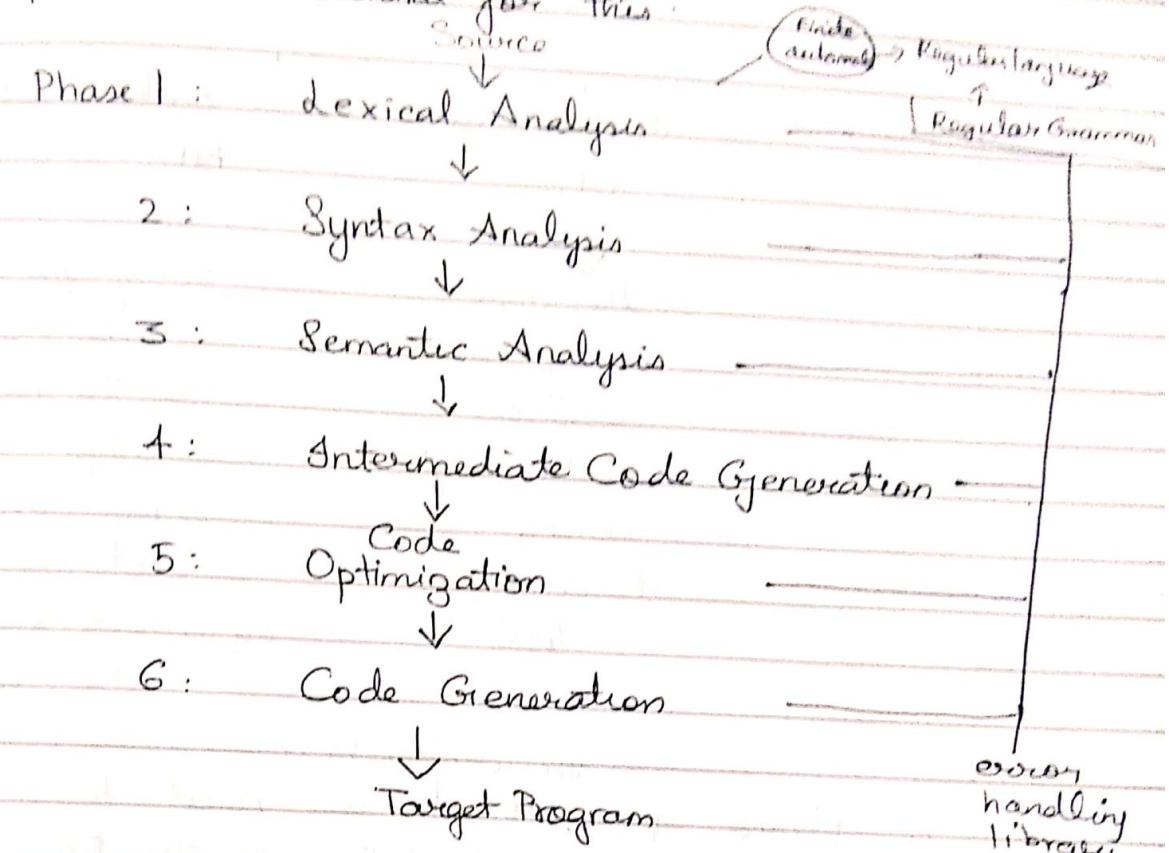
# Overview

Page No. Date: 7/6/19 YOUVA

Compilers perform 6 steps during processing.

A compiler is responsible for converting source code into object code.
6 phases are essential for this.




Phase 1: Reads character by character from A to R till it finds / identifies a delimiter.
This grouping between delimiters is a token.
The analyser should also verify if token is valid.

eg. Identifier in C: Consists of letters & numbers, not a keyword, starts with letter or underscore.

The lexical analyser designs a model for this verification.
This model is called finite automaton (plural of

Finite automata is specified by regular language
The rules for such regular language are specified by regular grammar (grammar of language)

Finite Automata

- Deterministic DFA → the single config can lead to another single config. (determinable output)
- Non-Deterministic NFA → set of possible moves / steps to the next config => not possible to determine exactly. eg. Move Queen in chess game

To optimize space, stored in which the FA is, we need to minimize the no. of states of the FA

=> DFA minimized (should minimize DFA)

NFA <-> DFA conversion is possible.

The token, if valid, will be emitted to syntax analyser.

The syntax analyser of C will group tokens till it finds;
It then verifies if the statement is syntactically correct.
It also verifies syntax of loops, branch, etc.

Uses model called Push Down Automata
based on Context Free language.
designed using → Rules are based on Context Free (CFG)

Page No.
Date
YOUVA

PushDown Automata can be implemented on a digital system using either:

i) Chomsky Normal Form or
ii) Greibach Normal Form

} CFG is converted to any of these forms to implement on system.

CYK algo is used to find if a given source can be CFG or not.

III Semantic analyser checks if the meaning of statement is valid.
Model used here: Linear Bounded Automata
Based on: Context-Sensitive Language
↓
defined by Context-Sensitive Grammar (CSG)

Also, the concepts of Turing Machine & Recursively Enumerable language will be covered.
i.e. how Turing Machine is modelled
(non-deterministic problems)

ASSIGNMENT:
Find a unique / distinctive application of ToC

TEXTBOOK
PETER-LINZ Introduction to Formal Languages & Automata
4th Edition (Green & Black)

Page No.
Date: 11/6/19 (Tuesday)

Set : A collection of elements
(no structure, no relationship)

Two forms:
1) $S = \{ -, -, - \}$
2) $S = \{ x : \text{property of } x \text{ in the set} \}$

Set operations:
1) Union
2) Intersection
3) Set difference
4) Complement
5) Size of the set

For sets $S_1$ & $S_2$,
$S_1 \cup S_2 = \{ x : x \in S_1 \text{ or } x \in S_2 \}$
$S_1 \cap S_2 = \{ x : x \in S_1 \text{ and } x \in S_2 \}$
$S_1 - S_2 = \{ x : x \in S_1 \text{ and } x \notin S_2 \}$
$\overline{S} = \{ x : x \in U \text{ and } x \notin S \}$
$|S| \Rightarrow$ no. of elements in the set

Universal set = set of all elements in that domain

Types of Set:

1) Singleton set - only 1 element
2) Empty set - no elements; $\{\}$ or $\phi$
3) Universal set - all elements in that domain
4) Subset - $S_1 \subseteq S \Rightarrow$ part of a set
5) Proper subset - $S_1 \subset S \Rightarrow$ same set S cannot be $S_1$
6) Power set - Set of all subsets of the set. $S \Rightarrow$ size is $2^n$ where $n = |S|$
7) Cartesian product - set-con

$S = S_1 \times S_2 = \{ (x, y) : x \in S_1 \text{ and } y \in S_2 \}$
contains ordered pairs

Power set is represented as $2^S$

Page No.:
Date:
YOUVA

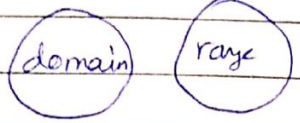
## Function:

Def: A rule that maps an element in one set (domain) to a unique element in another set (range)

(or)

$f: x \rightarrow y$

$x \in$ domain, $y \in$ range




eg. $A = \{1, 2, 3\}$ domain
$B = \{2, 4, 6\}$ range

$f = 2^x$ When $x = 1, y = 2$ } exist in range
$x = 2, y = 4$
$x = 3, y = 8$

Partial function: only part of domain is mapped to range

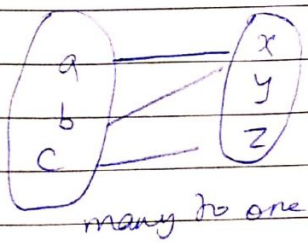
Total function: Every element in domain A has a unique corresponding mapping to set B (range)

eg. $A: \{1, 2, 3\}$
$B: \{3, 5, 7, 8\}$

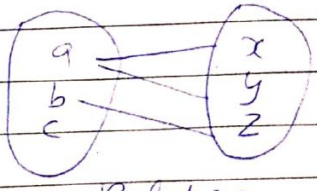
$f = 2x + 1$ $x = 1$ $y = 3$
$x = 2$ $y = 5$
$x = 3$ $y = 7$

Every element in domain has a mapping in range so it is a total function

Many-to-one is a function (but) one-to many is not. It is a relation.




many to one




Relation

Properties of relation

1) Reflexive
2) Symmetric
3) Transitive
4) Antisymmetric

1) Let R be a relation on a set A
then if $(a,a) \in R \quad \forall a \in A$,
R is reflexive

2) If $(a,b) \in R \quad \forall a,b \in A$
then $(b,a) \in R$
R is symmetric

3) If $(a,b) \in R$, $(b,c) \in R$ then $(a,c) \in R$

4) If $(a,b) \in R$ but $(b,a) \notin R$
R is antisymmetric

If a set satisfies R, A, T => poset / partial ordering relation
R, S, T => equivalence relation

I) eg. $A = \{2,3,4\}$
$R = \leq$ in $N$

Refl: $2 \leq 2$ ✓
$3 \leq 3$ ✓ => Reflexive
$4 \leq 4$ ✓

i.e. For all a in A,
$a \leq a$
=> Reflexive

Sym: $2 \leq 3$ but $3 \not\leq$ is not $\leq 2$
=> $(2,3) \in R$ but $(3,2) \notin R$
=> Not symmetric

=> antisymmetric

I. $2 \leq 3$ & $3 \leq 4 \Rightarrow 2 \leq 4$

i.e. For $(a, b)$ where $a \leq b$ and $(b, c)$ where $b \leq c$,

$\Rightarrow a \leq b$ and $b \leq c$

$\Rightarrow a \leq c$

$\Rightarrow (a, c) \in R$

Transitive

Poset

II) $x = \{1, 2 \ldots 7\}$

$R = x - y$ divided by 3

$R = \{(1,4), (1,7), (2,5), (3,6), (4,1), (7,1), (5,2), (6,3), (1,1), (4,7), (7,4), (7,7), (2,2), (3,3), (4,4), (5,5), (6,6)\}$

Reflexive: let $a \in A$

$a - a = 0$ which is divisible by 3

$\Rightarrow R$ is reflexive

Symmetric: If $a - b = 3m$, then $-(b - a) = 3m$

Transitive: If $x - y = 3m$ and $y - z = 3n$

$x - y + y - z = 3m + 3n$

$x - z = 3(m + n)$

Transitive

$\Rightarrow$ Equivalent

Page No.:
Date:

Languages are defined as a set.
Grammar is a quadruple (4-tuples)

Languages ——> $\Sigma$ represents alphabet set
eg. $\Sigma$ {a, b} if alphabet set only has a, b

→ Note: Book used specifies all lowercase letters to be used in alphabet set except u, v, w because may be used to name strings. (represent name of string)

Combination of elements in alphabet set is called a string.

eg. u = ab
v = aba
w = aabb

Concatenation uv = ababa

Substring is a continuous sequence of elements within the string.
Prefix, proper prefix & Suffix, proper suffix are correspondent to subset & proper subset.

Empty string represented by $\lambda$ or $\epsilon$ (epsilon)
Empty language represented by $\phi$

Kleene (Additive)
$\Sigma^+$ → ~~Kleene~~-closure = { a, b, ab, ba, aa, ... } (any combination of alphabets)
$\Sigma^*$ → Kleene star closure = { $\lambda$, $\Sigma^+$ }

Subset of $\Sigma^*$ is language (any subset of star closure)
↳ subset of strings from an alphabet

Eg → no of strings in language is defined

Page No.:
Date:

Finite $L = \{ab, ba, aba\}$

Infinite $L = \{a^n b^n, n \geq 0\}$

languages can be
1) Finite
2) Infinite

↳ rules; but n ranges from 0 to ∞

You can have many languages associated w/ same alphabet set.

12/6/19 Let $L_1$ & $L_2$ be 2 finite languages

$L_1 = \{ab, abb, aba\}$ $L_2 = \{ba, bbb, bab\}$

$L_1 \cup L_2 = \{ab, abb, aba, ba, bbb, bab\}$

$L_1 \cap L_2 = \phi$

Reverse of a string $\omega = \omega^R$

eg. $\omega = abaa$

$\omega^R = aaba$

⇒ Reverse of a language $L^R = \{\omega^R, \forall \omega \in L\}$

Complement of language: $\bar{L}$

For min language, $\bar{L} \cup L = \Sigma^*$

$\Rightarrow \bar{L} = \Sigma^* - L$

Concatenation of ~~Reg~~ languages

$L_1 L_2$:

$\{abba, abbbb, abbab, abbba, abbbbb, abbbab, ababa, ababbb, ababab\}$

$L_1^2 = \{abab, abbabb, abaaba, ababb, ababa, abbab, abbaba, abaab, ababbb\}$

ab, aabb, aaabbb
abaabb

Page No.:
Date:

$L = \{ a^n b^n : n \geq 0 \}$

Find $L^2, L^R, \bar{L}$

$\Sigma = \{a, b\}$ Expanding this language,

$L = \{ \lambda, ab, aabb, aaabbb, \ldots \}$

$L^2 = \{ ab, abab, abaabb, abaaabbb, \ldots \}$

$= \{ a^n b^n a^m b^m : n \geq 0, m \geq 0 \}$

$L^R = \{ b^n a^n : n \geq 0 \}$

$\bar{L} = \Sigma^* - \{ a^n b^n : n \geq 0 \}$

i.e. It includes all patterns of a & b that do not follow the above pattern.

GRAMMAR:

<sentence> → <nounphrase> <predicate>

<nounphrase> → <article> <noun>

<predicate> → <verb>

Replacement happens in above fashion.

Grammar is the most powerful notation for a language to validate whether the language is correct or not

Quadruple:

a) grammar

$G = (V, T, S, P)$

- V: non-empty finite set of variables
- T: non-empty finite set of terminal symbols
- S: starting symbol of grammar or production form
- P: production set

All 4 tuples are non-empty.

V & T are disjoint sets

Production is of the form

$x \rightarrow y$

$x \rightarrow (V \cup T)^+$ i.e. at least 1 element from either V or T should be there

$y \rightarrow (V \cup T)^*$ may have 0 or more occurrences from V or T

Derivation

eg. $U \Rightarrow xyz$

Production rule: $y \rightarrow a$

Applying rule to U,

$V \Rightarrow xaz$

A language can have many kinds of grammar

Language can be derived from grammar (generated from grammar)

Production gives rules to be able to transform a string from one form to another

Example:

(1) $G = (\{S\}, \{a, b\}, S, P)$

P is: $S \rightarrow aSb \mid \lambda$

$\Rightarrow S \rightarrow aSb$

$S \rightarrow \lambda$

Strings:

1) $S \overset{2}{\Rightarrow} \lambda$ → applying rule 2

2) $S \overset{1}{\Rightarrow} aSb$ → applying rule 1

$\Rightarrow ab$

Sb  Aab  Abab  Sb
Sbb
S -> Aa  Aabb

3) $S \Rightarrow aSb$
$\Rightarrow aaSbb$
$\Rightarrow aabb$

$L = \{\lambda, ab, aabb \ldots\}$

$\Rightarrow L = \{a^n b^n : n \geq 0\}$

Variables are caps & terminal symbols are lowercase except U, V, W

3rd tuple is a special symbol part of variable set

Derivation starts with starting symbol, here S. We apply the production rules as necessary on that variable.

If you cannot find a pattern that can be described mathematically, write descriptively.

Another grammar: $S \to aA \mid \lambda$
$A \to Sb$

Qn 2:

$L = \{a^n b^{n+1} : n \geq 0\}$

b
abb
aabbb
aaabbbb

$S \to Sb$

Qn 3: Find grammar w/ exactly one a

Qn 4: " w/ atleast one a

5: " exactly three a's

6: " at least 3 a's

2) $L = \{ a^n b^{n+1} : n \geq 0 \}$

Pattern: b
abb
aabbb

$S \rightarrow aSb \mid b$

$S \rightarrow aSb$
$S \rightarrow b$

$G = \{ V, T, S, P \}$

$V = \{ S, A \}$
$T = \{ a, b \}$
$S = \{ S \}$ ~~$b \rightarrow abb$~~
$P = \{ S \rightarrow Ab, A \rightarrow aAb \mid \lambda \}$

$S \rightarrow Ab \mid$
$A \rightarrow aAb \mid \lambda$

abb
aabbb
aaabbbb

(or)

$P = \{ S \rightarrow aSb \mid b \}$

(or)

$P = \{ S \rightarrow Ab, A \rightarrow aB \mid \lambda, B \rightarrow Ab \}$

a(aB)b
a(Ab)bb

Ab
(aB)b
a(Ab)b
a(aB)bb
aaAbbb

3) Exactly one a
$L = \{ a, ab, ba, abb, bab, bba, \ldots \}$

$G = \{ V, T, S, P \}$

$V = \{ S, A \}$
$T = \{ a, b \}$
$S = \{ S \}$
$P = \{ S \rightarrow AaA, A \rightarrow bA \mid \lambda \}$

bAa

(or)

$P = \{ S \rightarrow AaB, A \rightarrow bA \mid \lambda, B \rightarrow bB \mid \lambda \}$

(or) $S \rightarrow Sb \mid Aa$
$A \rightarrow Ab \mid \lambda$

aAb
aAbbb

4) $P = \{ S \to AaA \ , \ A \to bA / \lambda / aA \}$

(or)

$R = \{ S \to AaA, \ A \to aAb / bBA / \lambda \}$

$P = \{$

$S \to aA / Sb$

$A \to Aa / \lambda$

bAa

$P = \{ S \to AaB$

$A \to aA / bA / \lambda$

$B \to aA / bA / \lambda$

$\}$

no pos... in ...

5) $L = \{ aaa, abaa, aaba, aaab, baaa \ldots \}$

$P = \{ S \to AaAaAaA, \ A \to bA / \lambda \}$

bA

6) $P = \{ S \to AaAaAaA, \ A \to bA / aA / \lambda \}$

More questions:

1) $L_1 = \{ a^n b^m : n \geq 0, m > n \}$
2) $L_2 = \{ a^n b^{2n} : n \geq 0 \}$
3) $L_1 \cup L_2$
4) $L_1 L_2$
5) $L_3 = \{ a^n b^{n-3} : n \geq 3 \}$
6) $L_4 = \{ a^{n+2} b^n : n \geq 1 \}$

Solutions:

① $L_1 = \{ b, bb, bbb, abb, abbb, aabbb, \ldots \}$

$P = \{ S \to AbB$

$A \to aAb / \lambda$

$B \to bB / \lambda$

$\}$

aAbb

bB

aAbb B

aaAbbbb

2) $L_2 = \{ \lambda, abb, aabbbb, aaabbbbbb \ldots \}$

$S \rightarrow aSbb / \lambda$ $\quad a(aSbb) \Rightarrow abb$

$\Rightarrow aa(aSbb)bb \Rightarrow aabbbb$

3) $L_1 \cup L_2 = \{ \lambda, b, abb, abbb, aabbb, aabbbb \ldots$
$bb, bbb, \} \ldots$

$P = \{ S \rightarrow AbB / \lambda$
$A \rightarrow aAb / \lambda$
$B \rightarrow bB / \lambda$
$\}$

4) $L_1 L_2 = \{ b, babb, baabbbb, bbbabb, \ldots$
$abb, abbabb, abbaabbbb, abbaaabbbb,$
$aabbbabb \ldots \}$

5) $L_3 = \{ a^n b^{n-3} : n \geq 3 \}$

$L_3 = \{ aaa, aaaab, aaaaabb \ldots \}$

$S \rightarrow aaaA$ $\quad aaa(aAb) \Rightarrow$
$A \rightarrow aAb / \lambda$

6) $L_4 = \{ a^{n+2} b^n : n \geq 1 \}$ $\quad aaa\lambda$ $\quad aaaaAb \Rightarrow aaaaaAbb$

(or) $L_4 = \{ aaab, aaaabb, aaaaabbb \ldots \}$

$S \rightarrow aaaAb$ $\quad aaaaAbb$
$A \rightarrow aAb / \lambda$

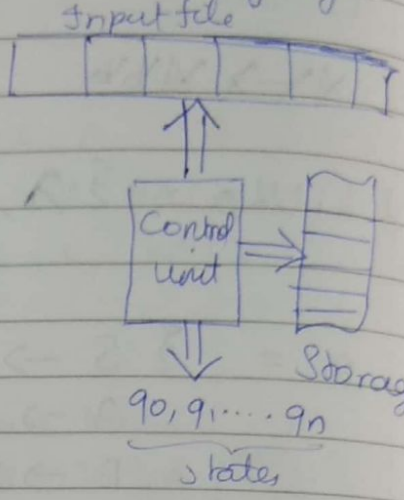
18/6/19

# Automata

It is a model to represent a language

Consists of:

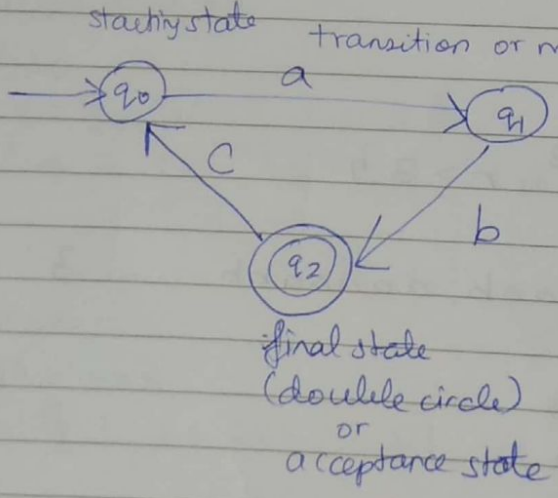
1. Control unit
   Specifies set of internal states
2. Input file / symbol
3. Storage symbol




Pushdown automata has a storage medium
Some automata do not have one.

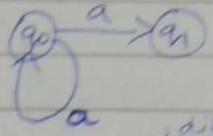
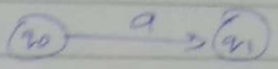
Control unit can read & change content of storage medium.

An automaton is represented as a graph w/ nodes representing internal states.




Automata based on action

| NFA | DFA |
|---|---|
| Non deterministic<br>• from each config, there are multiple possibilities<br>So impossible to determine the next config. | Deterministic<br>• each configuration uniquely determines the next configuration |

| | |
|---|---|
| • for each input symbol, there are different transitions (multiple possibilities) or transitions<br>eg.  | • each input symbol has a unique transition<br>  |
| • not so easy; difficult needs trial & error (String recognition is difficult) | • easier to recognize string |

Automata based on output response

| Accepter | Transducer | Grammar for an identifier |
|---|---|---|
| • Binary output (yes/no) | • Specific output | refer below for sample grammar |

Identifier:

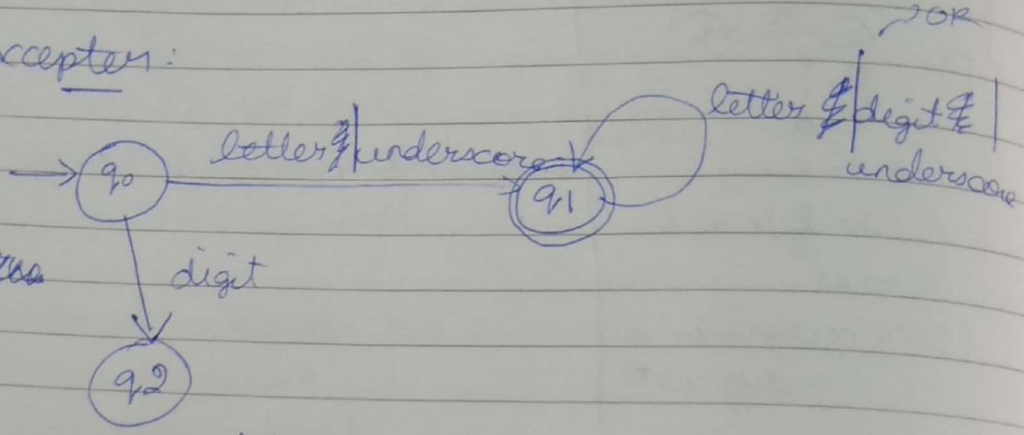
<identifier> → <letter><rest> | <underscore><rest>
<rest> → <letter><rest> | <digit><rest> | <underscore><rest> | λ.

<letter> → a | b | ... z | A | B | ... | Z
<digit> → 0 | 1 | ... 9
<underscore> → _

Note:

State names can be either 0, 1, 2, 3...
or $q_0, q_1, q_2 \ldots$
Then write $Q = \{q_1, q_2, \ldots\}$ as required

Acceptor:

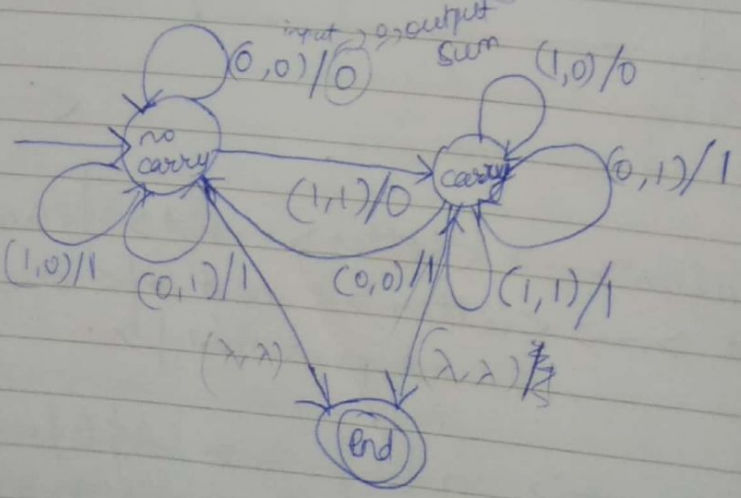



non-final state
(you cannot reach the final/acceptance state through this path)

Transducer:

eg. Binary adder

| | 0 | 1 |
|---|---|---|
| 0 | S = 0<br>C = 0 | S = 1<br>C = 0 |
| 1 | S = 1<br>C = 0 | S = 0<br>C = 1 |

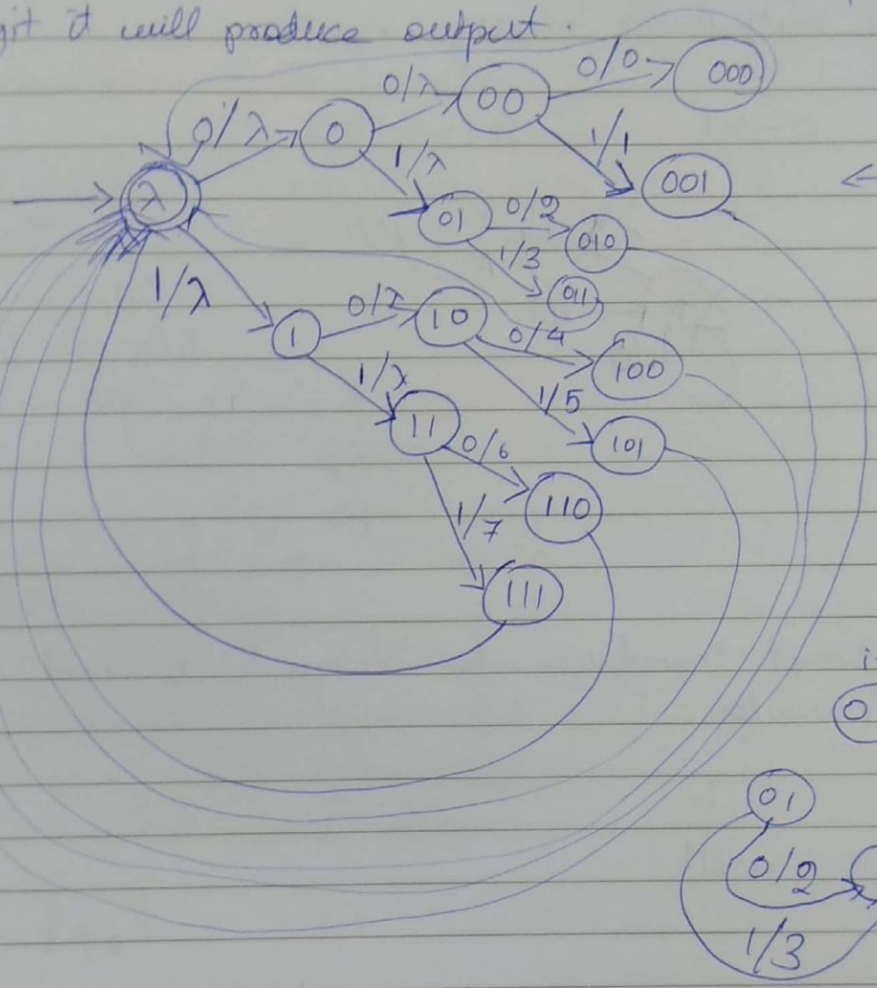


Construct an automaton to convert binary to octal

Type: Transducer

eg. 001 010 110 : 126

Till 3rd digit it should produce λ, then after 3rd digit it will produce output.




← slightly incorrect. Stop branch after 2 digits & then loop up the inputs.

i.e. after




Question:

0110 → λ0110

(or)

abba → λabba

(produces same o/p as input one time signal later)

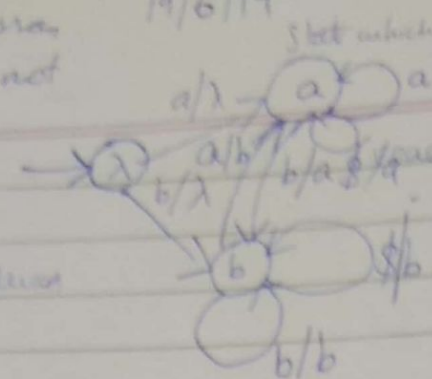
when reading 1st symbol, o/p = λ

and symbol, o/p = 1st symbol & so on

Final state necessary for acceptor, not transducer

Acceptor shows yes/no response only after finite state

Transducer

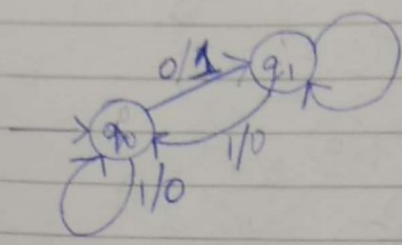



Note: $ Special ch... appended at ... of ... indicate ...

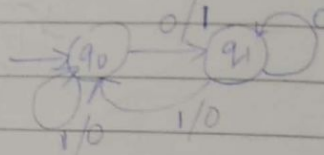
- Design transducer to produce 1's complement as output for binary input

0 → 1  011 → 100

HW




$\delta(q_0,0) = 1$
$\delta(q_0,1) = 0$
$\delta(q_1,0) = 1$
$\delta(q_1,1) = 0$




- Attempt transducer for binary to decimal Converter.

$2^4 2^3 2^2 2^1 2^0$
1 1 0 0 1

HW

16 + 8 + 1

## Deterministic Finite Automata

Finite automata have no storage file/compared They only have Control Unit & Input File.

DFA represented by quintuple
M represents any automata

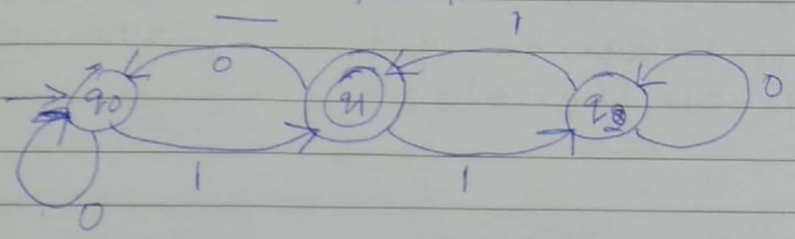
$M = (Q, \Sigma, \delta, q_0, F)$

Q = non-empty finite set of internal states
$\Sigma$ = set of alphabets
$\delta$ = transition function (it is a total function)
$q_0$ = initial state
F = set of final states

↓ every domain element has unique mapping in range

$\delta : Q \times \Sigma \rightarrow Q$

Deterministic Finite Acceptor:




$Q = \{q_0, q_1, q_2\}$
$\Sigma = \{0, 1\}$

$\delta$:
$\delta(q_0, 0) \Rightarrow q_0$
$\delta(q_0, 1) \rightarrow q_1$
$\delta(q_1, 0) \rightarrow q_0$
$\delta(q_1, 1) \rightarrow q_2$
$\delta(q_2, 0) \rightarrow q_2$
$\delta(q_2, 1) \Rightarrow q_1$

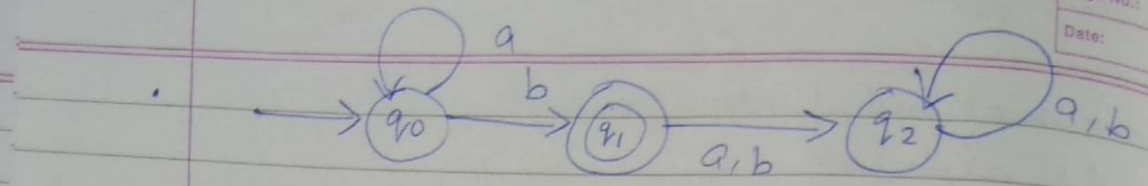
Transition function is a total function. So it is a DFA

$F = \{q_0\}$

We need to find what strings lead to finale state, or determine the language accepted by the automaton

We observe that strings ending w/ 0 are not accepted
&
ending w/ even no. of 1s not accepted.

Accepted: 101, 0111, 11001
Not accepted: 1100, 100, 1111




$Q = \{q_0, q_1, q_2\}$

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

$\delta$:

$\delta(q_0, a) = q_0$

$\delta(q_0, b) = q_1$

$\delta(q_1, a) = \delta(q_1, b) = q_2$

$\delta(q_2, a) = \delta(q_2, b) = q_2$

Total onto function
$\Rightarrow$ DFA

eg

$a^n b \; ; n \geq 0$

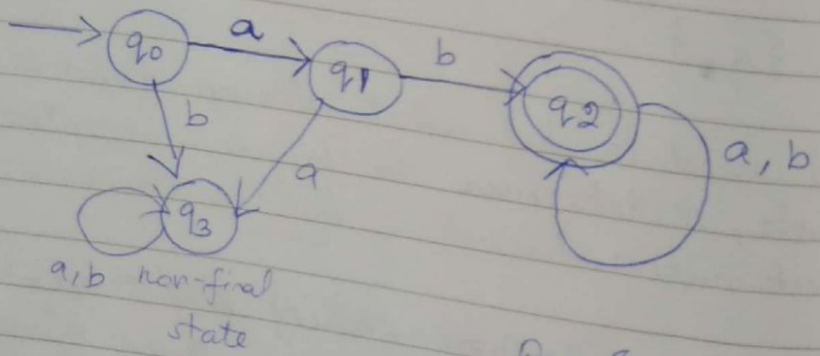
↓

which reaches final state

(aaab, ab, b, ...)

but abab is invalid

- Construct a DFA for all strings over $\Sigma \{a, b\}$ with the prefix ab



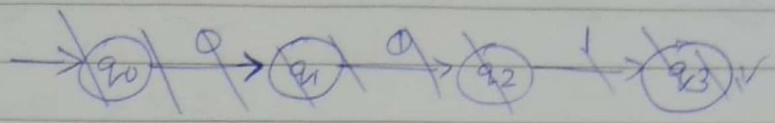

$Q = \{q_0, q_1, q_2, q_3\}$

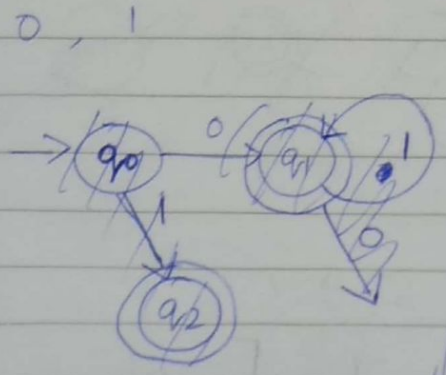
$\Sigma = \{a, b\}$

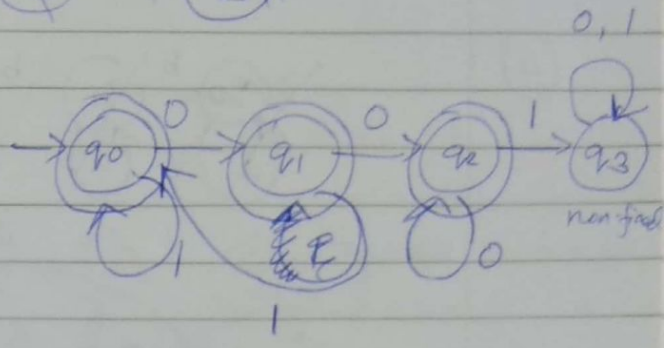
Saturday Test on Gramma, Transdu... DFA

Page No.:
Date:
YOUVA

1. Construct DFA that accepts all strings over $\Sigma = \{0,1\}$ except substring 001

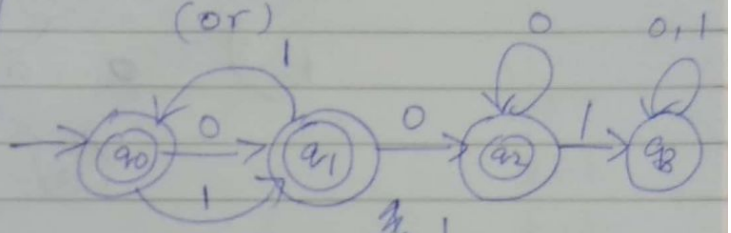



(or)

for instance



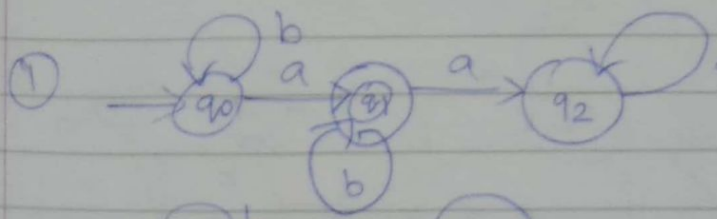

1010 not accepted so not correct

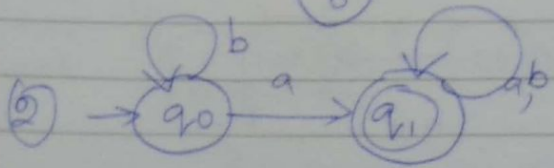
21/6/19 Problems:

DFA:

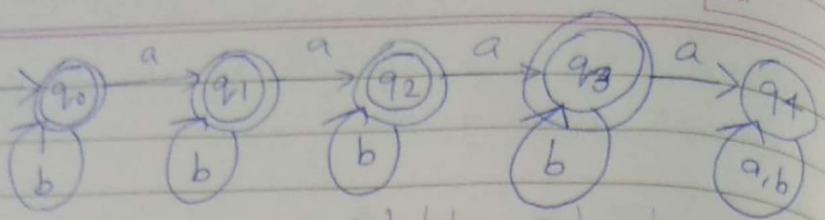
Construct DFA for the following language DFA
$\Sigma = \{a, b\}$

1) All strings w/ exactly one a
2) All strings with at least one a
3) All strings with no more than 3 ds
4) " " " atleast 1 a & exactly 2 bs
5) " " " 2 a's & more than one b



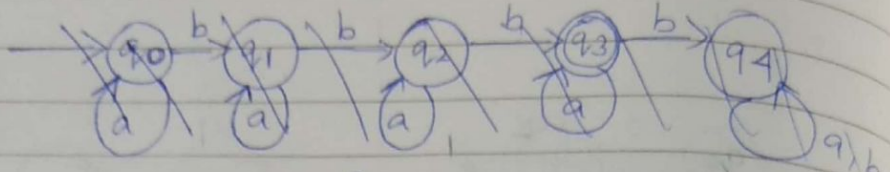



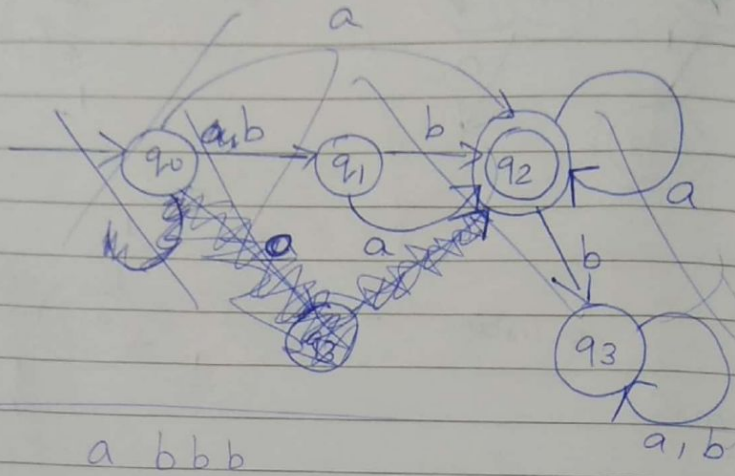
(3)




a b b b  a b a b

(4)




a b b b

Grammar:

1) Find gramma for the foll over $\Sigma\{a,b\}$

a) $L = \{ w : |w| \bmod 3 = 0 \}$  aaabbb

b) $L = \{ w : n_a(w) > n_b(w) + 1 \}$

(4)

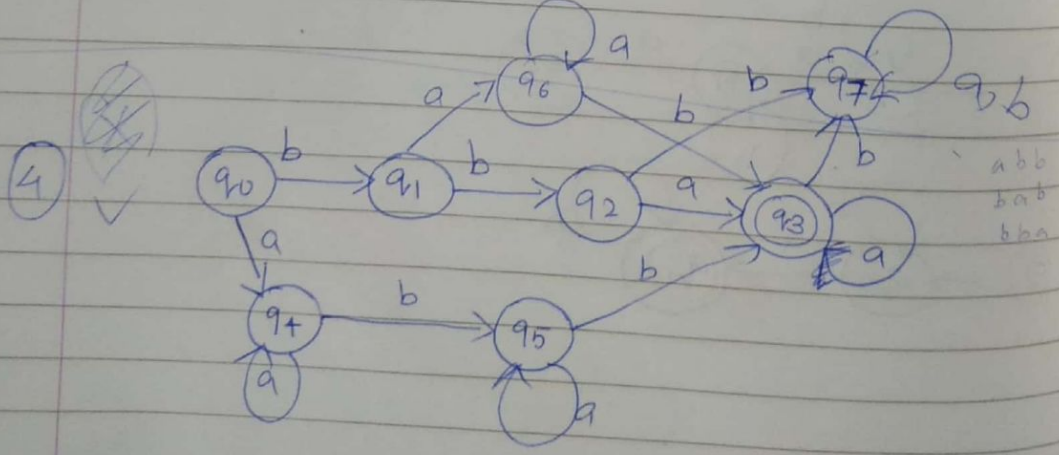


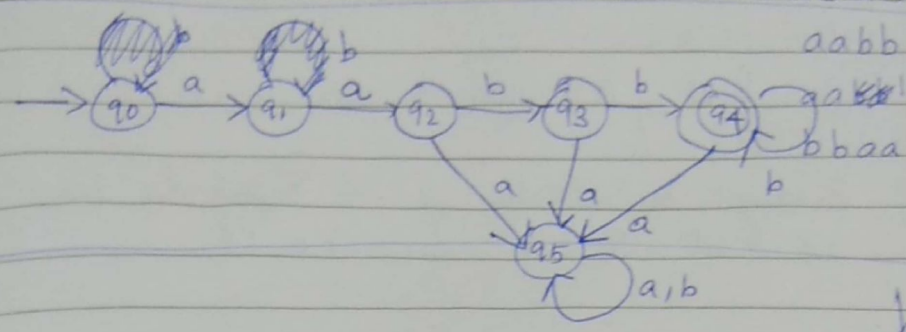

abb
bab
bba

Grammar

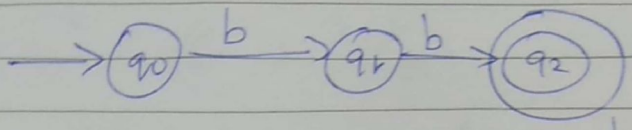
(a) $L = \{\lambda, aaa, abb, aab, bbb, \ldots\}$

$S \rightarrow ABABAB \mid \lambda$

(5)




ababab
aabb
bbaa

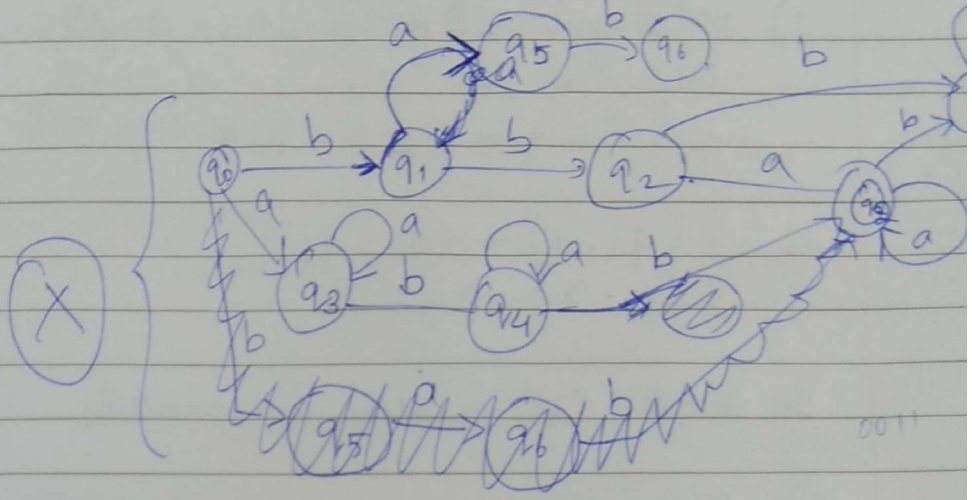



bba
abb
bab

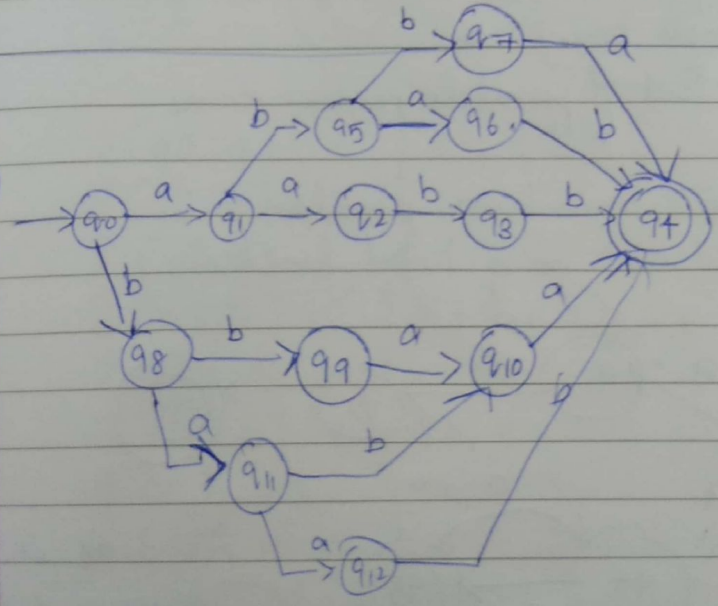
(X)

(4)

(X)




(5)




aabb
abab
abba
baba
bbaa
baab

# HW

Show that given language is regular, you can construct any finite automaton

(1) $L = \{ w : |w| \bmod 5 \neq 0 \}$

(2) $L = \{ w : n_a(w) \bmod 3 > 1 \}$

(3) $L = \{ a w_1 a a a w_2 a : w_1, w_2 \in \{a, b\}^* \}$

(4) Every substring of 1 symbols has at most 2 0s over $\{0, 1\}$

$L = \{ w : |w| \bmod 5 \neq 0 \}$

$L = \{ w : n_a(w) \bmod 3 > 1 \}$

$L = \{ a w_1 a a w_2 a : w_1, w_2 \in \{a,b\}^* \}$

Every substring of 1 symbols has at least 2 0s over $\{0,1\}$

## NFA

1) 2nd argument of transition function may be $\lambda$

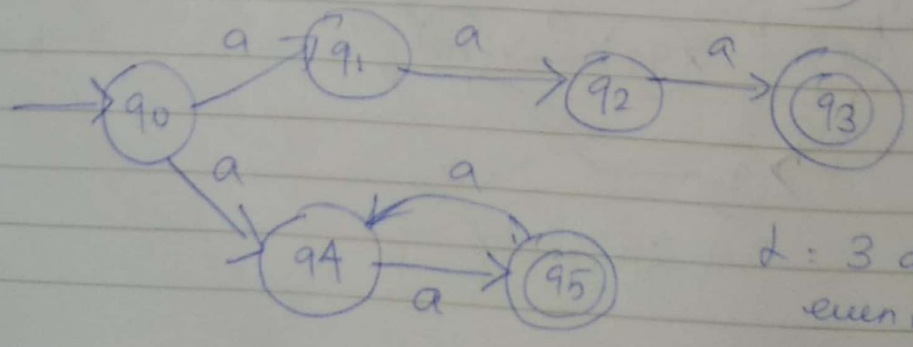
i.e. $\delta : Q \times \{\Sigma \cup \lambda\} \rightarrow 2^Q$

eg. $\delta(q_0, a) \rightarrow (q_1, q_2)$
$\delta(q_0, \lambda) \rightarrow q_2$

In DFA, $\lambda$ transition not possible

2) Multiple states can exist for a single transition
i.e. Set of possible moves exists

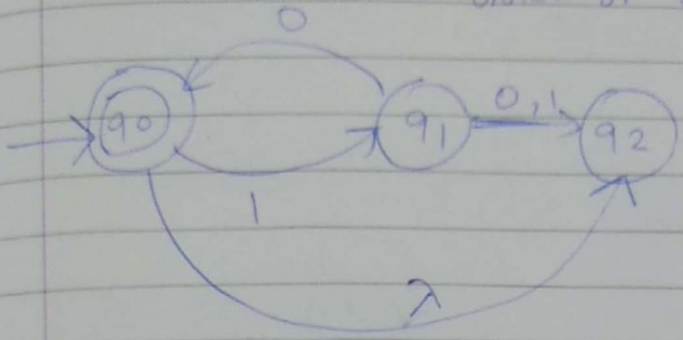
3) Undefined transition possible
(need not be a total function)




L : 3 as or even no. of as

$L = \{aaa\} \cup \{$ even no. of a's except 0 zero occurrence of a $\}$

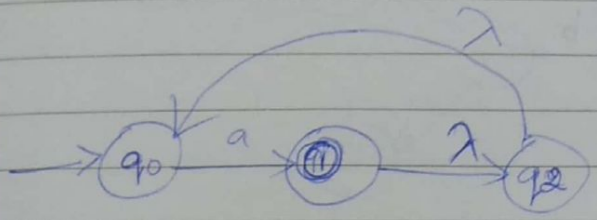
↑ when 2 possibilities reach acceptance state, use UNION or OR




This is an NFA

$L = \{(10)^n : n \geq 0\}$

↓ Starting state is itself final




$a\lambda = a$

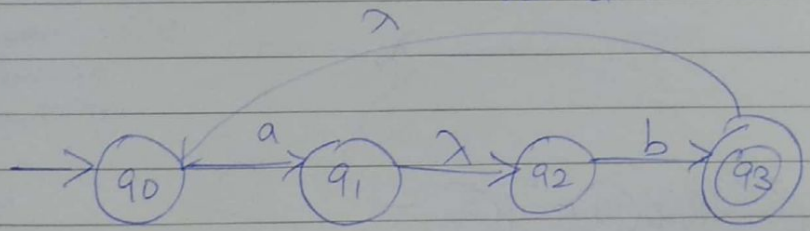
$a\lambda\lambda = a$

$\delta(q0, a) = \{q1, q2, q0\}$

$\delta(q1, a) = \{q1, q2, q0\}$

$\delta(q2, a) = \{q1, q2, q0\}$

↑ what transition are possible AFTER consuming an 'a'?




$\delta(q0, a) = \{q1, q2\}$

$\delta(q0, b) = \phi$

$\delta(q1, a) = \phi$

$\delta(q1, b) = \{q3, q0\}$

$\delta(q2, a) = \phi$

$\delta(q2, b) = \{q3, q0\}$

$\delta(q3, a) = \{q1, q2\}$

$\delta(q3, b) = \phi$

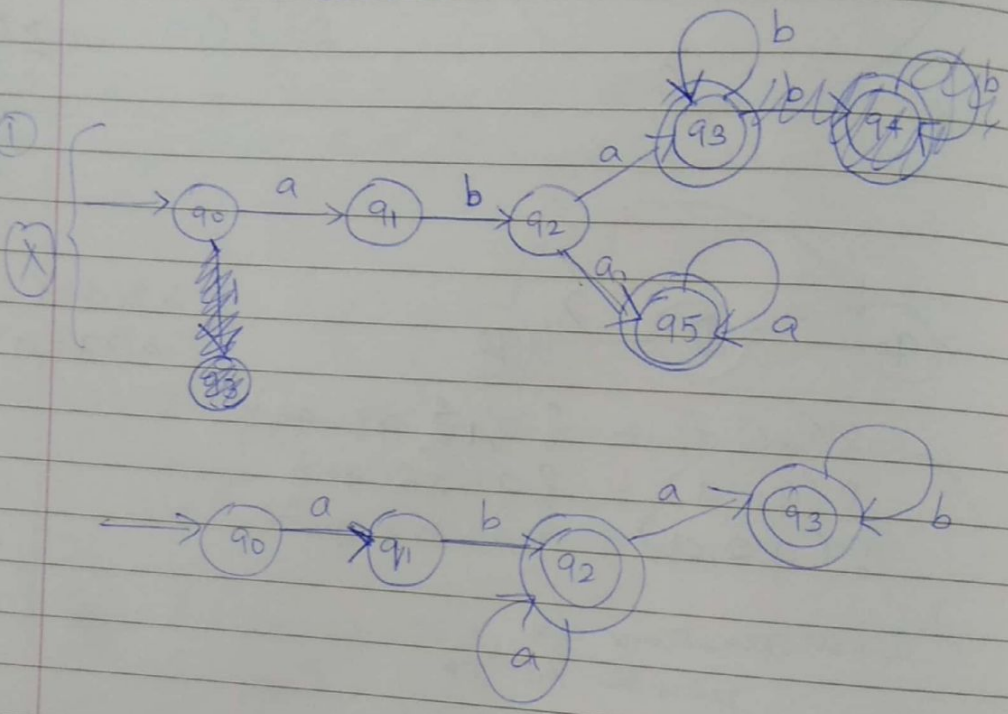
Page No.:
Date:

. Construct NFA for the following

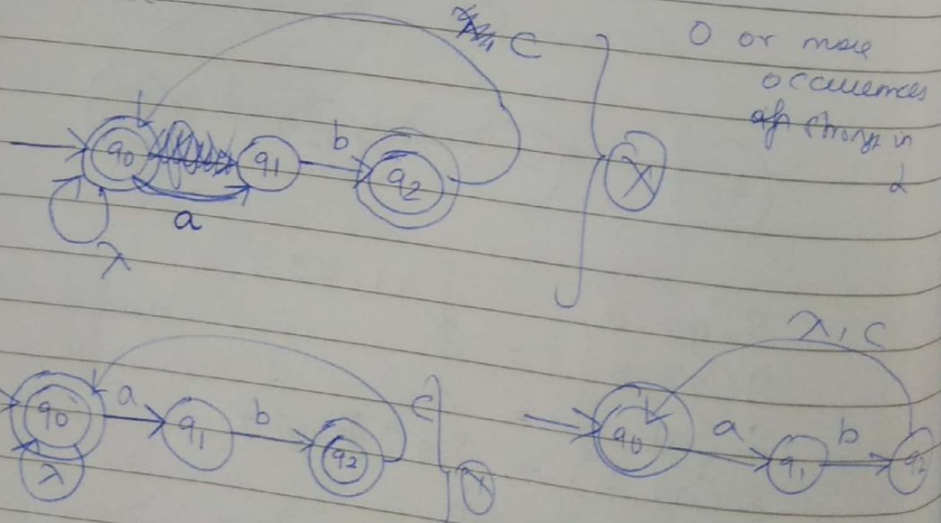
1) $\{abab^n : n \geq 0\} \cup \{aba^n : n \geq 0\}$

2) NFA w/ 3 states that accepts the language $\{ab, abc\}^*$

Note: If no. of states is specified, only 1 solution.



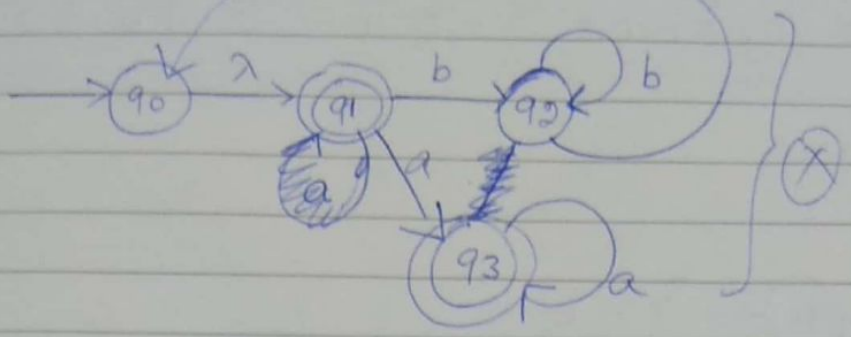

② λ, ab, abc

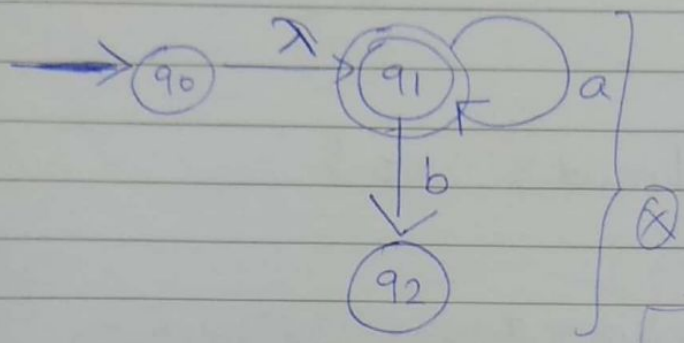


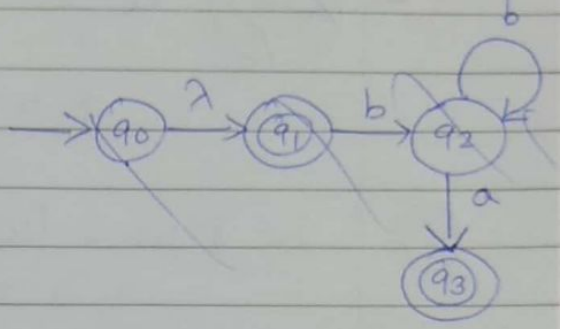

3) Find NFA w/ 4 states

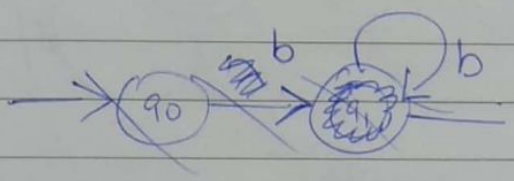
$L = \{a^n : n \geq 0\} \cup \{b^n a : n \geq 1\}$

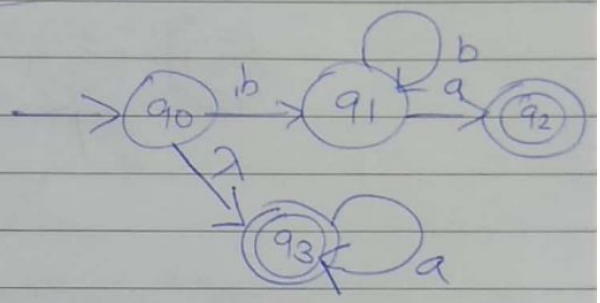
3)



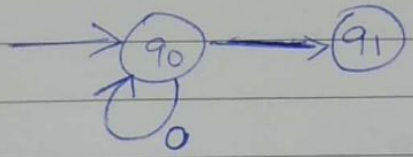



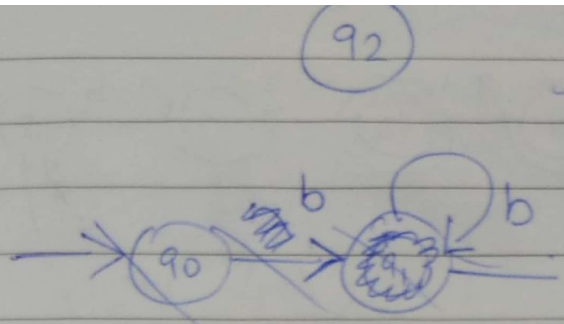



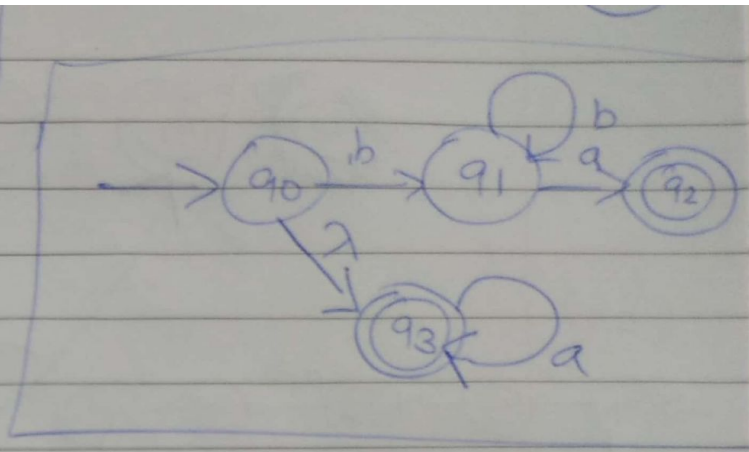




26/6/19 DFA Problem

1) Construct DFA for all strings containing 2 consecutive 0s followed by 2 consecutive 1s over alphabet {0, 1}

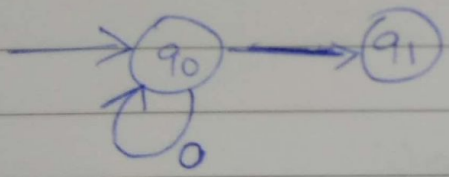





26/6/19 DFA Problem

1) Construct DFA for all strings containing 2 consecutive 0s followed by 2 consecutive 1s over alphabet {0, 1}

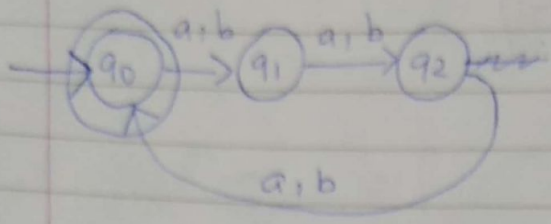


Hard-Boiled Wonderland & the End of the world

mention initial state in exam for DFA & NFA

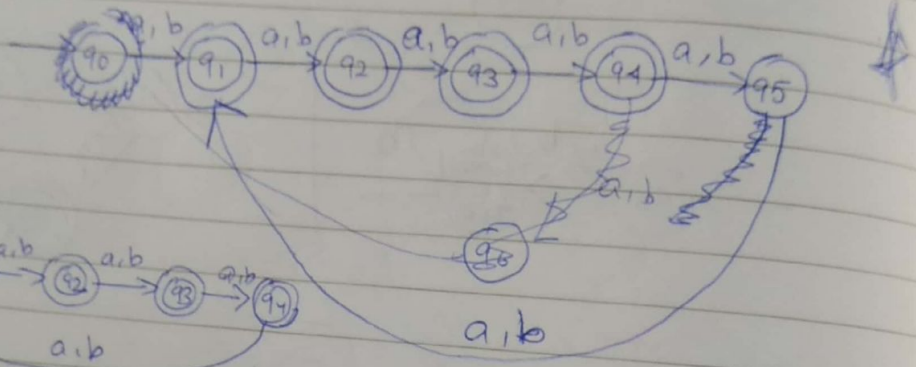
① $L = \{ w : |w| \bmod 3 = 0 \}$ $\Sigma \{a, b\}$

Diagram:




② $L = \{ w : |w| \bmod 5 \neq 0 \}$

1, 2, 3, 4



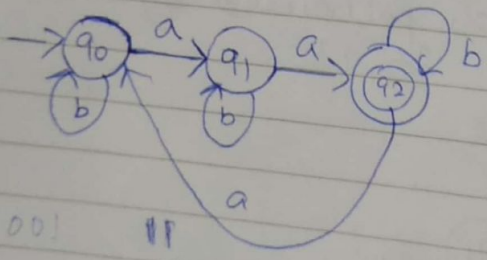

(or)





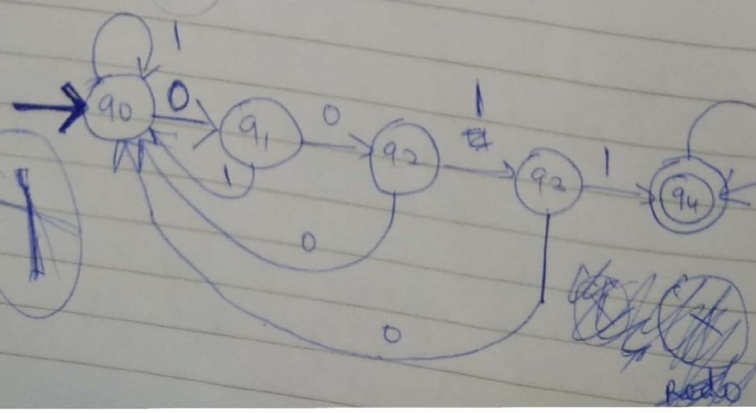


③ $L = \{ w : n_a(w) \bmod 3 > 1 \}$



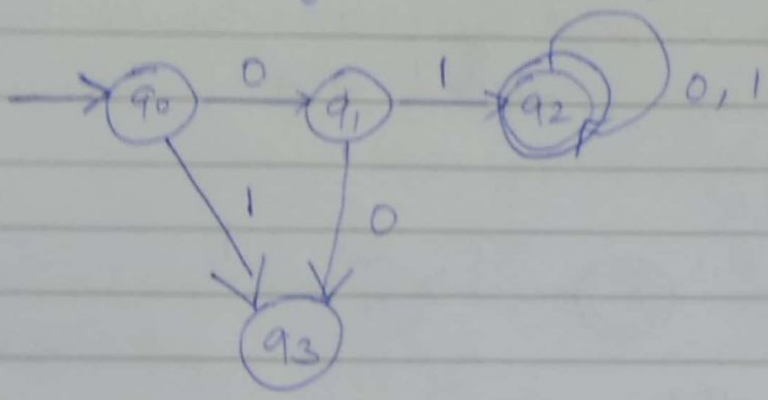

abab
babbab

④ $(0|1)^* 0011 (0/1)^*$ — 0011



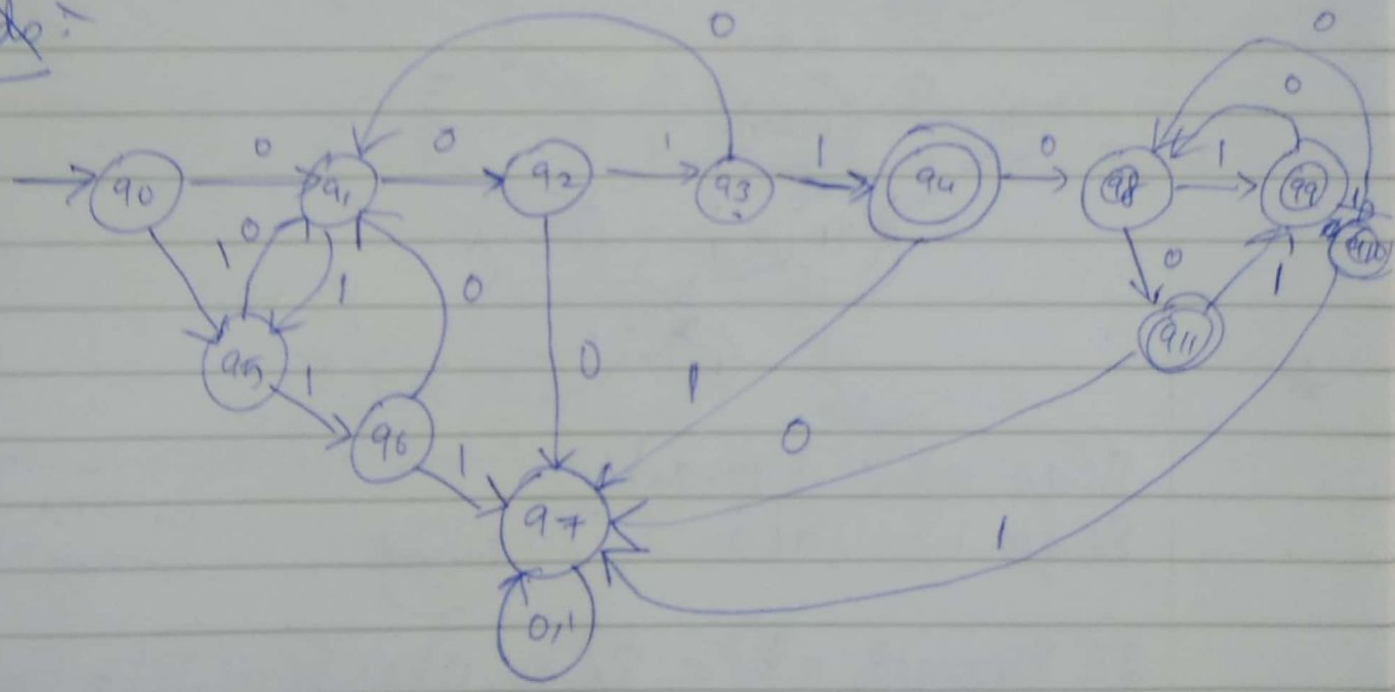
Page No.:
Date:
YOUVA

3) Starts with substring 01 & then any substring over $(0/1)^*$




Redo:

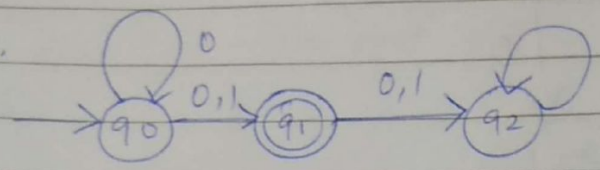
4)



28/6/19 NFA to DFA

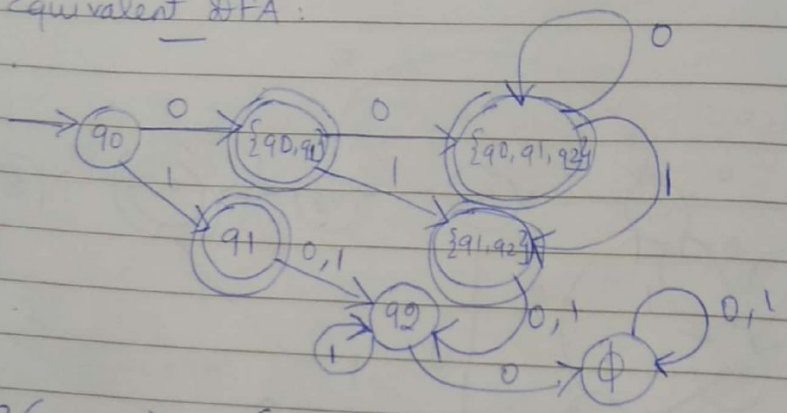
① Languages accepted by NFA should be accepted by converted DFA.

eg.




This is an NFA.

Equivalent DFA:




$\delta(q0, 0)$ : $\{q0, q1\}$ -> combine and make a new state for DFA (group them)

$\delta(q0, 1)$ : $\{q1\}$ -> subset

$\delta(\{q0, q1\}, 0) = \delta(q0, 0) \cup \delta(q1, 0) = \{q0, q1\} \cup$

$\delta(\{q0, q1\}, 1) = \delta(q0, 1) \cup \delta(q1, 1) = \{q0, q1, q$

$= \{q1\} \cup \{q2\}$

$= \{q1, q2\}$

$\delta(q1, 0) = q2$

$\delta(q1, 1) = q2$

$\delta(\{q0, q1, q2\}, 0) = \delta\{q0, 0\} \cup \delta\{q1, 0\} \cup \delta\{q2, 0\}$

$= \{q0, q1\} \cup \{q2\} \cup \phi$

$= \{q0, q1, q2\}$

$\delta(\{q0, q1, q2\}, 1) = \delta\{q0, 1\} \cup \delta\{q1, 1\} \cup \delta\{q2, 1\}$

$= q1 \cup q2 \cup q2$

$= \{q1, q2\}$

$\delta(\{q1,q2\},0) = \delta(q1,0) \cup \delta(q2,0)$
$= q2 \cup \phi = q2$

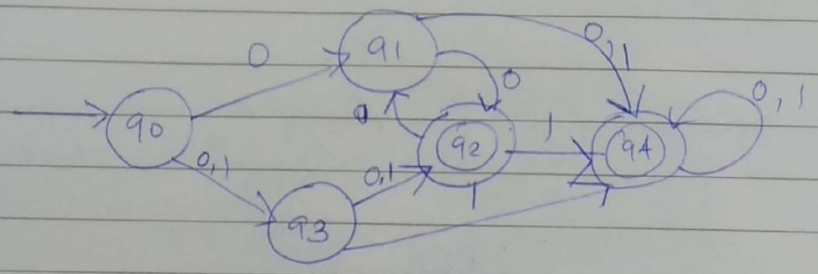
$\delta(\{q1,q2\},1) = \delta(q1,1) \cup \delta(q2,1)$
$= q2 \cup q2 = q2$

$\delta(q2,0) = \phi$
$\delta(q2,1) = q2$

$L = 0^*(0/1)$ (i.e. a path from start to end exists which has $\lambda$)

that is accepted in NFA

In case you have a $\lambda$ transition, you need to mate initial state as final state in the ~~DFA~~



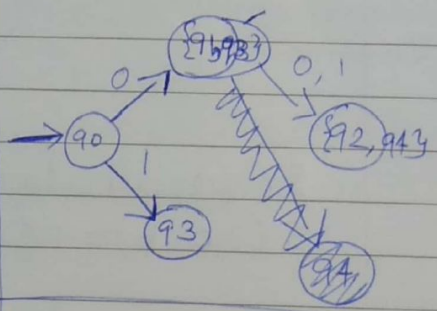

$\delta(q0,0) : \{q1,q3\}$
$\delta(q0,1) : \{q3\}$

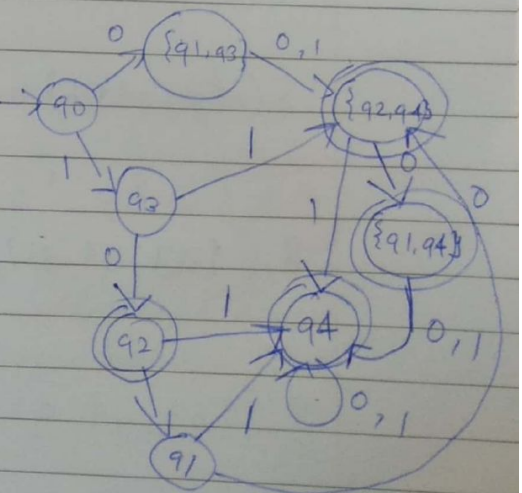
$\delta(\{q1,q3\},0) = \delta(q1,0) \cup \delta(q3,0)$
$= \{q2,q4\} \cup \{q2 = q2,q4\}$
$= q2 \cup q2 \cup q4$

$\delta(\{q1,q3\},1) = \delta(q1,1) \cup \delta(q3,1)$
$= \{q4\} \cup q4$
$= \{q2,q4\}$

$\delta(\{q2,q4\},0) = \delta(q2,0) \cup \delta(q4,0)$
$= q1 \cup q4$
$= \{q1,q4\}$

$\delta(\{q2,q4\},1) = \delta(q2,1) \cup \delta(q4,1)$
$= q4 \cup q4$



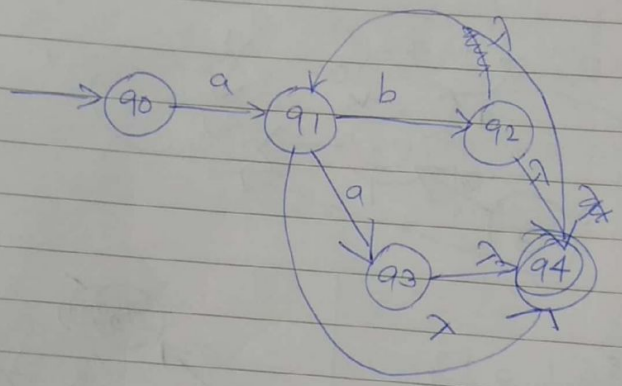
$\delta(q3,0) = q2$
$\delta(q3,1) = \{q1, q4\}$

$\delta(\{q1,q4\},0) = \delta(q1,0) \cup \delta(q4,0)$
$= \{q2,q4\} \cup q4$
$= \{q2,q4\}$
$\delta(\{q1,q4\},1) = \delta(q1,1) \cup \delta(q4,1)$
$= q4 \cup q4 = q4$

$\delta(q2,0) = q1$
$\delta(q2,1) = q4$

$\delta(q4,0) = q4$
$\delta(q4,1) = q4$

$\delta(q1,0) = \{q2,q4\}$
$\delta(q1,1) = \{q4\}$



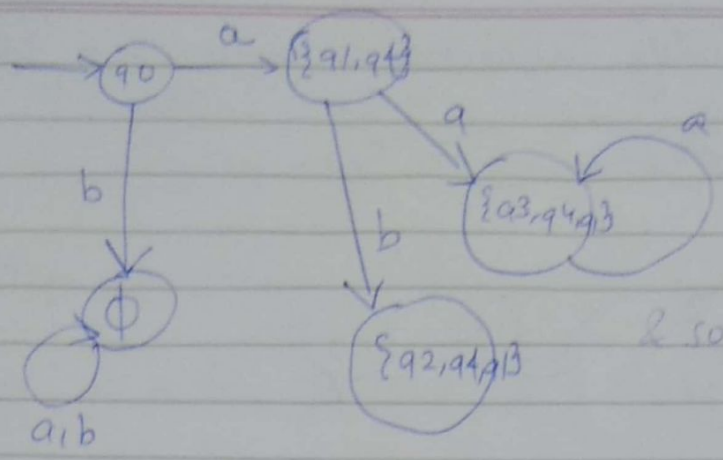

$\delta(q0,a) = \{q1,q4\}$
$\delta(q0,b) = \phi$

$\delta(\{q1,q4\},a) = \delta(q1,a) \cup \delta(q4,a) = \{q3,q4,q1\}$
$\cup \{q3,q4,q1\}$

$\delta(\{q1,q4\},b) = \delta(q1,b) \cup \delta(q4,b)$
$= \{q2,q4,q1\} \cup \{q2,q4\}$

$\delta(\{q3,q4,q1\},a) = \delta(q3,a) \cup \delta(q4,a) \cup \delta(q1,a)$
$= \{q3,q4,q1\} \cup \{q3,q4,\}$
$\{q3,q4,q1\}$

$\delta(\{q3,q4,q1\},b) = \delta(q3,b) \cup \delta(q4,b) \cup \delta(q1,b)$
$= \{q2,q4,q1\}$




& so on (Contd.)

29/6/19 DFA minimization

① Indistinguishable states
② Distinguishable states → equivalent states

① If you have 2 states P, Q
and $\delta^*(P, w) \in F$
( ) final state
↳ extended transition function
(2nd argument is a string, rather than a symbol as usual)
and $\delta^*(Q, w) \in F$
( ) taking same string from both P & Q leads to final state
then P & Q are indistinguishable states.

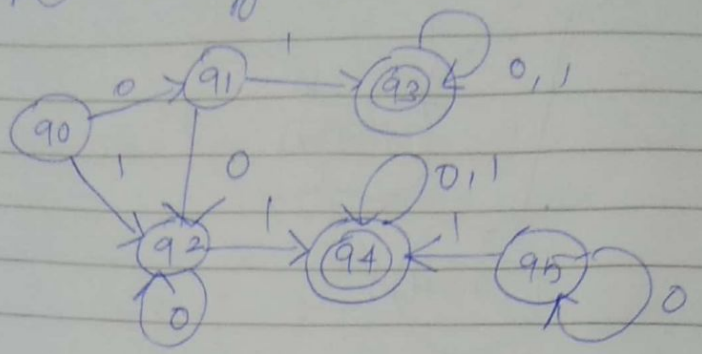
ALSO

If $\delta^*(p, w) \notin F$
( ) non-final state
and $\delta^*(q, w) \notin F$
then also p & q are indistinguishable states.

② If $\delta^*(p, w) \in F$ and $\delta^*(q, w) \notin F$
or
If $\delta^*(p, w) \notin F$ and $\delta^*(q, w) \in F$
then they are distinguishable.

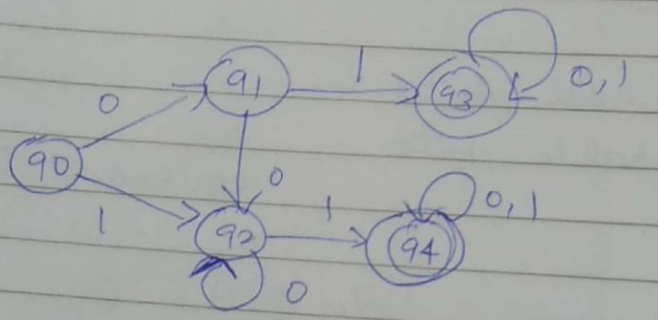
Step ①: Identify unreachable states

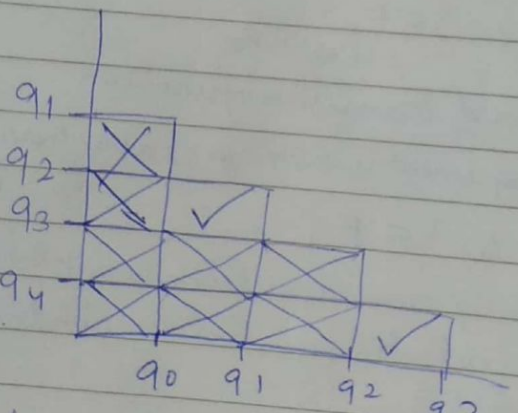
eg.




Here, this is q5. (no incoming edges only outgoing)

Step ②:




final-non-final combos to be maker

$\delta(q3, 0) = q3$ $\quad \delta(q3, 1) = q3$

$\delta(q4, 0) = q4$ $\quad \delta(q4, 1) = q4$

We have found a potential combination for q3, q4 because they are indistinguishable.

Page No.
Date

Since they are not equal, we check if q1 & q2 can have equal transitions

q0 q2:

$\delta(q0, 0) = q2$ $\delta(q2, 1) = q4$

$\delta(q0, 0) = q1$ $\delta(q0, 1) = q2$

They are distinguishable states.

If q1, q2 is satisfied & q2, q4 satisfied then we can say q0, q2 are equal (not possible as they are distinguishable)

Check: q1, q2

$\delta(q1, 0) = q2$ } even if the $\delta(q2, 1)$ $\delta(q1, 1) = q3$

$\delta(q2, 0) = q2$ } are equal $\delta(q2, 1) = q4$

=> not equal. Now, q3, q4 is equal

=> q1, q2 is equal

But q2, q4 not equal so q0, q2 is not equal

q0, q1

$\delta(q0, 0) = q1$ } equal

$\delta(q1, 0) = q2$ } not equal $\delta(q0, 1) = q2$

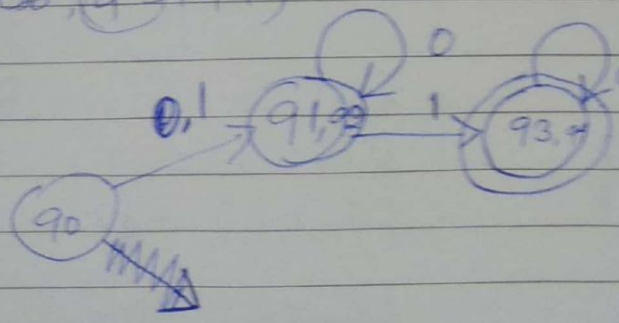
$\delta(q1, 1) = q3$

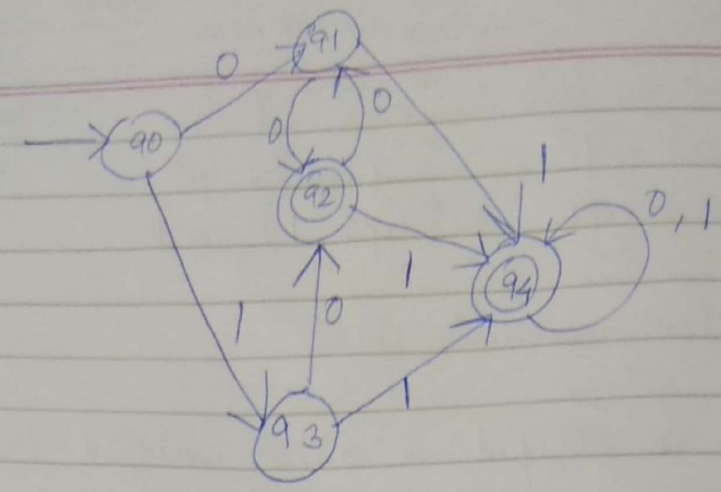
(.) different state

So q0, q1 is different.

From table we can combine (q1, q2) since you possess the lowest value (q1)

Also (q3, q4)

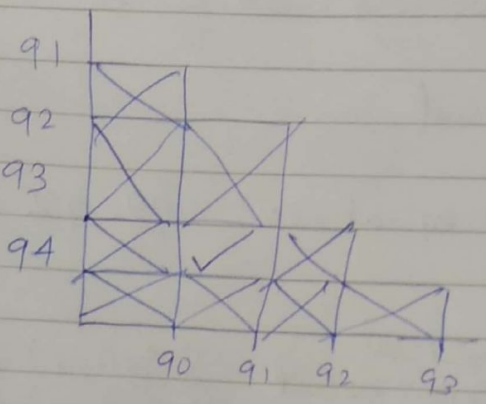


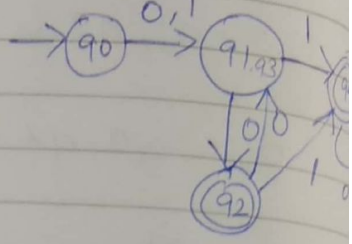



Step 1: Identify unreachable states

=> No unreachable states

Step 2:




q2, q4

$\delta(q2,0) = q1$
$\delta(q4,0) = q4$
} not equal => they are distinguishable

q3, q0

$\delta(q0,0) = q1$
$\delta(q3,0) = q2$
} distinguishable so not equal

q1, q2

$\delta(q1,0) = q2$
$\delta(q2,0) = q1$

q3, q1

$\delta(q3,0) = q2$
$\delta(q1,0) = q2$

$\delta(q3,1) = q4$
$\delta(q1,1) = q4$

q1, q0

$\delta(q0,0) = q1$
$\delta(q1,0) = q2$

q3, q2

$\delta(q3,0) = q2$
$\delta(q2,0) = q1$
} not equal