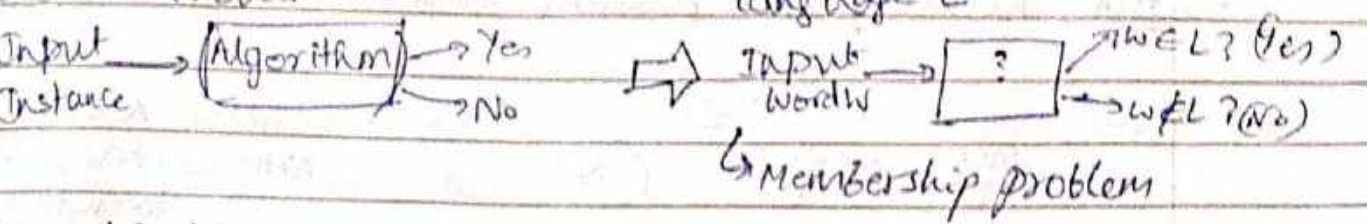
DOMS | Page No.
Date 14/12/2021

L-1

→ we try to encode problems as languages

Decision Problem



↳ Membership problem

→ Formal Definition of Language

- An alphabet is a finite nonempty set of letters or symbols.
  ↳ Denoted by Σ.
- word : A finite, possibly empty sequence of symbols from an alphabet Σ.
- EMpty word is ε. $|\varepsilon| = 0$ $|w|$ = size/length of word
- Concatenation of 2 words:

$$w_1 = a_1 a_2 \cdots a_n, \quad w_2 = b_1 b_2 \cdots b_k$$
$$w_1 \cdot w_2 = a_1 \cdots a_n b_1 \cdots b_k$$

- For ε, $w \cdot \varepsilon = \varepsilon \cdot w = w$
- $w^1 = w$, $w^{k+1} = w^k \cdot w$
- $(ab)^3 = (ab^2) \cdot (ab) = (ab) \cdot (ab) \cdot (ab) = ababab$

- Set of all words over an alphabet Σ is ~~dont~~ denoted by Σ^*.

- $\Sigma^0 = \{\varepsilon\}$ Σ^k = Set of all k-length words belonging to Σ.
- $\Sigma^{k+1} = \Sigma^k \cdot \Sigma$ $\Sigma^* = \bigcup_{i \in \{0,1,\ldots\}} \Sigma^i$

⁂ A language L is a subset of Σ^*. (i.e., $L \subseteq \Sigma^*$)

Ex) If $\Sigma^* = \{a, b\}$

possible languages: $L_1 = \{\varepsilon\} = \phi$ $L_3 = \{a, b, bb, bbb, \ldots\}$

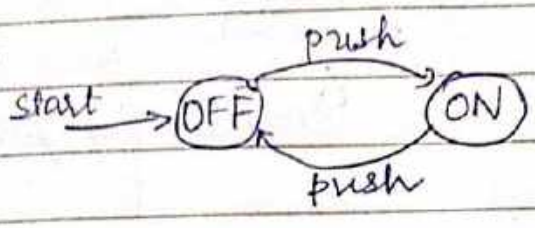
$L_2 = \{aa, bb\}$ . . . .

→ **Goal:** Finite Representations of Infinite languages

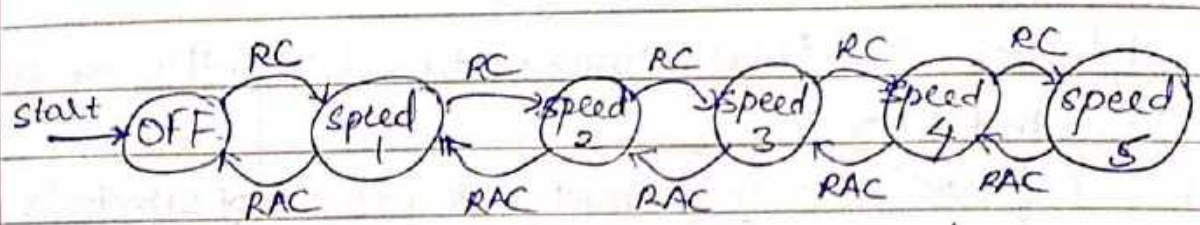
2

DFA EXAMPLES (Deterministic Finite Automata)

→ fan, initially off, pushing button changes states



circles = ~~Actions~~ States
Arrows = change in state due to actions/alphabet (transition)

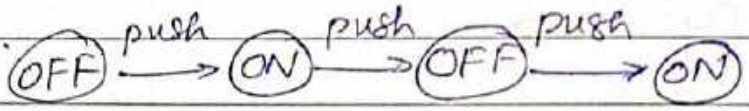


RC = Rotate clockwise
RAC = Rotate Anticlockwise

↓

No. of states are finite, so it is a finite state machine

Ex) What happens if I/P is w = (push)(push)(push) for OFF-ON case? (i.e. specify run of the machine)



→ ~~Let Σ~~

Consider an alphabet $\Sigma = \{a, b\}$. List ~~of~~ the language such that $L = \{w \mid 2 \mid |w|\}$.

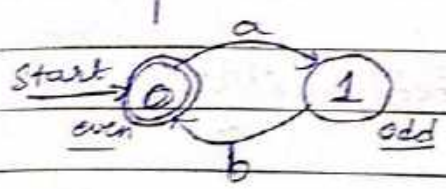
Sol^n

ε ∈ L
aa ∈ L
ab ∈ L
ba ∈ L
bb ∈ L
aaaa ∈ L

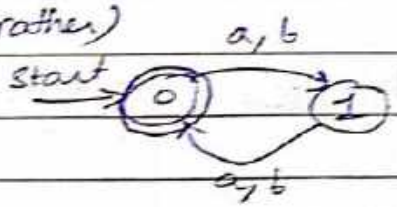
(all even strings)
length

Now, consider an automata system, of which I/P is from Σ^*

$w = ab$



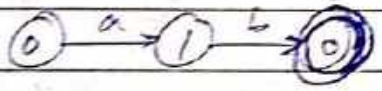
(or rather)



This accepts only even-length words!

- Run automata of characters of word as I/P, one by one.
- Double-circled state is a "final" or "accepting" state.
- When run terminates, if current state is accepting, then there is notion of acceptance

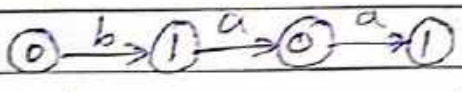
Ex) Run of $w = ab$ on above automata:



final state is accepting state
$\Rightarrow w \in L$

- Run that ends at accepting state is called 'accepting run'.

Ex) Run of $w = baa$ on above automata.



(No other possible runs)
final state is non-accepting.
$\Rightarrow w \notin L$

↓
'Rejecting run'

- Always begin runs at initial state, for new I/P - marked by arrow with no origin state.

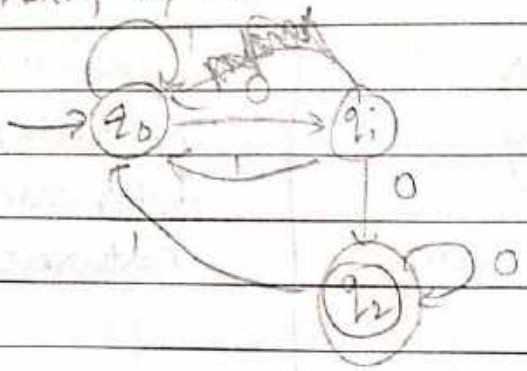
→ Deterministic Finite Automata

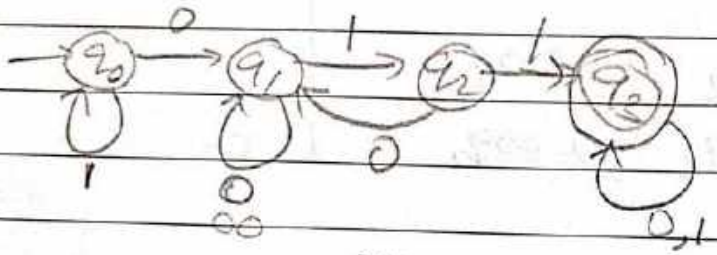
- It is given by $(Q, \Sigma, \delta, q_0, F)$
  - Q = Set of states (FINITE) → Non-empty.
  - Σ = Alphabet/actions by which we/environment interact w/ machine.
  - q_0 = Initial state ($q_0 \in Q$, mandatorily)
  - F = Set of final states, ($F \subseteq Q$, mandatorily)
  - $\delta : (Q \times \Sigma) \to Q$

ending w/ 00

0100

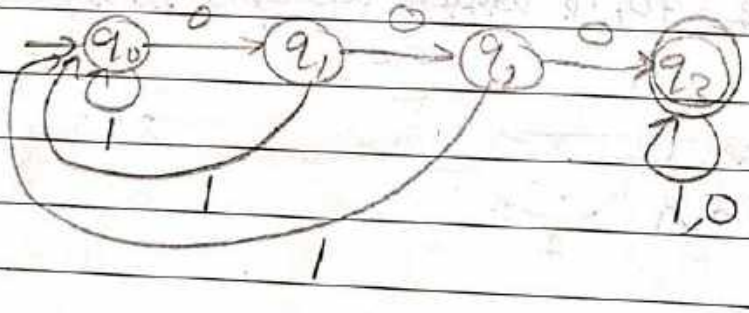
1000





0011

0010



L-3

$\delta : Q \times \Sigma \to Q$ DFA, $M = (Q, \Sigma, \delta, q_0, F)$

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

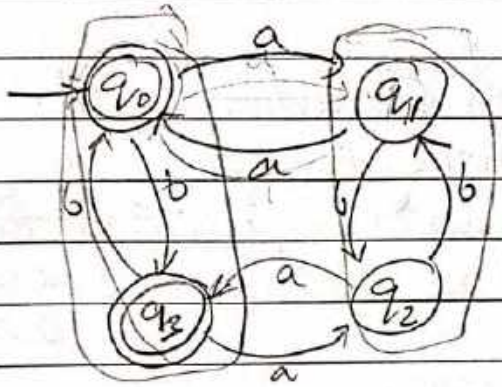
$Lang(M) = \{ w \in \Sigma^* \mid M \text{ has accepting run on } w \}$

$= \{ w \in \Sigma^* \mid \hat{\delta}(q_0, w) = q_f \in F \}$

$= \{ w \in \Sigma^* \mid \hat{\delta}(q_0, w) \cap F \neq \phi \}$

Ex) $\Sigma = \{a, b\}$

$L = \{ w \in \Sigma^* \mid w \text{ has even no. of a's} \}$.

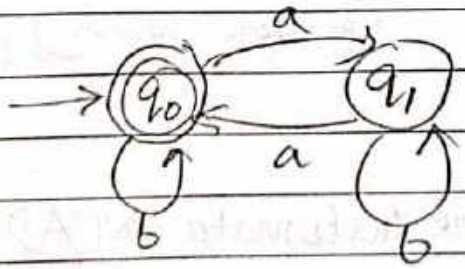


a b

aaaa ∈ q_0

q_0

aabaa



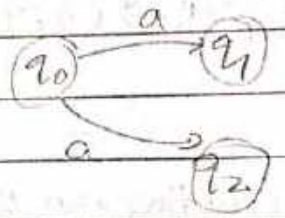
To reach final state, you have to cross even no. of a's!

→ Regular Languages

$L \subseteq \Sigma^*$ is a regular language if $\exists$ a DFA 'M' such that $Lang(M) = L$

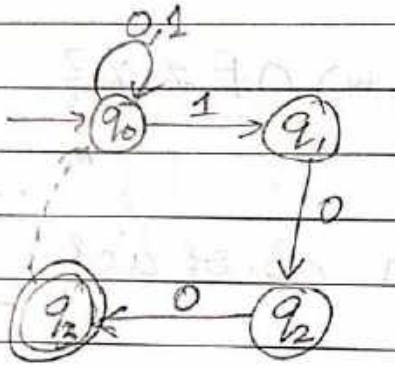
→

→ Non Deterministic



(not one-one)

Ex) $L = \{w \mid w \text{ ends with } 100\}$. Design NFA.
$(\Sigma = \{0, 1\})$



Accepting:
$q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_2 \xrightarrow{0} q_3$

Rejecting:
$q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_0$ ✗

- Multiple runs on same word.
- Atleast ONE accepting run is enough for a word to have an acceptance notion.

$L(M) = \{w \in \Sigma^* \mid \text{NFA } M \text{ has} \geq 1 \text{ accepting run over } w\}$

- Incomplete run (stuck) ⇒ leave it as it is (δ-def^n undefined)

NFA makes guesses

⇒ Non-Deterministic Finite Automata (NFA)

$M = (Q, \Sigma, \delta, q_0, F)$

$\delta : (Q \times \Sigma) \to \mathcal{P}(Q)$ → power set

$\delta(\text{current state}, \text{I/P})$ → multiple possibilities of final states.

$|P(Q)| = 2^{|Q|}$

δ	0	1
q_0	$\{q_0\}$	$\{q_0, q_1\}$
q_1	$\{q_2\}$	ϕ
q_2	$\{q_3\}$	ϕ
q_3	ϕ	ϕ

Transition Table for prev example.

$\delta : Q \times \Sigma \to P(Q)$

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

$\hat{\delta}(q, wa) = \hat{\delta}(q, w)\, \delta\, a$

$\hat{\delta}(q, \epsilon) = q$

$\hat{\delta}(q, a) = \delta(q, a)$

$\hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a)$

$\hookrightarrow \forall\, q' \in \hat{\delta}(q, w), \quad \delta(q', a)$

$$\hat{\delta}(q, wa) = \bigcup_{q' \in \hat{\delta}(q, w)} \delta(q', a)$$

$L(M) = \{ w \in \Sigma^* \mid \hat{\delta}(q_0, w) \cap F \neq \phi \}$

$\hat{\delta}(q_0, w)$ → set of all states where we end up at after processing 'w'.

$q' \in \hat{\delta}(q_0, w)$ → one specific run/ending state.

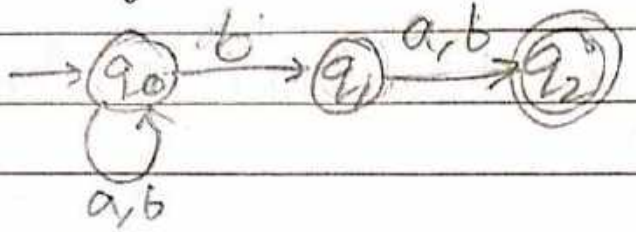
$\delta(q', a)$ → possible states if we process a after processing possible q' values

Date / /

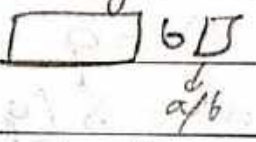
Ex) $\Sigma = \{a, b\}$

$L = \{w \in \Sigma^* \mid 2^{nd} \text{ last letter in } w \text{ is } b\}$

Design NFA.



only way to get to q_2,



aaba

$L = L(M)$

(language of this machine is the reqd. language)
(accepted)

DOMS | Page No.
Date 17/12/2021

L-4

- NFA has modelling advantages. But is it more powerful than DFA? (expressibility)

Does $\exists L \subseteq \Sigma^*$ | we can construct NFA but no DFA?

2^{Σ^*} languages for some Σ^*

All $L \subseteq \Sigma^*$ such that $\exists$ DFA for $L \Rightarrow$ Regular language.

Does $\exists L$: it has NFA but not DFA?

pf: T.P. DFA $\subseteq$ NFA (All DFAs are NFAs) (or vice-versa?)

$\Rightarrow$ For some DFA D, $lang(D) = L$.

$\Rightarrow \exists$ NFA N : $lang(N) = L$

pf:

For D, we have $D = (Q_D, \Sigma, \delta_D, q_0, F_D)$

~~We want~~ Also $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$

$\delta_D : Q_D \times \Sigma \to Q_D$

$\delta(q, a) \to q'$

$\delta_N : Q_N \times \Sigma \to 2^{Q_N}$ ($P(Q_N)$)

$(q, a) \to \{\ldots\}$ maps to multiple vertices

let $Q_N = Q_D$,

$\Sigma = \Sigma$

$q_0 = q_0$

$F_N = F_D$

$\delta_N(q, a) \to \{\delta_D(q, a)\}$

singleton set (one-one mapping)

$\hat{\delta}_D(q_0, w) = \hat{\delta}_N(q_0, w)$

$\rightarrow$ can we do it other way round? Probably not.

No

(So far, NFAs are at least as powerful as DFAs)

$F_D = \{S / S \cap F_N \neq \emptyset\}$

DOMS | Page No. | Date / /

Try NFA → DFA.

$N = (Q_N, \Sigma, \delta_N, q_0, F_N) \xrightarrow{?} D = (Q_D, \Sigma, \delta_D, q_0, F_D)$

$lang(N) = L$ $\quad$ $lang(D) = L$

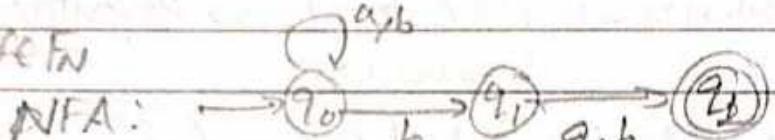
make new states that are subsets of Q.

$Q_D = 2^{Q_N}$ $\quad$ $S \subseteq 2^{Q_N}$

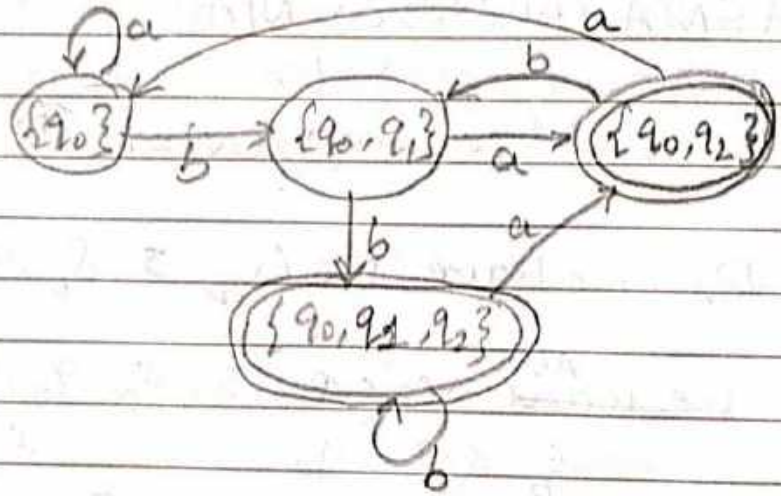
$\delta_D(S, a) = \bigcup_{p \in S} \delta_N(p, a)$

All states containing $f \in F_N$ would be final state in DFA

NFA:



DFA:



$q_2 \xrightarrow{a} X$

$q_2 \xrightarrow{b} X$

To proove $lang(N) = lang(D)$

It would be sufficient to prove that:

$\forall w \in \Sigma^*, \quad \hat{\delta}_N(\{q_0\}, w) = \hat{\delta}_D(\{q_0\}, w)$

If $w \in L(N)$, $\Rightarrow w \in L(D)$

$\hat{\delta}_N(q_0, w)$ $\quad$ $\hat{\delta}_D(\{q_0\}, w)$

$\hat{\delta}_N(q_0, w) = S$ $\quad$ $S \in F_D$ By def^n F_D is accepting state set

$S \in F_N$

Pf: By induction on $|w|$.

Base case: $|w| = 0$

$\hat{\delta}_N(q_0, \varepsilon) = \delta_N(q_0, \varepsilon) = q_0$

DOMS | Page No.
Date / /

$\hat{\delta}_D(\{q_0\}, \epsilon) = \delta_D(\{q_0\}, \epsilon) = \delta_N(q_0, \epsilon) = q_0$

Ind. hyp: If $|w| = n \Rightarrow$ To prove $|w| = n+1$ works.

Ind. step: Let $\hat{\delta}_N(q_0, w) = \hat{\delta}_D(\{q_0\}, w)$ — hyp.

where $|w| = n+1$

$w = xa$, $|x| = n$.

for $n+1$:

LHS: $\hat{\delta}_N(q_0, xa) = \delta_N(\hat{\delta}_N(q_0, x), a)$ — ①

$|xa| = n+1$.

RHS: $\hat{\delta}_D(\{q_0\}, xa) = \delta_D(\hat{\delta}_D(\{q_0\}, x), a)$ — ②

By ind. hyp, $\hat{\delta}_N(q_0, x) = \hat{\delta}_D(\{q_0\}, x)$ for $|x| = n$

$\Rightarrow \delta_N(\hat{\delta}_N(q_0, x), a) = \delta_D(\hat{\delta}_D(\{q_0\}, x), a)$

$\Rightarrow \hat{\delta}_N(q_0, xa) = \hat{\delta}_D(\{q_0\}, xa)$

i.e. $\hat{\delta}_N(q_0, x) = \{p_1, \cdots p_n\}$

$\delta_N(\{p_1, \cdots p_n\}, a) = \bigcup_{i=1}^{k} \delta_N(p_i, a)$ All possibilities

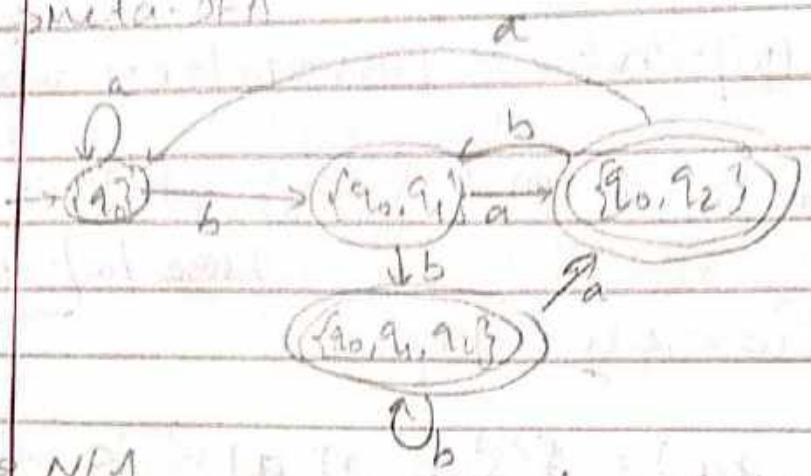
also

$\hat{\delta}_D(q_0, x) = \{p_1, \cdots p_k\}$

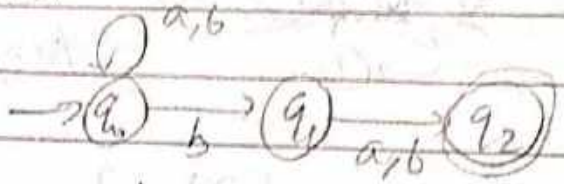
$\delta_D(\{p_1, \cdots p_k\}, a) = \bigcup_{i=1}^{k} \delta_N(p_i, a)$

So, $\delta_N(q, a) = \delta_D(q, a)$.

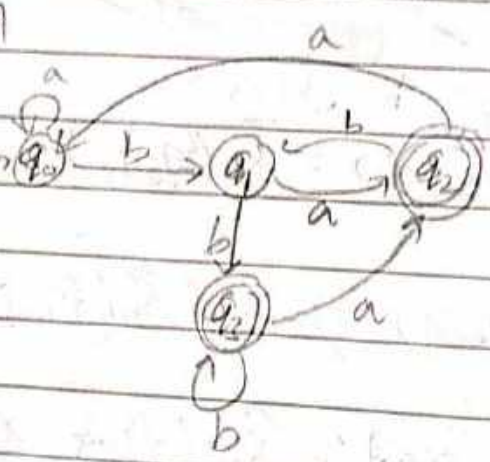
→ Given DFA



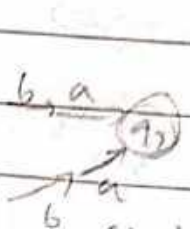
→ NFA



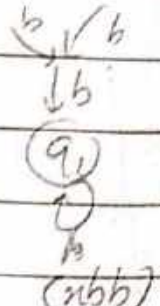
→ DFA



(Rename the sets)



(Aba)



(abb)

DOMS | Page No.
Date 21/12/2021

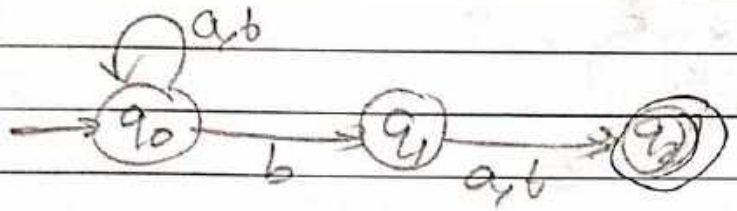
L-5

- NFAs and DFAs are equally powerful!
  Both of them accept regular languages.

→ subset construction: (NFA to DFA)

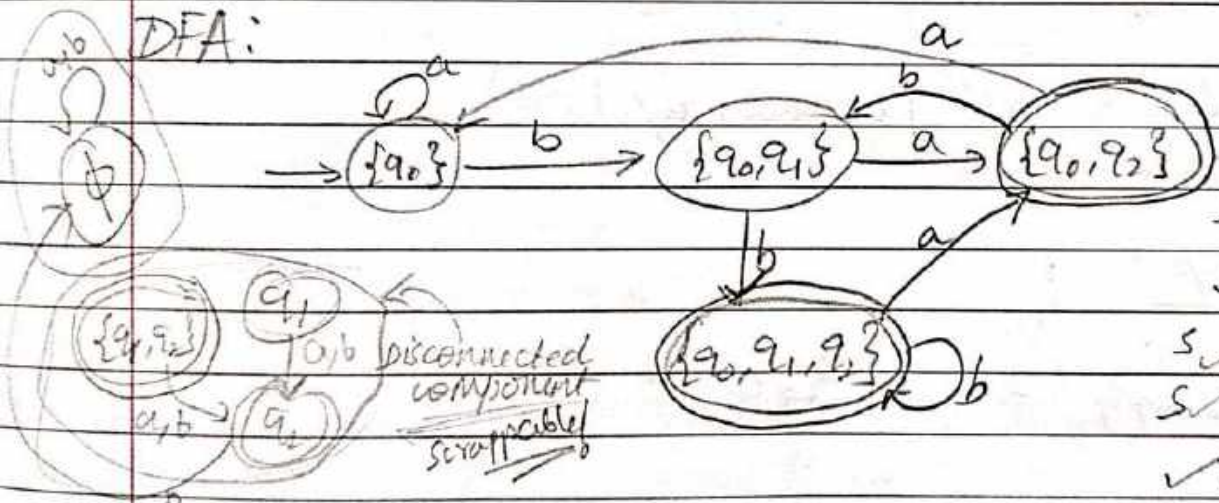
ex) $L = \{ w \in \{a,b\}^* \mid 2^{nd} \text{ last letter is } b \}$

NFA:



δ_N	a	b
$\to q_0$	q_0	q_0, q_1
q_1	q_2	q_2
$*q_2$	ϕ	ϕ

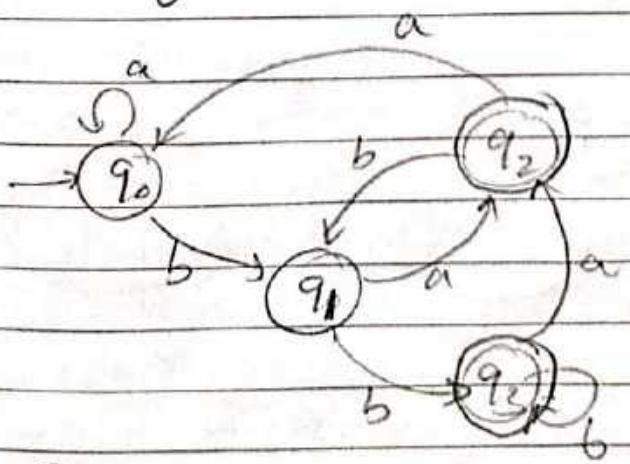
DFA:



δ_D	a	b
S ✓ ϕ	ϕ	ϕ
✓ $\{q_0\}$	$\{q_0\}$	$\{q_0,q_1\}$
S ✓ $\{q_1\}$	$\{q_2\}$	$\{q_2\}$
S ✓ $\{q_2\}$	ϕ	ϕ
✓ $\{q_0,q_1\}$	$\{q_0,q_2\}$	$\{q_0,q_1,q_2\}$
S $\{q_1,q_2\}$	$\{q_2\}$	$\{q_2\}$
✓ $\{q_0,q_2\}$	$\{q_0\}$	$\{q_0,q_1\}$
✓ $\{q_0,q_1,q_2\}$	$\{q_0,q_2\}$	$\{q_0,q_1,q_2\}$

S = scrap

Renaming:



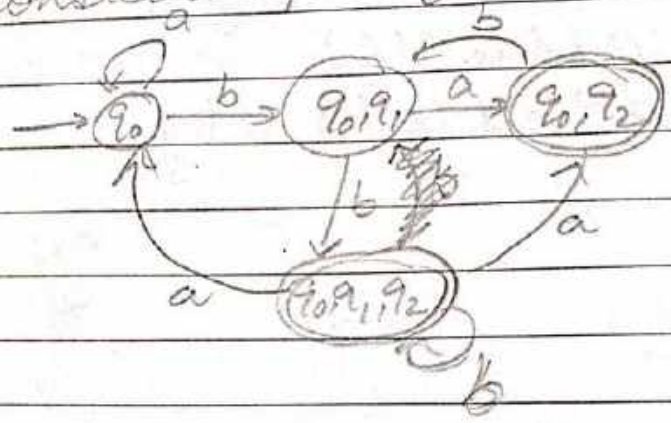
NFA → DFA
n states → 2^n states

NFA → DFA: some may be scrapped

→ Incremental construction:

- construct reachable states, as needed/encountered

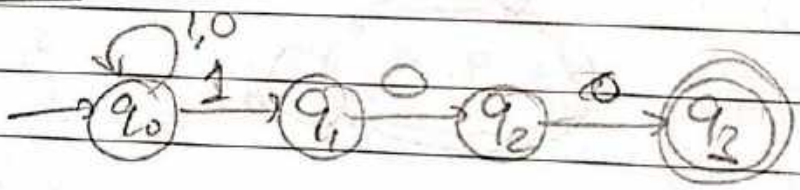
Ex) considering prev. automata,



$Q_D \subseteq 2^{Q_N}$ Only reachable subsets of Q_N

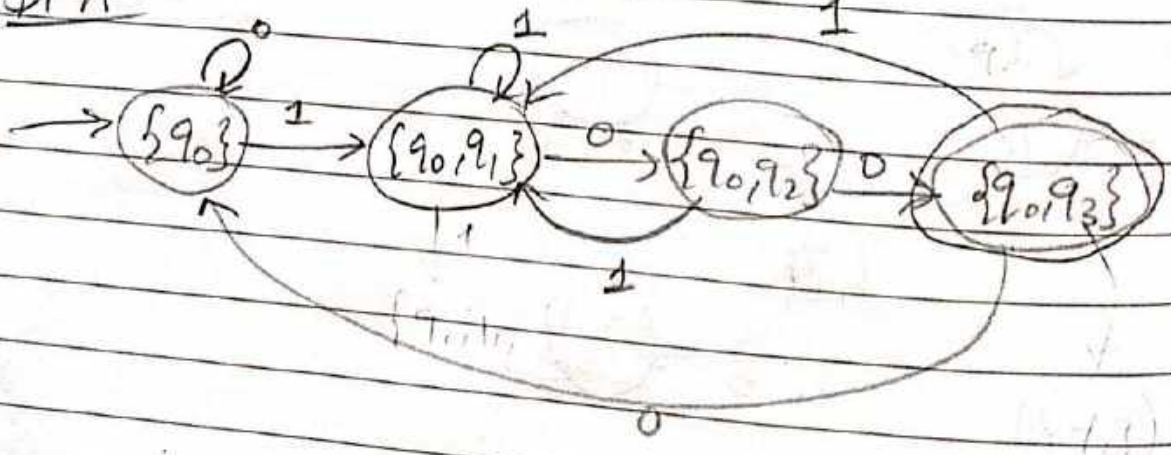
Ex) $L = \{w \in \{0,1\}^* \mid w \text{ ends w/ } 100\}$

NFA:



Subset construction → 16 states
Do incrementally.

DFA



→ Subset construction

$|Q_N| = n \rightarrow 2^n$ states to be constructed
NFA

→ Incremental construction

~~$|Q_N| = 2^n$~~

$|Q_N| = n \rightarrow \leq 2^n$ states (usually $< 2^n$)
NFA

NFA	DFA	
$n=3$	$\|Q_D\|=4$	$(2^n/2)$
$n=4$	$\|Q_D\|=8$	
n	$\rightarrow 2^{n-1}$	DFA has exponential size w.r.t its original NFA.
$n+1$	$\rightarrow 2^n$	
$O(n)$	$\rightarrow O(2^n)$	

→ Claim : For $(n+1)$-state NFA, there exists no equivalent DFA with less than 2^n states.
↳ for $|\Sigma|=2$ ↳ Lower bound

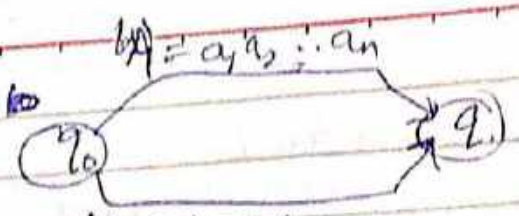
b^n.

Pf: by contradiction:
Assume $\exists$ DFA D' w/ $< 2^n$ states, corresponding to an $(n+1)$-state NFA, N'

consider n-length strings (2^n of them)
check behaviour of D on the n-length strings.
Let $(q_0 \xrightarrow{w} q)$
consider 2^n strings · but $< 2^n$ states we have given

So, atleast 2 Str are in same state.
Pigeonhole principle (contradiction) flows

Date / /



$L = \{ w \in \{a,b\}^* \mid n^{th}$-last letter is $b \}$

$\exists$! DFA w/ $< 2^n$ states, use contradiction.

Assume $a_i \neq b_i$.

$\Rightarrow a_i = a,\ b_i = b$

(or) $a_i = b,\ b_i = a$

But the words w_1 and w_2 are:

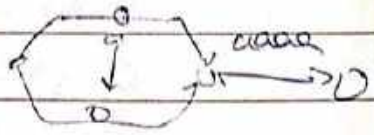
$w_1 = a_1 a_2 \ldots a \ldots a_n$

$w_2 = b_1 b_2 \ldots b \ldots b_n$

Only $w_2 \in L$ (n^{th} letter from last is b)

$\Rightarrow w \notin L$.

$\Rightarrow$ contradiction



consider:

$a_1 \ldots a_{i-1}\ a_i\ a_{i+1} \ldots a_n\ aa \ldots a$

$b_1 \ldots b_{i-1}\ b_i\ b_{i+1} \ldots b_n\ aa \ldots a$ ($i-1$ a's)

$a_i \neq b_i$

n^{th} from last

we want n^{th} letter from end to be equal to b.

but,

a $w_1 \notin F$. ($w_1 \in L$)

$w_2 \in F$ ($w_2 \in L$)

(can do this for a_1, b_1 for further)

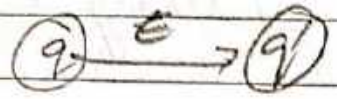
$a_1 = a$, $b_1 = b$ $\Rightarrow$ $a a_2 \ldots a_n \notin L \rightarrow w_1 \notin F$

$a b_2 \ldots b_n \in L \rightarrow w_2 \in F$

L-6

ε-NFA (i.e. NFA w/ ε-transitions)

$(\epsilon \notin \Sigma)$



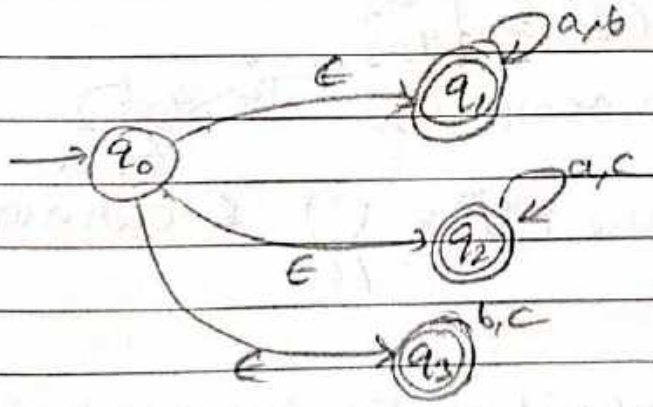
$(Q, \Sigma, \delta, q_0, F)$, $\delta : Q \times \Sigma \cup \{\epsilon\} \to 2^Q$

Ex) $\delta(q_0, a) = q_1$
$\delta(q_i, \epsilon) = q_j$

Ex) $\Sigma = \{a, b, c\}$ $\quad a\epsilon = a = \epsilon a$
word : aa ⓜ εεεεaεa

- we can change state without processing input.

Ex) $\Sigma = \{a, b, c\}$
$L = \{ w \in \Sigma^* \mid w \text{ does not use all letters of } \Sigma \}$.



- Doesn't have c ⇒ q_1
- Doesn't have b ⇒ q_2
- Doesn't have a ⇒ q_3

The only requirement is that there exists a letter in Σ not used in ω.

If w = cba

One of the Runs are :-

$q_0 \xrightarrow{\epsilon} q_1 \xrightarrow{c} X$

$q_0 \xrightarrow{\epsilon} q_2 \xrightarrow{c} q_2 \xrightarrow{b} X$

$q_0 \xrightarrow{\epsilon} q_3 \xrightarrow{c} q_3 \xrightarrow{b} q_3 \xrightarrow{a} X$

$\hat{\delta}(q,\epsilon) = \delta(q,\epsilon) \cup \delta(q,\epsilon\epsilon) \cdots$

DOMS | Page No.
Date / /

→ ϵNFA is no more powerful than NFA.

→ ϵ-closure (q) = Set of All states that are reachable from q, only by using ϵ-transitions.
= $\{r \mid p \subseteq r$, where p is ϵ-closed under $q\}$
↳ $p \in \epsilon$-closure(q)

In prev. example,
ϵ-closure(q_0) = $\{q_1, q_2, q_3, q_0\}$

($q_i \in \epsilon$ is implicit.)

Firstly, ϵ-closure(q) = $\{q\}$
Then expand upon it using ϵ-transitions.

Ex) In prev. ex.
ϵ-closure(q_0) = $\{q_0, q_1, q_2, q_3\}$
(q_1) = $\{q_1\}$
(q_2) = $\{q_2\}$
(q_3) = $\{q_3\}$
ϵ-closure(q_1, q_2) = $\{q_1, q_2\}$

In general,

$$\epsilon\text{-closure}(S) = \bigcup_{p \in S} \epsilon\text{-closure}(p)$$

$\hat{\delta}(q_0, w)$ = set of all states reachable from q_0 after processing w.

$\hat{\delta}(q, \epsilon) = \epsilon$-closure($\delta(q, \epsilon)$)

$\hat{\delta}(q, w) = \hat{\delta}(q, xa)$ $\quad w = xa$.

Ind hyp: $\hat{\delta}(q_0, x) = \{p_1, \cdots p_k\}$

$\hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a)$

$\hat{\delta}(q, xa) = \bigcup_{i=1}^{k} \delta(A, a)$

$\bigcup_{j=1}^{m} \epsilon\text{-closure}(r_j)$ ← ϵ-closure ← $\{r_1, \ldots, r_m\}$.

(Each, take union)

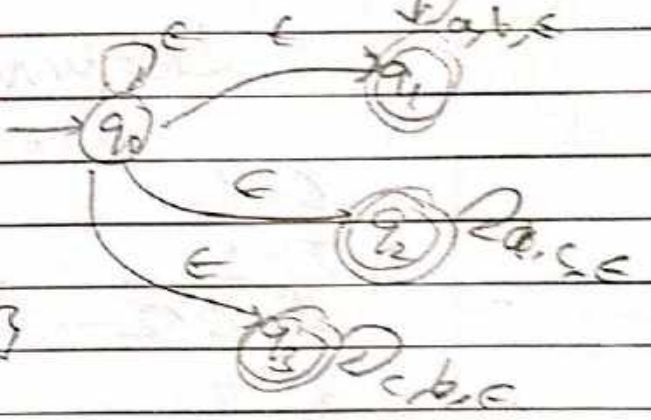
Consider an ϵ-NFA N.

$\text{lang}(N) = \{ w \in \Sigma^* \mid \hat{\delta}(q_0, w) \cap F \neq \phi \}$

Ex) $w = ab$.

$\hat{\delta}(q_0, ab) = ?$

$\hat{\delta}(q_0, \epsilon a) = ?$



$\hat{\delta}(q_0, \epsilon) = \{q_0, q_1, q_2, q_3\}$

$\Rightarrow \hat{\delta}(q_0, \epsilon a) = \delta(\hat{\delta}(q_0, \epsilon), a)$

$= \delta(\{q_0, q_1, q_2, q_3\}, a)$

$= \{q_1, q_3\}$ (with ϵ-closure)

$\Rightarrow \hat{\delta}(q_0, ab) = \delta(\hat{\delta}(q_0, a), b)$

$= \delta(\{q_1, q_3\}, b)$

$= \{q_1\}$ (with ϵ-closure)

Take $w = abc$

$\hat{\delta}(q_0, abc) = \delta(\hat{\delta}(q_0, ab), c)$

$= \phi$ (with ϵ-closure)

$\epsilon\text{-closure}(\phi) = \phi$.

Page No.
Date 29/12/2021

L-7

(N_ϵ) ϵ-NFA ⟶ NFA (N)

$(Q_\epsilon, \Sigma, \delta^\epsilon, q_0^\epsilon, F^\epsilon)$

$L = lang(N_\epsilon)$

$\delta^\epsilon : Q_\epsilon \times \Sigma \cup \{\epsilon\} \longrightarrow 2^{Q_\epsilon}$

$\Rightarrow \delta(s, a) = \delta(\epsilon\text{-closure}(s), a)$

Better: $\epsilon\text{-closure}(\delta(\epsilon\text{-closure}(s), a))$

↳ To eliminate all ϵ-transitions from an ϵ-NFA, to convert it into NFA

$Q = Q_\epsilon$
$\Sigma = \Sigma$
$\delta^\epsilon : Q \times \Sigma \longrightarrow 2^Q$
$q_0^\epsilon = q_0$
$F = F^\epsilon$.

$\Rightarrow lang(N_\epsilon) = lang(N)$

(I/P) $(Q_\epsilon, \Sigma, \delta^\epsilon, q_0^\epsilon, F^\epsilon)$ ⟶ (O/P) $(Q, \Sigma, \delta, q_0, F)$

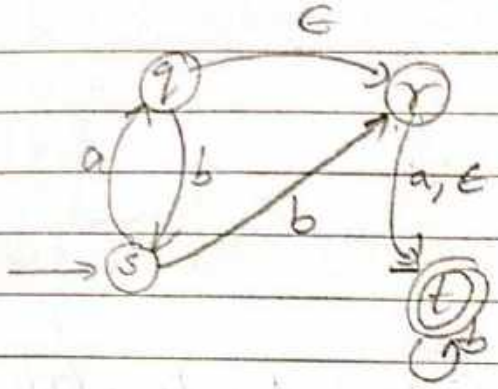
$\Sigma = \Sigma$
$Q = Q_\epsilon$
$q_0 = q_0^\epsilon$
$F = F^\epsilon$

$\delta : Q \times \Sigma \longrightarrow 2^Q$
$\delta^\epsilon : Q_\epsilon \times \Sigma \longrightarrow 2^{Q_\epsilon}$

Take $\epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q), a))$ for all $q \in Q$, $a \in \Sigma$

$\delta(q, a)$ ←

Q.1)



Convert ε-NFA to NFA

$\Sigma = \{a, b\}$

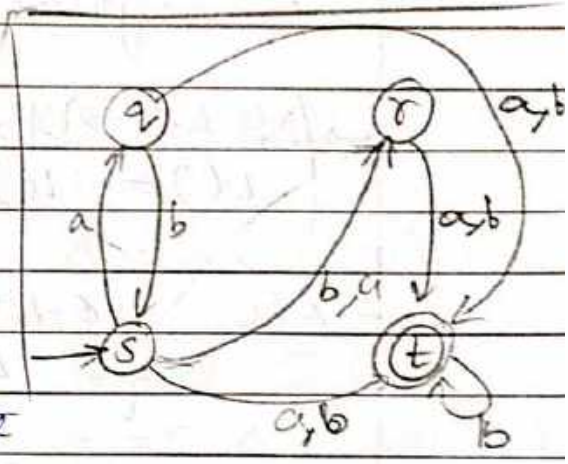
~~$\delta(s, a) = \{s\}$ ε-closure(~~

ε-closure (s) = {s}

ε-closure (q) = {q, r, t}

ε-closure (r) = {r, t}

ε-closure (t) = {t}



~~$\delta(s, a) = \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}$~~

~~δ closure~~

$\delta(\varepsilon\text{-closure}(s), a) = \delta(\{s\}, a) = \{q\}$

ε-closure ({q}) = {q, r, t} → δ(s, a)

$\delta(\varepsilon\text{-closure}(s), b) = \delta(\{s\}, b) = \{r\}$

ε-closure ({r}) = {r, t} → δ(s, b)

$\delta(\varepsilon\text{-closure}(q), a) = \delta(\{q, r, t\}, a) = \{t\}$

ε-closure ({t}) = {t} → δ(q, a)

$\delta(\varepsilon\text{-closure}(q), b) = \delta(\{q, r, t\}, b) = \{t\}$ → δ(q, b)

ε-cl

$\delta(\varepsilon\text{-closure}(r), a) = \delta(\{r, t\}, a) = \{t\}$ → δ(r, a)

ε-cl

$\delta(\varepsilon\text{-closure}(r), b) = \delta(\{r, t\}, b) = \{t\}$ → δ(r, b)

ε-cl

→ $\delta(\varepsilon\text{-closure}(t), a) = \delta(\{t\}, a) = \phi$ → δ(t, a)

ε-cl

$\delta(\varepsilon\text{-closure}(t), b) = \delta(\{t\}, b) = \{t\}$ → δ(t, b)

ε-cl

Proving L(N) = L(N')

→ Regular Languages:

→ Σ^* = word-set

→ $L \subseteq \Sigma^*$ is a language

→ 2^{Σ^*} is set of all languages

→ L is regular if $\exists$ DFA, D : $L = lang(D)$ or $L(D)$

→ $\left[\begin{matrix} NFA \longleftrightarrow DFA \\ L() \longleftrightarrow L() \end{matrix}\right]$ ↔ or NFA, N.

→ Also, $\left[\begin{matrix} \epsilon\text{-NFA} \longrightarrow NFA \longleftrightarrow DFA \\ L() \longrightarrow L() \longleftrightarrow L() \end{matrix}\right]$

→ Is L regular?
- construct DFA or NFA or ϵ-NFA

→ Set of regular langs $\subset$ set of langs

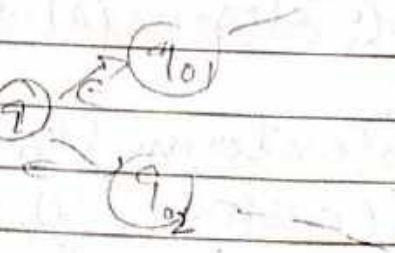
→ If L_1, L_2 are reglangs,
and
$L_1 \cup L_2 = \{w \mid w \in L_1 \text{ or } w \in L_2\}$

langs : L_1 (reg.), L_2 (reg.)

DFAs : D_1, D_2

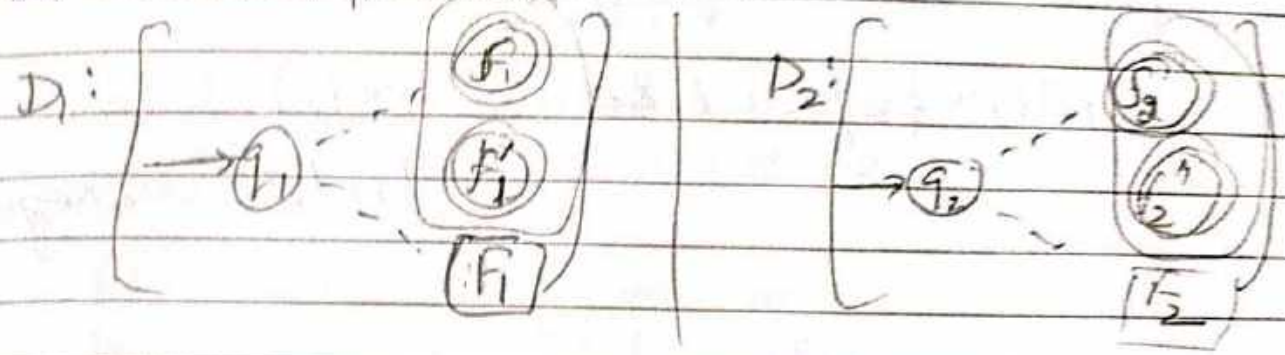
$lang(D_1) = L_1$ $lang(D_2) = L_2$

q_{01} (NFA) q_{02} (NFA)

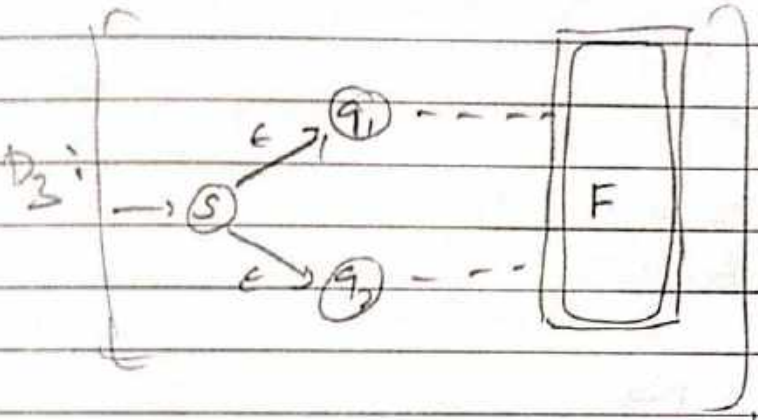


→ Union of Reglangs

~~$D_1 = (Q_1, \Sigma, \delta, q_0, F_1)$~~



For $L_1 \cup L_2$, we construct a DFA as:



✓ $F = F_1 \cup F_2$

Add a new state s.

✓ $Q = Q_1 \cup Q_2 \cup \{s\}$.

✓ $q_0 = s$ (new element)

✓ $\delta : Q \cup \{\epsilon\} \times \Sigma \to Q$

where $Q = Q_1 \cup Q_2 \cup \{s\}$

✓ $\Sigma = \Sigma_1 \cup \Sigma_2$

$L = L_1 \cup L_2 = \text{lang}(D_3)$

is regular

⇒ Reglangs are closed under union!

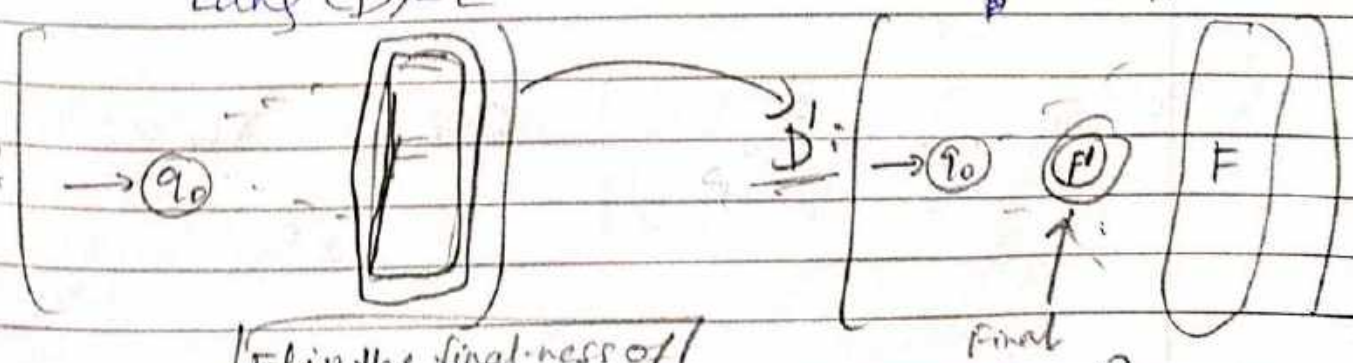
→ ~~last~~ Complement of Reglangs → Reglangs are closed under complemental

$L \subseteq \Sigma^*$ is regular

Its complement is $\overline{L} = \Sigma^* \setminus L$. Is it regular?

Lang (D) = L

$F_{D'} = Q \setminus F$



Flip the final-ness of all states

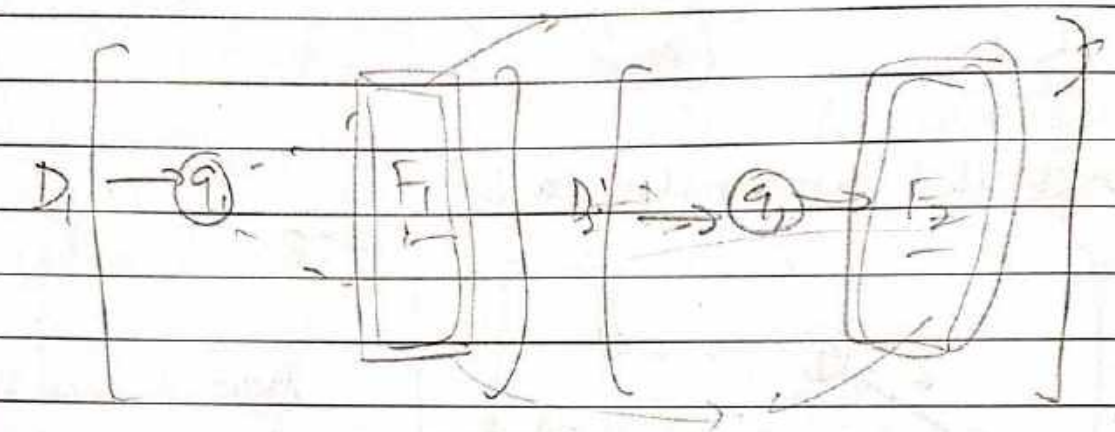
q: Final → non-FINAL

q: non-final → Final

→ Intersection of Reglangs

$L_1 \cap L_2 = \{w \mid w \in L_1 \text{ AND } w \in L_2\}$,

$L_1, L_2 \subseteq \Sigma^*$ are regular.



$$L_1 \cap L_2 = \overline{\overline{L_1 \cap L_2}} = \overline{\overline{L_1} \cup \overline{L_2}}$$

L_1 is reg $\Rightarrow \overline{L_1}$ is reg
L_2 is reg $\Rightarrow \overline{L_2}$ is reg
$\Rightarrow \overline{L_1} \cup \overline{L_2}$ is reg

$\Downarrow$

$\overline{\overline{L_1} \cup \overline{L_2}}$ is reg

$\Downarrow$

$L_1 \cap L_2$ is reg $\Leftarrow$ $\overline{\overline{L_1} \cup \overline{L_2}}$ is reg

Hence proved.

$L_1 : (\Sigma^*) \to L_1(\Sigma^* \setminus L_1)$
$L_2 : (\Sigma^*) \to L_2(\Sigma^* \setminus L_2)$
$\Rightarrow \overline{L_1} \cup \overline{L_2} : (\Sigma^* \setminus L_1 \cup \Sigma^* \setminus L_2)$

$\Downarrow$

$\overline{L_1 \cup L_2} : \Sigma^* \setminus (\Sigma^* \setminus L_1 \cup \Sigma^* \setminus L_2)$

$= (L_1 \cap L_2)$

DOMS | Page No.
Date / /

I-1

24/12/21

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

Σ^* = set of all binary strings.

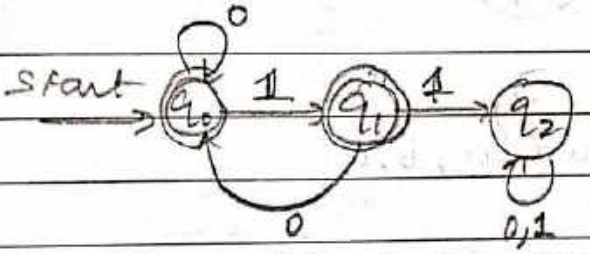
→ countably infinite

$|\Sigma^*| < |P(\Sigma^*)|$

we can use cantor's diagonalization generalized argument.

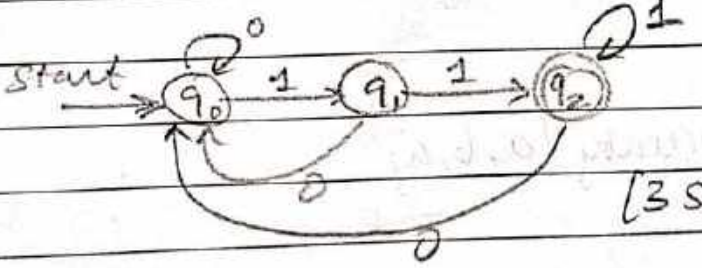
2) (a) and (b)

(i) NOT 11 contiguous subword



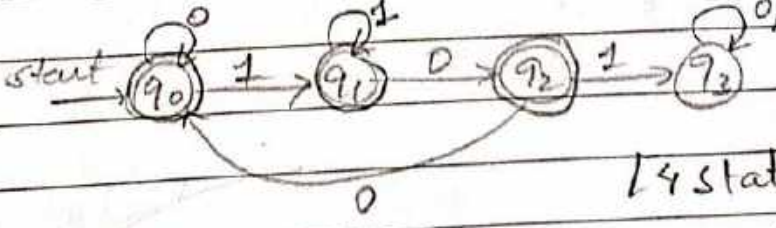
[3 states DFA]

(ii) Ends in 11



[3 states DFA]

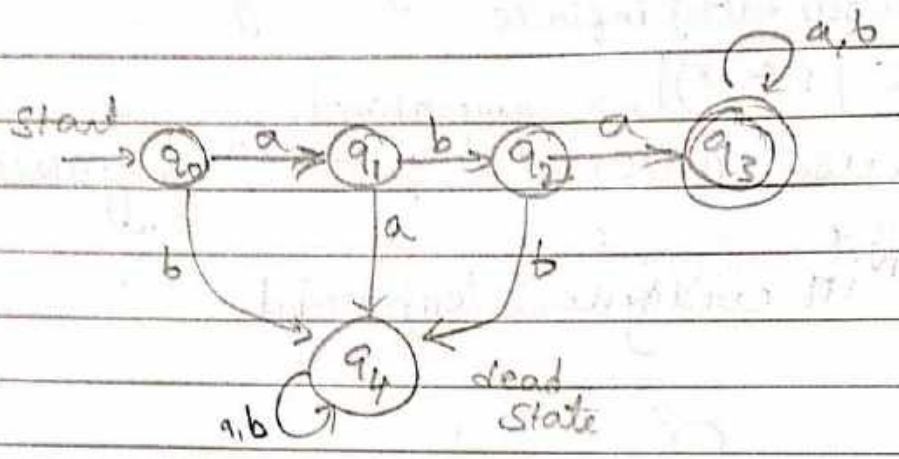
(iii) do not contain 101



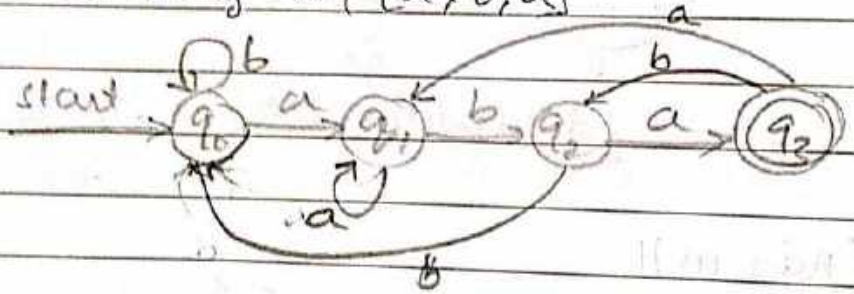
[4 states to MAKE 101, then negate/flip final-nes

3) $\Sigma = \{a, b\}$ DFAs!

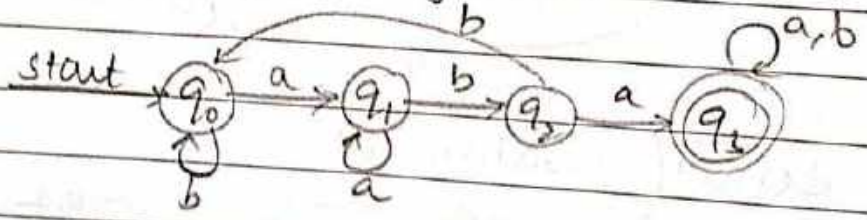
(a) starting w/ [a, b, a]



(b) ending w/ [a, b, a]



(c) containing [a, b, a]

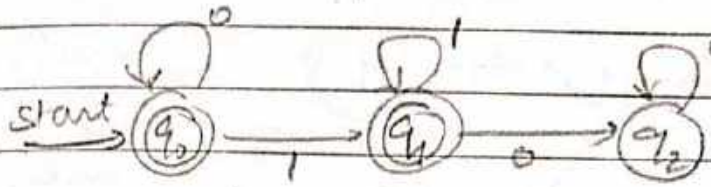


→

DOMS | Page No.
Date / /

4)

(a)

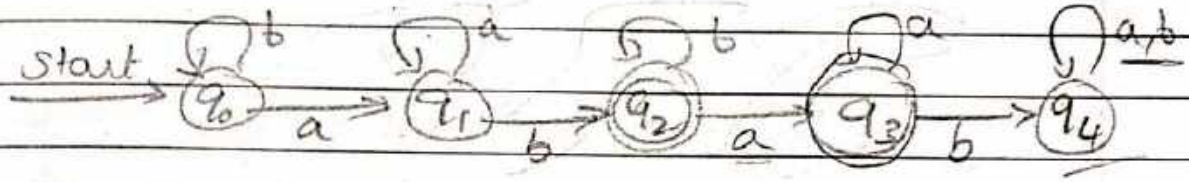


or 110

10

Does NOT contain [1,0]

(b)



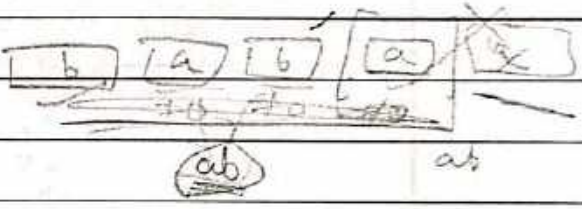
ab b---

aba a..

containing ab ___ only once

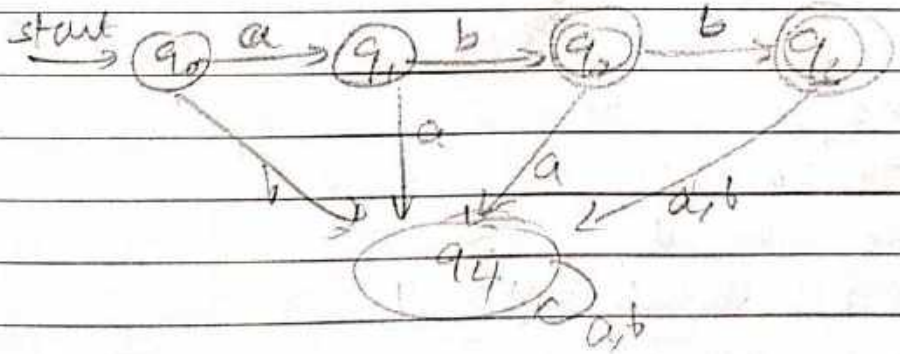
ab
aba
abba
abb

bbbab



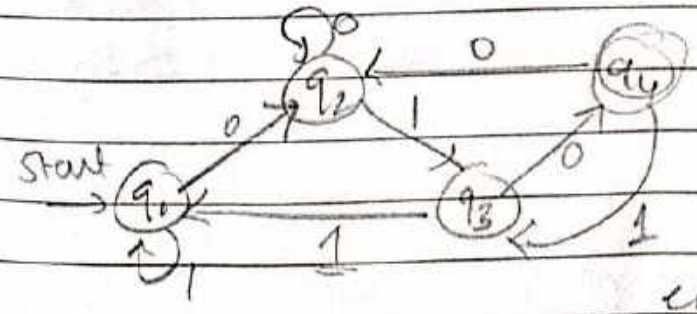
abbab

(c)



{ab, abb}

(d)

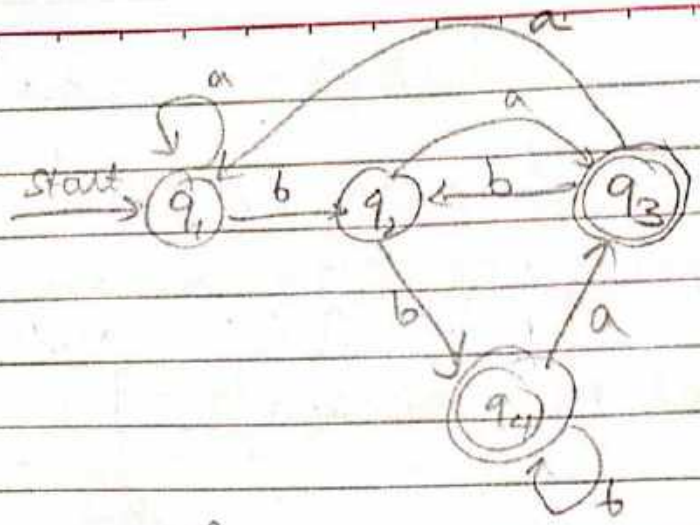


010

110

ends in 010.

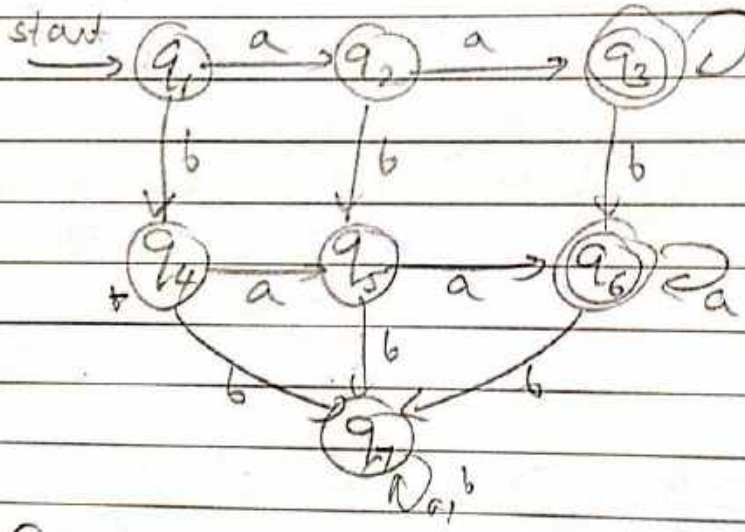
(f)



aba
aaaba
aabb

[Second last letter: b]

(g)



@ q_4 : b
@ q_2 : a
@ q_5 : ba, ab
@ q_7 : bb, bab, abb, ...
@ q_3 : aa[a...]
@ q_6 : aa[a..]b[a...], ba

(≥ 2 a, ≤ 1 b)

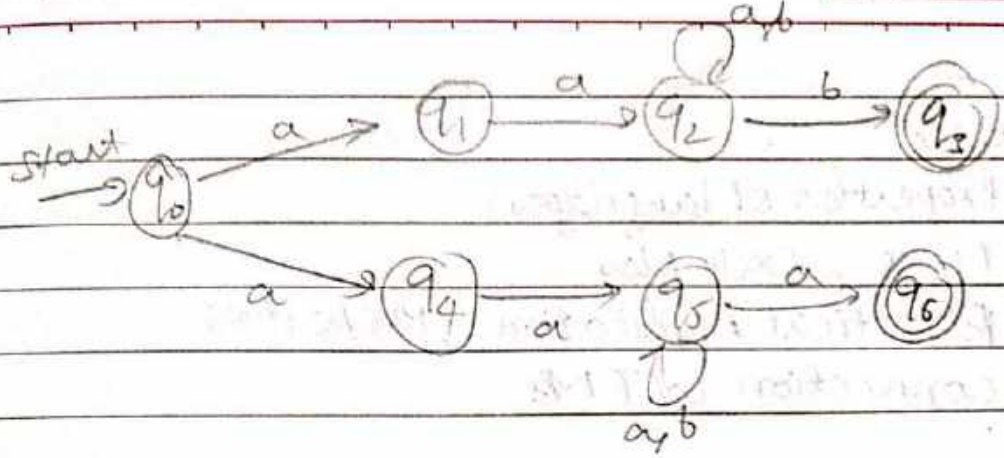
At most, 1b

b can be 1st / 2nd / 3rd char / or not there

aba → n
baa
aab

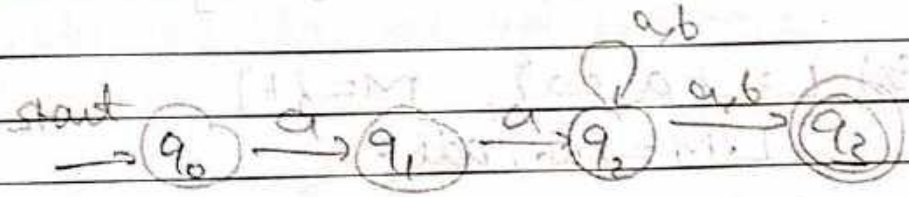
DOMS | Page No.
Date / /

g)



Starts w/ aa aa[]

~~Ea~~ has atleast 3 chars.



Page No.
Date 04/01/2022

L-8

Outline
- Properties of languages
- Regex, examples
- Practical Applications, Properties
- connection w/ T.Ms

→ concatenation of languages (Not commutative)
↳ but is associative

$M, L \subseteq \Sigma^*$

$L \cdot M = \{ xy \mid x \in L \wedge y \in M \}$

Ex) $L = \{a, aa\}$, $M = \{b\}$
$L \cdot M = \{ab, aab\}$

Ex) $L = \{a, aa\}$, $M = \{b, ab\}$
$L \cdot M = \{ab, aab, aaab\}$

Ex) $L = \{\} = \phi$, $M = \{a, ab\}$
$L \cdot M = \{\}$
$(\phi \cdot M = \phi)$

Ex) $L = \{\epsilon, ab\}$, $M = \{a, a^2, a^3, \ldots\}$
$L \cdot M = M \cup \{aa, aa^2, aa^3, \ldots\}$
$\cup \{ba, ba^2, ba^3, \ldots\}$
$= \{a, a^2, \ldots,\} \cup \{ba, ba^2, \ldots\}$

→ Kleene star/closure (unary operator)

$L^* = L^0 \cup L^1 \ldots = \boxed{\bigcup_{i=0}^{\infty} L^i}$

$\boxed{L^0 = \{\epsilon\}}$ By definition
$L^1 = L$
$L^2 = L \cdot L$
$L^3 = L \cdot L \cdot L$ ... etc.

Ex) $L = \{a\}$

$L^0 = \{\epsilon\}$

$L^* = \{a\}$

$L^2 = \{a^2\}$

$\vdots$

$L^* = \{\epsilon, a, a^2, \dots\}$

Ex) $L = \{a, b\}$

$L^0 = \{\epsilon\}$

$L^1 = L = \{a, b\}$

$L^2 = \{aa, ab, ba, bb\}$

$L^3 = \{aaa, aba, baa, bba, aab, abb, bab, bbb\}$

$\vdots$

Here, $\Sigma = \{a, b\}$.

clearly $(L^* = \Sigma^*)$ in this case

$$\boxed{L = \Sigma \Rightarrow L^* = \Sigma^*}$$

Ex) $L = \{\} = \phi$

$\boxed{L^0 = \{\epsilon\}}$

$L^1 = \{\epsilon\} \cdot \phi = \phi$

$\vdots$

(all ϕ)

$\Rightarrow \boxed{L^* = \{\epsilon\}}$

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

$L^0 = \{\epsilon\}$

$L^1 = \{\epsilon\}$

$\vdots$

(all $\{\epsilon\}$)

$\Rightarrow \boxed{L^* = \{\epsilon\}}$

Regular Expressions

For infinite languages, we have finite representations in terms of DFA/NFA/ε-NFA., as a Turing Machine. Regular expressions form an alternative way to represent languages. (Algebraic/Declarative Notation)

constants and variables are interpreted as being numbers. Arithmetic expressions ($2+3\times5$, $2x+5y$) also "denote" numbers. Similarly, Regular Expressions denote languages

Important points: (all defined over Σ)

① ϕ.
⇒ $Lang(\phi) = \{\} = \phi$

② ϵ
⇒ $lang(\epsilon) = \{\epsilon\}$

③ $a \in \Sigma$
$lang(a) = \{a\}$

④ r_1, r_2 are regular expressions (Inductive Assumption)
$(r_1 + r_2) \longrightarrow lang(r_1 + r_2) = lang(r_1) \cup lang(r_2)$
('+' is an 'OR' condition)

⑤ $r_1 \cdot r_2 \longrightarrow lang(r_1 \cdot r_2) = lang(r_1) \cdot lang(r_2)$

⑥ $r_1^* \longrightarrow lang(r_1^*) = (lang(r_1))^*$

So, $0^* = (lang(0))^* = \{\epsilon, 0, 00, 000, \ldots\}$

So, $10^* = \{1, 10, 100, 1000, \ldots\}$

$(1+0)0^* = 10^* + 0 \cdot 0^*$ $\quad lang(0+1) = \{0, 1\}$
$= 10^* + 0^*$ ~~$\{\epsilon\}$~~

[abc]

Ex) $r = a + b$

$lang(r) = lang(a) \cup lang(b)$

$= \{a\} \cup \{b\}$

$= \{a, b\}$

So, $r = a + b \Rightarrow lang(r) = \{a, b\}$

Ex) $r = a + b + c + d$

$\Rightarrow lang(r) = \{a, b, c, d\}$

grep = { global regular expression print }

Ex) $r = \phi \cdot a + b \cdot \epsilon$

$\Rightarrow lang(r) = lang(\phi a) \cup lang(b\epsilon)$

$= \phi \cup lang(b)$

$= \phi \cup \{b\} = \{b\}$

Ex) $L = \{w \mid w \text{ begins w/ aab}\}$ $\Sigma = \{a, b\}$

$r = aab(a+b)^*$ (i.e.) $aab\Sigma^*$

Ex) $L = \{w \mid w \text{ ends w/ aab}\}$ $\Sigma = \{a, b\}$

$r = (a+b)^* \cdot aab$

Ex) $L = \{w \mid w \text{ contains aab}\}$ $\Sigma = \{a, b\}$

$r = (a+b)^* aab (a+b)^*$

→ awk : Practical Application.

{patterns}	{actions}
emailid	{print}
Numbers	{delete}

→ compilers : lex / flex (lexical analyzer generator)

- keywords
- variables
- constants
- operators

→ (i) → (f) → "oh, now look for then clause & condition"

→ (i) → (n) → (t) → "A variable is about to be declared"

int abc = 10 - 1

KEY VAR CONST

operators

error ← (doesn't match pattern of a const)

DOMS | Page No.
Date 05/01/2022

L-9

Outline
- Regex, Examples
- Reglang → Regex, Examples.

Ex) $L = \{w \in \{0,1\}^* \mid w \text{ has exactly 3 1's}\}$

$r = 0^*10^*10^*10^* \quad (L = L(r))$

- Reglang = languages defined by DFAs, and hence, NFAs, ϵ-NFAs. $(L \subseteq \Sigma^*)$

[Set of Reglangs = Set of langs defined by Regex]

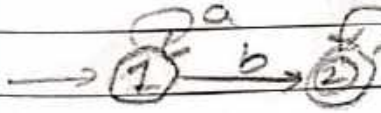
[1] [2] } req.

Pf for Reglang → RegEx (i.e.) Reglang ⊆ RegEx

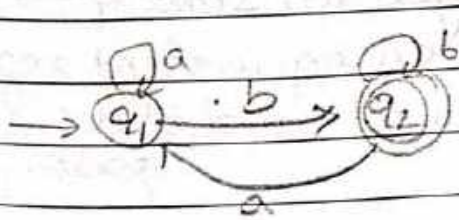
(i.e.) DFA → RegEx

$L(M) = a^*$

$r = a^* \rightarrow L(r) = a^*$



$r = a^*.bb^* = L(M)$



$r = (a^*bb^*)(aa^*bb^*)^*$

→

Consider some DFA, D'

If $w \in L(D)$, run goes from $q_1 \to q_{f_i}$ for some i.

Get regexp. for all w: $\hat{\delta}(q_1, w) = q_{f_1}$: R_{1,f_1}

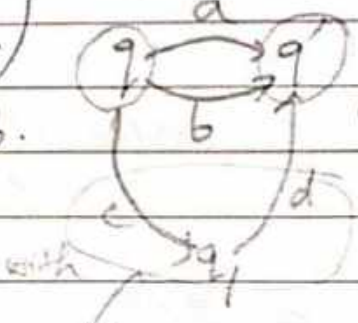
Then, $\forall w$: $\hat{\delta}(q_1, w) = q_{f_2}$: R_{1,f_2}

$\forall w$: $\hat{\delta}(q_1, w) = q_{f_n}$: R_{1,f_n}

Big picture

Finally, $\left(R_D = \sum_{i=1}^{n} R_{1,f_i}\right)$

consider some $\left(\begin{array}{l}\delta(q,a) = q' \\ \delta(q,b) = q'\end{array}\right)$ $\Leftrightarrow r = a+b$.



Rather than induction on path length, show that ~~set~~ ~~set of~~ ~~language/regex~~ DFA for some $q \to q'$ via any no. of paths has an equivalent regex.

(By induction on no. of states we're passing through)

DOMS | Page No.
Date / /

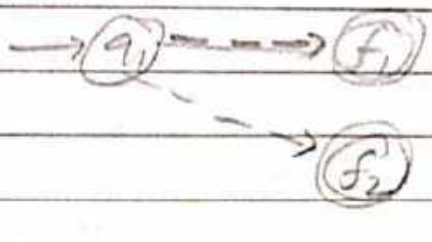
Given a DFA D w/ n states, number/label them arbitrarily as $1, 2, \ldots, n$ or $q_1, q_2, \ldots, q_n$.
consider a notion of comparison, where $q_i < q_j$ for $i < j$.
So, $q_1 < q_2 < q_3 < \ldots < q_n$.

(q_i) (q_j)

$R_{ij} = [\ldots]$ under construction. set of all words $q_i \to q_j$

R_{ij}^k = set of all words : $q_i \to q_j$ without passing through any state $> q_k$

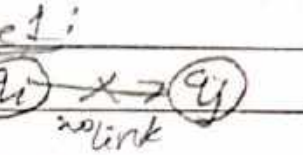
restricts no. of states we pass through during $q_i \to q_j$.



Induction on k:

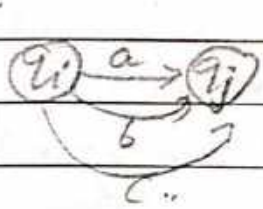
Base case: $k = 0$

case 1:



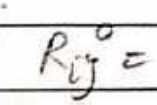
$R_{ij}^0 = \phi$

case 2:



$R_{ij}^0 = a + b + c$
(collect all 1-length transitions)

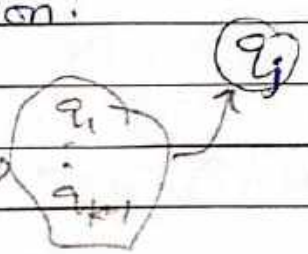
Case 3: $i = j$:



$R_{ij}^0 = \epsilon + a + b + \ldots$

→ Inductive Assumption:

R_{ij}^{k-1} :



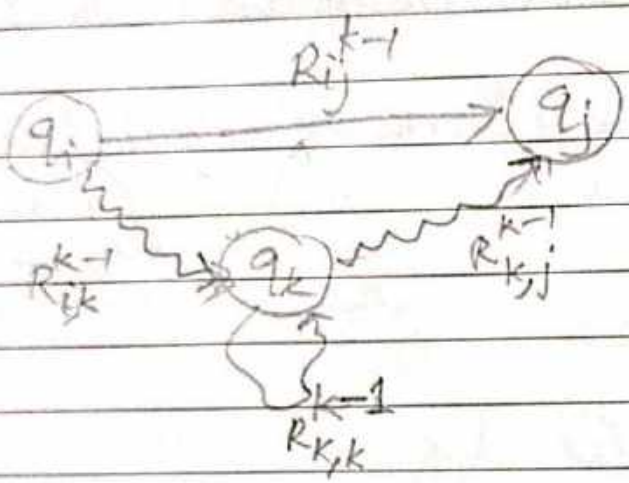
Date / /

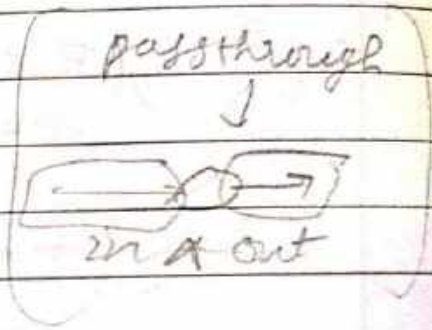
Now, for R_{ij}^k,

$q_i \rightarrow (q_1 \cdots q_k) \rightarrow q_j$

we are now allowed to pass through q_k.

without q_k, R_{ij}^k is set of paths from q_i to q_j





$$R_{ij}^k = R_{ij}^{k-1} + R_{ik}^{k-1}\left(R_{k,k}^{k-1}\right)^* R_{kj}^{k-1}$$

(weak induction)

parallel — series

Assume R_{ij}^{k-1} exists $\forall i,j$ (# is given)

Hence, we have R_{ij}^k using R_{ij}^{k-1}.

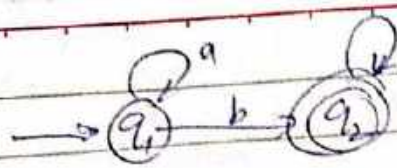
$\Rightarrow$ Regex exists for any $k \in \mathbb{N} \cup \{0\}$

($p(k-1) \Rightarrow p(k)$, Base: $p(0)$)

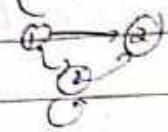
All paths $R_{ij} \equiv [R_{ij}^n] \rightarrow$ hence, $[R_{q_0 f}^n]$

Date / /

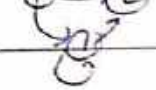
a^*bb^*



$$\left[R_{1,2}^{2} = R_{1,2}^{1} + R_{1,2}^{1}(R_{2,2}^{1})^* R_{2,2}^{1}\right]$$

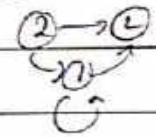


$$\left[R_{1,2}^{1} = R_{1,2}^{0} + (R_{1,1}^{0})(R_{1,1}^{0})^*(R_{1,2}^{0})\right]$$



$$\left[R_{2,2}^{1} = R_{2,2}^{0} + R_{2,1}^{0}(R_{1,1}^{0})^* R_{1,2}^{0}\right]$$

ϕ



$R_{1,2}^{0} = b$, $R_{1,1}^{0} = \epsilon + a$, $R_{2,1}^{0} = \phi$, $R_{2,2}^{0} = \epsilon + b$

$R_{2,2}^{1} = (\epsilon + b) + \phi = (\epsilon + b)$

$R_{1,2}^{1} = b + (\epsilon + a)(\epsilon + a)^*(b)$

$= b + (\epsilon + a)\, a^* b$

a^*

a^*b
a^*b

$= b + a^*b = (\epsilon + a^*)b = a^*b$ $(\epsilon \in a^*)$

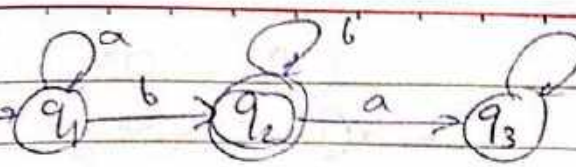
$R_{1,2}^{2} = a^*b + a^*b\,(\epsilon + b)^*(\epsilon + b)$

$= a^*b + a^*b\, b^*(\epsilon + b)$

b^*

$= a^*b + a^*b \cdot b^* = a^*b(\epsilon + b^*) = \boxed{a^*bb^*}$

b^*



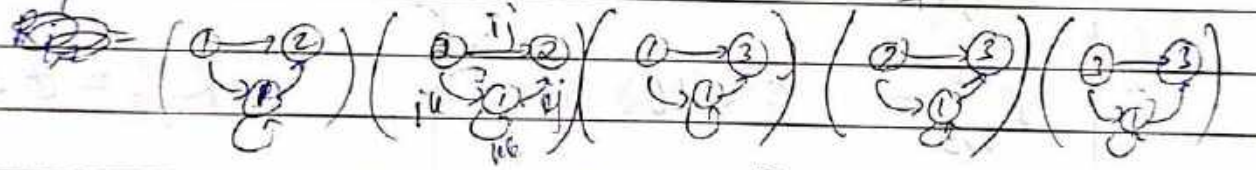
$$R_{1,2}^{3} = R_{1,2}^{2} + R_{1,3}^{2}(R_{3,3}^{2})^{*}R_{3,2}^{2}$$

$$\rightarrow R_{1,2}^{2} = R_{1,2}^{1} + R_{1,2}^{1}(R_{2,2}^{1})^{*}(R_{2,2}^{1})$$

$$R_{1,3}^{2} = R_{1,3}^{1} + R_{1,2}^{1}(R_{2,2}^{1})^{*}R_{2,3}^{1}$$

$$R_{3,3}^{2} = R_{3,3}^{1} + R_{3,2}^{1}(R_{2,2}^{1})^{*}R_{2,3}^{1}$$

$$\rightarrow R_{3,2}^{2} = R_{3,2}^{1} + R_{3,2}^{1}(R_{2,2}^{1})^{*}R_{2,2}^{1}$$



$$\rightarrow R_{1,2}^{1} = R_{1,2}^{0} + R_{1,1}^{0}(R_{1,1}^{0})^{*}R_{1,2}^{0}$$

$$\rightarrow R_{2,2}^{1} = R_{2,2}^{0} + R_{2,1}^{0}(R_{1,1}^{0})^{*}R_{1,2}^{0}$$

$$R_{1,3}^{1} = R_{1,3}^{0} + R_{1,1}^{0}(R_{1,1}^{0})^{*}R_{1,3}^{0}$$

$$R_{2,3}^{1} = R_{2,3}^{0} + R_{2,1}^{0}(R_{1,1}^{0})^{*}R_{1,3}^{0}$$

$$R_{3,3}^{1} = R_{3,3}^{0} + R_{3,1}^{0}(R_{1,1}^{0})^{*}R_{1,3}^{0}$$

$$\rightarrow R_{3,2}^{1} = R_{3,2}^{0} + R_{3,1}^{0}(R_{1,1}^{0})^{*}R_{1,2}^{0}$$

DOMS | Page No.
Date / /

{ϵ, a, b}

$R_{1,1}^0 = a(\epsilon + a)$

$\rightarrow R_{1,2}^0 = b$

$R_{1,3}^0 = \phi$

$\rightarrow R_{2,1}^0 = \phi$

$\rightarrow R_{2,2}^0 = \epsilon + b$

$R_{2,3}^0 = a$

$\rightarrow R_{3,1}^0 = \phi$

$\rightarrow R_{3,2}^0 = \phi$

$R_{3,3}^0 = \epsilon + a + b$

$R_{1,2}^1 = b + (\epsilon + a)(\epsilon + a)^* b$

$= b + (\epsilon + a) a^* b$

$= b + a^* b$

$= (\epsilon + a^*) b$

$= a^* b$

$R_{2,2}^1 = (\epsilon + b) +$

$R_{3,2}^1 = \phi + \phi = \phi$

$R_{1,2}^1 = b + (\epsilon + a)(\epsilon + a)^* b$

$= b + (\epsilon + a) a^* b$

$= b + a^* b$

$= (\epsilon + a^*) b = a^* b$

$R_{1,3}^1 = \phi + R_{1,1}^0 (\epsilon + a)(\epsilon + a)^* \phi = \phi$

$R_{2,2}^1 = (\epsilon + b) + \phi = \epsilon + b$

$R_{2,3}^1 = a + \phi = a$

$R_{3,3}^1 = (\epsilon + a + b) + \phi = \epsilon + a + b$

$$R_{1,2}^2 = R_{1,2}^1 + R_{1,2}^1 (R_{2,2}^1)^* R_{2,2}^1$$

$$= a^*b + a^*b(\epsilon+b)^*(\epsilon+b)$$

$$= a^*b + a^*bb^*(\epsilon+b)$$

$$= a^*b + a^*bb^*$$

$$= a^*b(\epsilon+b^*)$$

$$= a^*bb^*$$

$$R_{1,3}^2 = R_{1,3}^1 + R_{1,2}^1 (R_{2,2}^1)^* R_{2,3}^1$$

$$= \phi + a^*b(\epsilon+b)^* a$$

$$= a^*bb^*a$$

$$R_{3,3}^2 = R_{3,3}^1 + R_{3,2}^1 (R_{2,2}^1)^* R_{2,3}^1$$

$$= \epsilon + a + b + \phi = \epsilon + a + b$$

$(\epsilon+a)^* = a^*$

$a^*(\epsilon+a) = a^*$

$$R_{3,2}^2 = \phi + \phi = \phi$$

So,

$$R_{1,2}^3 = R_{1,2}^2 + R_{1,3}^2 (R_{3,3}^2)^* R_{3,2}^2$$

ϕ

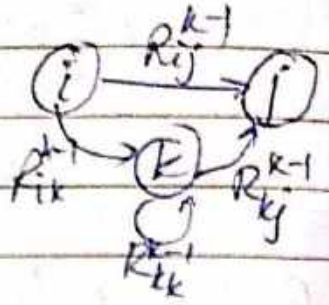
$$\boxed{R_{1,2}^3 = a^*bb^*}$$

$bb^* = b^*b = b^+$

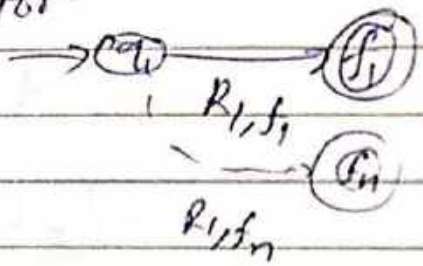
L-10

outline
- DFA ⇒ RegEx, Examples
- Complete DFA
- (properties of RegEx)

→ $R_{ij}^{k} = R_{ij}^{k-1} + R_{ik}^{k-1} (R_{kk}^{k-1})^{*} R_{kj}^{k-1}$



For



$\Sigma \cdot \Rightarrow \sum_{i=1}^{n} R_{1,f_i}$.

→ Fibonacci sequence

$f(n) = f(n-1) + f(n-2)$

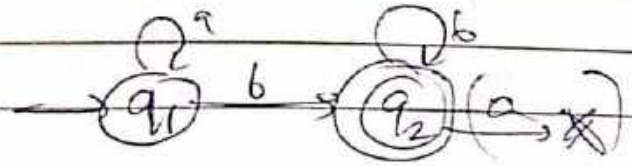
$f(0) = 0, f(1) = 1$

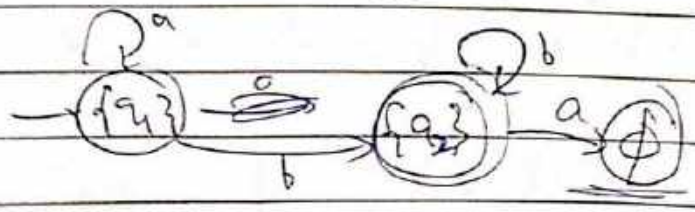
(Recursion until base cases)

we may reuse info during computation

→ Complete DFA (Dfa we were working with)

$\forall q \in Q, a \in \Sigma, \exists q' : \delta(q,a) = q'$.





$|\delta(Q \times \Sigma)| \leq 1$

↳ DFA

Page No.
Date 07/01/2022

L:11

outline
- RegEx properties, examples
- RegEx ⇨ RegLang, examples
- Non-regular languages.

Properties of RegEx
- $L+M = M+L$
- $(L+M)+N = L+(M+N)$
- $(L \cdot M) \cdot N = L \cdot (M \cdot N)$
- $L \cdot (M+N) = L \cdot M + L \cdot N$
- $(M+N) \cdot L = M \cdot L + N \cdot L$
- $\epsilon \cdot L = L \cdot \epsilon = L$
- $\phi + L = L + \phi = L$
- $\phi \cdot L = L \cdot \phi = \phi$
- $(L+\epsilon)^* = L^*$
- $L^{**} = L^*(L+\epsilon)$
- $L+L = L \rightarrow$ Idempotent
- $L^* \cdot L^* = L^*$

- Addition/Union = Associative, commutative
- Concatenation = Associative
- Identity = ϵ(concat), ϕ (additive)
- Distributivity of concat^n over union. (both dir^{ns})
- ↳ pf: use $x w = xy$, $x \in \text{lang}(L)$, $y \in \text{lang}(M+N)$ ⇒

$L^* = (L+\epsilon)^*$
ϕ

$\text{lang}(L^*) = \{\epsilon, L, L^2, L^3, \ldots\}$
$\text{lang}((L+\epsilon)^*) = \{\epsilon, L+\epsilon, (L+\epsilon)^2, \ldots\}$
$(L+\epsilon)^1 = \text{lang}(L) \cup \{\epsilon\} = L+\epsilon$.
$(L+\epsilon)^2 = (\text{lang}(L) \cup \{\epsilon\}) \cdot (\text{lang}(L) \cup \{\epsilon\})$
$= (L+\epsilon)(L+\epsilon)$
$= L^2 + \epsilon L + L\epsilon + \epsilon\epsilon$
$= L^2 + L + \epsilon$
$(L+\epsilon)^n = L^n + L^{n-1} + \ldots + \epsilon$.

Date / /

$$(R^*S^*)^* = (R+S)^*$$

$\text{Lang}((R^*S^*)^*)$

$= \{\epsilon, R^*S^*, (R^*S^*)^2, (R^*S^*)^3, \ldots\}$

$(R \mid S)^*$

Let $w \in \text{Lang}(R^*S^*)^*$

$w = (R^*S^*)(R^*S^*)\cdots(R^*S^*)$ n times

$w = w_1 \cdot w_2 \cdots w_n$

consider $w_i = R^*S^*$ for $i \in \{1, 2, \ldots n\}$

$w_i = x_i y_i, \quad x_i \in R^*, y_i \in S^*$

$(w = x_1 y_1 x_2 y_2 \cdots x_n y_n)$

$x_i \in R^*$

$\Rightarrow x_i \in (R+S)^*$ ①

$\text{Lang}(R^*) \subseteq (\text{Lang}(R^*) \cup \text{Lang}(S^*))^*$

$(R^* \subseteq (R+S)^*)$

$\{\epsilon, R, S, R^2, S^2, RS, SR, \ldots\} \in \{\epsilon, R, R^2, \ldots\}$

Also, $y_i \in S^*$

$\Rightarrow y_i \in (R+S)^*$ ②

(If $w = abc$ and $a, b, c \in L^*$ $\Rightarrow w \in L^*$)

① ② $\Rightarrow w \in (R+S)^*$

$\Rightarrow (R^*S^*)^* \subseteq (R+S)^*$

Consider $w \in (R+S)^*$.

$\Rightarrow w = x_1 x_2 \cdots x_n$, where $x_i \in (R+S)^* \ \forall i$

$\rightarrow$ $\boxed{(L^*)^* = L^*}$

↓

Proof:

$L^* = \{\epsilon, L, L^2, L^3 \ldots\}$

$(L^*)^2 = \{\epsilon, L, L^2, L^3 \ldots\}$

$(L^*)^3 = \{\epsilon, L, L^2, L^3 \ldots\}$

$\Rightarrow (L^*)^n = (L^*) \ \forall n \in \mathbb{N}$.

Since $(L^*)^n = (L^*) \ \forall n \in \mathbb{N}$,

$(L^*)^* = \{\epsilon, L^*, (L^*)^2 \ldots\}$

$= \{L^*\} = L^* \quad (\epsilon \in L^*)$

Let $w \in (L^*)^*$

$\Rightarrow w \in (L^*)(L^*) \cdots (L^*)$

$\Rightarrow w = w_1 w_2 \cdots w_n$ $\quad w_i \in L^* \ \forall i$

$w = (L \cdots L)(L \cdots L) \cdots (L \cdots L)$

$w \in L^*$

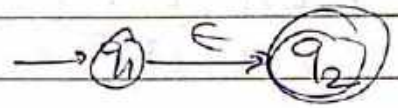
Last time: (Regular (DFA)) ⟹ (Regex)
This time: (Regular (DFA)) ⟸ (Regex)

→ Theorem: If L is denoted by a regular expression, then, L is regular.

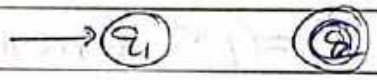
To prove:
L is accepted by any one of: (DFA, NFA, ϵ-NFA).

Base cases

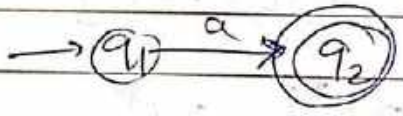
① $r = \epsilon$
$L(r) = \{\epsilon\}$



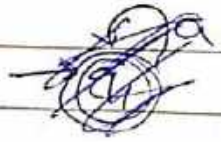
② $r = \phi$
$L(r) = \{\}$



③ $r = a \in \Sigma$



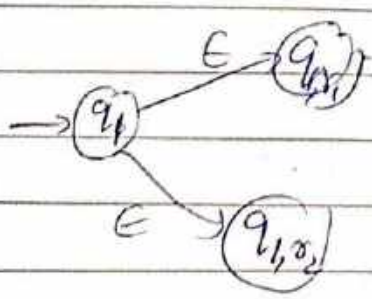
④ $r = a^*$



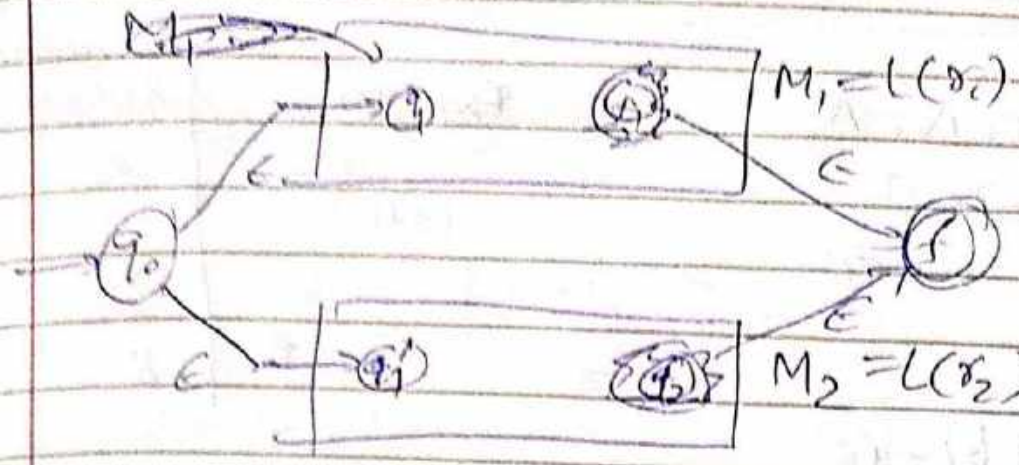
Inductive cases

① $r = r_1 + r_2$

Induction hypothesis,
r_1 has ϵ-NFA, M_1
r_2 has ϵ-NFA, M_2

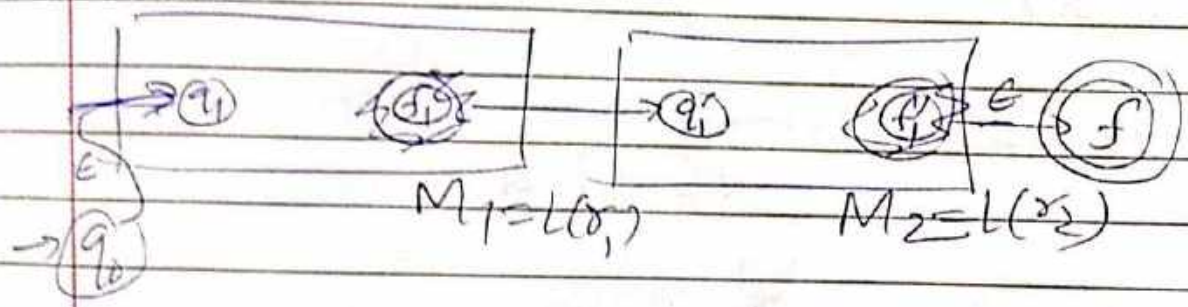


M_i has only 1 initial state
only 1 final state
no outgoing arcs from f
~~no incoming to initial~~ state

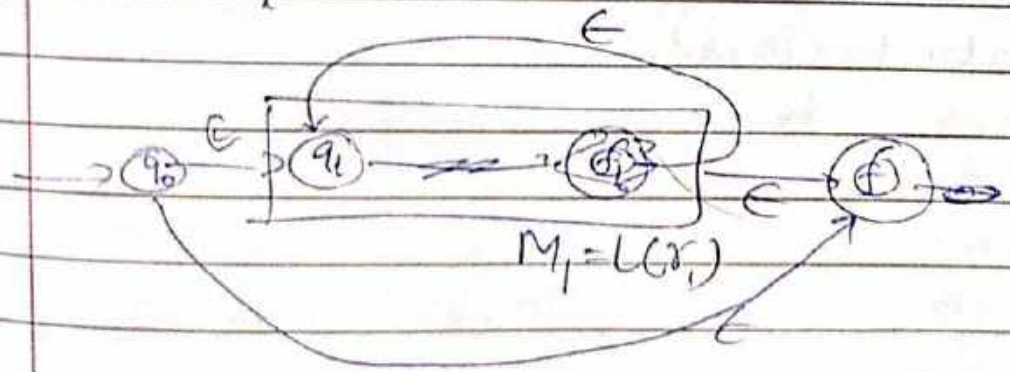


Hence, $lang(r_1 + r_2) = lang(r_1) \cup lang(r_2)$
$\therefore lang(r)$

(2) $r = r_1 \cdot r_2$



(3) $r = (r_1)^*$



Hence, regex ⇨ DFA.

Thus, regex, DFA have equal power

(ϵ-NFA = DFA = NFA = Regex)

→ Regular

Thompsons-Construction

expr → ε-NFA

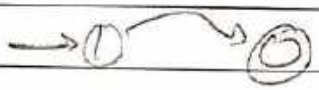
Expression	Size(ε-NFA) #states
a	2
(a+b)	6
a.b	6
(a+b)*	8

((a+b)*+b) → 6

Size(ε-NFA) = 2 × Size(expression)

DFA to Regex:

n → $\geq 2^n$



|w| = n (w ∈ lang (N ?))

Satisfiability by:	Time complexity
DFA	O(n)
NFA	$O(2^n)$
ε-NFA	$O(n^5)$

$O(n^3)$

(ε-closure matrix) (Graph(ε-NFA) Adj. Matrix) → (NFA adj. matrix)

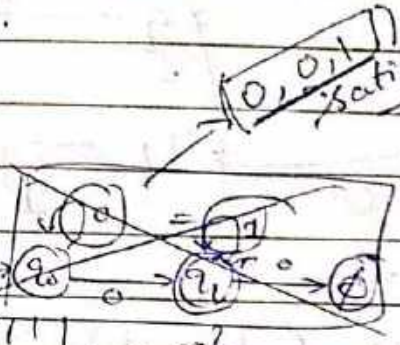
$O(n^3)$

Page No.
Date 11/01/22

L-12+13

Non-Regular Languages.

Ex) For $L = \{0^n 1^n \mid n \geq 1\}$



$L = \{01, 0011, 000111, \ldots\}$

Finite Automata not possible

(accepting run exists in NFA $\Leftrightarrow$ part of language)

$n=1$, D_1
$n=2$, D_2
$\vdots$
$n=k$, D_k
$\vdots$
infinite DFAs

$\Rightarrow$ union gives infinite automata

Pf by Contradiction:

Assume $\exists$ DFA D' : lang $(D) = L$

i.e., assume L is regular.

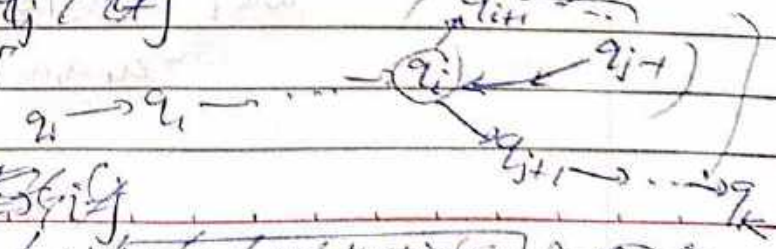
Let $D = (Q, \Sigma, \delta, q_0, F)$., where $|Q| = k$

consider just 0^k, $0^k = \underbrace{0 \ldots 0}_{k\text{-length}}$

$(q_0 \xrightarrow{0} q_1 \xrightarrow{0} \ldots \xrightarrow{0} q_k) \rightsquigarrow$ (k+1) distinct states.

$\Rightarrow \exists i, j$: $q_i = q_j$ / $i \neq j$

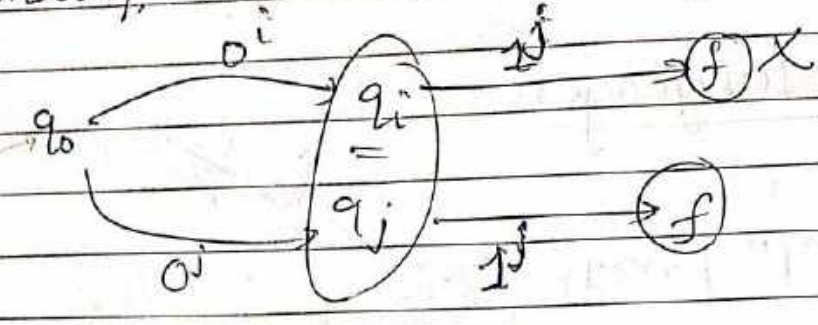
$\exists i, j$: $q_i = q_j$, $i \neq j$

$\Rightarrow$ 

$\Rightarrow$ You can reach q_i via 0^i & 0^j, but $i \neq j$! $\Rightarrow$ contradiction.

Assume $q_i = q_j$ but somehow $i \neq j$ even now.

Similarly, we can make an argument for 1^k.



$\Rightarrow$

But, 0^i1^j satisfies the DFA, where $i \neq j$.

$\Rightarrow 0^i1^j \in L$, when it should NOT

$\Rightarrow\Leftarrow 0^i1^j \notin L$ $(i \neq j)$

Pumping Lemma:

For any given Regular language L, there exists a constant $n \geq 1$, such that,

for any $w \in L$ $[|w| \geq n]$

[there exists strings $x, y, z \in \Sigma^*$;

$w = xyz$, $|y| \neq \epsilon$, $|xy| \leq n$]

such that

$w' = xy^kz \in L$, $k \geq 0$

i.e. $xy^*z \in L$

pf:

Given:

- L is regular

$\Rightarrow \exists$ DFA 'D' : lang(D) = L,

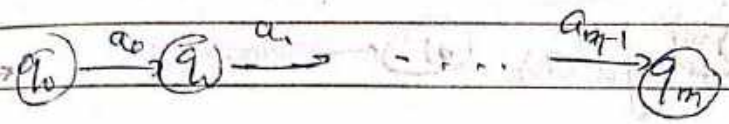
$D = (Q, \Sigma, \delta, q_0, F)$, where $|Q| = n$.

Consider

$w \in L$: $|w| = m \geq n$

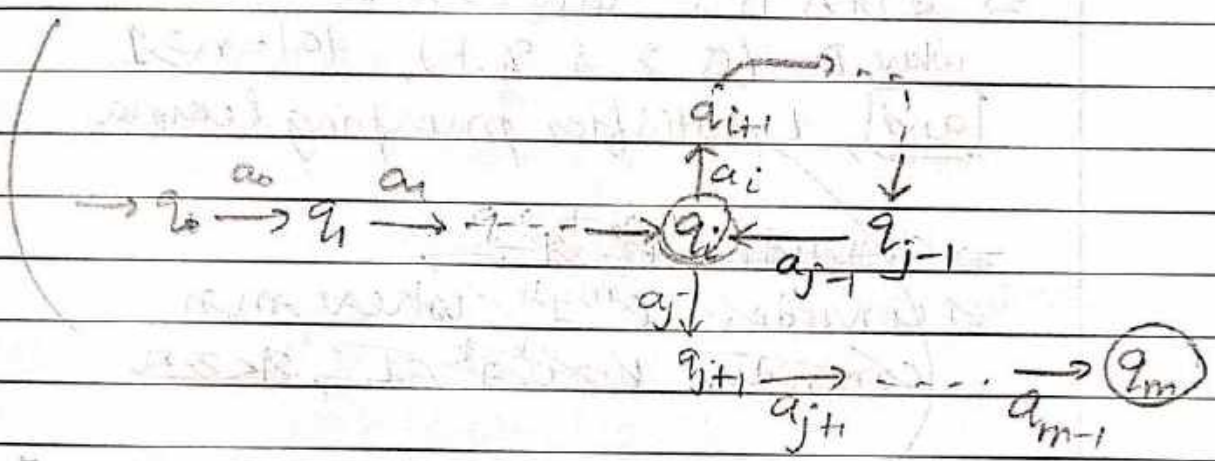
$w = a_0a_1a_2 \ldots a_{m-1}$

Examine run of D on w.



$$m \geq n = |Q|$$

$$\Rightarrow \exists\, i, j : q_i = q_j,\ i \neq j \quad \text{(PHP)}$$



The loop at q_i can run any number of times, even zero.

$$\Rightarrow w = \underbrace{a_0 a_1 \cdots a_{i-1}}_{x} \underbrace{(a_i a_{i+1} \cdots a_{j-1})^*}_{y^k} \underbrace{a_j a_{j+1} \cdots a_{m-1}}_{z}$$

$$\Rightarrow w = x \cdot y^k \cdot z, \quad k \geq 0$$

Worst-case scenario, $j = i+1$, it's a self loop.

$$\Rightarrow \underbrace{a_0 a_1 \cdots a_{i-1}}_{x} \underbrace{(a_i)^*}_{y^k} \underbrace{a_j a_{j+1} \cdots a_{m-1}}_{z}$$

$$k \geq 0$$

Ex) $L = \{0^n 1^n \mid n \geq 1\}$

claim: L is non-regular

Pf: By contradiction,
Assume L is regular.
$\Rightarrow \exists$ DFA 'D': lang(D) = L
where $D = (Q, \Sigma, \delta, q_0, F)$, $|Q| = n \geq 1$
and, L satisfies pumping lemma

~~Consider~~

~~Consider $0^m 1^n$, where m, n~~
~~Consider $w = 0^k 1^k \in L$, $k \geq n$~~

Consider $0^n 1^n$.
clearly $0^n 1^n \in L$.
But, does $0^n 1^n$ satisfy DFA D?
$|0^n 1^n| = 2n \geq n$
$\Rightarrow$ By pumping lemma,
$\exists x, y, z$: $w = xyz \in L \Rightarrow w' = xy^k z \in L \; \forall k \geq 0$
$(|y| \geq 1$
$|xy| \leq n$

$\Rightarrow w = \underbrace{00\ldots0}_{n \text{ zeroes}} \underbrace{11\ldots1}_{n \text{ 1's}}$

$y \in 0^* 1^*$ in its general form

Either: 00...01|1...1
x y z
x y z
x y z
x y z

$|x|$: 0 to $n-1$

$n = |y|$: 1 to n

$|z|$: n to $2n-1$

Consider $x\ y\ z$

$n-m \quad m \quad n$

$\Rightarrow xy^k z \in L, \quad k \geq 0$

$\Rightarrow 0^{n-m} 0^{km} 1^n, \quad k \geq 0$

$\Rightarrow 0^{n+(k-1)m} 1^n, \quad k \geq 0$

$\Leftrightarrow |w|_0 \geq |w|_1 \Rightarrow w \notin L$

contradiction

Also: $\exists k : x^k y^k z \notin L$.

$\Rightarrow\Leftarrow$ Pumping lemma.

→ (Pumping lemma applies to all DFAs for whom $\exists\ w \in \text{Lang}(D) : |w| \geq |Q|$)

Ex) $L = \{ w \in \{0,1\}^* \mid w \text{ is a palindrome} \}$. Is L not regular?

Pf. by contradiction:

Assume L is regular ⇒ Set some $n \geq 1$.

Consider $w \in L : |w| \geq n$

$\Rightarrow (w = 0^n 1 0^n)$

Try to decompose w into x, y, z : $|y| > 0$

$\Rightarrow y = 0^t$ for some $1 \leq t \leq n$ $\quad |xy| \leq n$

$\Rightarrow (x = 0^{n-t}, z = 10^n)$

Consider, $xy^k z$, with $k = 0$, $t > 0$

$\Rightarrow xz = 0^{n-t} 1 0^n \notin L$

$\Rightarrow \exists k : xy^k z \notin L$

$\Rightarrow\Leftarrow$ Pumping lemma contradicts.

Hence, L is not regular.

$x = 0^{n-m-k}$

$y = 0^m$ $\quad m \neq 0 \quad (m > 0)$

$z = 0^k 1 0^n$

Consider $k = 0 \Rightarrow y^k = \epsilon$

$xy^kz = 0^{n-m-k} \cdot \epsilon \cdot 0^k 1 0^n$

$= 0^{n-m} 1 0^n \notin L$ (not a palindrome)

$\Rightarrow \exists k : xy^kz \notin L \Rightarrow\Leftarrow$ Pumping lemma

Hence, Assumption incorrect.

($L \Rightarrow$ non-regular)

Ex) language : well-balanced parenthesis.

$\Sigma = \{(,)\}$

At any pt., [no. of (] $\geq$ [no. of)]

( ) ( ) ( ) (( ) ( ) ( ))

x y z

(((( ) ( ))))

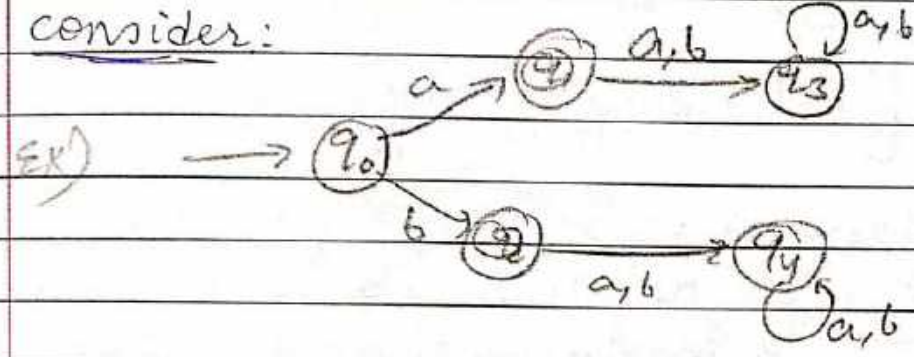
y^2

DOMS | Page No.
Date 12/01/22

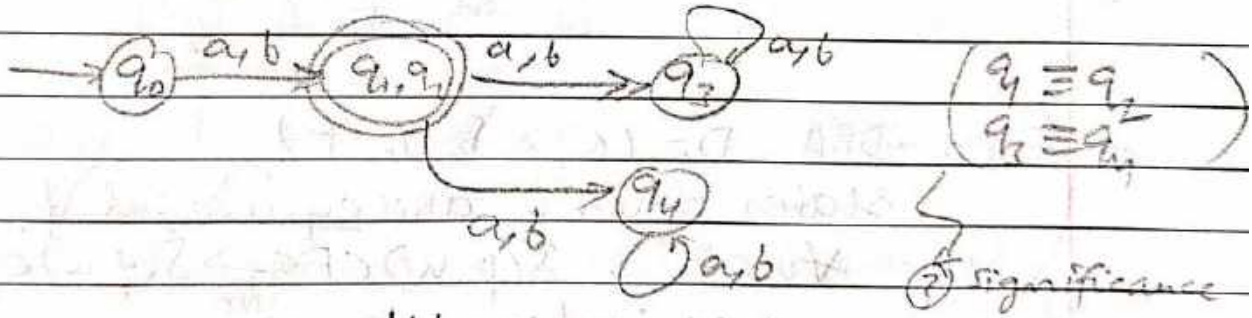
K-19.

outline
- Equivalence of 2 states in a DFA
- Equivalence, Distinguishability
- Algo. for distinguishing, Example, Correctness
- Applications

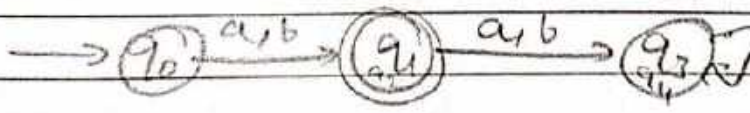
consider:



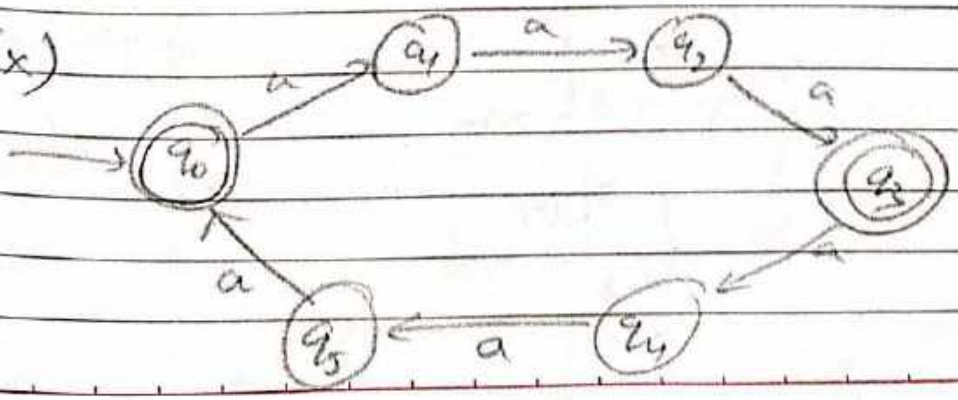
111



111

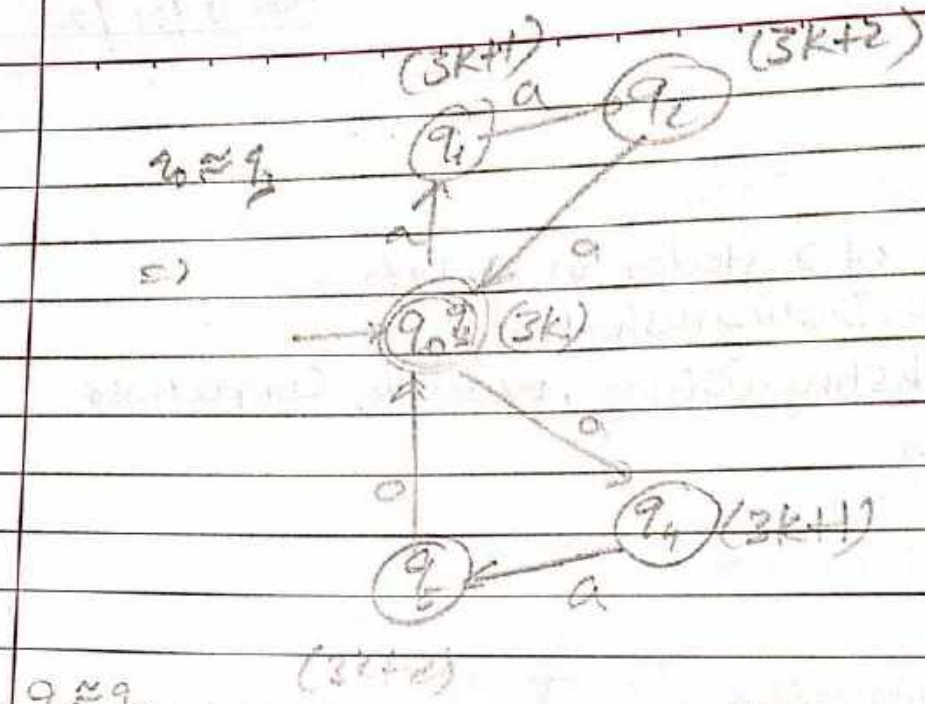


Ex)

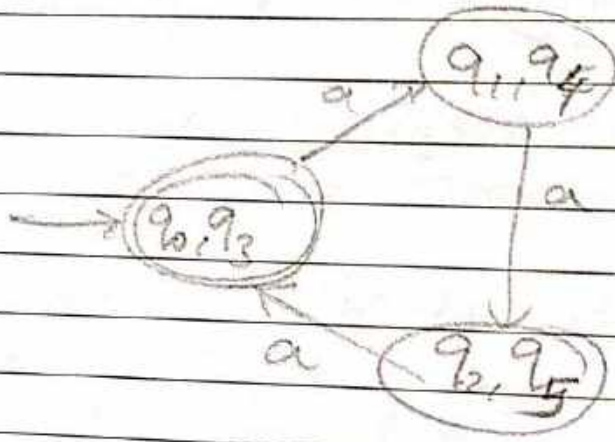


$(a^3)^*$

a^n, $n/3$

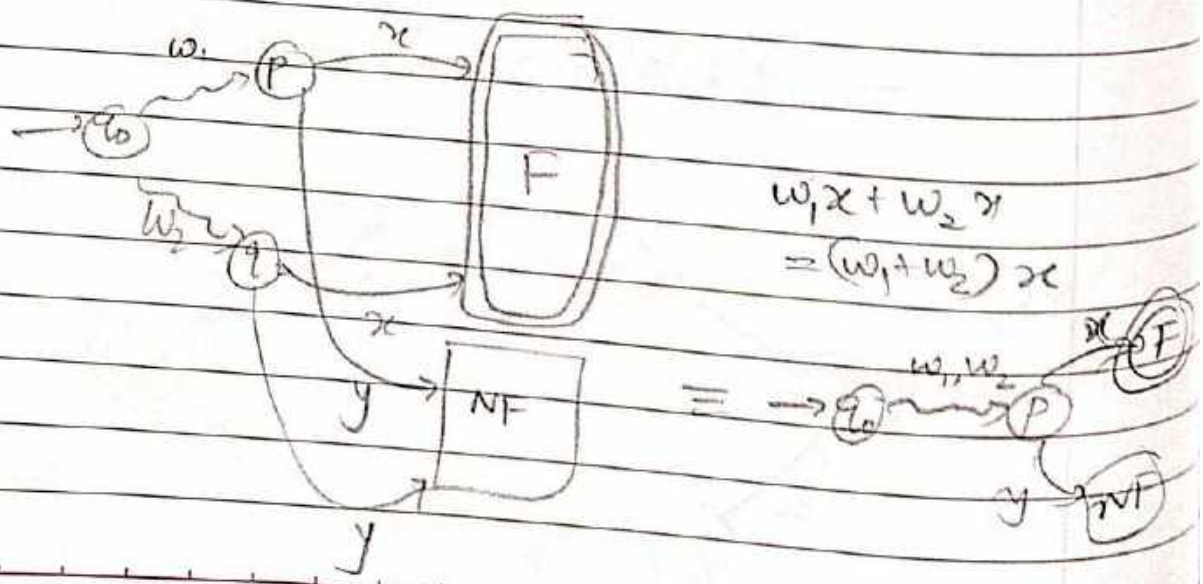


$q_1 \approx q_4$
$q_0 \approx q_2$



→ For a DFA, $D = (Q, \Sigma, \delta, q_0, F)$,
states p and q are equivalent if,
$\forall w \in \Sigma^*$, $\hat{\delta}(p, w) \in F \iff \hat{\delta}(q, w) \in F$
iff

↳ Notion of equivalence.



$\sim(p \leftrightarrow q)$

$= \sim(p \rightarrow q) \vee \sim(q \rightarrow p)$

$\sim(p \vee q) \vee \sim(\sim q \vee p)$

$(p \wedge \sim q) \vee (q \wedge \sim p)$

DOMS | Page No. | Date / /

→ Distinguishable states

For a DFA, $D = (Q, \Sigma, \delta, q_0, F)$, states p and q are distinguishable if:

$\exists w \in \Sigma^*$, $\hat{\delta}(p,w)$ (XOR) $\hat{\delta}(q,w)$ is true

(i.e)

($\hat{\delta}(p,w) \in F$ AND $\hat{\delta}(q,w) \notin F$)

(or)

($\hat{\delta}(p,w) \notin F$ AND $\hat{\delta}(q,w) \in F$)

→ Algorithm for distinguishability

Disproving equivalence is easier than proving it, so let us formulate an algorithm to check distinguish rather than equivalence.

(due to nature of Σ^*)

Algo:

① Take $|w| = 0$, $w = \epsilon$ (Base case)

check $\hat{\delta}(p,\epsilon)$ and $\hat{\delta}(q,\epsilon)$.

If $\hat{\delta}(p,\epsilon)$ XOR $\hat{\delta}(q,\epsilon)$ belongs to F,

$\equiv \delta(p,\epsilon)$ XOR $\delta(q,\epsilon)$

$\equiv p$ XOR q

(i.e) check p and q if one is final, other is not final.

② Take $|w| = n$, $w = x'$

Let $\hat{\delta}(p,w) \in F$,

$p \xrightarrow{a} p_1$

$q \xrightarrow{a} q_1$

If p_1, q_1 are disting., then p, q are disting.

(since $\hat{\delta}(p_1,a)$ XOR $\hat{\delta}(q_1,a) \in F$)

If ~~r~~ $r \neq s$,

then, $\exists w'$: ($r \xrightarrow{w'} (f)$, $s \xrightarrow{w'} (\neg f)$)

which would mean p, q are distinguishable.

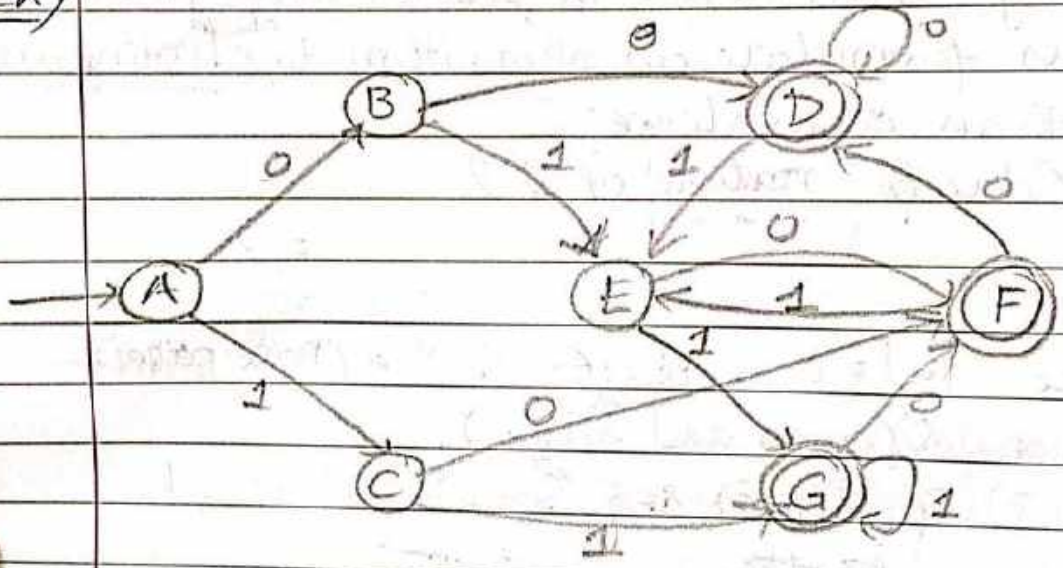
$p \xrightarrow{qw'} (f)$

$q \xrightarrow{qw'} (\neg f)$

$aw' \in \Sigma^*$

($\hat{\delta}(p, aw') \in F$, $\hat{\delta}(q, aw') \notin F$) (working backwards from final states)

Ex)



	A	B	C	*D	E	*F	*G
A	✓	X	X	X	X	X	X
B		✓	X	X	X	X	X
C			✓	X	✓	X	X
*D	X	X	X	✓	X	✓	X
E				X	✓	X	X
*F	X	X	X		X	✓	X
*G	X	X	X		X		✓

not reachable

DOMS | Page No. | Date / /

(1) construct a matrix Q × Q.

(2) All diag ele's = ∅ (✓)

(3) All (i, j) where i ∈ F, j ∉ F : = (X)

(4) consider leftover pairs: (with just 1-length words)

only 0:

A —0→ B
B —0→ *D
} X

B —0→ *D
E —0→ *F
can't say

A —0→ B
C —0→ *G
} X

C —0→ *F
E —0→ *F
can't say

A —0→ B
E —0→ *F
} X

D —0→ *D
F —0→ *D
can't say yet

B —0→ D
C —0→ F
can't say

D —0→ *D
C —0→ *F
can't say

F —0→ *D
G —0→ *E
can't say

only 1

B —1→ E
G —1→ *G
} X

D —1→ E
F —1→ E
(✓)

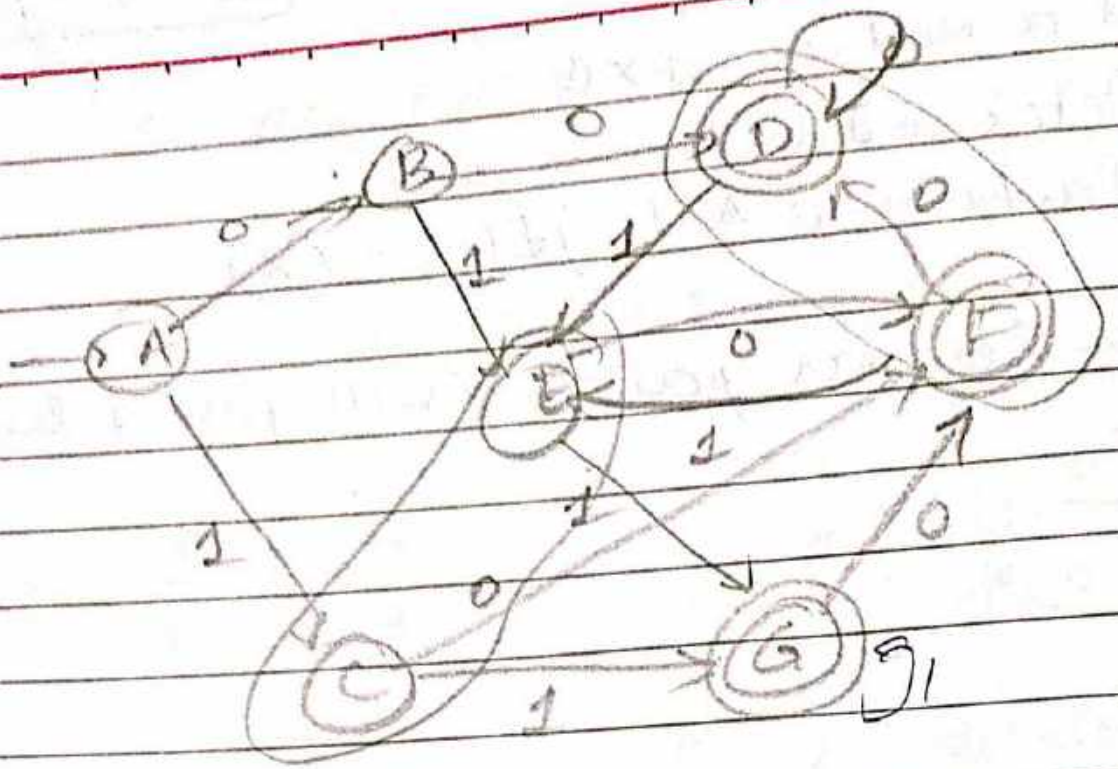
B —1→ E
E —1→ *G
} X

D —1→ E
G —1→ *G
} X

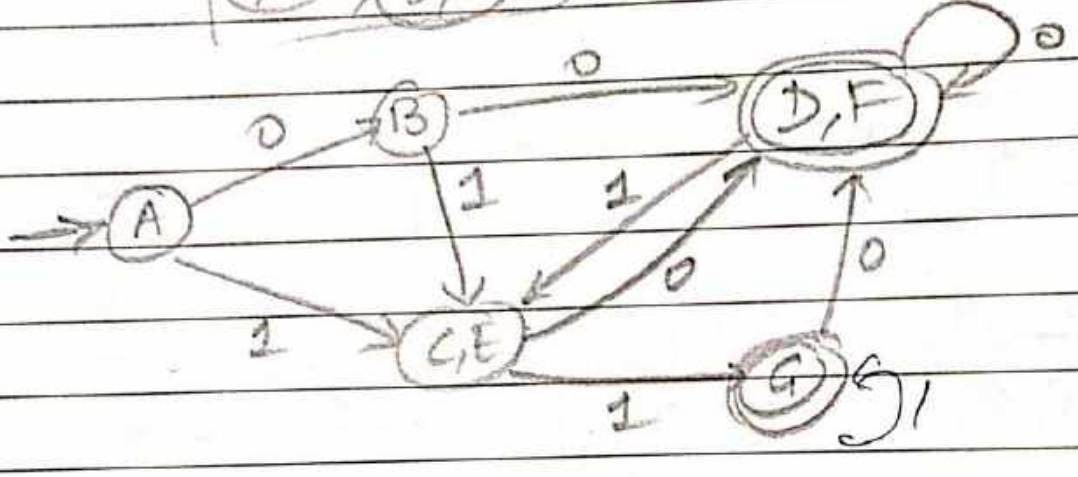
C —1→ *G
E —1→ *G
can't say

F —1→ E
G —1→ *G
} X

Non trivial pairs: (C, E), (D, F).



(C,E), (D,F)



Proof of correctness of Table-filling algorithm

(p, q are indistinguishable by algo ⇓ p, q are equivalent!) $\forall w \in \Sigma^*$

For our example, it worked out easily. May need a lot more time, but clears up things at breadth-level somewhat.

DOMS | Page No.
Date 13/01/22

L-15

outline
- checking equivalent states, distinguishable states
- table-filling algorithm. , correctness
- minimal automata.

→ DFA = (Q, Σ, δ, q_0, F)

p, q are equivalent if $\forall w \in \Sigma^*$, $\hat{\delta}(p, w) \in F$ iff $\hat{\delta}(q, w) \in F$

p, q are distinguishable if $\exists w$: ONE of $\hat{\delta}(p, w)$, $\hat{\delta}(q, w)$ belongs to F. Other to Q\F.

→ Table filling algorithm:
$w = \epsilon$
$|w| \geq 0, 1, 00, \ldots$

- when would problem occur?
  - ↳ If table says equivalent but IRL distinguishable
  - ↳ only tick-mark part is req'd to be verified as being equivalent.

Correctness

↳ If p, q are indistinguishable(✓) by TFAlgo, then, p and q must be equivalent.

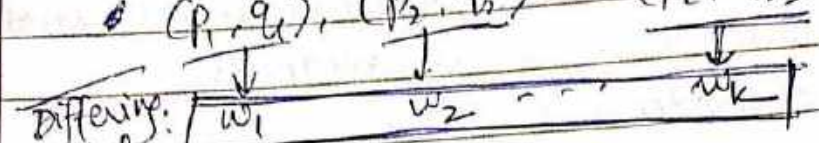
Ps: This statement would be disproven if

{ p, q are indistinguishable by algo (✓)
AND p, q are not equivalent IRL }

↳ $\exists w \in \Sigma^*$, such that $\hat{\delta}(p, w)$ and $\hat{\delta}(q, w)$ differ in their finality/acceptance.

(p, q) = { runcaught / bad pair
escaped pair

- There may be multiple bad pairs.
All bad pairs are distinguishable by some string.



Pick up shortest word, corresponding pair.

$w_j \rightarrow (p_j, q_j)$

where, $|w_j| = \min_{1 \le i \le k} \{|w_i|\}$

(p_j, q_j), $w = a_1 a_2 \cdots a_n$

$n = 0 \Rightarrow w = \epsilon \Rightarrow$ ~~$p_j \in F$ or $q_j \in F$~~

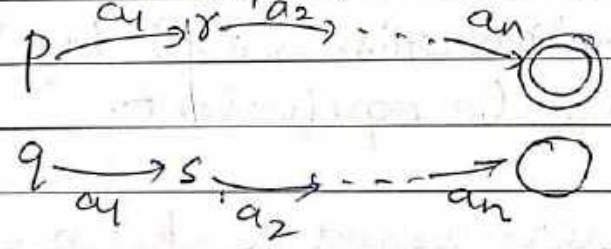
↳ one is final, one is nonfinal

* our algorithm would've caught it in its first step!

~~$n = 1$~~

$n \ge 1 \Rightarrow w = a_1 \cdots a_n$

~~$\hat{\delta}(p, w)$ and $\hat{\delta}(q, w)$~~



If p, q are distinguishable by w,

$\Rightarrow$ using $a_2 \cdots a_n$,

r and s are distinguishable.

~~(p, q)~~, (r, s)

→ Is (r, s) a bad pair?

If (r, s) is a bad pair, they are distinguishable by word of length $(n-1)$.

Our initial assumption that (p, q) are distinguishable by the shortest word ~~would~~ is contradicted.

DOMS | Page No.
Date / /

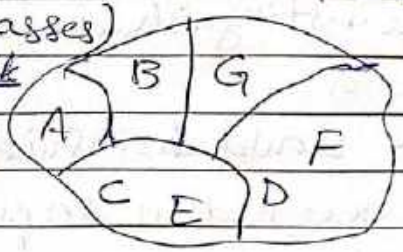
If (r,s) is not a bad pair,
Algo finds that (r,s) is distinguishable and (r,s) are not equivalent, and are disting.,
→ (r,s) is distinguishable + caught by algo.
⇒ By Inductive case of algo., (p,q) would be found to be distinguishable and caught by algo

~~Hence, Irrespective~~
Hence, either our assumption is contradicted, leading to falsifying the statement of incorrectness,
(OR)
The bad pairs are all caught by the algo., using its inductive nature, proving the statement of correctness.

(r,s) not caught ⇒ our choice of w,i is contradicted
(r,s) caught ⇒ (p,q) distinguishable + caught

→ On repeating the algo., we will eventually be able to reduce our automata to a minimal DFA, where each pair of states is distinguishable

Equivalent = No distinguishing string!
↳ Transitive, Reflexive, symmetric, ⇒ Equivalence Relation
⇒ It creates partitions in Q ($\subseteq Q \times Q$)
(equivalence classes)
In our ex, (or) block



blocks of partition / equivalence classes.

Note: Blocks of the partition are mutually exclusive and exhaustive.
↳ generated by algorithm of distinguishability

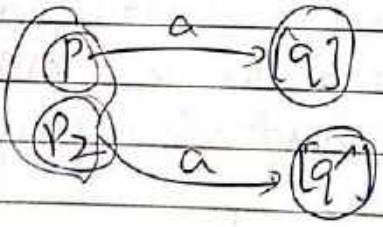
let our original machine be $M \rightarrow (Q, \Sigma, \delta, q_0, F)$,
states of our minimal DFA, M', will be the blocks from equivalence classes of Q, as Q'.
↳ $(Q', \Sigma, \delta', q_0', F')$

$\{A\} = [A]$
$\{B\} = [B]$
$\{C, E\} = [C] = [E]$
$\{D, F\} = [D] = [F]$
$\{G\} = [G]$

$q_0' = [q_0]$
F' = blocks containing atleast one final state of M
i.e. blocks containing final state of M

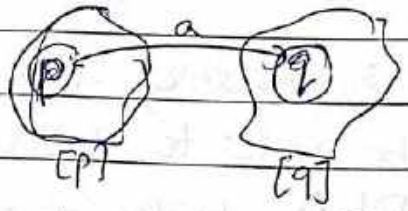
let $(p_1, p_2, \cdot) \xrightarrow{a} [q]$.

why is $[q'] = [q]$.



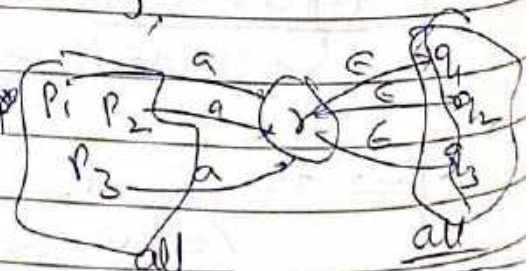
If q, q' are distinguishable, one of them leads to F', one does not.
If q, q' are distinguishable, by induction, aw would distinguish p_1, p_2
$\Rightarrow p_1, p_2$ are not in same equivalence class
$\Rightarrow$ Assumption incorrect
$\Rightarrow q, q'$ are equivalent $[q] = [q']$

$\delta : Q' \times \Sigma \rightarrow Q'$
$\delta'([p], a) \rightarrow [\delta(p, a)]$



δ' links $[p]$ to $[q]$.

It's like if,



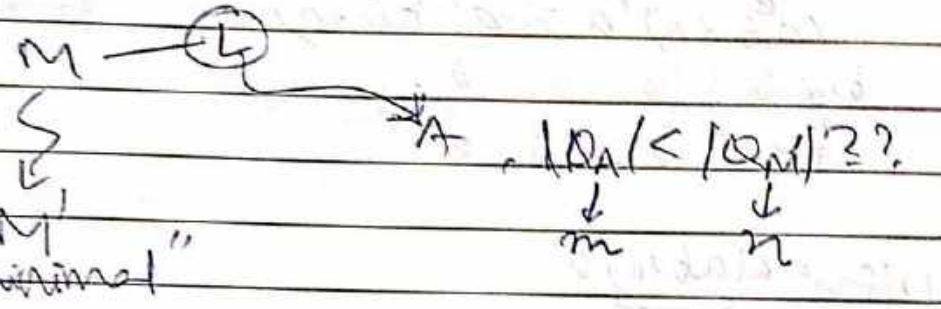
all from $[p] \xrightarrow{a}$ all from $[q]$

Set of states, Q', is set of blocks due to partition by equivalence classes.

Runs of M can be recreated in M' [and vice versa]. So lang(M) = lang(M')

$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} f$ ⟷ $[q_0] \xrightarrow{a_1} [q_1] \xrightarrow{a_2} \cdots \xrightarrow{a_n} [f]$ (quotient Automata)

Since all ~~equivalent~~ of the states are now distinguishable in M', it is (informally) a minimal automata.

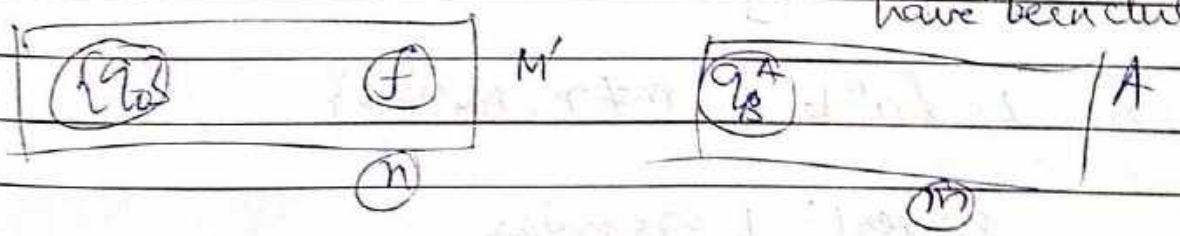


Assume that $\exists A : |Q_A| = m < n = |Q_M|$.

~~Since~~ Since $|Q_M| = n$, ~~∄~~ there are exactly 'n' equivalence classes in Q.

If $\exists A : |Q_A| = m < n$,

then $\exists q \in Q_A$: ~~elements of~~ ~~both~~ elements of diff. equivalence classes have been clubbed.



$\left(\begin{matrix} p \xrightarrow{w} q \\ \neq \\ p' \xrightarrow{w} q \end{matrix}\right)$ Then, ∃ two states going to q on w.

p eq. to q

p' eq. to q

(Goal: Pigeonhole principle)

~~But~~ $p \Rightarrow [p] = [p']$. That isn't possible ⟹ Minimal!

Quiz-1-Review (L-16)

Q1) $(ba)^*$

Q3) (a) $(r+s)^* = r^* + s^*$ (X)

(d) $(rs+s)^*r = r(sr+r)^*$ (X)

(b) $(rs+r)^*r = r(sr+r)^*$

$(ab+a)^*a = a(ba+a)^*$ ← $a \in r,\ b \in s$; $r \to a,\ s \to b$

begins: a a

ends: a a

LHS: $w \in (ab+a)^*a$

$(ab+a)(\cdots)(ab+a)a$

$= a(b+\epsilon)a(b+\epsilon)a \cdots a(b+\epsilon)a$

$= a(ba+a)(ba+a)\cdots(ba+a)$

$= a(ba+a)^*$ → RHS

Q2) L is regular

$L^R = \{w^R \mid w \in L\}$ → ? Reverse DFA

Q4) (a) $L = \{a^m b^n \mid m \neq n,\ m, n > 0\}$

Player 1: L = regular

Player 2: 'n' (no. of states)

Player 1: $w \in L$ (depends)

$w = a^{n+1}b^n$

DOMS | Page No.
Date / /

$\underbrace{aaa\ldots a}_{xy}\underbrace{a\,bbb\ldots b}_{z}$

$|y| \neq 0$

$|xy| \leq n$

$|y| = k$

$x = a^{n+1-k-b_1}$

$y = a^k$

$z = a^{k_1} b^n$

Player 1: pump y 'o' times

$\Rightarrow xz = a^{n+1-k-k} b^n$

$\Rightarrow xz = a^{n+1-k} b^n$

$k \geq 1$. If $k=1$, $a^n b^n \in L$

$k > 1$, $a^{\square} < n$.

$(\#a < \#b) \notin L$

(b) $L = \{ww^R | \ldots\}$

pick ex, etc...

1 P's win / Complete

Q.6) ~~b~~ Which make DFA

Q5) $L_1 \cup L_2$: regular

$a^m b^n$ $a^m b^n$ → ∪ : reg

∩ : reg

$a^m b^n b^p a^q$

$m > n$ $p > q$

$a^m b^{n+p} a^q$

$m=5$ $n=10$

$a b^3 a$

$ab \cdot b^2 a$

$a\, b^3\, 1$

$a\, b^1\, b^2\, a$

$a^2 b^3 a^1$

$a^1 \cdot b^3 a$

$ab^3 a$

Ex) a^n → a^*
prime not prime

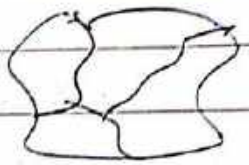
19-01-2022

L17

→ Equivalence Relations on Σ^*
- Reflexive, Symmetric, Transitive
- Partitions the set.

$R \subseteq \Sigma^* \times \Sigma^*$

then,



(A) $L \subseteq \Sigma^*$

$x, y \in \Sigma^*$

$x \sim_L y$ iff

~~(B)~~ $xw \in L$ iff $yw \in L$ ~~$\forall w \in L$~~ $\forall w \in \Sigma^*$

($x \in \in L$ iff $y \in \in L$)
⇒ ($x \in L$ iff $y \in L$)

($xz \in L$ iff $yz \in L$)

→ (Exercise: $\sim_L : \Sigma^* \times \Sigma^*$ is eq. rel^n
Check → refl, → Symm, → Trans)

Ex) $\Sigma = \{a, b\}$
$L \subseteq \Sigma^*$ $r = (ab + ba)^*$ $L(r) = L$

$\sim_L$ can be defined on any lang.
$x \sim_L y$ iff $\forall z$, $xz \in L$ iff $yz \in L$

① $[\varepsilon] = L = \{\varepsilon, ab, ba, abba, baab, \ldots\}$

Let $x=\epsilon$, $y=ab$. $\quad x, y \in L$

$xa = \epsilon \cdot a$. $\quad y = ab \cdot a$

$xa \notin L \longleftrightarrow ya \notin L$.

(or)

$xb = \epsilon \cdot b$ $\quad yb = ab \cdot b$

$xb \notin L \longleftrightarrow yb \notin L$

(or)

$x \cdot ab = \epsilon \cdot ab$ $\quad y \cdot ab = (ab)^2$

$(\in L) \longleftrightarrow (\in L)$

etc.

$x \cdot w \in L$ iff $y \cdot w \in L$.

$L \cdot a \xrightarrow{b \cdot L} \in L$

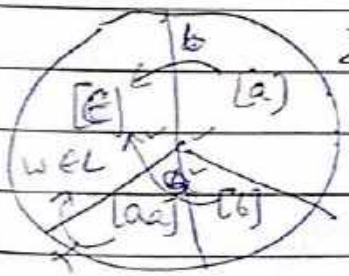
$L \cdot b \xrightarrow{a \cdot L} \in L$.

$(a+b)^* \setminus (aa+bb) \longrightarrow \notin L$.

$\longrightarrow a \cdot [a]$

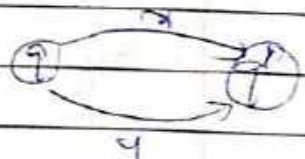
$\longrightarrow b \cdot [b]$

eq. class.



Defn: Given a DFA, $M=(Q, \Sigma, \delta, q_0, F)$, $L = lang(M)$, we can define an equivalence rel^n b/w 2 strings as $x \sim_M y$

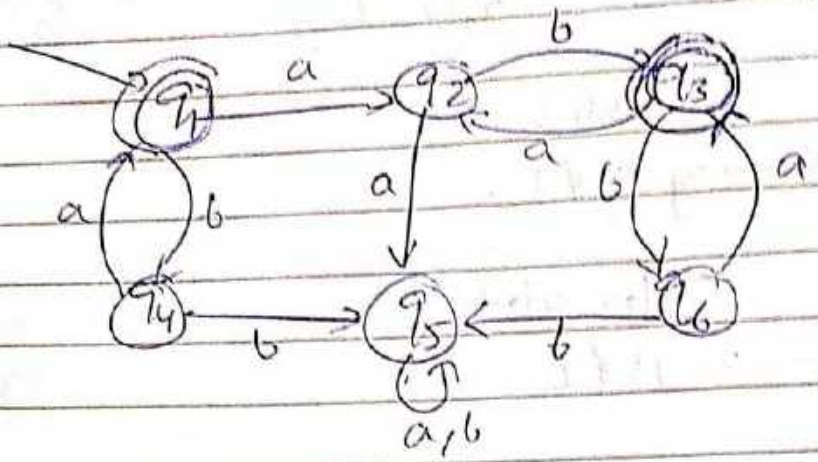
(x and y are related wrt M, if $\exists q \in Q : \hat{\delta}(q, x) = \hat{\delta}(q, y)$)



$\sim_M \subseteq \Sigma^* \times \Sigma^*$

(Exercise: $\sim_M$ is equivalence rel^n)

Ex)



The $q_4 - q_4$ loop : $a(ba)^*$

$q_1 \to (q_2 \to q_5)$: $ab(ab)^*$

$q_1 \to (q_1 \to q_4)(q_3)$: $ab((ab)^* + (ba)^*)$

$\boxed{(ab+ba)^*}$

① Eq_1 : $(ba)^*$ $q_1 \rightsquigarrow q_1$

② Eq_2 : $L \cdot a + a$ $q_2 \rightsquigarrow q_2$

③ Eq_3 : $ab \cdot L$ $q_3 \rightsquigarrow q_3$

④ Eq_4 : $b(ab)^*$ $q_4 \rightsquigarrow q_4$

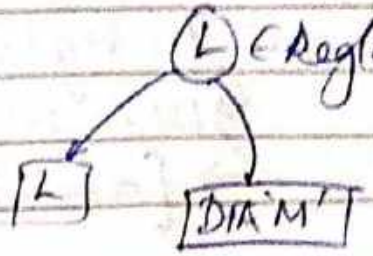
⑤ Eq_5 : $L \cdot (aa+bb)\Sigma^*$ $q_5 \rightsquigarrow q_5$

⑥ Eq_6 : $L \cdot b$ $q_6 \rightsquigarrow q_6$

Myhill Nerode Theorem

$x \sim_M y \Rightarrow x \sim_L y$
eq. wrt M eq. wrt L

($\sim_M$ Refines $\sim_L$)



($\sim_M$ refines $\sim_L$)

$x \sim_L y$ iff : $\forall z \in \Sigma^*, xz \in L \leftrightarrow yz \in L$.

Ex) $L = (ab + ba)^*$

① $[\epsilon] = L$ ⟶ (L)

② $[a] = L \cdot a \rightarrow b \cdot L \rightarrow$ (L)

③ $[b] = L \cdot b \rightarrow a \cdot L \rightarrow$ (L)

④ $[aa] = [(aa+bb)L] \rightarrow$ (X) $\rightarrow L$.

$M = (Q, \Sigma, \delta, q_0, F)$

$\sim_M \subseteq \Sigma^* \times \Sigma^*$

$x \sim_M y$ iff $\exists q \in Q : \hat{\delta}(q, x) = \hat{\delta}(q, y) = q'$.

- cleaner def^n

$x \sim_M y$ iff : $\exists q \in Q : \hat{\delta}(q_0, x) = \hat{\delta}(q_0, y) = q$

Fix source

DOMS | Page No.
Date 20/01/2022

L-18 Pg 4

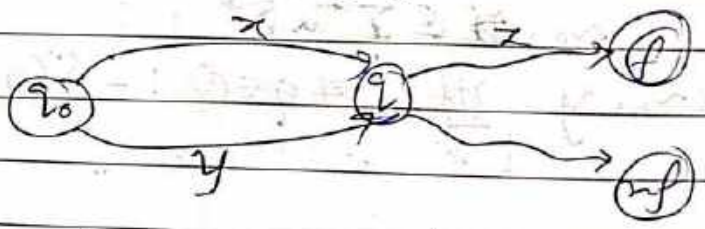
⇒ Given: L = reg lang $\rightsquigarrow (\sim_L \subseteq \Sigma^* \times \Sigma^*)$
↳ DFA, M : $L = L(M)$
↳ $[\sim_M \subseteq \Sigma^* \times \Sigma^*]$

Theorem ①

(Difference b/w $\sim_M$ and $\sim_L$?)
To prove $(x \sim_M y \longrightarrow x \sim_L y)$

Pf: consider $x \in \Sigma^*$.
Let $\hat{\delta}(q_0, x) = q$.
Consider some $z \in \Sigma^*$,
$\hat{\delta}(q_0, z)$ [$\in F$ or $\notin F$]

Given, $x \sim_M y$.
$\Rightarrow \hat{\delta}(q_0, y) = q$.
So, for any $z \in \Sigma^*$
$\hat{\delta}(q, z)$ [$\in F$ or $\notin F$],
since,



$\left(\begin{array}{l} xz \in L \Rightarrow yz \in L \\ yz \in L \Rightarrow xz \in L \end{array} \right) \forall z$

Hence, $\boxed{x \sim_L y}$

Hence, $\boxed{x \sim_M y \Rightarrow x \sim_L y}$

What does this mean?
"$\sim_M$ refines $\sim_L$"

→

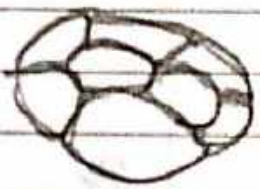
DOMS | Page No.
Date / /

Non-Technical Example

Ex) · Consider set of all cities. (Q)

$[x \; R_D \; y]$ iff $x, y \in$ same district

⇒ we get a districtwise partition.



Also,

$[x \; R_S \; y]$ iff $x, y \in$ same state

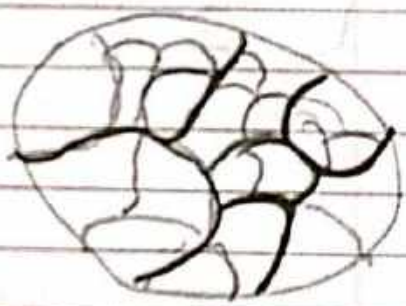
⇒ we get a statewise partition,
less refined.

same district ⟶ same state

$x R_D y \longrightarrow x R_S y$

$x R_S y \not\longrightarrow x R_D y$

R_D refines R_S



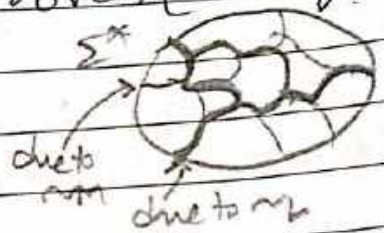
Eq. classes of R_D are inside eq. classes of R_S

districts ⟶ $|d| \geq |s|$ ⟵ states

$|\text{eq. classes}(R_D)| \geq |\text{eq. classes}(R_S)|$

Refinement = breaks existing eq. classes into smaller eq. classes.

→ Proves [$\sim_M$ refines $\sim_L$]



$\#eq.cl(\sim_M) \geq \#eq.cl(\sim_L)$

$\#states \geq \#eq.cl(\sim_L)$

From example,

$Eq_1 : (ba)^*$

$Eq_2 : La + a$

$Eq_3 : ab \cdot L$

$Eq_4 : b(ab)^*$

$Eq_5 : L(aa+bb)\Sigma^*$

$Eq_6 : ab \cdot L \cdot b$

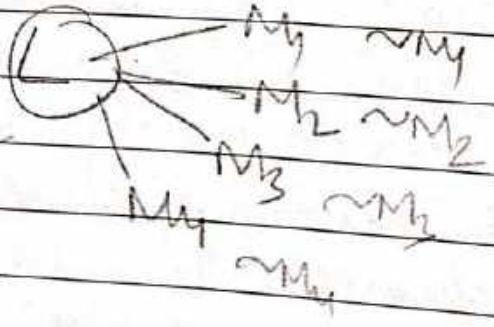
$[\epsilon] = L$

$[a] = L \cdot a$

$[b] = L \cdot b$

$[aa] = L(aa+bb)\Sigma^*$

$b(ab)^* = (ba)^* b$



#states in M_L ($\sim_L$) $\geq$ # Eq.cl of $\sim_L$

↑ lowerbound

∃ DFA, where $|Q| = |\#\text{Eq.cl of } \sim_L|$
(minimal automata)
Myhill-Nerode: Gave the lower bound for $|Q|$.

Pf: $M = (Q, \Sigma, \delta, q_0, F)$

$Q = \{[x] : x \in \Sigma^*\}$ $\sim_L$

$\Sigma = \Sigma$

$q_0 = [\epsilon]$

$F = \{[x] \mid x \in L\}$

$\delta([x], a) = [xa]$ $(x \in \Sigma^*, a \in \Sigma)$

→ $\text{lang}(M) = L$. #

Pf:

① M is a DFA

(How many states in M?
$|Q|$ is finite? why?
why #eq-cl($\sim_L$) is finite?)

Assume L is regular ⇒ ∃ DFA, D: lang(D) = L
⇒ construct $\sim_D$ for D.
⇒ $\sim_D$ refines $\sim_L$
⇒ $|\#\text{eq cl}(\sim_D)| \geq |\#\text{eq. cl}(\sim_L)|$
$|\#\text{states in D}| \rightarrow |Q| \geq |\#\text{states in M}|$
(finite) ⟹ finite

② Why is δ well defined?

$[x] \xrightarrow{a} [xa]$
$x' \in [x]$? $[x'a]$
$x' \xrightarrow{a} [x'a]$

Date / /

Consider $x, x' \in [x]$

$\Rightarrow x \sim_L x'$ (iff $xa \sim_L x'a$)

By def^n of $\sim_L$,

$x \sim_L x'$, iff $\forall z \in \Sigma^*$, $xz \sim_L x'z$

$\Rightarrow x \sim_L x' \Longrightarrow xa \sim_L x'a$

$\Rightarrow [xa] = [x'a]$

③ $L = lang(M)$?

Step ① $\forall x, y \in \Sigma^*$

$\hat{\delta}([x], y) = [xy]$

$[x] \xrightarrow{y} [xy]$

Pf: Ind^n on length of y.

Base case: $|y| = 0$. $y = \varepsilon$

$\hat{\delta}([x], \varepsilon) = [x]$ (by def^n)

Inductive Hypothesis: $|y| = n$,

$\hat{\delta}([x], y) = [xy]$.

Inductive Step $|y| = n+1$.

let $y = y'a$. $|y'| = n$, $|y| = n+1$.

$\hat{\delta}([x], y) = \hat{\delta}([x], y'a)$

$= \delta(\hat{\delta}([x], y'), a)$

By ind hyp.

$= \delta([xy'], a)$

$= [xy'a]$ by def^n

Step ② $x \in L(M)$, when $\delta([\epsilon], x) \in F$

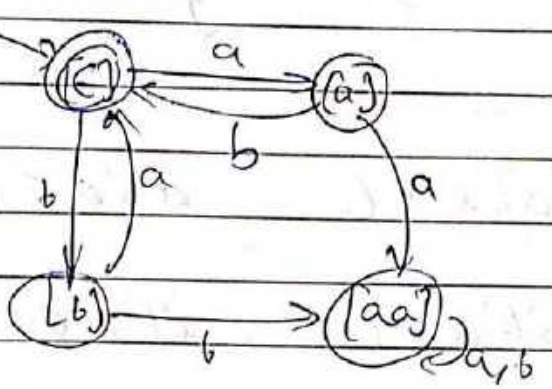
$\hat{\delta}([\epsilon], x) = [x]$, by step ①.

where, $[x]$ is a final state

iff $x \in L$

$\Rightarrow L = \text{Lang}(M)$

Ex) For $(ab + ba)^*$



Corollary

L is regular iff $\sim_L$ has finitely many eq. classes

Pf ($\Rightarrow$)

Assume L is regular

$\Rightarrow \exists$ DFA D for L.

$\#\text{eq cl}(\sim_D) \geq \#\text{eq. cl}(\sim_L)$

$\text{lang}(M) = L \Rightarrow L$ is regular

Regular L $\longrightarrow$ Minimal automata

$\hookrightarrow$ DFA 'D' collapse

$\#\text{eq-cl}(\sim_L)$ is finite

$\hookrightarrow$ rel^n is of finite index

T-2 + L-19B

21-01-2022

Ex) $L = \{a^i b^i \mid i \geq 1\}$

- Pumping lemma
- Myhill Nerode Theorem
- Sufficient to show that #eq.cl($\sim_L$) is infinite

$$\frac{a^1 b^1}{x} \quad \underset{\sim_L}{\overset{?}{=}} \quad \frac{a^2 b^2}{y}$$

$z = \epsilon$

$a^1 b^1 \epsilon \in L$, $a^2 b^2 \epsilon \in L$

$z = a$

$a^1 b^1 a \notin L$, $a^2 b^2 a \notin L$

$z = b$.

$a^1 b^2 \notin L$, $a^2 b^3 \notin L$

Behaves similarly for diff z'.

Strings in L,

$(a^1 b^1, a^2 b^2, a^3 b^3) \rightarrow$ eq-cl.

Pick up $x = a$, $y = b$. $(x, y \in \Sigma^*)$

$\sim_L \subseteq \Sigma^* \times \Sigma^*$

$z = \epsilon$, $a\epsilon \notin L$ $b\epsilon \notin L$

$z = a$, $a^2 \notin L$ $ba \notin L$

$z = b$ $\boxed{ab \in L \quad b^2 \notin L}$

$\Rightarrow [a] \neq [b]$

[a] [b]

Consider $x = a^2$, $b = a^3$

$z = b^2$, $a^2 b^2 \in L$, $a^3 b^2 \notin L$

$\Rightarrow [a^2] \neq [a^3]$ in general, $[a^n] \neq [a^{n+1}]$

$[a^n] \neq [a^k]$ for $n \neq k$.

Hence, there are infinite eq. classes,

$[a^0], [a^1], [a^2], \ldots$ since no. of values of n is infinite

($x = a^i$, $y = a^j$ $\quad i \neq j$

$\underline{z = b^i}$ differentiates them. ($a^i b^i \in L$, $a^j b^i \notin L$))

$\Rightarrow$ $[a]$ $[a^2]$ $[a^3]$ . . . . . . etc,

infinite no. of eq. cl ($\sim_L$)

MyhNT: **L is regular iff $\sim_L$ has finite index**

$\Rightarrow L = \{a^i b^i \mid i \geq 1\}$ is not regular

Q) $L = \{a^p \mid p \text{ is prime}\}$ is not regular

use Myhill-Nerode Theorem.

consider: $x = a^2$, $y = a^3$

Is $x \sim_L y$? $(x \in L, y \in L)$

let $z = a$.

$x = a^3$, $y = a^4$ $(x \in L, y \notin L)$

$\Rightarrow [a^2] \neq [a^3]$

consider: $x = a^3$, $y = a^5$ $(x \in L, y \in L)$

Is $x \sim_L y$?

let $z = a$.

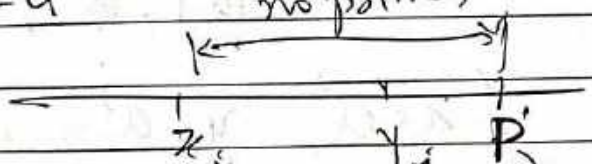
$x = a^4$, $y = a^6$ $(x, y \notin L)$

let $z = a^2$, $x = a^5$, $y = a^7$ $(x, y \in L)$

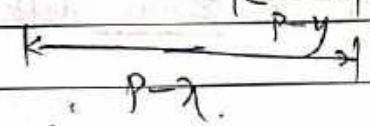
let $z = a^4$, $x = a^7$, $y = a^9$ $(x \in L, y \notin L)$ $\Rightarrow [a^3] \neq [a^5]$

Consider 2 primes ~~e~~ i, j.

$x = a^i$, $y = a^j$



we can find a^p such that



$z = a^{p-j}$.

$\Rightarrow z = a^{i+p-j} \notin L$

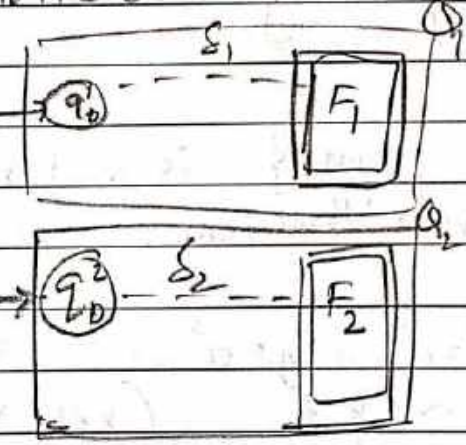
$y = a^{j+p-j} \in L$

$\Rightarrow [a^i] \neq [a^j]$, for $i \neq j$

$\Rightarrow \sim_L$ has infinite index.

Hence, L is not regular.

Q3) DFA for intersection



ϵ-NFA for $L_1 \cap L_2$:

$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$

We could also just do:

$Q = Q_1 \times Q_2$ $\quad q_0 = (q_0^1, q_0^2) = \{q_0^1, q_0^2\}$

$\Sigma = \Sigma$ $\quad F = F_A \times F_B$

$\delta : (Q \times Q_2) \times \Sigma \to (Q_1 \times Q_2)$

$(Q \times \Sigma \to Q)$

where,

$$\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$$

$D_1 = (Q_1, \Sigma, \delta_1, q_0^1, F_1)$

$D_2 = (Q_2, \Sigma, \delta_2, q_0^2, F_2)$

↓

$D_3 = (Q_3, \Sigma, \delta_3, q_0^3, F_3)$

$Q_3 = Q_1 \times Q_2$ (all ordered tuples)

$q_0^3 = \{q_0^1\} \times \{q_0^2\} = (q_0^1, q_0^2)$

$L(D_1 \times D_2) = L(D_3)$

$w \in L_1 \cap L_2 \Rightarrow w \in L(D_3)$

↓

$w \in L_1$ and $w \in L_2$

↓

$\hat{\delta}_1(q_0^1, w) \in$ lang (D_1)

$\hat{\delta}_2(q_0^2, w) \in$ lang (D_2)

$\Rightarrow \hat{\delta}_3((q_0^1, q_0^2), w) \in$ lang (D_3).

$L_1 \cap L_2 = \{w \mid w \in L_1 \wedge w \in L_2\}$

$$\delta_3((q_1, q_2), a) = (\underbrace{\delta_1(q_1, a)}_{q_1'}, \underbrace{\delta_2(q_2, a)}_{q_2'})$$

So, δ_3: ~~$Q_3 \times \Sigma \to Q_3$~~

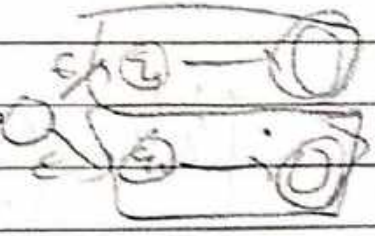
$Q_3 \times \Sigma \to Q_3$

$\Sigma = \Sigma$

$F_3 = F_1 \times F_2$ Tuples containing a final state from F_1 AND final state from F_2

Q4) DFA for union

ϵ-NFA for $L_1 \cup L_2$: simple



OR) direct DFA -

just make $F_3 = (Q_1 \times F_2) \cup (Q_2 \times F_1)$

$w \in lang(D_1)$ and $w \in lang(D_2)$

There exists runs of D_1/D_2 on w, $r_1(w)$, $r_2(w)$

$w = a_1 a_2 \cdots a_n$.

$(q_0^1 \overset{w}{\rightsquigarrow} f_1 \in F_1)$

$(q_0^2 \overset{w}{\rightsquigarrow} f_2 \in F_2)$

pf by Induction on length of w

Base case: $|w| = 0$, $w = \epsilon$

$\Rightarrow r_1(\epsilon) : q_0^1 \xrightarrow{\epsilon} q_0^1$

$r_2(\epsilon) : q_0^2 \xrightarrow{\epsilon} q_0^2$

$\Downarrow$

$r_3(w) : (q_0^1, q_0^2) \xrightarrow{\epsilon} (q_0^1, q_0^2)$

Inductive hypothesis: $|w| = n$, $w = a_1 \cdots a_n$

$\Rightarrow r_1(w) : q_0^1 \xrightarrow{a_1} \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n^1$

$r_2(w) : q_0^2 \xrightarrow{a_1} \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n^2$

On appending a_{n+1}, $\begin{pmatrix} q_n^1 \\ q_n^2 \end{pmatrix} \xrightarrow{a_{n+1}} \begin{pmatrix} q_{n+1}^1 \\ q_{n+1}^2 \end{pmatrix}$

$\delta_1(q_n^1, a_{n+1}) = q_{n+1}^1$

$\delta_2(q_n^2, a_{n+1}) = q_{n+1}^2$

$\Rightarrow r_3(w) : \begin{pmatrix} q_0^1 \\ q_0^2 \end{pmatrix} \xrightarrow{a_1} \xrightarrow{a_2} \cdots \xrightarrow{a_n} \begin{pmatrix} q_n^1 \\ q_n^2 \end{pmatrix} \xrightarrow{a_{n+1}} \begin{pmatrix} q_{n+1}^1 \\ q_{n+1}^2 \end{pmatrix}$

$\left[r_3(w) : \begin{pmatrix} q_0^1 \\ q_0^2 \end{pmatrix} \xrightarrow{w} \begin{pmatrix} f_1 \\ f_2 \end{pmatrix} \right]$ Accepting, for $\begin{pmatrix} q_0^1 \xrightarrow{w} f_1 \\ q_0^2 \xrightarrow{w} f_2 \end{pmatrix}$

$\Leftrightarrow w \in L(M_3)$

Stitching of individual runs to get run of product machine

Other way:

$w \in \text{lang}(D_3)$

To prove, $w \in L_1 \cap L_2$.

Simple pf.

$w \in \text{lang}(D_3)$:

$\exists\, r_3(w)$: $\left(\begin{pmatrix} q_0^1 \\ q_0^2 \end{pmatrix} \xrightarrow{w} \begin{pmatrix} f_1 \\ f_2 \end{pmatrix} \in (F_1 \times F_2)\right)$

Constructing runs for D_1, D_2 as:

$$\begin{pmatrix} D_1 : r_1(w),\ q_0^1 \xrightarrow{w} f_1 \in F_1 \\ D_2 : r_2(w),\ q_0^2 \xrightarrow{w} f_2 \in F_2 \end{pmatrix}$$

Projecting $r_3(w)$ on component 1 and 2.
↓ ↓
$r_1(w)$ $r_2(w)$

Both $r_1(w)$, $r_2(w)$ are accepting runs

$\Rightarrow w \in L_1,\ w \in L_2$

$\Rightarrow \boxed{w \in L_1 \cap L_2}$

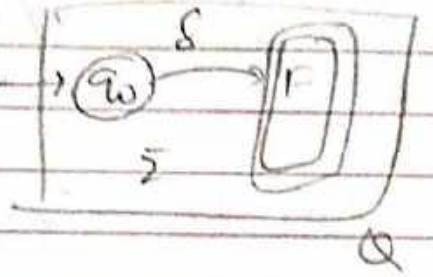
→

Q4) For union,

$$L_3 = (Q_1 \times \Gamma_2) \cup (Q_2 \times \Gamma_1)$$

Q5) First Halves

(Red pebble / Green pebble / Blue pebble) Diff Initiating pts in same machine.



Pebble guessed if $q_0 \xrightarrow{x} p \xrightarrow{y} f$

p → guessed

Is the guess correct?

~~Moves~~: Keep B static at some state, and let G be at that state
Keep R at initial state

As R moves 1 step forward, according to δ,
G moves 1 step non-deterministically.

Win: (R, B @ same state, G ∈ F)

Formal NFA construction:

$N = (Q', \Sigma, \delta', q_0, F')$

$Q' = Q^3 \cup \{q_0\}$

↳ additional state

DOMS | Page No.
Date / /

RGB.

$\delta'(q_s, \epsilon) = \{(q_0, q, q) \mid q \in Q\}$

$\delta'((p,q,r), a) = \{(\delta(p,a), q, \delta(r,b)) \mid b \in \Sigma\}$

$q_0' = q_s$

$F' = \{(q,q,f) \mid q \in Q, f \in F\}$

$\Sigma = \Sigma$

pf of correctness.

Is firstHalves(L) = Lang(N) ?

For $x \in \Sigma^*$, $\delta(q,x)$ = set of states (state) from q on x, ones $\delta(q_0, x)$

forward: firsthalves(L) $\subseteq$ Lang(N)

let $x \in$ firsthalves(L).

By def^n, $\exists\, y \in \Sigma^* \mid xy \in L, \wedge |x| = |y|$

let $\delta(q_0, x) = r$, $\delta(q_0, xy) = f$.

Since $xy \in L \Rightarrow f \in F$

$\Rightarrow \delta(r, y) = f$

Now. $(q_0, r, r) \in \delta'(q_s, \epsilon)$,

and $(\delta(q_0, x), x, \delta(r, y)) \in \delta'((q_0, r, r), x)$

$\Rightarrow (r, r, f) \in \delta'((q_0, r, r), x)$

By def^n of F', $(r, r, f) \in F'$

$\Rightarrow x \in$ Lang(N)

backward: let $x =$ lang(N). , $\exists\, r \in Q, f \in F$; $\delta'((q_0, r, r), x)$ (r, r, f)

$\Rightarrow \delta(q_0, x) = r$

By def^n of δ', for each step of first coord, the third coord also takes one step. Hence, $\exists\, y \in \Sigma^*$ / $|x| = |y|$, $\delta(r, y) = f$. $\Rightarrow \delta(q_0, xy) = f$

$\Rightarrow x \in$ firsthalves(L).