22/01/18

Theory of Automata & Formal Languages (RCS-403)

Course Objectives: The aim of this course is,

- To define mathematical methods of computing devices, called abstract machines, namely (viz.) Finite Automata, Pushdown Automata, and Turing Machines.
- To study the capabilities of these abstract machines.
- To classify machines by their power to recognize languages.
- Employ finite state machines to solve problems in computing.
- Explain deterministic and non-deterministic machines.
- Identify different formal language classes and their relationships.
- Design grammars and recognizers for different formal languages
- Determine the decidability and intractability of computational problems.

Unit-I (TAFL)

FUNDAMENTALS :

- **Symbol :—** An atomic unit, such as a digit, character, lower-case letter, etc.. Sometimes a word.
  (Note : formal language does not deal with the "meaning" of the symbols.)
- **Alphabet :—** A finite set of symbols, usually denoted by Σ.
  $\Sigma = \{a, b\}$, $\Sigma = \{0, 1, 2\}$, $\Sigma = \{0, a, 9, \}$, $\Sigma = \{a, b, c, d\}$ etc.
- **String :—** A finite length sequence of symbols, presumably from some alphabet.
  $w = abba$, $u = 0112102$, $v = 0a99aa$, $x = abbccd$, $y = aaa$
- Special String : ε (also denoted by λ) called empty string i.e. a string with no symbols at all.
- **Concatenation :—** Given two strings w and v. concatenation of w and v is the string obtained by appending the symbols of v to the right end of w, i.e.,

  if $w = a_1 a_2 \ldots a_n$ ← (right end of w)

  & $v = b_1 b_2 \ldots b_m$,

  then $wv = a_1 a_2 \ldots a_n b_1 b_2 \ldots b_m$. (concatenation)
- **Reversal :—** The reverse of a string is obtained by writing the symbols in reverse order.

  e.g. $w = a_1 a_2 \ldots a_n$

  then $w^R = a_n \ldots a_2 a_1$.
- **Length :—** Number of symbols in the string, denoted by $|w|$.

  e.g. if. $w = 01101$, then $|w| = 5$

  $w = aba$, then $|w| = 3$ etc.

  (Note : $|\lambda| = 0$.) & $\lambda w = w\lambda = w$ $\forall$ w.
- **Substring :—** Any string of consecutive symbols in some w is said to be a substring of w. If $w = uv$, u & v are substrings of w.

  $u \rightarrow$ prefix & $v \rightarrow$ suffix of w.

  e.g., if $w = abbab$, then $\{\lambda, a, ab, abb, abba, abbab\}$ is the set of all prefixes of w, while bab, ab, b are some of its suffixes.

- ω^n : string obtained by ω n times.
  e.g. if $\omega = abb$, then $\omega^3 = \underbrace{abb}_{\omega}\underbrace{abb}_{\omega}\underbrace{abb}_{\omega}$
  special case, $\omega^0 = \lambda$, $\forall\ \omega$.

- Some special sets of strings:-
  Σ^* : All strings of symbols from Σ [countably infinite set]
  $\Sigma^+ = \Sigma^* - \{\lambda\}$
  (Note: While Σ is finite by assumption, Σ^* and Σ^+ are always infinite since there is no limit on the length of the strings in these sets.)

  Σ^n : set of all strings over Σ whose length is n.
  $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \ldots$
  $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \ldots$

  e.g. $\Sigma = \{0, 1\}$
  $\Sigma^* = \{\lambda, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$
  $\Sigma^+ = \{0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$

- Language :- (1) A set of strings from some alphabet (finite or infinite). In other words,
  (2) A language is any subset L of Σ^*.

  e.g., if $\Sigma = \{a, b\}$
  $L_1 = \{\lambda, a, b\}$
  $L_2 = \{ab, aabb, aaabbb, \ldots\}$
  $L_3 = \{\omega \in \Sigma^* \mid |\omega|_a = |\omega|_b\}$
  $L_4 = \{a^n b^n : n \geq 0\}$

  L_1 is finite language & L_4 & L_2 & L_3 are infinite languages.

- Some special languages:

ϕ or $\{\}$: The empty set/language, containing no string.
$\{\lambda\}$: A language containing one string, the empty string.

- Since languages are sets, the union, intersection, and difference of two languages are immediately defined.

- Complement of a language:
  $\bar{L} = \Sigma^* - L$.

- Reverse of a Language: The reverse of a language is the set of all strings reversals, i.e.
  $L^R = \{\omega^R : \omega \in L\}$.

24/01/18

Recursive Definition of Length

for any letter: $|a| = 1$

for any string $\cdot wa$: $|wa| = |w| + 1$ $[\because |wx| = |w| + |x|]$

Ex: $|abba| = |abb| + 1$
$= |ab| + 1 + 1$
$= |a| + 1 + 1 + 1$
$= 1 + 1 + 1 + 1$
$= 4.$

- Length of concatenation
$|ux| = |u| + |x|$

Ex: $u = aab$, $|u| = 3$
$v = abaab$, $|v| = 5$
$|uv| = |aababaab| = 8$
$|uv| = |u| + |v| = 3 + 5 = 8.$

Operations on Languages

The usual set operations

$\{a, ab, aaaa\} \cup \{bb, ab\} = \{a, ab, bb, aaaa\}$
$\{a, ab, aaaa\} \cup \{bbb, aba\} = \{a, ab, bbb, aba, aaaa\}$
$\{a, ab, aaaa\} \cap \{bb, ab\} = \{ab\}$
$\{a, ab, aaaa\} - \{bb, ab\} = \{a, aaaa\}$

- Complement: $\bar{L} = \Sigma^* - L$

Ex: $L = \{a, ba\}$, then
$\bar{L} = \{\lambda, b, aa, ab, bb, aaa, \ldots\}.$

- Reverse: Def^n: $L^R = \{w^R : w \in L\}$

Ex: $\{ab, aab, baba\}^R = \{ba, baa, abab\}$
$L = \{a^n b^n : n \geq 0\}$
$L^R = \{b^n a^n : n \geq 0\}.$

- Concatenation: $L_1 L_2 = \{xy : x \in L_1, y \in L_2\}$

Ex: $L_1 = \{a, ab, ba\}$, $L_2 = \{b, aa\}$
$L_1 L_2 = \{ab, aaa, abb, abaa, bab, baaa\}$

Def^n: $L^n = \underbrace{LL \ldots L}_{n}$.

Ex: $\{a,b\}^3 = \{a,b\}\{a,b\}\{a,b\} =$
$\{aaa, aab, aba, abb, baa, bab, bba, bbb\}$

Special case: $L^0 = \{\lambda\}$

Ex: $\{a, bba, aaa\}^0 = \{\lambda\}$

Ex: $L = \{a^n b^n : n \geq 0\}$
$L^2 = \{a^n b^n a^m b^m : n, m \geq 0\}$
$aabbaaabbb \in L^2$

24/01/18

Star-closure (Kleene *) or concatenation closure

Def^n: $L^* = L^0 \cup L^1 \cup L^2 \ldots$ or $L^* = \bigcup_{i=0}^{\infty} L^i$

Ex:

$$\{a, bb\}^* = \left\{ \begin{array}{l} \lambda, \\ a, bb, \\ aa, abb, bba, bbbb, \\ aaa, aabb, abba, abbbb, \ldots \end{array} \right\}$$

- **Positive closure:**

Def^n: $L^+ = L^1 \cup L^2 \cup \ldots$ or $L^+ = \bigcup_{i=1}^{\infty} L^i$

$L^+ = L^* - \{\lambda\}$

$$\{a, bb\}^+ = \left\{ \begin{array}{l} a, bb, \\ aa, abb, bba, bbbb, \\ aaa, aabb, abba, abbbb, \ldots \end{array} \right\}$$

25/01/18

Language:

$L \subseteq \Sigma^*$

A Language is any subset of Σ^*.

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

$\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$

Some Languages are

$\{\lambda\}$, $\{a, aa, aab\}$, $\{\lambda, abba, baba, aa, ab, aaaaa\}$ etc.

An infinite Language

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

λ
ab
$aabb$
$aaaaabbbbb$
$\Big\}$ $\in L$, $\quad abb \notin L$.

25/01/18

Languages can be defined

- Descriptive definition
- Recursive definition
- using Regular Expressions (RE)
- using Finite Automaton (FA)
- -etc.

Descriptive definition of language:

The language is defined, describing the conditions imposed on its words.*

*words: Def^n: words are strings belonging to some language.

Ex: If $\Sigma = \{x\}$, then $L = \{x^n : n = 1,2,3,\ldots\}$ or $L = \{x, xx, xxx, \ldots\}$. Here $x, xx, \ldots$ are the words of L.

*Note: All words are strings, but not all strings are words.

Ex: The language L of strings of odd length, defined over $\Sigma = \{a\}$, can be written as

$L = \{a, aaa, aaaaa, aaaaaaa, \ldots\}$

Ex: The language L of strings that does not start with a, defined over $\Sigma = \{a, b, c\}$, can be written as

$L = \{\lambda, b, c, ba, bb, bc, ca, cb, cc, \ldots\}$

Ex: The language L of strings of length 2, defined over $\Sigma = \{0, 1, 2\}$, can be written as

$L = \{00, 01, 02, 10, 11, 12, 20, 21, 22\}$

Ex: The language L of strings ending in 0, defined over $\Sigma = \{0, 1\}$,

$L = \{0, 00, 10, 000, 010, 100, 110, \ldots\}$

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

$L = \{x \mid x \text{ is in } \Sigma^* \text{ and } x \text{ contains an even number of 0's}\}$

$L = \{00, 010, 100, 001, 0000, \ldots\}$

Ex: The language EQUAL, of strings with number of a's equal to number of b's, defined over $\Sigma = \{a, b\}$, can be written as

$\{\lambda, ab, ba, aabb, abab, baba, abba, \ldots\}$

Ex: The language EVEN-EVEN, of strings with even number of a's and even number of b's, defined over $\Sigma = \{a, b\}$, can be written as $\{\lambda, aa, bb, aaaa, aabb, abab, abba, baab, baba, bbaa, bbbb, \ldots\}$

Ex: The language INTEGER, of strings defined over $\Sigma = \{-, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, can be written as

$INTEGER = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$

Ex: The language EVEN, of strings defined over $\Sigma = \{-, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, can be written as

$EVEN = \{\ldots, -4, -2, 0, 2, 4, \ldots\}$.

Ex: The language $\{a^n b^n\}$, of strings defined over $\Sigma = \{a, b\}$, as
$L = \{a^n b^n : n = 1, 2, 3, \ldots\}$, can be written as
$L = \{ab, aabb, aaabbb, aaaabbbb, \ldots\}$.

Ex: The language $\{a^n b^n a^n\}$, of strings defined over $\Sigma = \{a, b\}$, as
$L = \{a^n b^n a^n : n = 1, 2, 3, \ldots\}$, can be written as
$L = \{aba, aabbaa, aaabbbaaa, aaaabbbbaaaa, \ldots\}$

Ex: The language factorial, of strings defined over
$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ i.e.
$L = \{1, 2, 6, 24, 120, \ldots\}$

Ex: The language FACTORIAL, of strings defined over $\Sigma = \{a\}$, as
$L = \{a^{n!} : n = 1, 2, 3, \ldots\}$, i.e.
$L = \{a, aa, aaaaaa, \ldots\}$

DOUBLE FACTORIAL, of strings defined over $\Sigma = \{a, b\}$, as
$L = \{a^{n!} b^{n!} : n = 1, 2, 3, \ldots\}$, i.e.
$L = \{ab, aabb, aaaaaabbbbbb, \ldots\}$

Ex: The language PRIME, of strings defined over $\Sigma = \{a\}$, as
$L = \{a^p : p \text{ is prime}\}$, i.e.
$L = \{aa, aaa, aaaaa, aaaaaaa, aaaaaaaaaaa, \ldots\}$.

31/01/18

Finite Automata

- DFA — Deterministic Finite Automata
- NFA — Nondeterministic Finite Automata

DFA:- There are a fixed number of states and we can only be in one state at a time.

NFA:- There are a fixed number of states but we can be in multiple state at one time.

Deterministic Finite Automata

- A DFA is a five-tuple : $M = (Q, \Sigma, q_0, F, \delta)$

Q	A finite set of states
Σ	A finite set of input ~~opt~~ symbols (Alphabet)
q_0	The initial/starting state, $q_0 \in Q$
F	A set of final/accepting states, $F \subseteq Q$.
δ	A transition function, which is a total function $Q \times \Sigma \rightarrow Q$.

$\delta : (Q \times \Sigma) \rightarrow Q$. δ is defined for any q in Q ($q \in Q$) and s in Σ, i.e. $s \in \Sigma$, and

$\delta(q, s) = q'$ is equal to another state q' in Q i.e. $q' \in Q$.

Intuitively, $\delta(q, s)$ is the state entered by M after reading symbol s while in state q.

Ex: $Q = \{q_0, q_1\}$

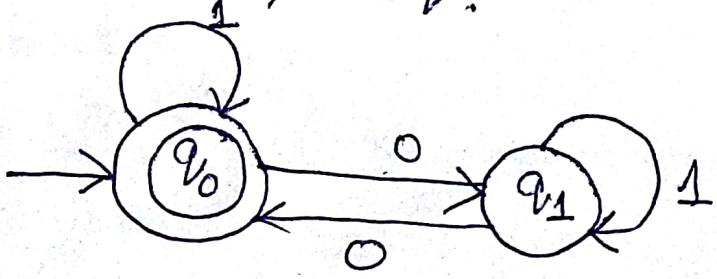
$\Sigma = \{0, 1\}$

Start state is q_0

$F = \{q_0\}$

δ :

	0	1
q_0	q_1	q_0
q_1	q_0	q_1



DFA 'M'

- Let $M=(Q,\Sigma,q_0,F,\delta)$ be a DFA and let w be in Σ^*. Then w is <u>accepted</u> by M iff $\delta(q_0,w)=p$ for some state p in F.
- Let $M=(Q,\Sigma,q_0,F,\delta)$ be a DFA. Then the language accepted by M is the set:

$$L(M)=\{w \mid w \text{ is in } \Sigma^* \ \delta(q_0,w) \text{ is in } F\}$$

Another equivalent definition:

$$L(M)=\{w \mid w \in \Sigma^* \text{ and } w \text{ is accepted by } M\}$$

- Let L be a language. Then L is a <u>regular</u> language iff there exists a DFA M such that $L=L(M)$.
- Let $M_1=(Q_1,\Sigma_1,q_0,F_1,\delta_1)$ and $M_2=(Q_2,\Sigma_2,p_0,F_2,\delta_2)$ be DFAs.

Then M_1 and M_2 are <u>equivalent</u> iff $L(M_1)=L(M_2)$.

<u>Notes</u>:

- A DFA $M=(Q,\Sigma,q_0,F,\delta)$ partitions the set Σ^* into two sets $L(M)$ and $\Sigma^*-L(M)$.
- If $L=L(M)$ then L is a subset of $L(M)$ and $L(M)$ is a subset of L.
- Similarly, if $L(M_1)=L(M_2)$ then $L(M_1)$ is a subset of $L(M_2)$ and $L(M_2)$ is a subset of $L(M_1)$.
- Some languages are regular, other are not. for example if

$L_1=\{x \mid x$ is a string of 0's and 1's containing an even number of 1's$\}$ and

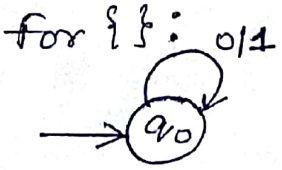
$L_2=\{x \mid x=0^n1^n$ for some $n\geq 0\}$

then L_1 is regular but L_2 is not.

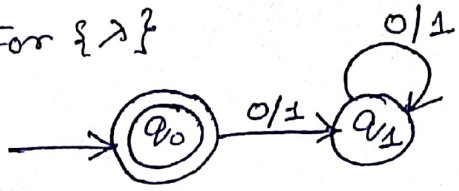
31/01/18

- Let $\Sigma = \{0, 1\}$. Give DFAs for $\{\}$, $\{\lambda\}$, Σ^* and Σ^+.

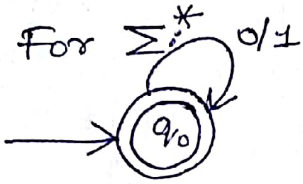
For $\{\}$:



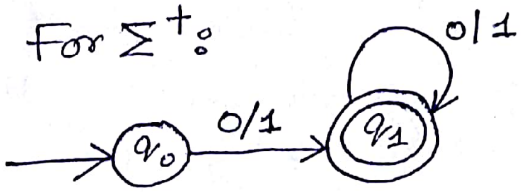
For $\{\lambda\}$



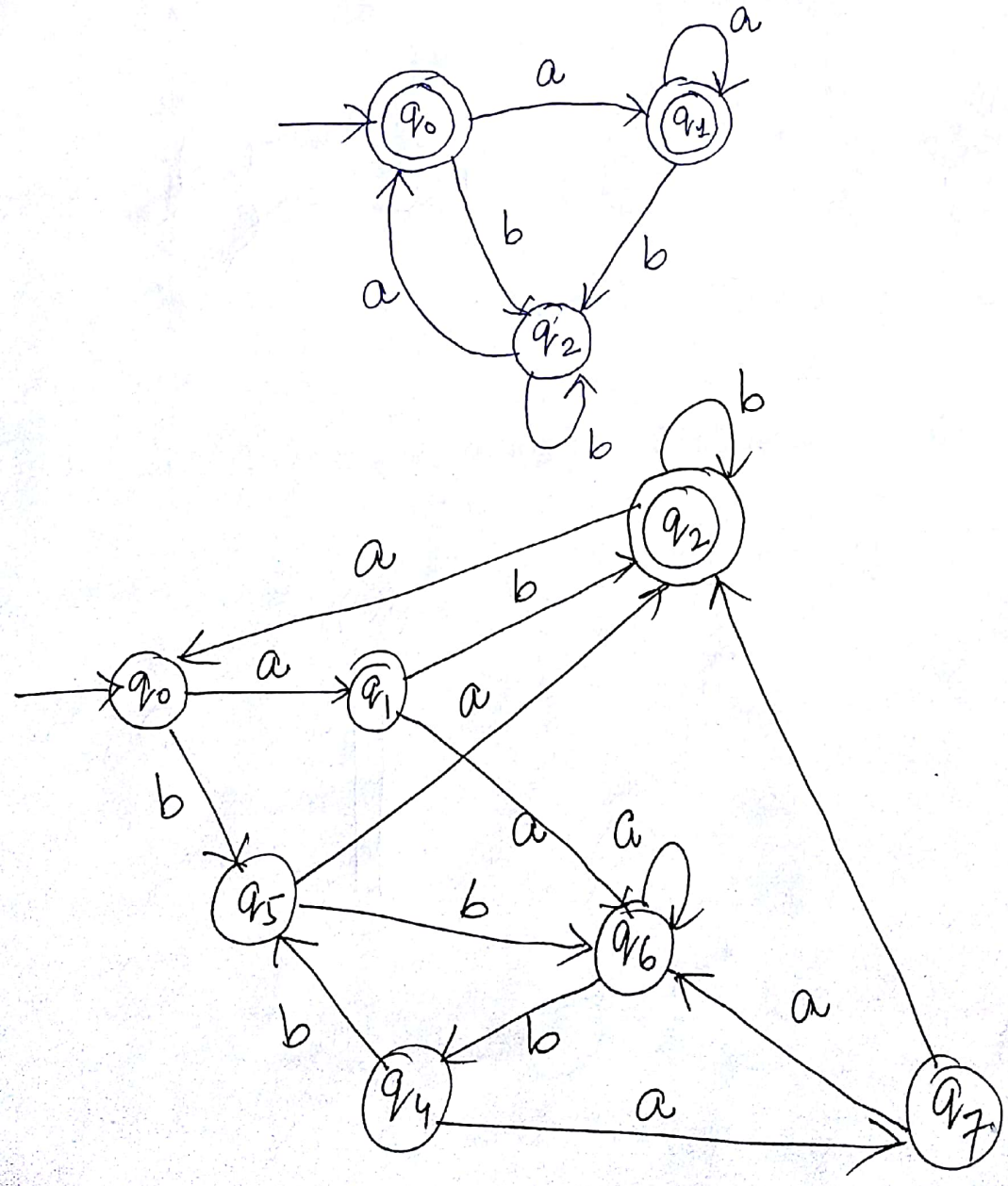
For Σ^*:



For Σ^+:

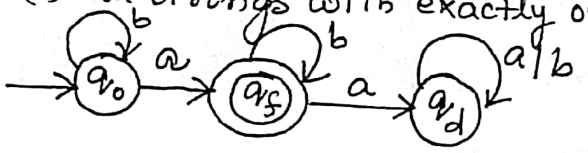


Ex: $L = \{\omega \in \Sigma^* \mid \omega \text{ is } \lambda \text{ or } \omega \text{ ends in } a\}$



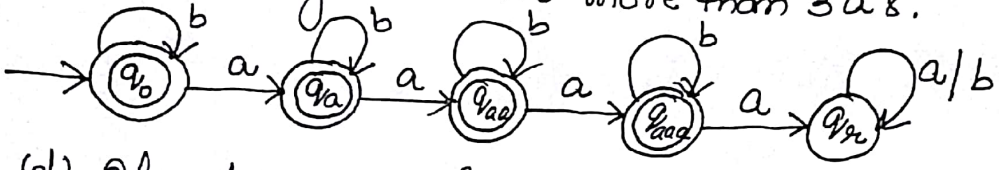
4/2/18 for $\Sigma = \{a, b\}$ construct DFA's

(a) all strings with exactly one a.

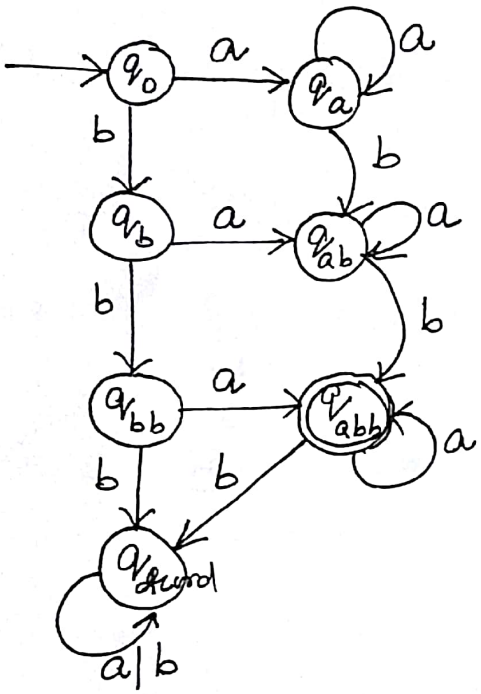


(b) all strings with @least one a

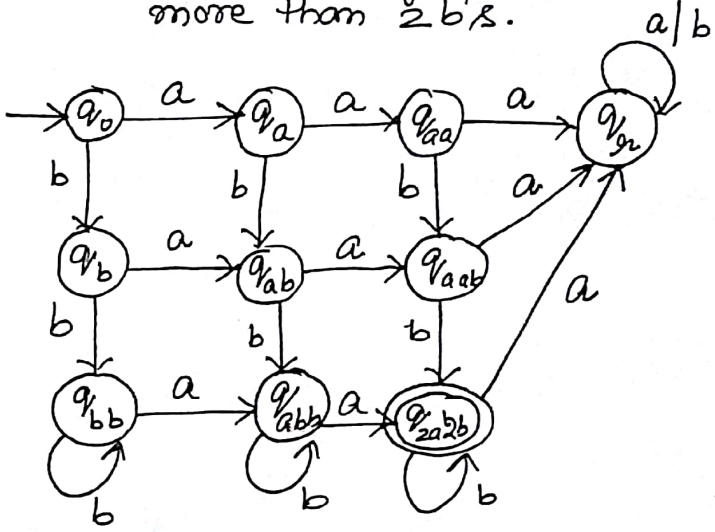
(c) all strings with no more than 3 a's.



(d) @least one a & exactly 2 b's.



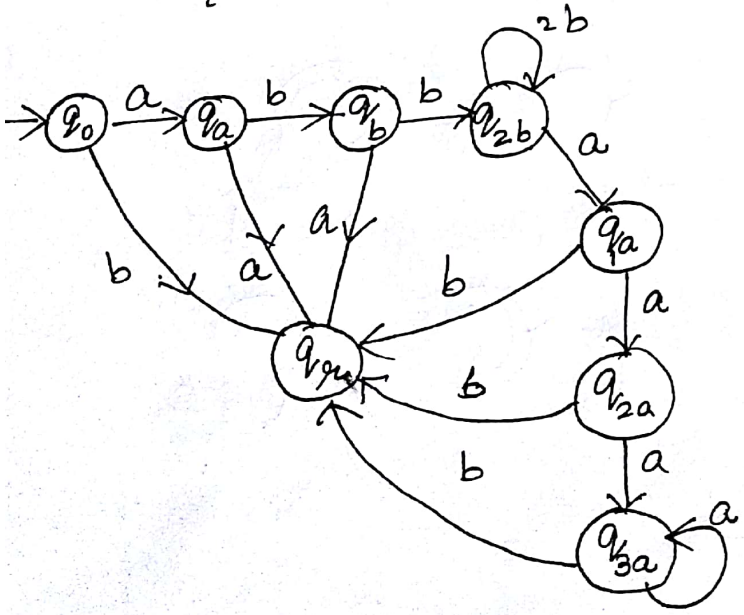
(e) all strings with 2 a's & more than 2 b's.



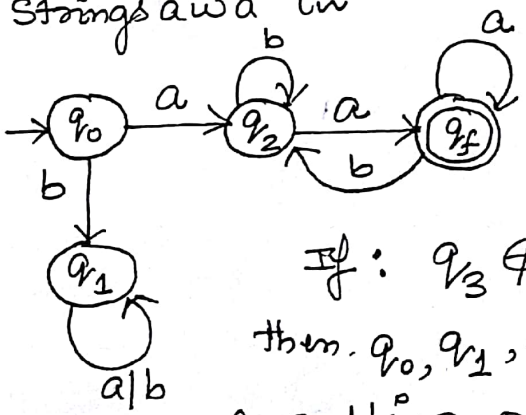
(f) $L = \{awa : w \in \{a,b\}^*\}$

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

(g) $L = \{a b^n a^m : n \geq 2, m \geq 3\}$



Strings awa in



If: $q_3 \notin F$

then $q_0, q_1, q_2 \in F$

resulting DFA accepts $\bar{L}$.

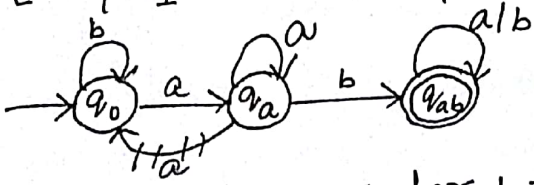
$M = (Q, \Sigma, q_0, F, \delta)$

& $\hat{M} = (Q, \Sigma, q_0, Q - F, \delta)$

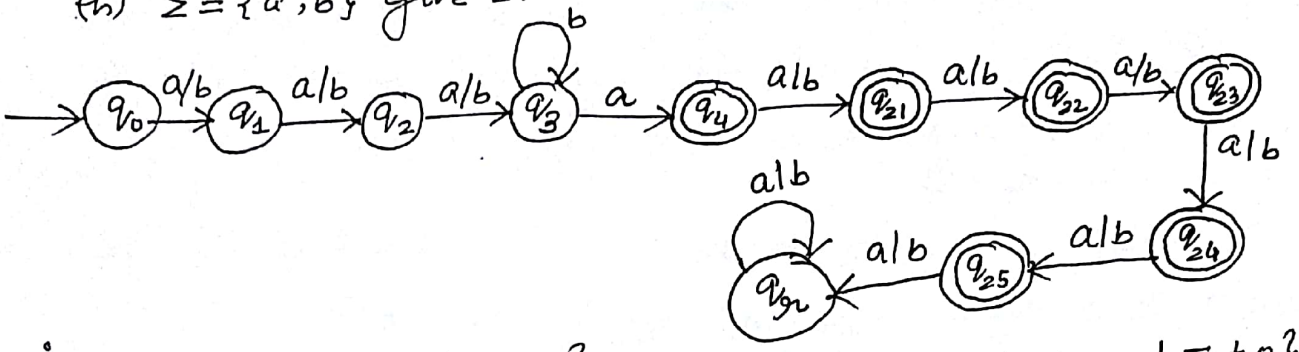
then $\overline{L(M)} = L(\hat{M})$.

02/02/18

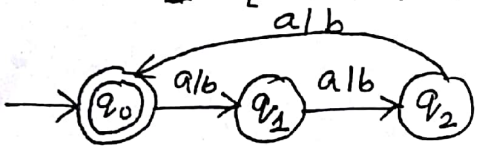
(g) $L = \{ w_1 ab w_2 : w_1 \in \{a,b\}^*, w_2 \in \{a,b\}^* \}$



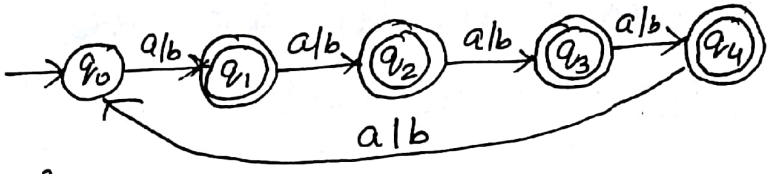
(h) $\Sigma = \{a, b\}$ give DFA for $L = \{ w_1 a w_2 : |w_1| \geq 3, |w_2| \leq 5 \}$



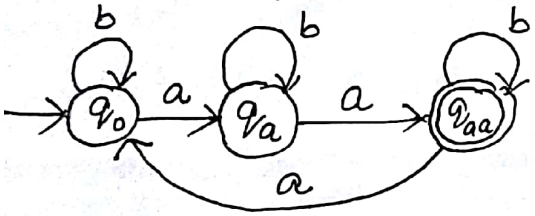
(i) $L = \{ w : |w| \bmod 3 = 0 \}$



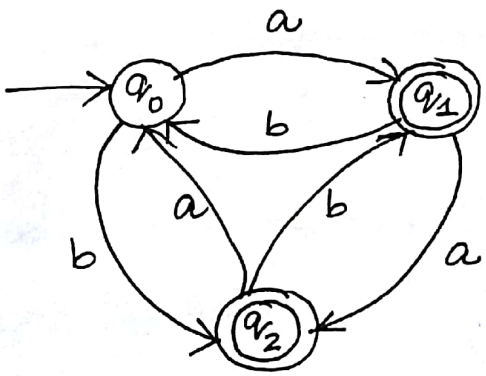
(j) $L = \{ w : |w| \bmod 5 \neq 0 \}$



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



(l) $L = \{ w : (n_a(w) - n_b(w)) \bmod 3 > 0 \}$



02/02/18 **Nondeterministic Finite Automata (NFA)**

- An NFA is a five-tuple: $M = (Q, \Sigma, q_0, F, \delta)$, where

Q: A finite set of states
Σ: A finite set of symbols (Alphabet set)
q_0: The initial/starting state, $q_0 \in Q$.
F: A set of final/accepting states, $F \subseteq Q$.
δ: A transition function, which is a total function

$$Q \times \Sigma \rightarrow 2^Q$$

$\delta : (Q \times \Sigma) \rightarrow 2^Q$, 2^Q is the power set of Q,
$\delta(q, s)$: The set of all states p such that there is transition labeled s from q to p. $s \in \Sigma$

- Let $M = (Q, \Sigma, \delta, q_0, F)$ be an NFA and let ω be in Σ^*. Then ω is accepted by M iff $\delta(\{q_0\}, \omega)$ contains at least one state in F.
- Let $M = (Q, \Sigma, q_0, F, \delta)$ be an NFA. Then the language accepted by M is the set:

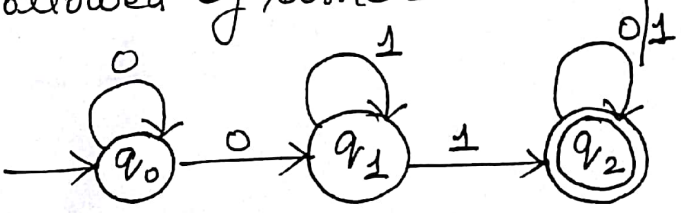
$$L(M) = \{\omega \mid \omega \in \Sigma^* \text{ and } \delta(\{q_0\}, \omega) \text{ contains at least one state in } F\}$$

- Another Equivalent definition:

$$L(M) = \{\omega \mid \omega \in \Sigma^* \text{ and } \omega \text{ is accepted by } M\}.$$

Example: Some 0's followed by some 1's.

$Q = \{q_0, q_1, q_2\}$
$\Sigma = \{0, 1\}$
Start state is q_0
$F = \{q_2\}$

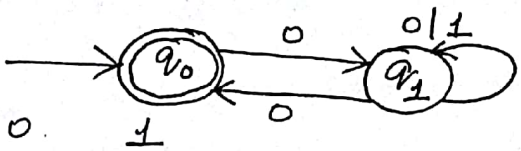


δ:

	0	1
q_0	$\{q_0, q_1\}$	$\{\}$
q_1	$\{\}$	$\{q_1, q_2\}$
q_2	$\{q_2\}$	$\{q_2\}$

05) 02/02/18 Ex: Empty string or start and end with 0.

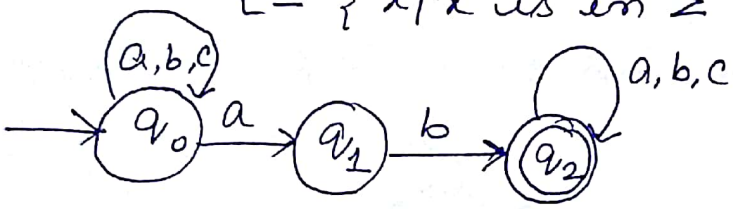
$Q = \{q_0, q_1\}$
$\Sigma = \{0, 1\}$
Initial state: q_0
$F = \{q_1\}$



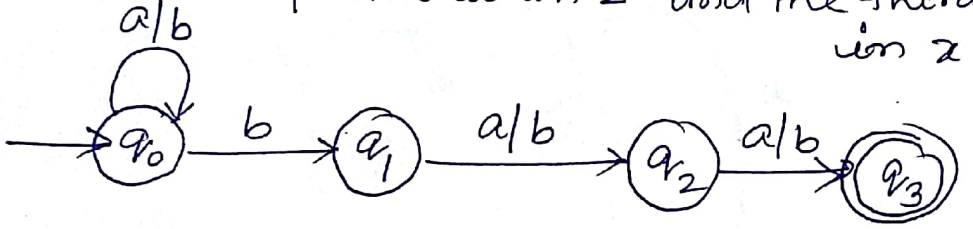
δ:

	0	1
q_0	$\{q_1\}$	$\{\}$
q_1	$\{q_0, q_1\}$	$\{q_1\}$

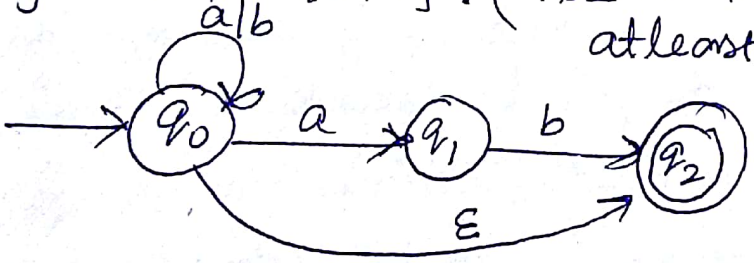
Ex: Let $\Sigma = \{a, b, c\}$. Give an NFA M that accepts:
$L = \{x \mid x \text{ is in } \Sigma^* \text{ and } x \text{ contains } ab\}$



Ex: Let $\Sigma = \{a, b\}$. Give an NFA M that accepts
$L = \{x \mid x \text{ is in } \Sigma^* \text{ and the third-to-the last symbol in } x \text{ is } b\}$



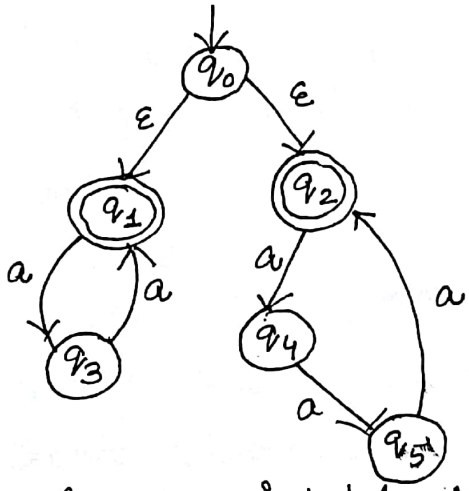
Ex: Let $\Sigma = \{a, b\}$. Give an NFA that accepts language defined is $\{a, b\}^*$. (Assume that it must contains atleast a and b)



05/02/18. # NFA's with ε-Transitions

- We extend the class of NFAs by allowing instantaneous(ε) transitions:
  1. The automaton may be allowed to change its state without reading the input symbol.
  2. In diagrams, such transitions are depicted by labelling the appropriate arcs with ε.
  3. Note that this does not mean that 'ε' has become an input symbol. On the contrary, we assume that the symbol ε does not belong to any alphabet.

Example: $L = \{a^n \mid n \text{ is even or divisible by } 3\}$



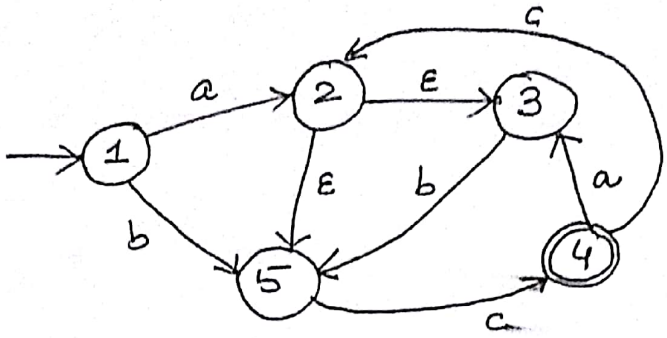
- A ε-NFA is a quintuple $M = (Q, \Sigma, \delta, q_0, F)$, where
  - Q is the set of states
  - Σ is the alphabet of input symbols
  - $q_0 \in Q$ is the initial/starting state
  - $F \subseteq Q$ is the set of final states
  - $\delta : Q \times \Sigma_\varepsilon \longrightarrow P(Q)$ is the transition function
- Note 'ε' is never a member of Σ.
- Σ_ε is defined to be $(\Sigma \cup \{\varepsilon\})$

ε-NFA

- ε-NFAs add a convenient feature but (in sense) they bring us nothing new: they do not extend the class of languages that can be represented. Both NFAs and ε-NFAs recognize exactly the same languages.

05/02/18

• Ex:



This ε-NFA accepts strings such as ac, abc, ...

Equivalence of NFA to DFA

Problem Statement: Let $X=(Q_x, \Sigma, q_0, F_x, \delta_x)$ be an NFA which accepts the language $L(X)$. We have to design an equivalent DFA $Y=(Q_Y, \Sigma, q_0, F_Y, \delta_Y)$ such that $L(Y)=L(X)$. The following procedure converts the NFA to its equivalent DFA —

Algorithm:

Input: An NFA
Output: An equivalent DFA

Step 1: Create state transition table from the given NFA.

Step 2: Create a blank state table under possible input alphabets for equivalent DFA.

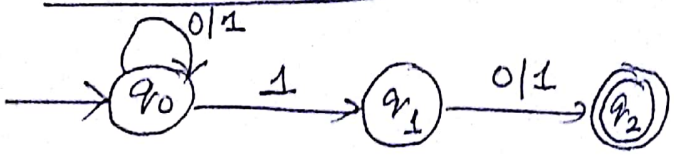
Step 3: Mark the start state of the DFA by q_0 (Same as the NFA)

Step 4: find out the combination of States $\{q_0, q_1, \ldots, q_n\}$ for each input alphabet symbol.

Step 5: Each time we generate a new DFA state under the ~~state~~ input alphabet symbol columns, we have to repeat step 4 again, otherwise go to step 6.

Step 6: The States which contain any of the final states of the NFA are the final states of the equivalent DFA.

05/02/18 Ex: NFA to DFA



For NFA

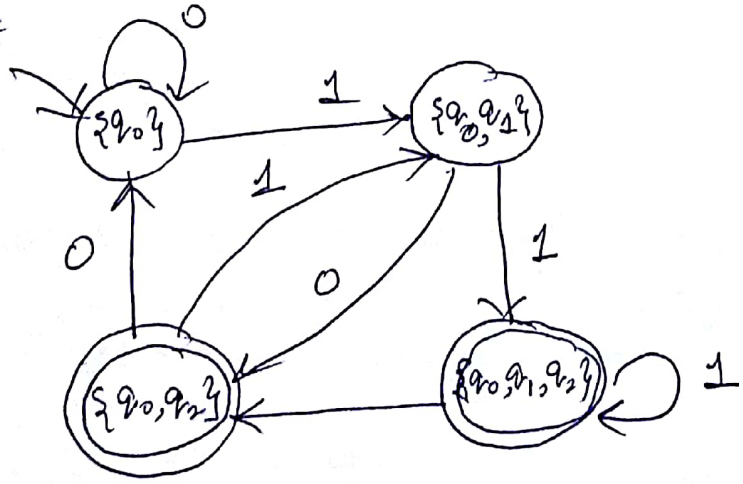
δ:

	0	1
q_0	$\{q_0\}$	$\{q_0, q_1\}$
q_1	$\{q_2\}$	$\{q_2\}$
q_2	$\{\}$	$\{\}$

For DFA

States	0	1
$\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$
$\{q_0, q_2\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$

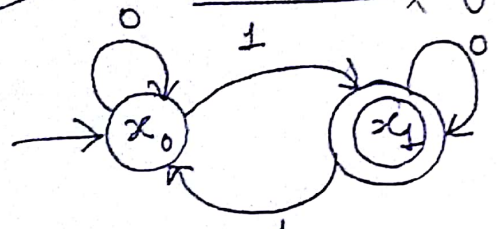
Equivalent DFA



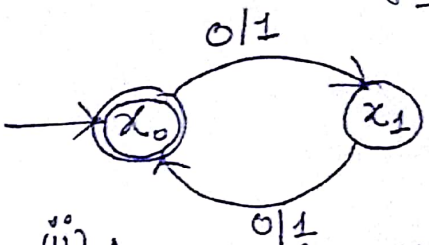
06/02/18

Some DFAs regarding restrictions on length

for reference



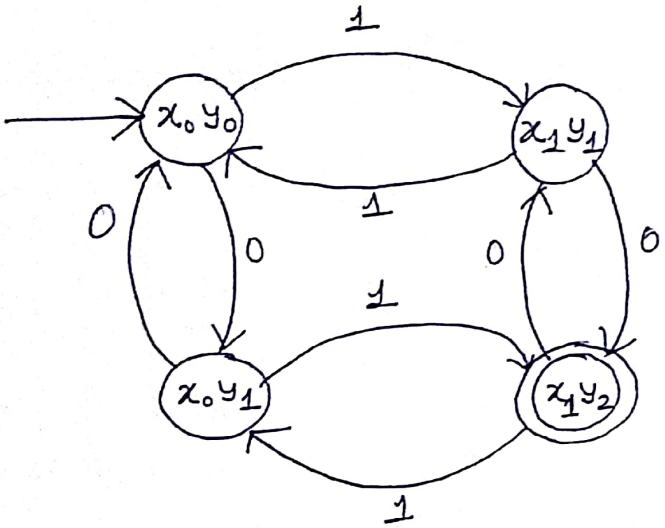
(i) An odd number of 1's



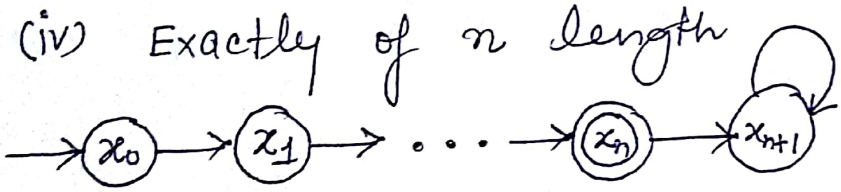
(ii) An even length

Combination

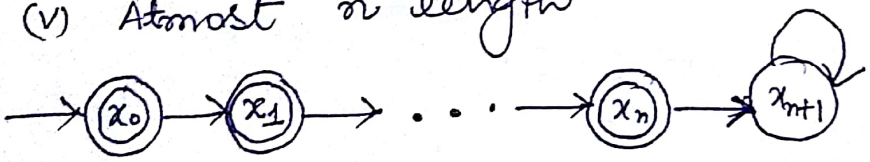
(iii) An odd number of 1's and even length



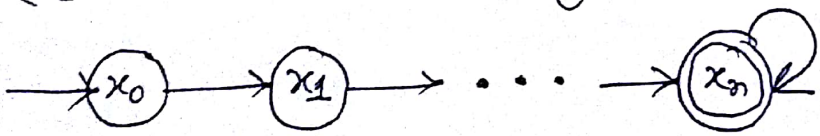
(iv) Exactly of n length



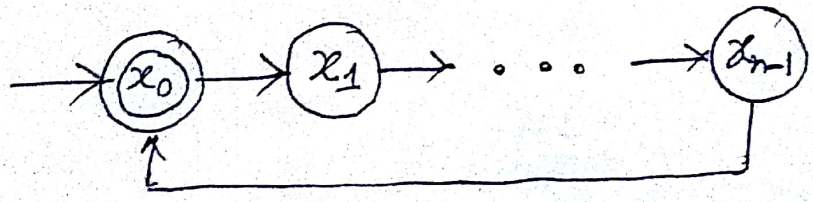
(v) Atmost n length



(vi) At least n length (@least)



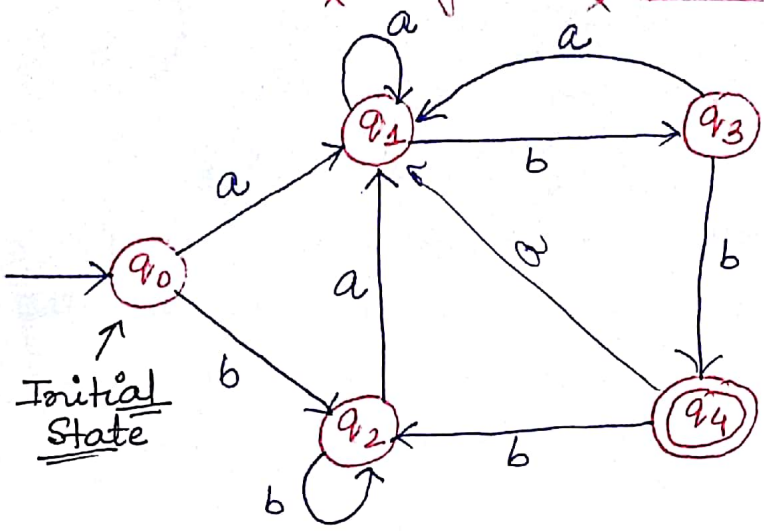
(vii) Divisble by n



15/02/18

Minimization of DFA by Partition Method

Ex:



$Q = \{q_0, q_1, q_2, q_3, q_4\}$

Divide Q in

final State $\{q_4\}$

Non-final State $\{q_0, q_1, q_2, q_3\}$

$\delta : Q \times \Sigma \to Q$

Transition Table

	a	b
→ q_0	q_1	q_2
q_1	q_1	q_3
q_2	q_1	q_2
q_3	q_1	$*q_4$
$*q_4$	q_1	q_2

Step 1:

$\pi_0 \to \{q_4\}\ \{q_0, q_1, q_2, q_3\}$

Step 2:

$\pi_1 \to \{q_4\}\ \{q_0, q_1, q_2\}\ \{q_3\}$

$q_2 \xrightarrow{a} q_1$

$q_2 \xrightarrow{b} q_2$

$q_0 \xrightarrow{a} q_1$ | $q_0 \xrightarrow{b} q_2$ — NF ✗

$q_3 \xrightarrow{a} q_1$ | $q_3 \xrightarrow{b} q_4$ F

	a	b
→ $[q_0, q_2]$	$\{q_1\}$	$\{q_0, q_2\}$
$\{q_1\}$	$\{q_1\}$	$\{q_3\}$
$\{q_0, q_2\}$	$\{q_1\}$	$\{q_0, q_2\}$
$\{q_3\}$	$\{q_1\}$	$\{q_4\}$
$\{q_4\}$	$\{q_4\}$	$\{q_0, q_2\}$

Step 3:

$\pi_2 \to \{q_4\}\ \{q_3\}\ \{q_0, q_2\}\ \{q_1\}$

$q_0 \xrightarrow{a} q_1$ ✓

$q_1 \xrightarrow{a} q_1$ ✓

$q_0 \xrightarrow{b} q_2$

$q_1 \xrightarrow{b} q_3$ ✗

Step 4:

$\pi_3 \to \{q_4\}\ \{q_3\}\ \{q_1\}\ \{q_0, q_2\}$

No need of separation

$q_0 \xrightarrow{a} q_2$ ✓

$q_2 \xrightarrow{a} q_1$ ✓

$q_0 \xrightarrow{b} q_2$ ✓

$q_2 \xrightarrow{b} q_2$ ✓

Stopping condition: output of $\pi_n = \pi_{n+1}$

	a	b
→ D	C	D ✓
C	C	B ✓
Same D	C	D
B	C	*A ✓
*A	C	D ✓

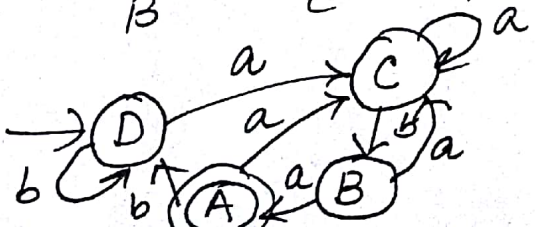
only once

$\{q_4\}, \{q_3\}, \{q_1\}, \{q_0, q_2\}$

↓A ↓B ↓C ↓D

$q_0 \to [q_0, q_2]$

$q_2 \to [q_0, q_2]$



Minimal-DFA

19/02/18

Distinguishing One String from Another

Def^n: Let L be a language in Σ^* i.e., $L \in \Sigma^*$.
Two strings x and y in Σ^* are distinguishable w.r.t. to L if there is a string $z \in \Sigma^*$ (which may depend on x and y) so that exactly one of the strings xz and yz is in L.
The string z is said to distinguish x and y with respect to L.

- we may say that x and y are indistinguishable with respect to L if there is no such string z;
- In other words, if for every z, both xz and yz have the same status — either both in L or both not in L.

e.g. $L = \{x \in \{0,1\}^* \mid x \text{ ends with } 10\}$

the strings 01011 and 100 are distinguishable w.r.t. L, because for $z = 0$, $01011z \in L$ and $100z \notin L$.

The strings 0 and 100 are indistinguishable with respect to L.

Lemma: Suppose that $L \subseteq \Sigma^*$ and $M = (Q, \Sigma, q_0, F, \delta)$ is any FA recognizing L.
If x and y are two strings in Σ^* for which $\delta^*(q_0, x) = \delta^*(q_0, y)$, then x and y are indistinguishable with respect to L.

Proof: Let z be any string in Σ^*, and consider strings xz and yz.

$$\delta^*(q_0, xz) = \delta^*(\delta^*(q_0, x), z)$$

$$\delta^*(q_0, yz) = \delta^*(\delta^*(q_0, y), z)$$

and therefore, by our assumption,

$$\delta^*(q_0, xz) = \delta^*(q_0, yz).$$

Since, M is assumed to recognize L, these two strings are either both in L or both not in L.

Therefore, x and y are indistinguishable w.r.t. L.

E.g. $L \subseteq \{0,1\}^*$: strings ending with 11.

Here string $z = 1$, distinguish strings 011 (x) and 001 (y), because $xz \in L$ i.e. $011z \in L$ and $yz \notin L$ i.e. $001z \notin L$.

Recursive Definitions of δ^*

1. $\delta^*(q, ax) = \delta^*(\delta(q, a), x)$

   $\delta^*(q, \lambda) = q$

2. $\delta^*(q, xa) = \delta(\delta^*(q, x), a)$

   $\delta^*(q, \lambda) = q$

E.g.: $\delta^*(q_0, aab) = \delta(\delta^*(q_0, aa), b) = \delta(\delta(\delta^*(q_0, a), a), b) =$

$\delta(\delta(\delta(\delta^*(q_0, \lambda), a), a), b) = \delta(\delta(\delta(q_0, a), a), b) =$ ...

20/02/18 **Regular Expressions**

Def^n: A regular expression is recursively defined as follows:

1. ϕ is a regular expression denoting an empty language.
2. ε-(epsilon) is " " indicates the language containing an empty string.
3. a is a regular expression which indicates the language containing only $\{a\}$.
4. If α and β are regular expressions, then
   (a) $\alpha+\beta$ is a R.E. corresponding to the language $L_\alpha \cup L_\beta$.
   (b) $\alpha \cdot \beta$ " " " " " " $L_\alpha \cdot L_\beta$.
   (c) α^* " " " " " " L_α^*.

Note — Nothing else is regular expression.

A Regular Language is represented by regular expression.

- The set R on an alphabet Σ is regular language defined as follows:

$\phi \subseteq R$, $\{\varepsilon\} \subseteq R$, $\{a\} \subseteq R \ \forall\ a \in \Sigma$

if $L_1, L_2 \subseteq R$, then $L_1 \cup L_2 \subseteq R$, $L_1 \cdot L_2 \subseteq R$ and $L_1^* \subseteq R$.

Ex: The Language $\{0, 00, 01, 000, 001, 010, 011, 0000, \ldots\}$ which consists all binary strings is regular. It can be constructed using regular expression $\{0\} \cdot (\{0\} \cdot \{1\})^*$.

Note: Expressions obtained by applying any of the rules from 1–4 are regular expressions.

20/02/18

Regular expressions	Meaning
$(a+b)^*$	set of strings of a's and b's of any length including the NULL string.
$(a+b)^*abb$	set of strings of a's and b's ending with the string abb
$ab(a+b)^*$	Set of strings of a's and b's starting with the string ab.
$(a+b)^*aa(a+b)^*$	set of strings of a's and b's having a substring aa.
$a^*b^*c^*$	Set of string consisting of any number of a's (may be empty string also) followed by any number of b's (may include empty string) followed by any number of c's (may include empty string).
$a^+b^+c^+$	set of string consisting of at least one 'a' followed by string consisting of at least one 'b' followed by string consisting of at leat one 'c'.
$aa^*bb^*cc^*$	" as above "
	" " "
$(a+b)^*(a+bb)$	set of strings of a's and b's ending with either a or bb
$(aa)^*(bb)^*b$	set of strings consisting of even number of a's followed by odd number of b's.
$(0+1)^*000$	set of strings of 0's and 1's ending with three consecutive zeros
$(11)^*$	Set of strings consisting of even number of 1's.