with same common difference.

To contruct a DFA that accepts a particular arithmetic Progression of the form $a, a|d, a|2d, \ldots$, we can use the following Procedure:

- Let the arithmetic progression be $a, a|d, a|2d, \ldots$
- Define 'd' states in the DFA as $Q = \{q_0, q_1, q_2, \ldots, q_{d-1}\}$, where q_n represents the no. that give a remainder of 'n' when divided by d.
- Set the start as q_0, and set the accepting state as q_j, where $j = a \bmod d$.
- Start reading the binary string from left to right. For each digit:
  - Multiply the current value by 2.
  - Add the value of the current digit to it.
  - Take the result mod. d.
  - Transition to the state corresponding to this modulo value.

Formally, the DFA is defined as follows:

$Q = \{q_0, q_1, q_2, \ldots, q_{d-1}\}$ $\quad \Sigma = \{0, 1\}$, $S = q_0$

$F = \{q_j\}$, where $j = a \bmod d$.

The transition function δ is define as:

$\therefore \delta(q_j, l) = q_{(2j+l) \bmod d}$ for $l \in \{0, 1, 2, \ldots, d-1\}$ and $l \subset \Sigma$.

Here:

- q_j: current state
- l: current input symbol (either 0 or 1).
- $2j|l$: Update the current value of doubling in (left-shifting) and adding the current input symbol.
- $(2j+l) \bmod d$: Take the modulo d of the updated value to determine the next state.

Then we can create finite no. of DFA each of Arithmetic progression and DFA for each entry is that finite set. Then we can take the union of those DFA to get the resulting DFA. Hence binary(x) is a regular language.

Hence this proposition is true.

Q6. Give a language $L \subseteq \{a,b\}^*$ such that neither L nor $\{a,b\}^* - L$ contain an infinite regular set.

Consider the language $L \subseteq \{a,b\}^*$ defined as follows

$L = \{w \in \{a,b\}^* \mid$ the length of w (denoted $|w|$) has an odd no. of digits in binary representation$\}$.

clearly, we can define L^c (the complement of L) as:

$L^c = \{w \in \{a,b\}^* \mid$ the length of w has an even number of digit B.R$\}$

Claim: Both L and L^c do not contain an infinite regular set.

Proof: We observing that both 'L and L^c' are infinite regular set. Because there are an infinite number of odd and even no. which allows us to construct an infinite number of strings in L and L^c respectively.

- Now, we will showing that 'L' does not contain an infinite regular set. So we Assume contradiction, that L contains an infinite regular set. we say R. Since R is regular, it must satisfy the ultimate periodicity criteria of regular languages, which says that the set of lengths of strings in R must be ultimately Periodic.

  $\{|w| \mid w \in R\}$ must eventually form an arithmetic progression. Since $R \subseteq L$, the set len(R) must consist of lengths whose binary representation have an odd no. More formally, the set $len(R) \subseteq$ {natural no. with an odd no. of binary digits.}.

  Now Assume len(R) forms an infinite arithmetic progression Let the arithmetic progression be represented as $\{a+nd \mid n \in \mathbb{N}\}$, where a is the starting point and d is the common difference

  Let the no. of binary digit in d be m.

  Since we have infinite increasing sequence of no., we can always find a no. p with K binary digit where:

  - $K = m+2$ if m is odd, or
  - $K = m+1$ if m is even

  we will counter a number of sequence with K+1 binary digit where K+1 is an even no. But this contradiction len(R) only contains no. with odd no.

  Therefore the set of len(R) cannot form an infinite arithmetic Progression, implying that R cannot be infinite regular set.

- Now we will showing that L^c does not contain an infinte regular set.

L^c contained an infinite regular set R', then the set of string length $n(R')$ would also be ultimately periodic. Since $R' \subseteq L^c$, the string lengths would be the no. of binary representative have an even no.

→ As before, an arithmetic progression of no. with an even no. of binary digit will eventually include number with an odd no. of digits as the sequence grows, which contradicts the fact $len(R')$ only contains numbers with an even no. only

Hence Contradiction shows that L^c cannot contain an infinite regular set.

Thus, we shown that Both L and L^c does not contain an infinite regular set. Therefore language $L = \{w \in \{a,b\}^* \mid$ the length of w has an odd no. of digits in binary representation$\}$

Q7. Let L be an arbitrary subset of $\{a\}^*$. Prove that L^* is regular.

→ Proving: L^* is regular we will use following definition of Ultimate periodicity of a language given in the class.

Now,

There exist $n_0 \geq 0, p \geq 1 \in \mathbb{N}$, such that for all $m \geq n_0$

$\therefore \; m \in X \Longrightarrow m+p \in X$

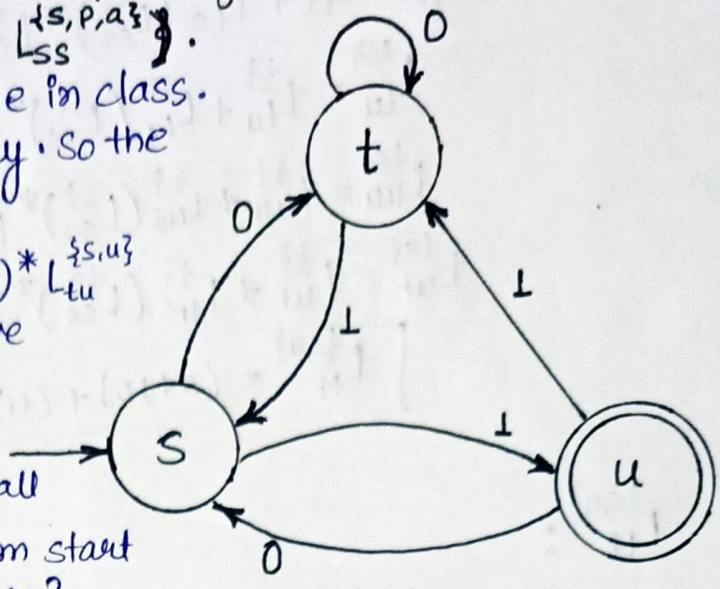
For case where $L = \phi$ where is obvious as $L^* = \phi^* = \{\epsilon\}$ which is regular. But for the $L \neq \phi$. Let's choose a string that has least positive length in L or L^* which will give the same element, so let that element be $a^t = z$ where $len(a^t) = t$, now let's take $p = t$ and $n_0 = t$. Let's construct a len(L) and len(L^*) where each element represents the length of the element in the respective set (L) and (L^*) such that the sets len(L) and len(L^*) will be the subsets of $\mathbb{N}$ Now, let's take $m \in len(L^*)$ and $m \geq n_0$ such that $\exists x \in L^*$ whose length $(x) = m$.

Now, since x is in L^* then their also $\exists y \in L^*$ such that y is made from concatenating x and z, which will also exist in the l^* as this set is made from the concatenation of the elements of L and L^*,

Hence $m+t \in len(L^*)$ and hence L^* is regular.

10

6. Use the Construction done in class to construct a regular expression corresponding to the language accepted by the DFA below (i.e. the expression corresponding to $L_{ss}^{\{s,p,q\}}$).



we will follow the construction done in class. we will remove the state 't' initially. So the expression will look like:

$$L_{su}^{\{s,t,u\}} = L_{su}^{\{s,u\}} + L_{st}^{\{s,u\}} \left(L_{tt}^{\{s,u\}}\right)^* L_{tu}^{\{s,u\}}$$

we can solve this using the recursive method discussed in class.

L_{su}^{x} = {all strings $x \mid \hat{\delta}(s,x)$ = u for $x \in X$ and all the intermediate states (apart from start and end state) belong to the set X}

$$L_{su}^{\{\}} = \{x \in \Sigma : \hat{\delta}(s,x) = u\} \text{ if } s \neq u$$

$$L_{su}^{\{\}} = \{x \in \Sigma : \hat{\delta}(s,x) = s\} \cup \{\epsilon\} \text{ if } s = u$$

Then:

$$L_{su}^{\{s,t,u\}} = L_{su}^{\{s,u\}} + L_{st}^{\{s,u\}} \left(L_{tt}^{\{s,u\}}\right)^* L_{su}^{\{s,u\}}$$

(10)

Term $L_{su}^{\{s,u\}}$:

$$L_{su}^{\{s,u\}} = L_{su}^{\{s\}} + L_{su}^{\{s\}} \left(L_{uu}^{\{s\}}\right)^* L_{uu}^{\{s\}}$$

$$L_{su}^{\{s\}} = L_{su}^{\{\}} + L_{ss}^{\{\}} \left(L_{ss}^{\{\}}\right)^* L_{su}^{\{\}} = \boxed{1 + \epsilon^* 1 = 1}$$

$$L_{uu}^{\{s\}} = L_{uu}^{\{\}} + L_{us}^{\{\}} \left(L_{ss}^{\{\}}\right)^* L_{su}^{\{\}} = \boxed{\phi + 0\epsilon^* 1 = 01}$$

$$L_{su}^{\{s,u\}} = 1 + 1(01)^* 01 = \boxed{1(01)^*}$$

$L_{st}^{\{s,u\}}$:

$$L_{st}^{\{s,u\}} = L_{st}^{\{s\}} + L_{su}^{\{s\}} \left(L_{uu}^{\{s\}}\right)^* L_{ut}^{\{s\}}$$

$$L_{st}^{\{s\}} = L_{st}^{\{\}} + L_{ss}^{\{\}} \left(L_{ss}^{\{\}}\right)^* L_{st}^{\{\}} = \boxed{0 + \epsilon(\epsilon)^* 0 = 0}$$

$$L_{su}^{\{s\}} = L_{su}^{\{\}} + L_{ss}^{\{\}} \left(L_{ss}^{\{\}}\right)^* L_{su}^{\{\}} = \boxed{1 + \epsilon(\epsilon)^* 1 = 1}$$

$$L_{uu}^{\{s\}} = L_{uu}^{\{\}} + L_{us}^{\{\}} \left(L_{ss}^{\{\}}\right)^* L_{su}^{\{\}} = \boxed{\epsilon + 0(\epsilon)^* 1 = \epsilon + 01}$$

$$L_{ut}^{\{s\}} = L_{ut}^{\{\}} + L_{us}^{\{\}} \left(L_{ss}^{\{\}}\right)^* L_{st}^{\{\}} = \boxed{1 + 0(\epsilon)^* 0 = 1 + 00}$$

$$\boxed{L_{st}^{\{s,u\}} = 0 + 1(\epsilon + 01)^*(1 + 00)}$$

Term $L_{tu}^{\{s,u\}}$:

$$L_{tt}^{\{s,u\}} = L_{tt}^{\{s\}} + L_{tu}^{\{s\}} (L_{uu}^{\{s\}})^* L_{ut}^{\{s\}}$$

$$L_{tt}^{\{s\}} = L_{tt}^{\{\}} + L_{ts}^{\{\}} (L_{ss}^{\{\}})^* L_{st}^{\{\}} = \boxed{0 + 1(\epsilon)^* 0 = 0 + 10}$$

$$L_{tu}^{\{s\}} = L_{tu}^{\{\}} + L_{ts}^{\{\}} (L_{ss}^{\{\}})^* L_{su}^{\{\}} = \boxed{\phi + 1(\epsilon)^* 1 = 11}$$

$$L_{uu}^{\{s\}} = L_{uu}^{\{\}} + L_{us}^{\{\}} (L_{ss}^{\{\}})^* L_{su}^{\{\}} = \boxed{\phi + 0(\epsilon)^* 1 = 01}$$

$$L_{ut}^{\{s\}} = L_{ut}^{\{\}} + L_{us}^{\{\}} (L_{ss}^{\{\}})^* L_{st}^{\{\}} = \boxed{1 + 0(\epsilon)^* 0 = 1 + 00}$$

$$\boxed{L_{tt}^{\{s,u\}} = (0+10) + (11)(01)^*(1+00)}$$

$L_{tt}^{\{s,u\}}$:

$$L_{tu}^{\{s,u\}} = L_{tu}^{\{s\}} + L_{tu}^{\{s\}} (L_{uu}^{\{s\}})^* L_{uu}^{\{s\}}$$

$$L_{tu}^{\{s\}} = L_{tu}^{\{\}} + L_{ts}^{\{\}} (L_{ss}^{\{\}})^* L_{su}^{\{\}} = \boxed{\phi + 1(\epsilon)^* 1 = 11}$$

$$L_{uu}^{\{s\}} = L_{uu}^{\{\}} + L_{us}^{\{\}} (L_{ss}^{\{\}})^* L_{su}^{\{\}} = \boxed{\phi + 0(\epsilon)^* 1 = 01}$$

$$\boxed{L_{tu}^{\{s,u\}} = 11 + 11(01)^* 01 = 11(01)^*}$$

Finally

$$\boxed{L_{su}^{\{s,t,u\}} = 1(01)^* + (0 + 1(01)^*(1+00))((0+10) + (11)(01)^*(1+00))^*(11(01)^*)}$$