DFA

deterministic
→ two arrows
→ 1 or 0

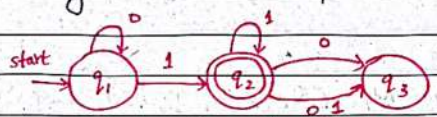
→ simplest model of computation
→ very limited memory

* no storage in automata.

DETERMINISTIC FINITE AUTOMATA

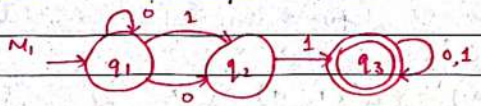
Date:

→ A finite automaton is a model for computation.
→ examples include: automatic doors which swing open when they sense some person.



→ Call it M_1 it has 3 states, q_1, q_2 and q_3
→ Initial state: q_1 (with an incoming arrow)
→ Accepting State: q_2 (with double circle)
→ Arrows shows transitions
→ Input sequence: 1101 to M_1 (Accepted)

→ A finite automata is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where
Q is a finite set called states.
Σ is a finite set called alphaset symbols. $\{0,1\}$
$\delta: Q \times \Sigma \to Q$ is a transition function.
q_0 is the start state.
F is a set of accept states (or final states) $F \subseteq Q$



$M_1 = (Q, \Sigma, \delta, q_1, F)$
$Q = \{q_1, q_2, q_3\}$
$\Sigma = \{0, 1\}$
$F = \{q_3\}$

$\delta =$

	0	1
q_1	q_1	q_2
q_2	q_1	q_3
q_3	q_3	q_3

Power of Σ $\quad \Sigma^n = 2^n$

Σ^0 = Set of all strings with length '0' = λ, ε $\Rightarrow 2^0$

Σ^1 = " " " $1 = \{a, b\}$ $\Rightarrow 2^1$

Σ^* (Kleene Closure) → Infinite Language. Date: _____

→ A string is finite sequence of symbol in Σ

→ A Language is a set of strings (finite or infinite)

→ The empty string ε is the string of length 0

→ The empty language. ϕ is the set with no strings.

→ If A is the set of all strings that machine M accepts we say that A is the language of machine M.

→ We write $L(M) = A$ & say M accepts A or M recognizes A.

→ A machine may accept several strings but it recognizes only one language.

$$L(M) = A = \{w \mid M \text{ accepts } w\}$$

→ The concatenation of strings x and y is written xy.

→ x^k denotes the concatenation of x with itself k times

$01^5 0 = 0111110$

→ Union & intersection

$A \cup B = \{x \mid x \in A \text{ or } x \in B\}$

$A \cap B = \{x \mid x \in A \text{ and } x \in B\}$

→ Complement

$\bar{A} = \{x \in \Sigma \mid x \notin A\}$

→ Concatenation

$AB = \{xy \mid x \in A \text{ and } y \in B\}$

→ $A^* = A^0 \cup A^1 \cup A^2 \cup \ldots$

$\Sigma^* = \Sigma^+ + \Sigma^0$ OR $\Sigma^+ + \varepsilon$

$\Sigma^* - \varepsilon = \Sigma^+$

Finite automata

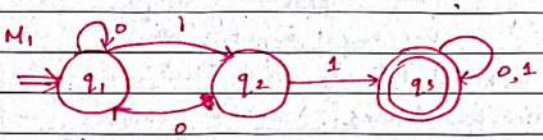
- FA with output
  - Moore
  - Mealy
- FA without output
  - DFA
  - NFA
  - ε-NFA

Regular Languages → DFA ✓

So what languages are NOT REGULAR?
→ lang. which are not recognized by any FSM
→ which require memory.

Date:

→ A language is called regular language if some finite automata recognizes it.



$L(M_1) = \{w \mid w \text{ contain substring } 11\} = A$

Therefore A is regular.

Regular Operations

Let A, B be languages:

UNION $A \cup B = \{w \mid w \in A \text{ or } w \in B\}$

CONCATENATION $A \circ B = \{xy \mid x \in A \text{ and } y \in B\} = AB$

STAR (Kleene closure) $A^* = \{x_1 \dots x_k \mid \text{each } x_i \in A \text{ for } k \geq 0\}$

Regular Expressions

→ Built from Σ, members $\Sigma, \phi, \varepsilon$ [Atomic]
→ By using $\cup, \circ, *$ [composite]

Examples

→ $(0 \cup 1)^* = \Sigma^*$ gives all strings over Σ.
→ $\Sigma^* 1$ gives all strings that ends with 1
→ $\Sigma^* 11 \Sigma^*$ = all strings that contain $11 = L(M_1)$

★ The class of regular languages is closed under:
① Union Operation ② Concatenation Operation.

$\Sigma^* 1$

R R R Date:

THEOREM 1.25 $A \cup B = C$

The class of regular languages is closed under union operation. In other words, if A_1 and A_2 are regular languages, so is $A_1 \cup A_2$

PROOF:

Let M_1 recognize A_1, where $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and M_2 recognize A_2, where $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$.

Construct M to recognize $A_1 \cup A_2$, where $M = (Q, \Sigma, \delta, q_0, F)$

(1) $Q = \{(r_1, r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$

This set is **Cartesian Product** of sets Q_1 and Q_2 and is written $Q_1 \times Q_2$. It is the set of all pairs of states, the first from Q_1 and the second from Q_2. - (tells maximum number of states)

(2) Σ, the alphaset is the same as in M_1 and M_2. In this theorem and in all subsequent similar theorems we assume for simplicity that both M_1 and M_2 have same input alphaset Σ. The theorem remains true if they have different alphasets, Σ_1 and Σ_2. We would then modify the proof to let $\Sigma = \Sigma_1 \cup \Sigma_2$

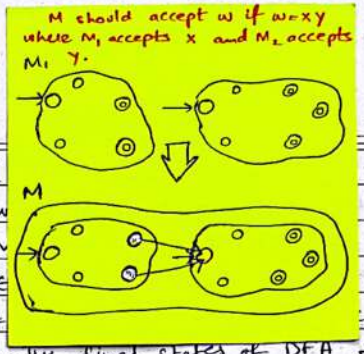
(3) δ, the transition function, is defined as follows. For each $(r_1, r_2) \in Q$ and each $a \in \Sigma$, let

$$\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$$

Hence δ gets a state of M (which actually is pair of states from M_1 and M_2) together with an input symbol and returns M's next state.

* cannot make repeating states.

(CONCATE-NATION) →



(4) q_0 is the pair (q_1, q_2). (

(5) F is the set of pairs in w

accept state of M_1 or M

$F = \{(r_1, r_2) \mid r_1 \in$

This expression is same as $F =$: that it is not same as $F = F_1 \times F_2$). The final states of DFA should contain final state of M_1 and M_2.

THEOREM 1.26

The class of regular languages is closed under concatenation operation.

In other words, if A_1 and A_2 are regular languages then so is $A_1 \circ A_2$.

* DFA does not have memory.

PROOF:

To prove this theorem, let's try something along the lines of proof of union case. As before we can start with finite automata M_1 and M_2 recognizing the regular languages A_1 and A_2. But now, instead of constructing automaton M to accept its input if either M_1 or M_2 accepts, it must accept if it's input can be broken into two pieces where M_1 accepts first piece and M_2 accepts second piece. The problem is that M doesn't know where to break it's input.

* if NFA has n states and DFA has m states, then $m \leq 2^n$, depending on the number of unreachable states in DFA.

* Nondeterminism doesn't correspond to physical machine we can build. However, it is useful mathematically.

NON-DETERMINISTIC FINITE AUTOMATA

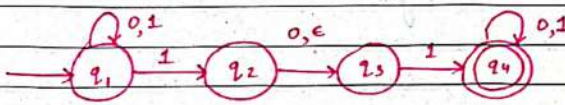
Date:

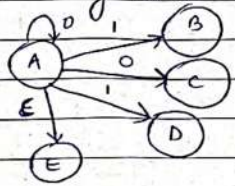
→ Nondeterminism is a useful concept.

→ In our automata every step of ~~comparison~~ computation follows in a unique way from preceding step.

→ Given a state and the next input symbol we know exactly what will be next state of machine

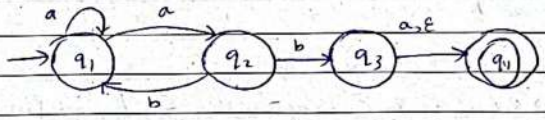
→ In nondeterministic machines several choices may exist for next state at any point.





(1) multiple paths possible (0, 1 or many at each step)
(2) ε-transitions is a "free" move without reading input
(3) Accept input if some path leads to (O) accept
(4) Next state may be chosen at random

Example Inputs
- ab (accept)
- aa (rejects)
- aba (accept)
- abb. (reject)



→ A nondeterministic ~~function~~ finite automaton (NFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$

→ all same as before except δ

→ $\delta: Q \times \Sigma_\varepsilon \rightarrow P(Q) = \{R \mid R \subseteq Q\}$

Σ_ε = $\Sigma \cup \{\varepsilon\}$

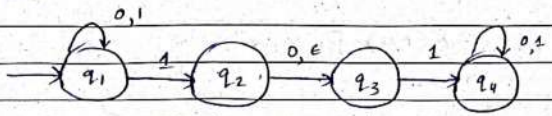
power (subset of Q)

2^Q

if two states then 4 possible states

Date:

EXAMPLE NFA



1) $Q = \{q_1, q_2, q_3, q_4\}$
2) $\Sigma = \{0, 1\}$
3) Transition function δ:

	0	1	ϵ
q_1	$\{q_1\}$	$\{q_1, q_2\}$	ϕ
q_2	$\{q_3\}$	ϕ	$\{q_3\}$
q_3	ϕ	$\{q_4\}$	ϕ
q_4	$\{q_4\}$	$\{q_4\}$	$\emptyset$

4) q_1 is the start state and
5) $F = \{q_4\}$

→ Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA
→ $w = y_1 y_2 \cdots y_m$ over same alphabet Σ_ϵ
→ $r_0, r_1, \ldots, r_m$ exists in Q such that:
  1) $r_0 = q_0$
  2) $r_{i+1} \in \delta(r_i, y_{i+1})$ for $i = 0, \ldots, m-1$ and.
  3) $r_m \in F$
→ Then we say N accepts w

?

Date:

DIFFERENCE B/W DFA & NFA

1. More than one transition from same symbol is possible.
2. Empty transitions are allowed in NFA.
3. Not all transitions need to be drawn.

THEOREM 1.39.

Every nondeterministic finite automaton has an equivalent deterministic finite automaton.

PROOF:

1. Assuming no ϵ transitions.

Let $N = (Q, \Sigma, \delta, q_0, F)$ be the NFA recognizing some language A.

We construct a DFA, $M = (Q', \Sigma, \delta', q_0', F')$ recognizing A.

- $Q' = P(Q)$
- For $R \in Q'$ and $a \in \Sigma$, let $\delta'(R,a) = \{q \in Q \mid q \in \delta(r,a)$ for some $r \in R\}$.

$$\delta'(R,a) = \bigcup_{r \in R} \delta(r,a)$$

- $q_0' = \{q_0\}$
- $F' = \{R \in Q' \mid R$ contains an accept state of $N\}$

Date:_____

COROLLARY 1.40

A language is regular if and only if some nondeterministic finite automaton recognizes it.

CONVERTING NFAs to DFAs.

THEOREM.

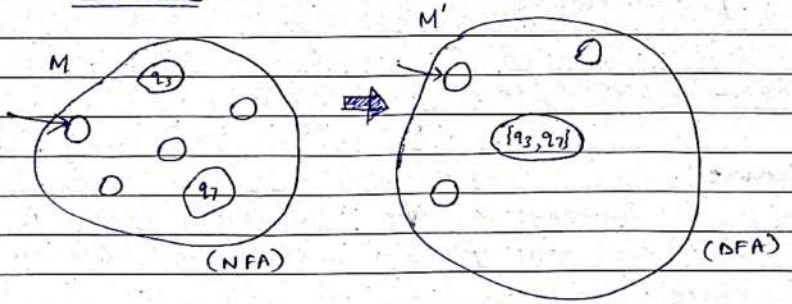
If an NFA recognizes A then A is regular.

PROOF:

Let NFA $M = (Q, \Sigma, \delta, q_0, F)$ recognize A. Construct DFA $M' = (Q', \Sigma, \delta', q_0', F')$ recognizing A.

(Ignore the ε-transitions, can easily modify to handle them)

IDEA: DFA M' keeps track of the subset of possible states in NFA M.



Q) If M has n states, how many states does M' have by this construction?

Date: _____

Construction of M':

$Q' = P(Q)$

$\delta'(R, a) = \{q \mid q \in \delta(r, a) \text{ for some } r \in R\}$

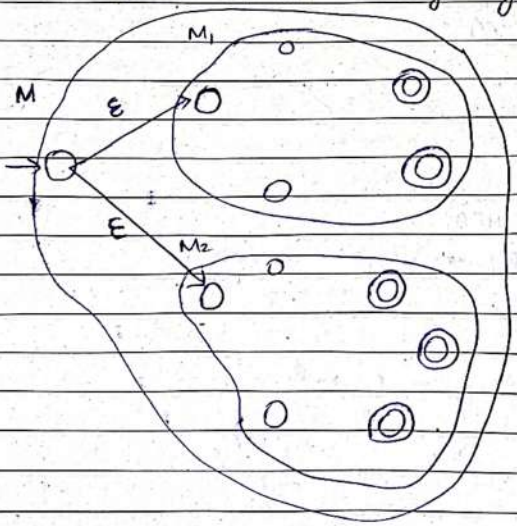
($R \in Q'$)

$q_0' = \{q_0\}$

$F' = \{R \in Q' \mid R \text{ intersects } F\}$

RECALL If A_1, A_2 are regular languages, so is $A_1 \cup A_2$ (The class of regular languages is closed under union)

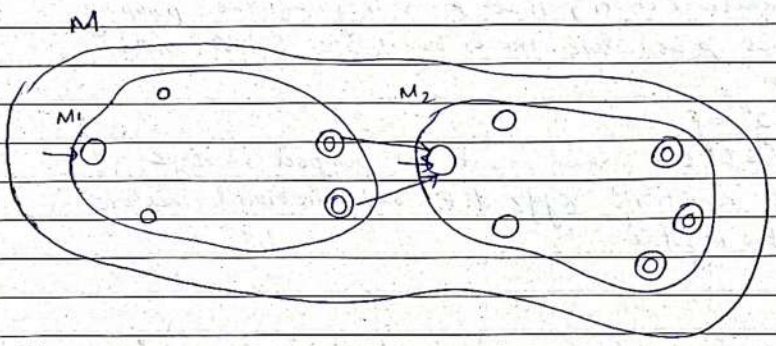
New Proof: Given DFAs M_1 and M_2 recognizing A_1 and A_2, construct NFA M recognizing $A_1 \cup A_2$



Date: _____

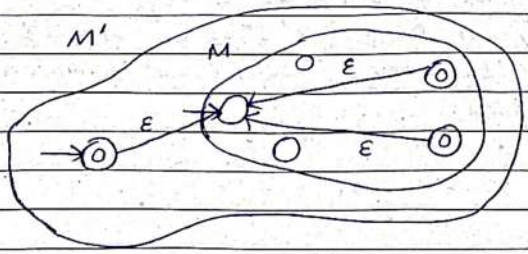
RECALL If A_1, A_2 are regular languages, so is $A_1 A_2$.

Proof Sketch: Given DFAs M_1 and M_2 recognizing A_1 and A_2.
Construct NFA M recognizing $A_1 A_2$



Theorem.
If A is regular language, so is A^*

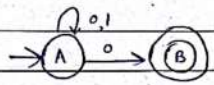
PROOF: Given DFA M recognizing A. Construct NFA M' recognizing A^*



Make sure M' accepts ε

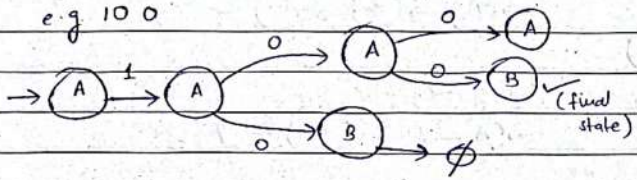
Date:

NFA Example

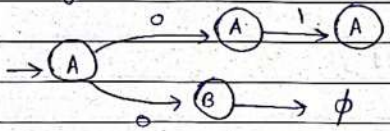


L = { set of all strings that end with 0 }.

e.g 100



e.g 01



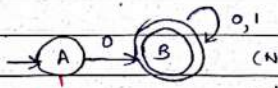
*not a final state.

X (not accepting)

* if there is any way to run the machine that ends in any set of states out of which atleast one state is final state, then the NFA accepts.

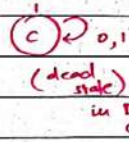
Example 2

L = { set of all strings that start with 0 }

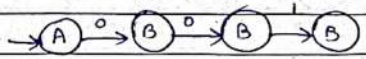


(NFA)

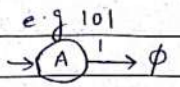
→ doesn't matter if we don't take input 1 for A.



e.g 001



e.g 101



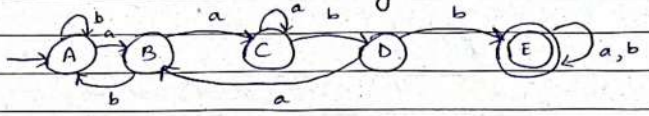
dead configuration.

Date:_____

DFA Example

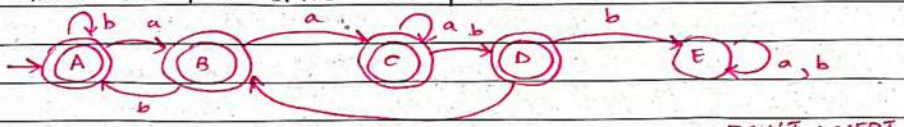
MUSTHILL

Construct a DFA that accepts any string over {a,b} that does not contain string aabb in it.



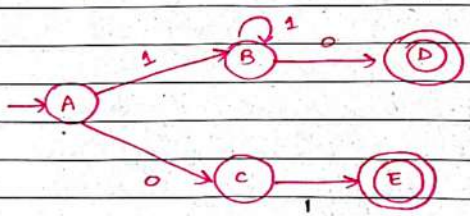
ACCEPTS "aabb"

→ flip states
→ make final states into non final state
→ make non final state into final states.



DOESN'T ACCEPT "aabb"

How to ~~recognize~~ figure out what DFA recognizes?



Accept States

10 ✓

111110 ✓ ∵ 11*0

A B D

01 ✓

L = {Accepts the string 01 or a string of atleast one '1' followed by a '0'}.

Date:_____

→ Every **DFA is an NFA** but not vice versa.

DFA $\delta = Q \times \Sigma \to Q$ NFA $\delta = Q \times \Sigma \to 2^Q$.

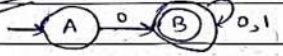
$Q \leq 2^0$

NFA ≅ DFA $m \leq 2^n$

L = {Set of all strings over (0,1) that starts with '0'}

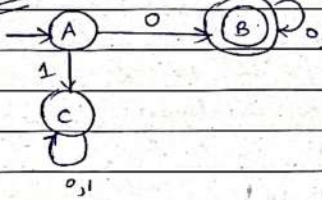
$\Sigma = \{0, 1\}$

NFA



	0	1
A	B	ϕ → dead configuration.
B	B	B

DFA



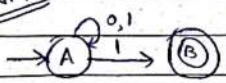
	0	1
A	B	C
B	B	B
C	C	C

C - dead state / trapped

L = {Set of all strings over (0,1) that ends with '1'}

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

NFA



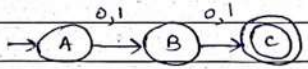
	0	1
A	{A}	{A,B}
B	ϕ	ϕ

Date:_____

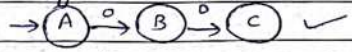
Q) Construct an NFA that accepts sets of all strings over {0,1} of length 2.

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

$L = \{00, 01, 10, 11\}$

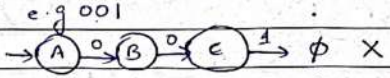


e.g 00



✓

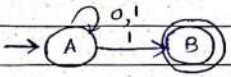
e.g 001



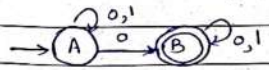
✗

Example 3

① L1 = {set of all strings that ends with '1'}

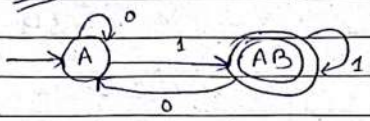


② L2 = {set of all strings that contain '0'}



Date: _____

DFA



	0	1
A	{A}	~~{A,B}~~ {AB}
AB	{A}	{AB}

AB - single state.

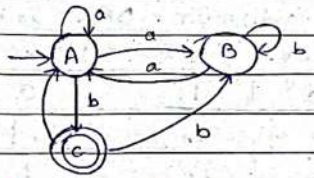
* we cannot reach B state

perform union of A and B (from NFA transition table)

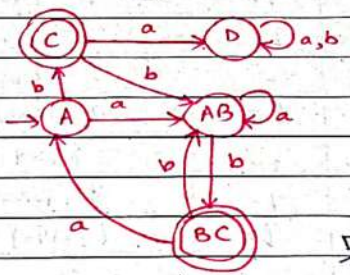
Q) Find the equivalent DFA for NFA given by $M = [\{A,B,C\}, (a,b), \delta]$ (Q, Σ) q_0 A, {C}, F where δ is given by:

NFA

	a	b
→ A	A,B	C
B	A	B
(C)	–	A, B



	a	b
→ A	AB	C
AB	AB	BC
(BC) final	A	AB
(C) final since it contains C	D	AB
D	D	D



DFA

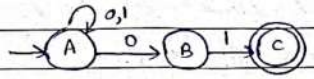
Date:

Example 4:

L = {Set of all strings over (0,1) that ends with '01'}

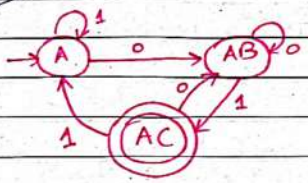
Construct equivalent DFA

NFA



	0	1
→ A	A,B	A
B	ϕ	C
(C)	ϕ	ϕ

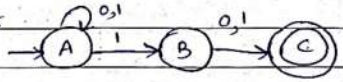
DFA



	0	1
→ A	AB	A
AB	AB	AC
(AC)	AB	A

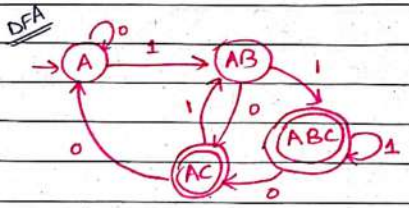
Q) Design an NFA for a language that accepts all strings over {0,1} in which the second last symbol is always '1'. Then convert it to it's equivalent DFA.

NFA



	0	1
→ A	{A}	{A,B}
B	{C}	{C}
(C)	ϕ	ϕ

DFA



	0	1
A	{A}	AB
AB	AC	ABC
(AC)	A	AB
(ABC)	AC	ABC

Date:

MINIMISATION OF DFA

Minimisation of DFA is required to obtain minimal version of any DFA which consists of minimum number of states possible.

DFA — 5 states — 4 states

00 000

equivalent

Two states 'A' and 'B' are said to be equivalent if

$\delta(A, X) \rightarrow F$ and $\delta(B, X) \rightarrow F$ OR $\delta(A, X) \not\rightarrow F$ and $\delta(B, X) \not\rightarrow F$ where 'X' is any input string.

→ If $|X| = 0$, then A and B are said to be ^{0}equivalent

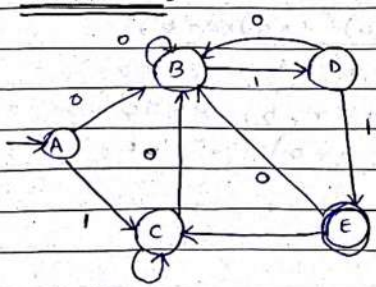
→ If $|X| = 1$, then A and B are said to be 1 equivalent

→ If $|X| = 2$, then A and B are said to be 2 equivalent

⋮

→ If $|X| = n$, then A and B are said to be n equivalent.

EXAMPLE 1:



	0	1
→ A	B	C ✓
B	B	D
C	B	C
D	B	E
(E)	B	C

Date: _____

0 Equivalence : {A, B, C, D} {E}

1 Equivalence : {A, B, C} {D} {E}

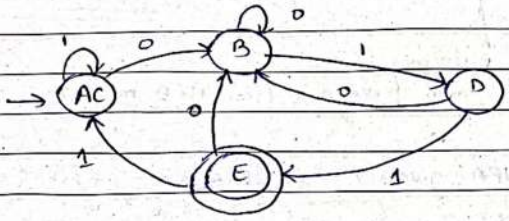
2 Equivalence : {A, C} {B} {D} {E}

3. Equivalence : {A, C} {B} {D} {D} {E}

A, B ✓ because C and B are in same set

A, C ✓

C, D ✗ because E is outside



(4 STATES)

Q) Castruct a minimum DFA equivalent to the DFA described by.

	0	1
→ q_0	q_1	q_5
q_1	q_6	q_2
(q_2)	q_0	q_2
q_3	q_2	q_6
q_4	q_7	q_5
q_5	q_2	q_6
q_6	q_6	q_4
q_7	q_6	q_2

0 Equivalence

{$q_0, q_1, q_3, q_4, q_5, q_6, q_7$} {$q_2$}

1 - equivalence.

{q_0, q_4, q_6} {q_1, q_7}

{q_3, q_5} {q_2}.

2 Equivalence.

{q_0, q_4} {q_6} {q_1, q_7} {q_3, q_5} {q_2}

3 Equivalence

{q_0, q_4} {q_6} {q_1, q_7} {q_3, q_5} {q_2}

Date:_____

REGULAR EXPRESSIONS

Regular Expression are used for representing certain sets of strings in an algebraic fashion.

$(a, b, c, \ldots \wedge, \phi)$ (1) Any terminal symbol ie symbols $\in \Sigma$ including $\wedge$ and ϕ are regular expressions.

R_1, R_2 (R_1+R_2) (2) The Union of two regular expressions is also regular expression

$R_1, R_2 \rightarrow (R_1 \cdot R_2)$ (3) The Concatenation of two regular expressions is also regular expression.

$R \rightarrow R^*$ (4) The iteration (or closure) of regular expression is also regular expression.

(5) The regular expression over Σ are precisely those obtained recursively by application of above rules once or several times.

IDENTITES OF REGULAR EXPRESSIONS

(1) $\phi + R = R$

(2) $\phi R + R\phi = \phi$

(3) $\epsilon R = R\epsilon = R$

(4) $\epsilon^* = \epsilon$ and $\phi^* = \epsilon$

(5) $R + R = R$

(6) $R^* R^* = R^*$

(7) $RR^* = R^*R$

(8) $(R^*)^* = R^*$

(9) $\epsilon + RR^* = \epsilon + R^*R = R^*$

(10) $(PQ)^* P = P(QP)^*$

(11) $(P+Q)^* = (P^*Q^*)^* = (P^* + Q^*)^*$

(12) $(P+Q)R = PR + QR$ and $R(P+Q) = RP + RQ$

Date:

Q) Prove that $(1+00^*1)+(1+00^*1)(0+10^*1)^*(0+10^*1)$ is equal to $0^*1(0+10^*1)^*$

LHS

$(1+00^*1)+(1+00^*1)(0+10^*1)^*(0+10^*1)$

$= (1+00^*1)[\epsilon + (0+10^*1)^*(0+10^*1)]$ $\because \epsilon + R^*R = R^*$

$= (1+00^*1)(0+10^*1)^*$

$= (\epsilon \cdot 1 + 00^*1)(0+10^*1)^*$ $\because \epsilon \cdot R = R$

$= (\epsilon + 00^*)1(0+10^*1)^*$

$= 0^*1(0+10^*1)^*$ = RHS

DESIGN REGULAR EXPRESSION

Q) Design regular expression for following languages over $\{a,b\}$

1. Language accepting strings of length exactly 2
2. Language accepting strings of length atleast 2
3. Language accepting strings of length atmost 2

1) $L_1 = \{aa, ab, ba, bb\}$

$R = aa + ab + ba + bb$

$= a(a+b) + b(a+b)$

$= (a+b)(a+b)$

2) $L_1 = \{aa, ab, ba, bb, aaa, \dots\}$

$R = (a+b)(a+b)(a+b)^*$

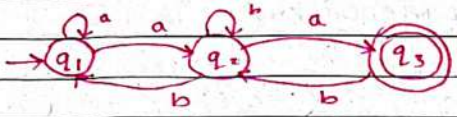
3) $L_1 = \{\epsilon, a, b, aa, ab, ba, bb\}$

$R = \epsilon + a + b + aa + ab + ba + bb$

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

Date:

Q) Find the Regular Expression for following NFA.



$q_3 = q_2 a \rightarrow (1)$

$q_2 = q_1 a + q_2 b + q_3 b \rightarrow (2)$

$q_1 = \epsilon + q_1 a + q_2 b \rightarrow (3)$

(1) $q_3 = q_2 a$

$= (q_1 a + q_2 b + q_3 b) a$

$= q_1 aa + q_2 ba + q_3 ba$

(2) $q_2 = q_1 a + q_2 b + q_3 b$ Putting value of q_3 from (1)

$= q_1 a + q_2 b + (q_2 a) b$

$= q_1 a + q_2 b + q_2 ab$

$q_2 = q_1 a + q_2 (b + ab)$

$q_2 = (q_1 a)(b + ab)^*$

(3) $q_1 = \epsilon + q_1 a + q_2 b$

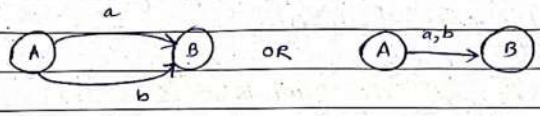
$q_1 = \epsilon + q_1 a + ((q_1 a)(b + ab)^*) b$

$\vdots$ Left TBD

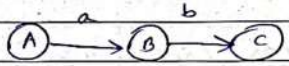
Date:_____

CONVERSION OF REGULAR EXPRESSION TO FINITE AUTOMATA.

(a+b)



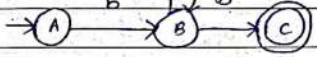
(a·b)



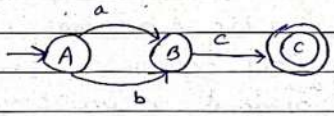
a*

(1) ba*b.

bb, bab, baab, ...

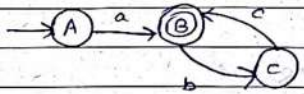


(2) (a+b)c.



(3) a(bc)*

a, abc, abcbc, abcbcbc, ...



Date:

PUMPING LEMMA

→ Pumping lemma is used to prove that a language is NOT REGULAR

→ it cannot be used to prove that language is Regular.

If A is a Regular language, then A has pumping length 'P' such that any string 'S' where $|S| \geq P$ may be divided into 3 parts $S = xyz$ such that following conditions must be true:

(1) $xy^iz \in A$ for every $i \geq 0$

(2) $|y| > 0$

(3) $|xy| \leq P$.

To prove that a language is not Regular using PUMPING LEMMA follow below steps:

(We prove using contradiction).

→ Assume that A is Regular

→ It has to have Pumping length (say P).

→ All strings longer than P can be pumped $|S| \geq P$

→ Now find a string 'S' in A such that $|S| \geq P$

→ Divide S into xyz.

→ Show that $xy^iz \notin A$ for some i.

→ Then consider all ways that S can be divided into xyz.

Date:

→ Show that none of these can satisfy all the 3 pumping conditions at same time.
→ S cannot be Pumped == CONTRADICTION.

EXAMPLE 1:

Use Pumping lemma prove that the language $A = \{a^n b^n \mid n \geq 0\}$ is NOT REGULAR.

Proof:

Assume that A is Regular

Pumping Length = P

$S = a^P b^P \Rightarrow$ aaaaaaabbbbbbb

x y z

P = 7

CASE 1: The y is in the 'a' part

aaaaaaabbbbbbb

x y z

CASE 2: The y is in the 'b' part

aaaaaaa bbbbbbb

x y z

CASE 3: The y is in the 'a' and 'b' part

aaaaaaabbbbbbb

x y z

CASE 1

$xy^i z \Rightarrow xy^2 z$

aa aaaa aaaa abbbbbbb

$11 \neq 7$ does not lies in lang.

CASE 2

$xy^i z \Rightarrow xy^2 z$

aaaaaaabb bbbb bbbb b

$7 \neq 11$ does not lie in lang.

CASE 3

$xy^i z \Rightarrow xy^2 z$ $a^n b^n$

aaaaa aabb aabb bbbbb

does not follow $a^n b^n$ so does not lie in language.

CASE 1 $|XY| \leq P \rightarrow |6| \leq 7$

CASE 2 $13 \leq 7$ X

Date: _____

1) $xy^iz \in A$ for all $i \geq 0$ $\quad y^i = \underbrace{yy\cdots y}_{i}$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

$x = 0^a, y = 0^b, z = 0^{p-a-b}1^p$

$i = 2 \rightarrow s' = xy^2z$

$0^a 0^{2b} 0^{p-a-b} 1^p$

$0^{a+2b+(p-a-b)} 1^p$

$0^{p+b} 1^p$ ∴ has excess 0s.

EXAMPLE 1:

Let $D = \{0^k 1^k \mid k \geq 0\}$

Show: D is not regular.

Proof By contradiction:

Assume (to get contradiction) that D is regular. The pumping lemma gives p as above. Let $s = 0^p 1^p \in D$. Pumping lemma says that can divide $s = xyz$ satisfying the 3 conditions. But $xyyz$ has excess 0s and thus $xyyz \notin D$ contradicting the pumping lemma. Therefore our assumption (D is regular) is false. We conclude that D is not regular.

Date:_____

EXAMPLE 2:

Let $F = \{ww \mid w \in \Sigma^*\}$. Say $\Sigma^* = \{0, 1\}$.

Show: F is not regular.

Proof By Contradiction:

Assume (for contradiction) that F is regular. The pumping lemma gives p as above. Need to choose $s \in F$. Which s?

Try $s = 0^p 0^p \in F$

Try $s = 0^p 1 0^p \in F$. Show cannot be pumped $s = xyz$ satisfying 3 conditions $xyyz \notin F$ contradiction! Therefore F is not regular.

EXAMPLE 3:

Let $B = \{w \mid w$ has equal numbers of 0s and 1s$\}$

Show: B is not regular.

Proof By Contradiction:

Assume (for contradiction) that B is regular.

We know that 0^*1^* is regular so $B \cap 0^*1^*$ is regular (closure under intersection). But $D = B \cap 0^*1^*$ & we already showed D is not regular. contradiction! Therefore our assumption is false, so B is not regular.

CHAPTER 2

Date:

~~Regular~~ Grammar

A Grammar "G" can be formally described using 4 tuples as $G = (V, T, S, P)$ where

- V = set of variables or non-terminal symbols
- T = set of Terminal Symbols.
- S = Start Symbol
- P = Production rules for Terminals and Non-Terminals.

A production rules has form $\alpha \rightarrow \beta$ where α and β are strings on $V \cup T$ and atleast one symbol of α belongs to V.

Example: $G = (\{S, A, B\}, \{a, b\}, S, \{S \rightarrow AB, A \rightarrow a, B \rightarrow b\})$

$V = \{S, A, B\}$

$T = \{a, b\}$.

$S = S$

$P = S \rightarrow AB, A \rightarrow a, B \rightarrow b$

Eg. $S \rightarrow AB$

$\rightarrow aB$

$\rightarrow ab$

Date:

EXAMPLE 2:

Using Pumping Lemma prove that the language $A = \{yy \mid y \in \{0,1\}^*\}$ is Not Regular.

Proof:

Assume that A is Regular

then it must have Pumping Lugth = P

$S = 0^P 1 0^P 1$

$P = 7$

x y z

$xy^iz \Rightarrow xy^2z$

00 0000 0000 01 00000001

$\notin A$

0000000100000001

x y z

$|xy| \leq P$

$6 \leq 7$ ✓

A is NOT REGULAR.

Date:_____

Regular Grammar

Regular Grammar can be divided into two types:

Right Linear Grammar

A grammar is said to be Right Linear if all productions are of the form

$A \to xB$

$A \to x$

where $A, B \in V$ and $x \in T$

Left Line Grammar

A grammar is said to be Left Linear if all productions are of form

$A \to Bx$

$A \to x$

where $A, B \in V$ and $x \in T$

Eg. $S \to abS \mid b$ → Right linear grammar.

$S \to Sbb \mid b$ → Left linear grammar

Derivations from a Grammar

The set of all strings that can be derived from a Grammar is said to be LANGUAGE generated from that Grammar.

Example 1 Consider the Grammar $G1 = (\{S, A\}, \{a, b\}, S, \{S \to aAb, aA \to aaAb, A \to \varepsilon\})$

$S \to aAb$ (by $S \to aAb$)

$\to aaAbb$ (by $aA \to aaAb$)

$\to aaaAbbb$ (by $aA \to aaAb$)

$\to aaabbb$ (by $A \to \varepsilon$)

Date:

Example 2 $G2 = (\{S, A, B\}, \{a, b\}, S, \{S \to AB, A \to a, B \to b\})$

$S \to AB$

$\to ab$

$L(G2) = \{ab\}$

CONTEXT FREE LANGUAGE

→ In formal language theory, a context free language is a language generated by some context free Grammar.

The set of all CFL is identical to set of languages accepted by Pushdown Automata.

Context free Grammar is defined by 4 tuples as

$G = \{V, \Sigma, S, P\}$ where

V = Set of variables or Non-Terminal Symbols.

Σ = Set of Terminal Symbols

S = Start Symbol.

P = Production Rule.

Context free Grammar has Production Rule of form

$A \to a$.

where, $a = \{V \cup \Sigma\}^*$ and $A \in V$.

Date:

Example:

For generating a language that generates equal number of a's and b's in the form $a^n b^n$, the CFG will be defined as.

$$G = \{(S, A), (a, b), (S \to aAb, A \to aAb \mid \epsilon)\}$$

$S \to aAb$

$\to aaAbb$ (by $A \to aAb$)

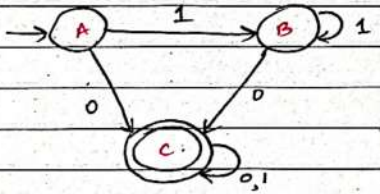
$\to aaaAbbb$ (")

$\to aaabbb$ (by $A \to \epsilon$)

$\to a^3 b^3 \Rightarrow a^n b^n$

REGULAR LANGUAGES AND FINITE AUTOMATA

Q) Consider the DFA A given below:



$C = A0 + B0 + C0 + C1$ — (I)

$B = A1 + B1$ — (II)

$A = \epsilon$ — (III)

Q) Which of the following are FALSE?

✓ (1) Complement of L(A) is context free

✓ (2) $L(A) = L((11^*0 + 0)(0+1)^*0^*1^*$

✗ (3) For language accepted by A, A is minimal DFA.

✗ (4) A accepts all strings over {0, 1} of length at least 2.

$B = \epsilon 1 + B1$

$B = 1 + B1$

R Q R P = QP*

$B = 11^*$

$C = 0 + 11^*0 + C(0+1)$

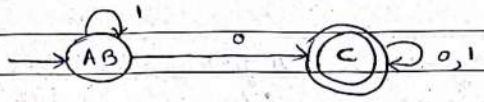
R Q R P

$C = 0 + 11^*0\,(0+1^*)^*$

Date:_____

(3) 0 Equivalence: {A, B} {C}

1 Equivalence: {A, B} {C}



Q) Which of the following languages over alphaset {0,1} is described by the regular expression: (0+1)* 0 (0+1)* 0 (0+1)* ?

0 OR 1

✗ (A) The set of all strings containing the substring 00.

✗ (B) The set of all strings containing at most two 0's

✓ (C) The set of all strings containing at least two 0's

✗ (D) The set of all strings that begin & end with either 0 or 1.

Date:_____

METHOD TO FIND WHETHER A STRING BELONGS TO GRAMMAR OR NOT

① Start with start symbol & choose the closest production that matches to the given string.

② Replace the variables with its most appropriate production Repeat the process until the string is generated or until no other productions are left.

Example: Verify whether the Grammar $S \to 0B \mid 1A$, $A \to 0 \mid 0S \mid 1AA$, $B \to 1 \mid 1S \mid 0BB$ generates string 00110101

$S \to 0B$ ($S \to 0B$)
$\to 00BB$ ($B \to 0BB$)
$\to 001B$ ($B \to 1$)
$\to 0011S$ ($B \to 1S$)
$\to 00110B$ ($S \to 0B$)
$\to 001101S$ ($B \to 1S$)
$\to 0011010B$ ($S \to 0B$)
$\to 00110101$ ($B \to 1$)

Example: Verify whether the Grammar $S \to aAb$, $A \to aAb \mid \wedge$ generates the string aabbb.

$S \to aAb$
$\to aaAb$ ($A \to aAb$)
$\to$