DFA + NFA ⇝ Push down Automata

Regular langu. ⇝ Context free Langu.

apsara

Date: __.__.____

31/8/22

Pf A is regular. So ∃ DFA M s.t. L(M) = A.

Let K-1 be the no. of states of M.

Then ∀ x, y, z s.t. xyz ∈ A with |y| ≥ K

By Pigeon hole ∃ a ~~string~~ string w and alphabets c_1, c_2 s.t. $c_1 w c_2 \subseteq y$ & state of M is same at c_1 and c_2 when xyz is read.

Let $y = u c_1 w c_2 w'$. Let $v = c_1 w$ and $\omega = c_2 w'$ $(v \neq \varepsilon)$

observe that if $xyz \in A$

Then so does $xuv^i\omega z$. $\forall\ i \geq 1$

$\Sigma = \{0, 1\}$

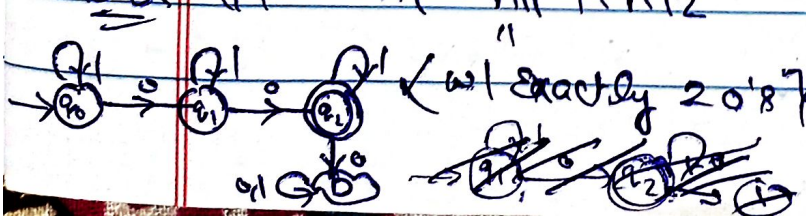
Problem: Each of the following language is the intersection of two simpler languages. Construct DFAs for simpler languages, then combine them using 'intersection DFA Construction' to get the actual DFA:

(i) $\{w \mid$ w has exactly two 0's and atleast two 1's$\}$

(ii) $\{w \mid$ w has even no. of 0's and each 0 is followed by 1$\}$

HW Compare DFA's for $A_1 \cup A_2$ and $A_1 \cap A_2$ where A_i is a regular set with DFA M_i.

Sol (i) $A_1 = A_{11} \cap A_{12} = \{w \mid$ at least two 1's$\}$



Draw product automata.

1/9/22 Q. $A_1 = L(M_1)$ & $A_2 = L(M_2)$. Find M such that $L(M) = A_1 \cap A_2$

Sol^n $M_1 = (Q_1, \Sigma, \delta_1, s_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, s_2, F_2)$

Recall $M' = (Q_1 \times Q_2, \Sigma, \delta, (s_1, s_2), F_1 \times F_2)$ Date:

$\delta^*((q_1, q_2), x) = (\delta_1^*(q_1, x), \delta_2^*(q_2, x))$

$L(M') = A_1 \cap A_2$.

Q. How to find M s.t. $L(M) = A_1 \cup A_2$?

$M(Q_1 \times Q_2, \Sigma, \delta, (s_1, s_2),$ }

Check: final state of $(A_1^c \cap A_2^c)^c$

$=$ Non final states of $A_1^c \cap A_2^c$

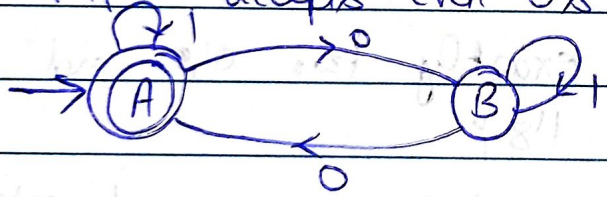
$= Q_1 \times Q_2 \setminus$ final states of $A_1^c \cap A_2^c$

$= Q_1 \times Q_2 \setminus ((Q \setminus F_1) \times (Q_2 \setminus F_2))$

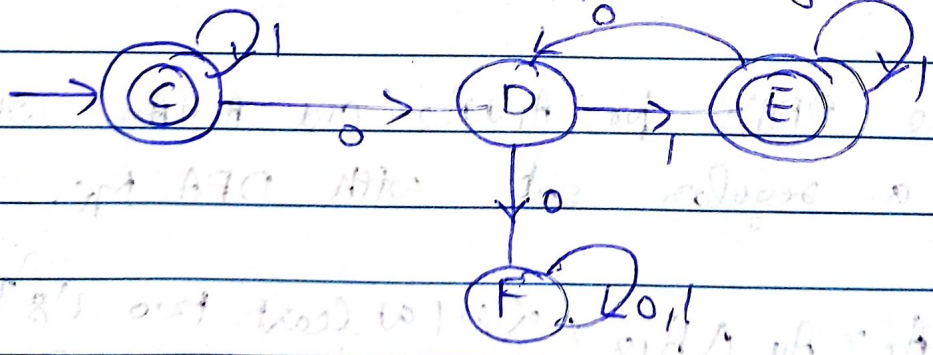
Set theorem: $(F_1 \times Q_2) \cup (Q_1 \times F_2)$

Q. $\Sigma = \{0, 1\}$. Find M s.t. $L(M) = \{w \mid w$ has even no. of 0's and each zero is followed by a 1$\}$

Sol^n $M_1 \rightarrow$ accepts even 0's



$M_2 \rightarrow$ each 0 is followed by 1



Now take intersection.

	0	1
(AC)	(BD)	(AC)
(AD)	BF	AE
(AE)	BD	AE
(AF)	BF	AF
(BC)	AD	BC
(BD)	AF	BE
(BE)	AD	BE
(BF)	AF	BF

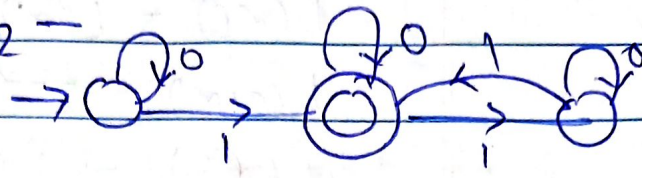
Q. $\Sigma = \{0,1\}$

$D = \{w \mid w$ contain even no. of 0's, odd no of 1's and doesn't contain 01$\}$

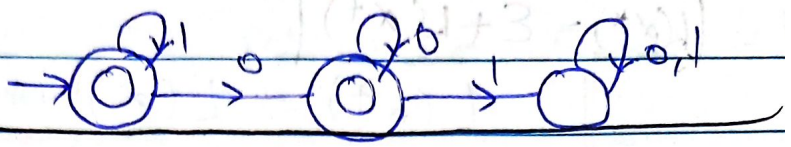
Date: __.__.____

M_1 – accept even 0's

M_2 –



MS – doesn't contain 01



* **Patterns :–** will be defined inductively. For a pattern α we define $L(\alpha) = \{x \in \Sigma^* \mid x \text{ matches } \alpha\}$

* **Atomic pattern :–** $\forall\ a \in \Sigma \quad L(a) = \{a\}$

→ ε matched by ε only → $L(\varepsilon) = \{\varepsilon\}$

→ $L(\phi) = \phi$

→ $L(\#) = \Sigma$

→ $L(@) = \Sigma^*$

→ So all alphabets, ε, ϕ, #, $, @ are patterns (atomic pattern)

* **Compound pattern :–** Uses Binary operations → +, ∩, ·
unary operations → +, *, ~

→ If α and β are patterns then we are → $\alpha+\beta$, $\alpha \cap \beta$, $\alpha \cdot \beta$, α^*, α^+, $\sim\alpha$ as well.

→ x matches $\alpha+\beta$ if x matches either α or β.

Page No.: _____

→ $x \in L(\alpha\beta)$ if $x \in L(\alpha) \cap L(\beta)$

$x \in L(\alpha\beta)$ if $\exists\ y, z$ with $x = yz$ and $y \in L(\alpha)$, $z \in L(\beta)$

$L(\alpha\beta) = L(\alpha)L(\beta)$

$L(\sim\alpha) = \Sigma^* \setminus L(\alpha)$

$L(\alpha^*) = L(\alpha^0) \cup L(\alpha^1) \cup L(\alpha^2) \ldots$

$= \{ x_1 \ldots x_n \mid n \geq 0 \mid x_i \in L(\alpha) \quad 1 \leq i \leq n \}$

$\boxed{L(\alpha^*) = \varepsilon + L(\alpha^+)}$

$L(\alpha^+) = L(\alpha)^1 \cup L(\alpha)^2 \cup \ldots \cup L(\alpha)^n$

$= \{ x_1 \ldots x_n \mid n \geq 1 \quad x_i \in L(\alpha) \quad 1 \leq i \leq n \}$

Basically Patterns are just strings of symbols from $\Sigma, \cup, \varepsilon, \phi, \#, @, +, \cap, \sim, *, \dagger, (,)$

Ex- $\Sigma^* = L(@) = L(\#^*)$

↑ verify $L(\#) = \Sigma$

$L(\#^*) = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \ldots$

$= \Sigma^* = L(@)$

* Note $x_1 \ldots x_m \in \Sigma^*$

Then $\{ x_1, \ldots, x_m \} = L(x_1 + \ldots + x_m)$

$L(x_1) = \{x_1\}$

$L(x_2) = \{x_2\}$

$\vdots$

$L(x_m) = \{x_m\}$

$L(x_1 + x_2 + \ldots + x_n)$

$= \{x_1\} \cup \{x_2\} \cup \ldots \{x_n\}$

$= \{x_1, x_2, \ldots x_n\}$

$L(x_1 + x_2 + x_3) = L(x_1 + (x_2 + x_3))$

$= L(x_2 + (x_1 + x_3))$

$= L(x_3 + (x_1 + x_2)) \ldots$

2/9/22 **Remark 1:** As objects patterns ε & ϕ are different from ε (the null string) & ϕ (the empty set). The patterns ε & ϕ are just symbols in the language of patterns.

Def^n:- Two patterns α, β are equivalent if $L(\alpha) = L(\beta)$.

Remark 2: Some pattern symbols we used last time are redundant. for example the pattern ε is equivalent to $\sim(\#@)$

Q. why? { Also @ is equivalent to $(\#^*)$; $\forall\ \alpha, \alpha^*$ is equivalent to $\alpha\alpha^*$ (?) i.e., $L(\alpha^*) = L(\alpha\alpha^*)$ (?)

$L(\varepsilon) = \{\varepsilon\}$

$L(\sim(\#@)) = \Sigma^* \setminus L(\#@)$

$= \Sigma^* \setminus L(\#)L(@)$

$= \Sigma^* \setminus \Sigma\cdot\Sigma^*$

$= \Sigma^* \setminus \Sigma^+ = \{\varepsilon\}$

$L(\alpha\alpha^*) = L(\alpha)L(\alpha^*)$
$= L(\alpha)\cdot[L(\alpha^0) \cup L(\alpha^1) \cup \dots]$
$= L(\alpha)[\varepsilon \cup L(\alpha) \cup L(\alpha^2) \cup \dots]$
$= L(\alpha^*)\{\varepsilon\}$

$L(@) = \Sigma^* \quad L(\#) = \Sigma$

$L(\#^*) = L(\#)^0 \cup L(\#)^1 \cup L(\#)^2 \cup \dots$

$= \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \dots$

$= \Sigma^*$

Q. Can you think of some more redundancies?

Ans — (1.) ϕ & $\sim@$

$L(\phi) = \phi$ and $L(\sim@) = \Sigma^* \setminus L(@)$

$= \Sigma^* \setminus \Sigma^* = \phi$

(2.) (?) $L(\alpha \cap \beta) = L(\sim(\sim\alpha + \sim\beta))$ $\forall\ \alpha, \beta$

(?) **Remark 3:** $\sim$ is also redundant (Hard).

Date: ___.___.___

Regular expression :-

Defn:- A regular expression is a (pattern) build from $a \in \Sigma$, Σ, ϕ (patterns) & operators $+$ (binary), $*$ (unary) & $\cdot$ (binary).

Thm:- Let $A \subseteq \Sigma^*$. The following are equivalent.

(1.) A is regular i.e. $A = L(M)$ for some DFA M.

(2.) $A = L(\alpha)$ for some pattern α

(3.) $A = L(\alpha)$ for a regular expression α.

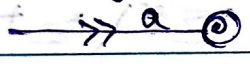
PF TST (1) $\Leftrightarrow$ (2) $\Rightarrow$ (3) $\Rightarrow$ (2)

(3) $\Rightarrow$ (2) is Trivial. (?)

(2) $\Rightarrow$ (1) and then (1) $\Rightarrow$ (3)

first show that some basic sets are regular (related to atomic pattern)

- $\{a\}$ is regular $\forall\, a \in \Sigma$



- $\{\varepsilon\}$ regular

- ϕ regular

Regular sets closed under $\cap, \cup, \sim, \cdot, *$.

So for regular A & B $A \cap B$, $A \cup B$, $\sim A$, $A \cdot B$, A^* and A^+ are regular.

we use them to show (3) $\Rightarrow$ (1) inductively.

If α is one of the following:

(i) a, $a \in \Sigma$
(ii) ε
(iii) ϕ
(iv) $\#$
(v) $@$
(vi) $\beta + \gamma$ β, γ - Pattern
(vii) $\beta \cap \gamma$
(viii) $\beta\gamma$
(ix) $\sim\beta$
(x) β^*
(xi) β^+

(i) — (v) related to atomic patterns. (i), (ii) & (iii) are regular ✓
& @ are redundant, so can be dealt by others.

(xi) Note β^+ is redundant.

(vi) $L(\beta + \gamma) = L(\beta) \cup L(\gamma)$ & regular sets are closed under $\cup$.

(vii), (viii), (ix), (x) - similar.

ends (2) $\Rightarrow$ (1)

what if α is not one of the above?

Now (1) $\Rightarrow$ (3) (To get a regular expression from an automata.)

Given an NFA, $M = (Q, \Sigma, \Delta, S, F)$ a subset $X \subseteq Q$ & $u, v \in Q$. we show how to construct regular expression α^*_{uv} representing the set of all strings x s.t. $v \in \hat{\Delta}(u, x)$ and all states along the way lies in X (with possible exception of u & v).

we define α^*_{uv} inductively: $x = \phi$, $\langle a_1, a_2, \ldots, a_k \rangle \subseteq \Sigma$ s.t. $v \in \Delta(u, a_i)$.

for $u \neq v$.

$$\alpha^{\phi}_{uv} \overset{\text{defn}}{=} \begin{cases} a_1 + \cdots a_k & \text{if } k \geq 1 \\ \phi & \text{if } k = 0 \end{cases}$$

$u = v$

$$\alpha^{\phi}_{uv} \overset{\text{defn}}{=} \begin{cases} a_1 \cdots + a_k + \varepsilon & ; k \geq 1 \\ \varepsilon & ; k = 0 \end{cases}$$ ✓

At

for nonempty X, choose $q \in X$ and take

$$\alpha^{x}_{uv} \overset{\text{defn}}{=} \left(\alpha^{x-\{q\}}_{uv}\right) + \left(\alpha^{x-\{q\}}_{uq}\left(\alpha^{x-\{q\}}_{qq}\right)^{*}\left(\alpha^{x-\{q\}}_{qv}\right)\right)$$

Note that any path from u to v with all intermediary states in X

either (i) does n't visit q ($\alpha_{uv}^{x-\{q\}}$).

(ii) visit q for first time (hence $X_{uq}^{x-\{q\}}$) followed by finite number of time. (Hence $(\alpha_{qq}^{x-\{q\}})^*$ followed by a path from q to v (hence $\alpha_{qv}^{x-\{q\}}$)

The sum of all expressions α_{sf}^{Q} represents the set of strings accepted by M whose SES & FEF

Date: 7/9/

Page No.:

Σ - Alphabet set

Thm :- $A \subseteq \Sigma^*$ Then $A = L(M)$ (i)
$\Leftrightarrow A = L(\alpha)$ (ii)
$\Leftrightarrow A = L(\alpha)$ (iii)

apsara
Date: Some pattern of some regular expression.

2

Pf (3) $\Rightarrow$ (2) ✓

(2) $\Rightarrow$ (1) Routine

(1) $\Rightarrow$ (3)

NFA: $M = (Q, \Sigma, \Delta, S, F)$

for $X \subseteq Q$ & $u, v \in Q$

Define :- α_{uv}^{*X} represents all strings x s.t. $\exists$ a path from u to v in M by x & all states in that path lie in X with possible exception of u & v.

$(v \in \hat{\Delta}(u, x))$

$X = \phi$

$a_1, a_2, \ldots, a_k \in \Sigma$ s.t. $v \in \Delta(u, a_i)$

if $u \neq v$

$$\alpha_{uv}^{\phi} \overset{\text{defn}}{=} \begin{cases} a_1 + \cdots + a_k & ; k \geq 1 \\ \phi & ; k = 0 \end{cases}$$

if $u = v$

$$\alpha_{uv}^{\phi} = \alpha \begin{cases} a_1 + \cdots + a_k + \varepsilon & k \geq 1 \\ \varepsilon & k = 0 \end{cases}$$

$X \neq \phi$ choose $q \in X$

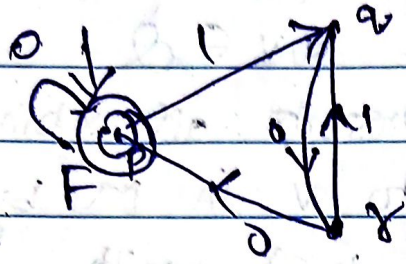
$$\alpha_{uv}^{X} \overset{\text{defn}}{=} \alpha_{uv}^{X - \{q\}} + \alpha_{uq}^{X - \{q\}} \left(\alpha_{qq}^{X - \{q\}}\right)^* \alpha_{qv}^{X - \{q\}}$$

Page No. 1

(*) Any path from u to v with all integer states in X either (1) never inside (2) visit q once followed by loop to q, then never to q.

Date: __.__.____

Exp:- Set of all α_{sf}^{Q} whose S start state, f final states, Q all states



Inductively we want $\alpha_{pp}^{<p,q,r>}$

$$= \alpha_{pp}^{<p,r>} + \left(\alpha_{pq}^{<p,r>} \cdot \left(\alpha_{qq}^{<p,r>}\right)^{*} \cdot \alpha_{qp}^{<p,r>}\right)$$

Paths going from p to p staying in $<p,r>$?

$$\alpha_{pp}^{<p,r>} = 0^{*} = \{0, 00, 000, \ldots\}$$

$$\alpha_{pq}^{<p,r>} = 0^{*} \cdot 1$$

$$\alpha_{qq}^{<p,r>} = \varepsilon + 0 \cdot 1 + 00^{*}1$$

$$= \varepsilon + 0 \cdot (\varepsilon + 00^{*}) \cdot 1$$

$$\alpha_{qp}^{<p,r>} = 00^{*}$$

$$\alpha_{pp}^{<p,q,r>} = 0^{*} + 0^{*}\left[(\varepsilon + 00^{*}1)^{*} 00^{*}\right.$$

$$= L(M)$$

Page No.:_____

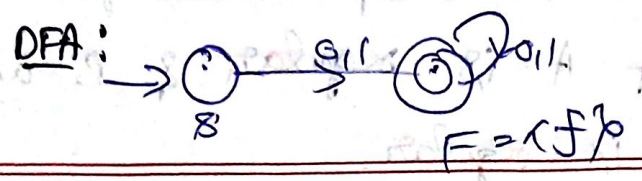
Problem :- 1. Let $A/B = \{w \in B \mid wx \in A \text{ for some } x \in B\}$
Show that if A is regular & B any language
then A/B is regular.

2. Let N be an NFA with K states that recognises A. Show that $A \neq \phi \Rightarrow \exists$ a string $x \in A$ s.t $|x| \leq K$.

8/9/22

No dead state in DFA.

DFA:

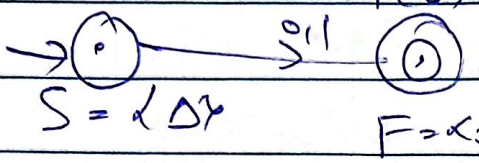


NFA: $(Q, \Sigma, \Delta, S, F)$

$\Delta : Q \times \Sigma \to 2^Q$

$\hat{\Delta} : 2^Q \times \Sigma^* \to 2^Q$

$2^Q = P(Q)$ (Power set)



$\Delta(f,0) = \Delta(f,1) = \phi$ (Allowed in NFA)

For Exam :- Read equality DFA ⇔ NFA

Language And Regular set

Language are subsets of Σ^*. Language A regular or A regular set if $A = L(M)$ (for some M).

$\Leftrightarrow$ $A = L(\alpha)$ for some pattern α

$\Leftarrow$ $A = L(\alpha)$ α is regular expression

ϵ pattern $\neq$ ϵ string

(Notion: ϵ)

ϕ pattern $\neq$ ϕ set

(Notion: Φ)

Σ : Alphabet $\forall\ a \in \Sigma$ $L(a) = \{a\}$

$L(\epsilon) = \{\epsilon\}$ string

$L(\Phi) = \phi$ $L(\#) = \Sigma$

$L(@) = \Sigma^*$

binary $+, \cap, \cdot$

unary $+, *, \sim$

apsara

Date: __.__.____

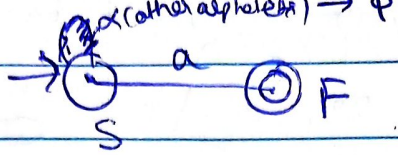
$L(\alpha+\beta) = L(\alpha) \cup L(\beta)$

$L(\alpha \cap \beta) = L(\alpha) \cap L(\beta)$

L(reg. expression) is reg. set

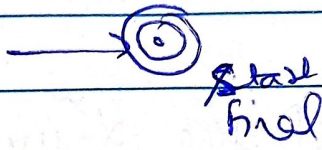
$L(a) = \{a\}$ $\quad a \in \Sigma$ $\quad \Delta(S, b) = \phi$ $\quad b \in \Sigma \setminus \{a\}$

Not DFA



Not DFA

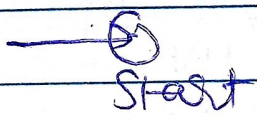
$L(\epsilon) = \{\epsilon\}$



$\Delta(S, \alpha) = \phi \quad \forall \alpha \in \Sigma$

Not DFA

$L(\phi) = \phi$



A reg. set & B reg set then $A \cap B$ reg. set

$L(B \cap \gamma) = L(\beta) \cap L(\gamma)$

reg reg

Automata ⇒ Reg. Expression $M = (Q, \Sigma, \Delta, S, F)$

NFA, Goal: find α, reg expression s.t. $L(M) = L(\alpha)$

$X \subseteq Q$

α_{uv}^{*} $\quad u, v \in Q$

α_{uv}^{*} is the pattern matched by $x = x_0 \ldots x_k$ has the prop. $v \in \hat{\Delta}(u, x)$ & $\exists$ a path of states $\in X \cup \{u, v\}$

$\forall x_0, \ldots x_k \in \Sigma$

Define / observe Inductively:

apsara

Date: / /

$u \neq v$ $\quad a_1, a_2, \ldots, a_k \in \Sigma$ with $a_i \in \Delta(u, q_i)$

$$\alpha^{\phi}_{uv} = \begin{cases} a_1 + \cdots + a_k & k \geq 1 \\ \phi & \text{if } k = 0 \end{cases}$$

$u = v$

$$\alpha^{\phi}_{uv} = \begin{cases} a_1 + \cdots + a_k + \varepsilon & k \geq 1 \\ \varepsilon & \text{if } k = 0 \end{cases}$$

Let we have define $\alpha^* \forall X$ with size $|X| - 1$

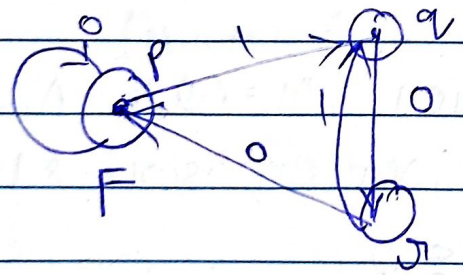
→ Define α^{X}_{uv}

$q \in X$

$$\alpha^{*}_{uv} \overset{\text{defn}}{=\!=} \alpha^{X-\{q\}}_{uv} + \alpha^{X-\{q\}}_{uq} \left(\alpha^{X-\{q\}}_{qq}\right)^{*} \alpha^{X-\{q\}}_{qv}$$

EXERCISE = Submission will be Considered for taken extra credit.

~~Prove~~ Prove that (#) independent of choice of q.



$$\alpha^{\{p,q\}^*}_{qq} = \varepsilon + (01)^* + 000^*1$$

why this?

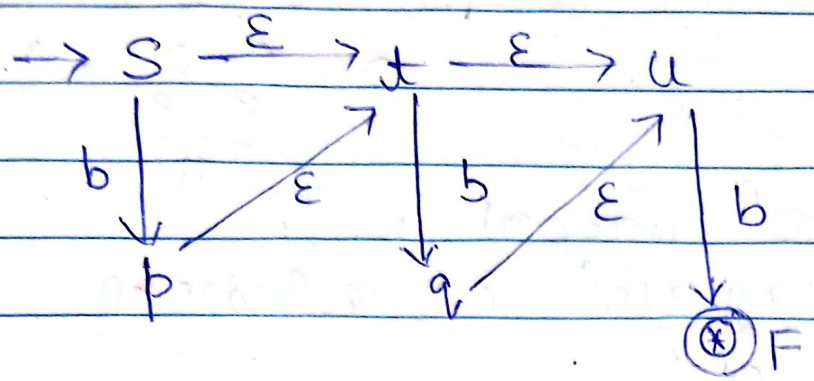
Q. Take all $x \in \{0,1\}^*$ which are palendrome show that this set is not regular.

Page No.: _____

Date: __.__.____

9/9/22 **A special NFA: ε-transition** → An ε-transition $p \xrightarrow{\varepsilon} q$ is a possible transition between states p and q that can happen any time.

Exm



If the machine is in state s & the input symbol is b then it may:

- read b and move to p.

or · slide to t without reading an input symbol, then read b and move to q.

or · Slide to t without reading a symbol then slide to u without an input symbol then read the b and move to state q.

Exc: for any such ε-transition NFA show that ∃ an equivalent usual NFA and hence an equivalent DFA.

$$p \xrightarrow{\varepsilon} q_i \quad b \in \Sigma$$

ε-transition

$$\Delta'(p,b) = \Delta(p,b) \cup \{q_1, \ldots, q_n\} \quad \forall b.$$

Exc: Show that $\{x \in \{a\}^* \mid |x| \text{ is divisible by } 3 \text{ or } 5\}$ is regular

Exc: $\Sigma = \{0,1\}$. $A = \{w \in \Sigma \mid w \text{ is palindrome}\}$
Show that A is not regular.

apsara

Date:

$\in \in A$, $0 \in A$, $1 \in A$, $010 \in A$, $0110 \in A$,
$00111 \notin A$

$0^n 1 0^n \in A$. If $A = L(M)$ where $M = (Q, \Sigma, \delta, \ldots)$

Take $n > |Q|$ so for $0^n 1 0^n$

$0 \ldots 0 - 0 - - 0 \; x \; yz \in 0^n$

$x y^i z 1 0^n$ accepted by M

But $\underbrace{x y^i z}_{0^{n+k}} 1 0^n$ not palindrome.

Remark :- A can generated by simple recursion !!

Define if $A \subseteq \Sigma^*$ 1. $\in, 0, 1 \in A$

2. if $x \in A$ then $0 \cdot x \cdot 0$ and $1 \cdot x \cdot 1 \in A$

Q. Can you similarly construct $\{0^n 1^n\}$?

Context free grammer :- Notation for expressing recursive def^n of language.

CFG for palindrome :- 1. $P \to \in$

2. $P \to 0$

3. $P \to 1$

4. $P \to 0P0$ [If $P \to x$ then $P \to 0x0$]

5. $P \to 1P1$ [If $P \to x$ then $P \to 1x1$]

0110
01011010

apsara

Date: __.__.____

Defⁿ of CFG :- 1. ∃ a finite set of symbols that form the strings of the language being formed (∑0,1? for the palindrome example) called terminals.

2. ∃ another finite set of variables, where each variable represent a language (for palindromes it is ∑P but in general can be a larger set)

3. One of the non-terminals is specified as a start symbol, which represents the language being defined (P for palindrome again)

4. ∃ a finite set of production rules that represents the recursive define of a language. Each production consist of

   (a) A variable that is being defined by the production.

   (b) Production symbol ⟶

   (c) A string of zero or more terminal and variables called the body of the production represents one way to form strings in the language of the variables of the head In doing so we leave the terminals unchanged a substitute for each variable of the body by any string that is known to be in the language.

Defⁿ :- Languages generated by context free grammers are called context free languages (CFL)

Ex - Palindromes.

Thm - Pumping Lemma **Imp for Exm** for all CFL