Regular Languages: outline

- Regular grammars
- Automata
  - Finite State Automata (FSA)
- Regular expressions
- Minimization of DFSA
- Identification of non-regular languages
  - Pumping Lemma for Regular languages

Regular Grammars

- Definition: a PSG $G = (N, T, P, S)$ is regular provided that:
  1. If there exists a λ-production, then it is of the form $S \rightarrow \lambda$ and S does not appear on the right hand side of any production.
  2. All other productions are of the form:
     - $\alpha \rightarrow \beta$, $\alpha \in N$, $\beta \in T$, OR
     - $\alpha \rightarrow \beta\gamma$, $\alpha, \gamma \in N$, $\beta \in T$
- A language generated from a regular grammar is called a **Regular Language**.
- Ex1. G_1:

$$S \rightarrow aS \mid aB$$

$$B \rightarrow bB \mid b$$

Is G_1 a regular grammar?

Regular Grammars: cont'd

- Ex2. Write a grammar G_2 such that $L(G_2) = L(G_1) \cup \{\lambda\}$
- <u>Theorem</u>: Given a λ-free regular grammar $G = (N, T, P, S)$ such that $L = L(G)$, we can construct a λ-free grammar $G^+ = (N, T, P^+, S)$ such that $L(G^+) = L^+$
- <u>Theorem</u>: If L_1 and L_2 are two regular languages, then:
  a. $L_1 \cup L_2$ is also a regular language
  b. L_1L_2 is also a regular language
  c. $L_1{}^*$ is also a regular language

Transition diagrams

- A regular grammar G=(N, T, P, S) can be represented by a transition diagram (a directed graph with labeled arcs as follows:
  - The nodes of the graph are to contain non-terminals
  - The arcs are labeled with terminals
  - One of the nodes is an **initial node** which is designated with a pointer
  - One (or more) of the nodes is designated as **final node** which is either a square or a double circle
  - If A→aB is in P, then the arc from A to B is labeled with a
  - If A→a is in P, then the arc from A to a final state is labeled with a

Transition diagrams: cont'd

- Example:

  Let G=(N, T, P, S) be a regular grammar with

  P: $S \rightarrow aA|bB$

  $A \rightarrow aA|a$

  $B \rightarrow bB|b$

  Draw a transition diagram that represents G

Automata

- Abstract machines

  Characteristics
  - **Input**: input values (from an input alphabet ∑) are applied to the machine
  - **Output**: outputs of the machine
  - **States**: at any instant, the automation can be in one of the several states
  - **State relation**: the next state of the automation at any instant is determined by the present state and the present input
  - **Output relation**: output is related to either state only or to both the input and the state

Automata: cont'd

- Types of automata
  - Final State Automata (FSA)
    - Deterministic FSA (DFSA)
    - Nondeterministic FSA (NFSA)
  - Push Down Automata (PDA)
    - Deterministic PDA (DPDA)
    - Nondeterministic PDA (NPDA)
  - Turing Machine (TM)

Deterministic FSA (DFSA)

- Definition

  A DFSA is a 5-tuple $M=(Q, \Sigma, t, q_o, F)$ where:

  i. Q is a finite set of states

  ii. Σ is an input alphabet

  iii. t is a, possibly partial, function

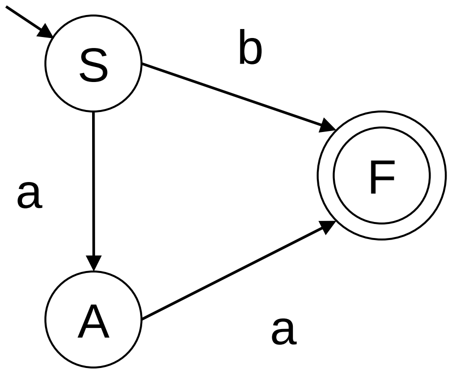
  $$t: Q X \Sigma \rightarrow Q$$

  iv. $qo \in Q$ is the initial state

  v. $F \subseteq Q$ is a finite set of final states

DFSA: cont’d

- Example:



M=(Q, ∑, t, q_o, F)
Q={S, A, F}
∑={a, b}
Q_o=S
F={F}<u>C</u>Q

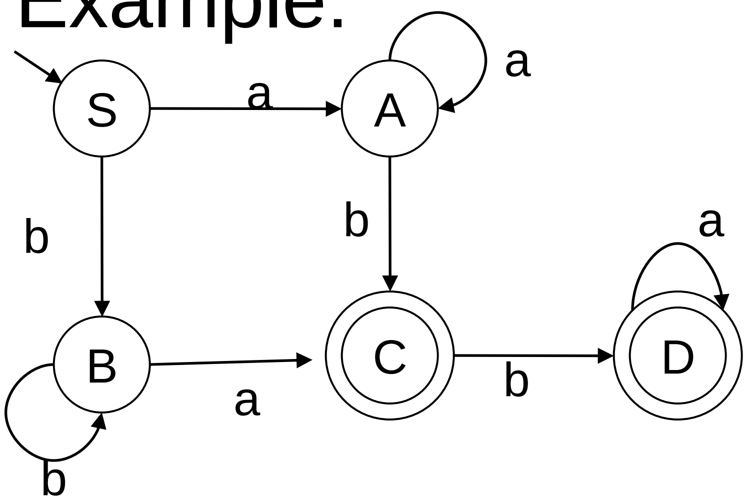
<u>State</u>	<u>Input</u>	<u>Next state</u>
S	a	A
S	b	F
A	a	F

DFSA: cont'd

- Acceptance of strings by DFSA
  - A string w in Σ^* is accepted by a DFSA M if
    - There exists a path which originates from some initial state, goes along the arrows, and terminates at some final state, and
    - The path value obtained by concatenation of all edge-labels of the path is equal to w.

DFSA: cont'd

- Example:



Q = {S, A, B, C, D}
∑ = {a, b}
qo = S
F = {C, D}

State	Input	Next state
S	a	A
S	b	B
A	a	A
A	b	C
B	b	B
B	a	C
C	b	D
D	a	D

Check whether the following strings are accepted or not:

- ab
- ba
- bbaba
- aa
- aaabbaaa

DFSA: cont'd

- Extensions of t

  t needs to be extended as:

  t: Q X Σ* → Q

  let x ε Σ*, then

  i) x = λ

  ii) x = ay, a ε Σ, y ε Σ*

  Then i) t(q, λ) = q

  ii) t(q, x) = t(q, ay)

  = t(t(q, a), y)

  Eg. a) t(S, ab)

  b) t(S, bbba)

  c) t(S, aaa)

DFSA: cont'd

- Given DFSA M, the set of strings accepted by M is given by L(M)

  $L(M) = \{x \varepsilon \Sigma^* \mid t(qo,x) \varepsilon F\}$

Nondeterministic FSA (NFSA)

- Definition

  An NFSA is a 5-tuple $M=(Q, \sum, t, q_o, F)$ where:

  i. Q is a finite set of states

  ii. $\sum$ is an input alphabet

  iii. t is a, total, function

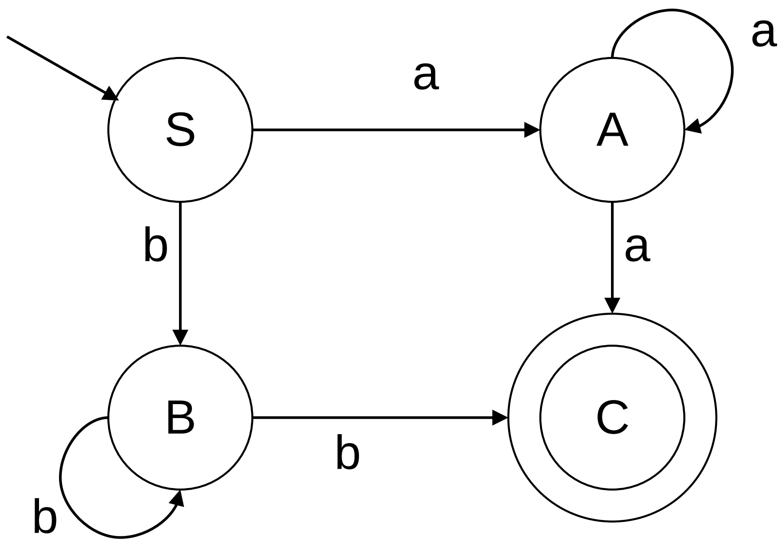
  $$t: Q X \sum \rightarrow 2^Q$$

  iv. $q_o \in Q$ is the initial state

  v. $F \subseteq Q$ is a finite set of final states

NFSA: cont'd

Example



State	Input	Next state(s)
S	a	{A}
S	b	{B}
A	a	{A, C}
A	b	{}
B	a	{}
B	b	{B, C}
C	a	{}
C	b	{}

The strings aa, bbb are both accepted by the NFSA.

NFSA: cont'd

String Acceptance by NFSA

- Extensions of t:

1. t: $Q \times \Sigma \rightarrow 2^Q$ to t: $2^Q \times \Sigma \rightarrow 2^Q$ by defining

   $t(Q', a) = \bigcup t(q, a), q \in Q', Q' \subseteq Q, a \in \Sigma$

1. t: $2^Q \times \Sigma \rightarrow 2^Q$ to t: $2^Q \times \Sigma^* \rightarrow 2^Q$

   let $x \in \Sigma^*$, then

   i) $x = \lambda \Rightarrow t(Q',x) = Q'$

   ii) $x = ay, a \in \Sigma, y \in \Sigma^*$

   $\Rightarrow t(Q', x) = t(Q', ay) = t(t(Q', a), y)$

<u>Example</u>: Evaluate the following using the NFSA in the previous slide

a) t(S, aaa)

b) t(S, aaba)

NFSA: cont'd

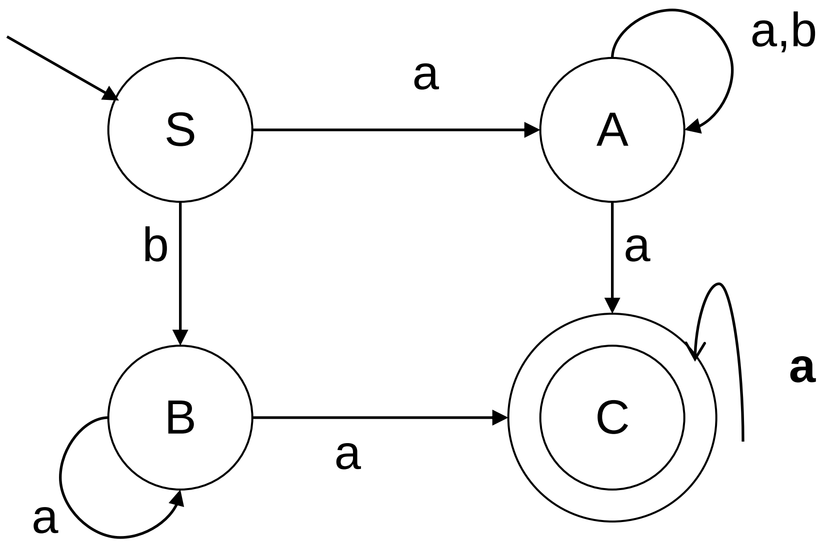
- Let M be an NFSA,

  L(M) = set of strings accepted by M

  $= \{x \in \Sigma^* \mid t(q_o, x) \cap F \neq \emptyset\}$

NFSA: cont'd

Example



State	Input	Next state(s)
S	a	{A}
S	b	{B}
A	a	{A, C}
A	b	{A }
B	a	{B,C}
B	b	{ }
C	a	{C}
C	b	{ }

Consider the above NDFSA, find

i) t({A,C},abba)
ii) Let x=aab

Equivalence of DFSA and NFSA

- Theorem: Let L be a language. L is accepted by a DFSA iff L is accepted by NFSA.

  proof:

  i) (=>): L is accepted by DFSA=>L is accepted by NDFSA
  ii) (<=): L is accepted by NFSA=>L is accepted by DFSA

  Any DFSA can have a total function by introducing a dummy state such that all undefined transitions are defined to that state

- i) Given a DFSA with a partial function,it is possible to convert it to a total function as follows:
- i) set $t(q,a)=\Delta$, a dummy state,where $t(q,a)$ is undefined
- ii) $t(\Delta,a)= \Delta$, for each $a \in \Sigma$
- Now assume that $M=(Q, \Sigma,t,q0,F)$ with a total function and construct NFSA

$M'(Q, \Sigma,t',q0,F)$ such that $L(M)=L(M')$

- Define

  $t'(q,x)=\{t(q,x)\}$

ii) (<=) L is accepted by NFSA => L is accepted by DFSA.

Equivalence: cont'd

Let $M = (Q, \sum, t, q_o, F)$ be NFSA, We construct DFSA M' such that $L(M)=L(M')$

1. Start {qo} and calculate t({qo},a), for all $a \in \sum$.

   ie. Obtain possible states that are reachable from qo, say K.

1. Calculate $t(K,a)$ for all $a \in \sum$
2. Repeat this process until no new subsets of Q are constructed

Thus, $M' = (Q', \sum, t', q_o', F')$ where

Q' = all subsets of Q reachable from q_o

$q_o' = \{q_o\}$

$F' = K' \underline{C}\ Q$ such that $K' \text{ n } F \neq \emptyset$

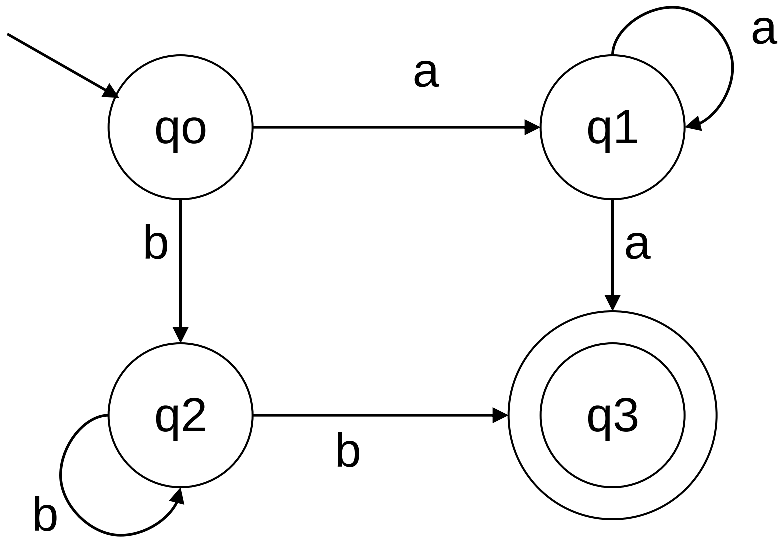
t' is an extension of t such that $t: 2^Q \times \Sigma \rightarrow 2^Q$

$L(M) = L(M')$

<u>Illustration</u>:

- Example:

Given the following NFSA, Construct its equivalent DFSA.


a
qo
a
q1
b
a
q2
b
q3
b

- Solution:
- Find reachable states from qo.

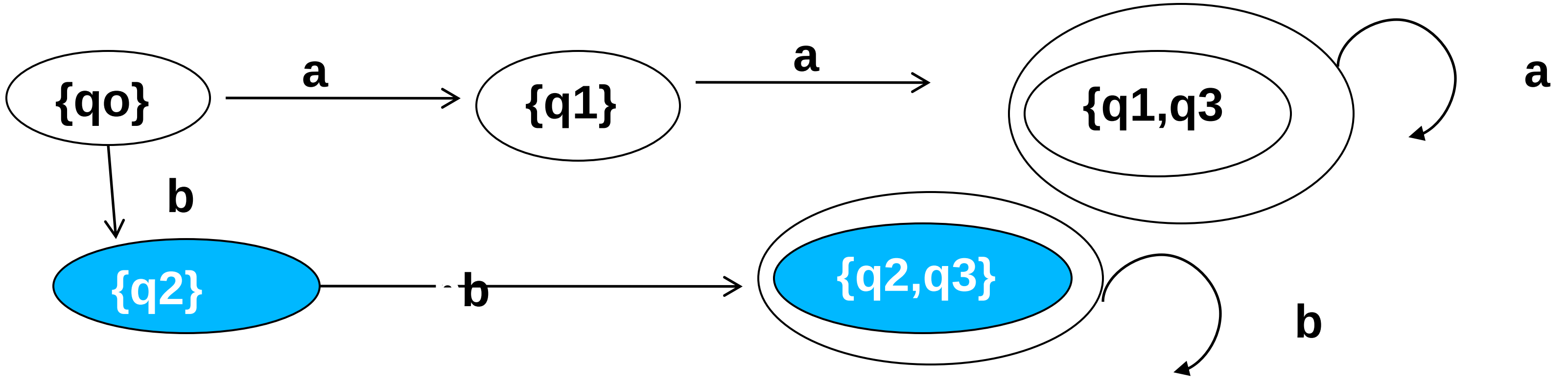
	t(q,a)	t(q,b)
State	a	b
{qo}	{q1}	{q2}
{q1}	{q1,q3}	{}
{q2}	{}	{q2,q3}
{q1,q3}	{q1,q3}	{}
{q2,q3}	{}	{q2,q3}

- Hence M'=(Q', Σ,t',qo',F')

Where Q'={{qo}, {q1}, {q2},{q1,q3},{q2.q3}}

qo={q0}

F' ={{q1,q3},{q2,q3}} and t is given by:

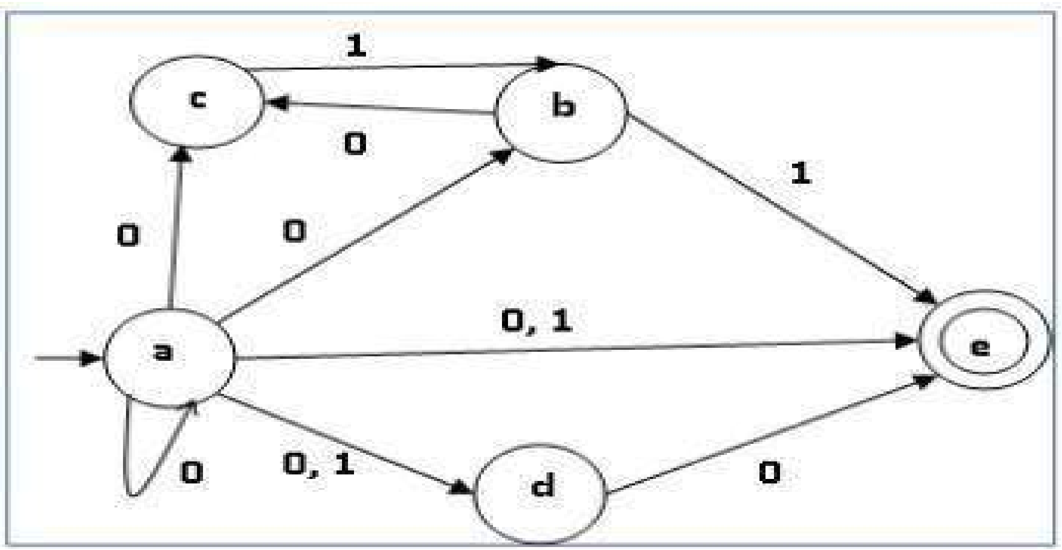


Example

The NDFA table is as follows −

q	δ(q,0)	δ(q,1)
a	{a,b,c,d,e}	{d,e}
b	{c}	{e}
c	∅	{b}
d	{e}	∅
e	∅	∅

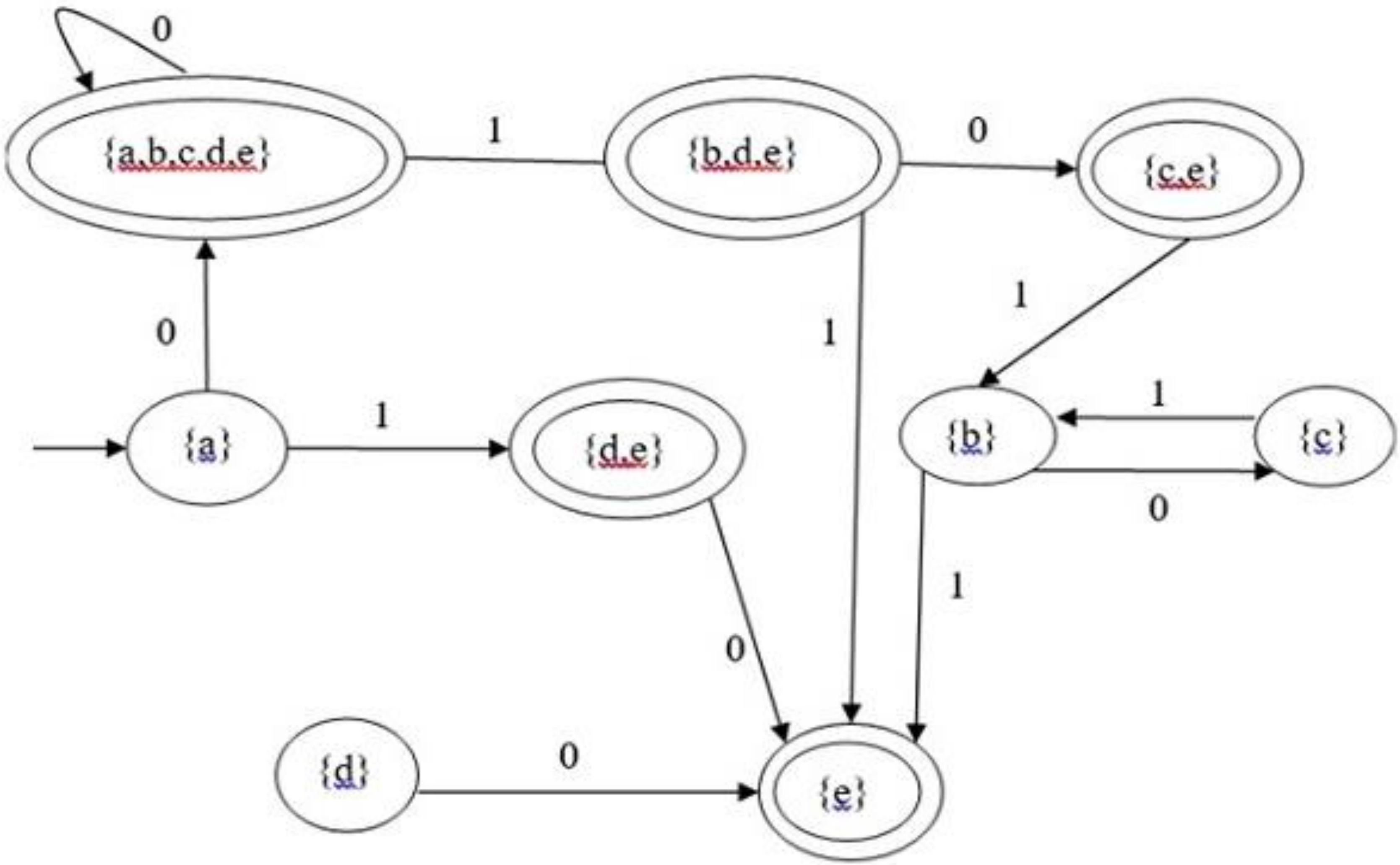
Let us consider the NDFA shown in the figure below.



Using above algorithm, we find its equivalent DFA. The state table of the DFA is shown in below.

q	δ(q,0)	δ(q,1)
a	{a,b,c,d,e}	{d,e}
{a,b,c,d,e}	{a,b,c,d,e}	{b,d,e}
{d,e}	e	∅
{b,d,e}	{c,e}	E
e	∅	∅
d	e	∅
{c,e}	∅	B
b	c	E
c	∅	B

The state diagram of the DFA is as follows −



Equivalence: cont'd

- Theorem: The following statements are equivalent:

  i. L is accepted by NFSA
  ii. L is accepted by DFSA
  iii. L is generated by a regular grammar

Proof: (i) => (ii) => (iii) => (i)

(i) => (ii) proved in the previous theorem.

(ii) => (iii)

Let L be a language accepted by DFSA M. Construct a regular grammar G such that L(M) = L(G)

Let M = $(Q, \sum, t, q_o, F)$

Construct of G = (N, T, P, S) as follows:

1. $N = Q$, $T = \sum$, $S = q_o$
2. P:

   qi → aqj, if t(qi, a) = qj

   qi → a, if t(qi, a) Є F

L(M) = L(G)

Equivalence: cont'd

(iii) => (i)

Let L be a language generated by regular grammar G. We want to construct NFSA M such that L = L(M) = L(G)

Let G = (N, T, P, S)

Construct M = (Q, ∑, t, q_o, F) as follows:

1. Q = N, ∑ = T, q_o= S, F = {xЄN| x → a Є P, a Є T}

   Q = N U {F_f}, F_f not in N

2. t is given by:

   if q_i → aq_j, then t({q_i}, a) n {q_j} ≠Ø

   q_i → a, then t({q_i}, a) n F ≠Ø

   L(M) = L(G)

Illustration:

Equivalence: cont'd

- Theorem: If L is a regular language, then so is L bar (L').

  proof:

  L is regular → there exists a DFSA M = (Q, ∑, t, q_o, F) such that L = L(M)

  x∈L → t(q_o, x) ∈ F

  x ∈L' → x is not in L → t(q_o, x) is not in F

  Hence, M' = (Q, ∑, t, q_o, Q\F) such that L' = L(M')

Equivalence: cont'd

- Theorem: If L_1 and L_2 are regular languages, then so is L_1 n L_2.

  proof:

  L_1 is regular → L_1' is regular

  L_2 is regular → L_2' is regular

  → L_1' U L_2' is regular

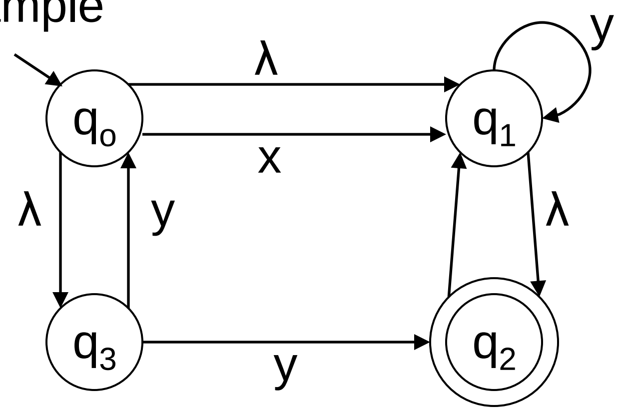
  → (L_1' U L_2')' is regular

  → L_1 n L_2 is regular

FSA with λ moves

- Definition: Let $M = (Q, \sum, t, q_o, F)$ be a FSA, M is said to be with λ-moves if $t{:}QX(\sum U\{\lambda\}) \rightarrow Q$

Example

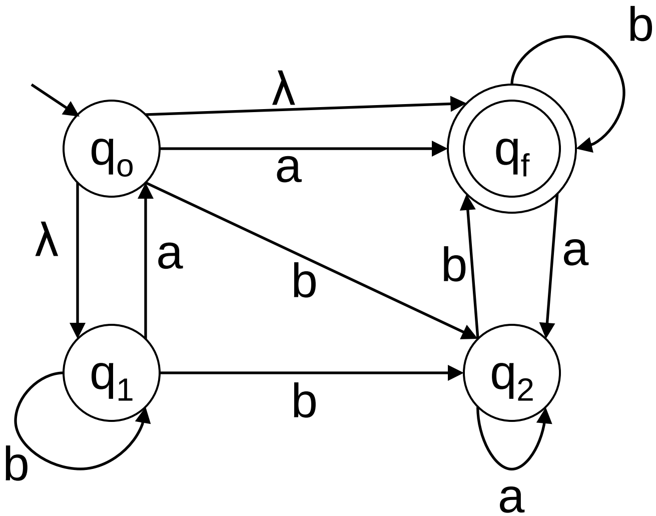


$R(q)$ = the set of λ-reachable states
For ex. $R(q_o) = \{q_o, q_1, q_2, q_3\}$

If $t(q_i, \lambda) = q_j$, then q_j is λ-reachable from q_i

For $Q' \underline{C} Q$, $R(Q') = UR(q'), q' \in Q'$

FSA with λ moves: cont'd



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

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

Let q, q' Є Q. If q' Є t(q, λ), then q' is λ-reachable denoted by $q \overset{*}{\underset{\lambda}{\rightarrow}} q'$

Let R(q) be the set of states that are reachable, $R(q) = \{q' \mid q \rightarrow q'\}$

If Q' $\subseteq$ Q then R(Q') = U R(q), q Є Q'

FSA with λ moves: cont'd

- Example: consider the previous NFSA

  R(qo) = ?

  R(q1) = ? R(q2) = ? R(q3) = ? R(qf) = ?

  R({qo, q1, q2, qf}) = ?

String acceptance by FSA with λ moves

- Extend t to t' such that
  t': Q X (∑U{λ}) → 2^Q defined as
  t'(q, λ) = R(q)
  t'(q, a) = UR(t(q', a)), q'ЄR(q), aЄ ∑U{λ}

  Ex. consider the previous NFSA
  t'(q_o, λ) = ?
  t'(q_o, a) = ?
  t'(q_o, b) = ?
- Extend t' such that t': 2^Q X (∑U{λ}) → 2^Q defined as
  t'(Q', λ) = R(Q'), Q' $\underline{C}$ Q
  t'(Q', a) = Ut'(q', a), q'ЄQ', a Є ∑

String acceptance: cont'd

- Extend t' such that t': 2^Q X ∑* → 2^Q defined as

  t'(Q', ax) = t'(t'(Q', a), x), a Є ∑, x Є ∑*

  Thus, L(M) = {x Є ∑*|t'(qo, x) n F ≠Ø}

  = set of strings accepted by NFSA with λ-moves

Cont'd

- Theorem: If L is accepted by NFSA with λ-moves, then L is a regular language.

  proof: It suffices to construct an equivalent NFSA without λ-moves

  Let $M = (Q, \sum, t, q_o, F)$ be NFSA with λ-moves

  To construct $M' = (Q, \sum, t', q_o, F')$, define

  $t'(q, a) = t(q, a)$, for all $a \in \sum$ and t is the 3rd extension

  $F' = F$ if $R(q_i) \cap F = \emptyset$

  $\quad = F \cup \{q_i\}$ otherwise

  $\quad L(M) = L(M')$

Cont'd

- <u>Example</u>: Construct an equivalent NFSA without λ-moves for the NFSA in the previous example.
  - For every state q, first check R(q) and calculate t(R(q), a) for a∈∑
  - Then check R(q') for the second time where q' is the result of t(R(q), a) for a∈∑

Regular Expressions

- Definition: A regular expression is a string over $\sum$ if the following conditions hold:
  1. λ, $\emptyset$, and $a \in \sum$ are regular expressions
  2. If α and β are regular expressions, so is $\alpha\beta$
  3. If α and β are regular expressions, so is $\alpha+\beta$
  4. If α is a regular expression, so is α^*
  5. Nothing else is a regular expression if it doesn't follow from (1) to (4)
- Let α be a regular expression, the language represented by α is denoted by $L(\alpha)$.

Regular Expressions: cont'd

- L satisfies:
  1. $L(\varnothing) = \varnothing$, $L(\lambda) = \{\lambda\}$, $L(a) = \{a\}$, $a \in \Sigma$
  2. If α and β are regular expressions, then:
     - $L(\alpha\beta) = L(\alpha)L(\beta)$
     - $L(\alpha+\beta) = L(\alpha)+L(\beta)$
     - $L(\alpha^*) = L(\alpha)^*$

- Example:

  $\alpha = a^*(b+c)$

  $\rightarrow L(\alpha) = L(a^*(b+c))$

  $= L(a^*)L(b+c)$

  $= L(a)^*(L(b) \cup L(c))$

  $= \{a\}^*(\{b\} \cup \{c\})$

  $= \{a\}^*(\{b,c\})$

Note: In the absence of parentheses, the hierarchy of operations is as follows: iteration, concatenation, and union.

Regular Expressions: cont'd

- Two regular expression P and Q are equivalent (P=Q) if P and Q represent the same set of strings.
- Identities for regular expressions
  - $\emptyset + R = R$
  - $\emptyset R = R\emptyset = R$
  - $\lambda^* = \lambda, \emptyset^*\lambda$
  - $R+R = R$
  - $R^*R^* = R^*$
  - $RR^* = R^*R$
  - $(R^*)^* = R^*$
  - $\lambda + RR^* = R^* = \lambda + R^*R$
  - $(PQ)^*P = P(QP)^*$
  - $(P+Q)^* = (P^*Q^*)^* = (P^*+Q^*)^*$
  - $(P+Q)R = PR + QR, R(P+Q) = RP + RQ$

Regular Expressions: cont'd

- Example: Give a regular expression for representing the set L of strings in which every 0 is immediately followed by at least two 1's.

- Theorem: If r is a regular expression, then L(r) is a regular language.

  proof: construct NFSA with λ-moves that accepts L(r).

  i. r = Ø, r = λ, r = a
  ii. r1 + r2
  iii. r1r2
  iv. r1*

  Let r be a regular expression, M(r) is a NFSA with λ-moves that accepts L(r)

Regular Expressions: cont'd

- Example: Construct NFSA that accepts $r=(a+b)^*ba(ba)^*$

- Exercise: Construct NFSAs equivalent to the following regular expressions:
  1. $(1+0)^*(00+11)(0+1)^*$
  2. $10+(0+11)0^*1$

Regular Expressions: cont'd

- Theorem (Arden's Theorem)

  Let P and Q be two regular expressions over Σ. If P does not contain λ, then the following equation in R,

  $R = Q + RP$

  has a unique solution given by $R = QP^*$

Regular Expressions: cont'd

- Algebraic method for finding the regular expression recognized by a FSA

  Assumptions

  1. No λ-moves
  2. There is only one initial state, say v_1
  3. Vertices are $v_1, \ldots, v_n$
  4. V_i is the regular expression representing the set of strings accepted by the system
  5. α_{ij} denotes the regular expression representing the set of labels of edges from v_i to v_j

Consequently, we can get the following set of equations in $v_1, \ldots, v_n$

$v_1 = v_1\alpha_{11} + v_2\alpha_{21} + \ldots + v_n\alpha_{n1} + \lambda$

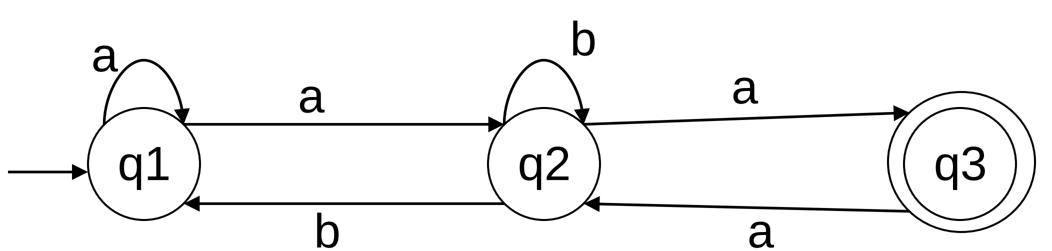
$v_2 = v_1\alpha_{12} + v_2\alpha_{22} + \ldots + v_n\alpha_{n2}$

…

$v_n = v_1\alpha_{1n} + v_2\alpha_{2n} + \ldots + {}_{vn}\alpha_{nn}$

Regular Expressions: cont'd

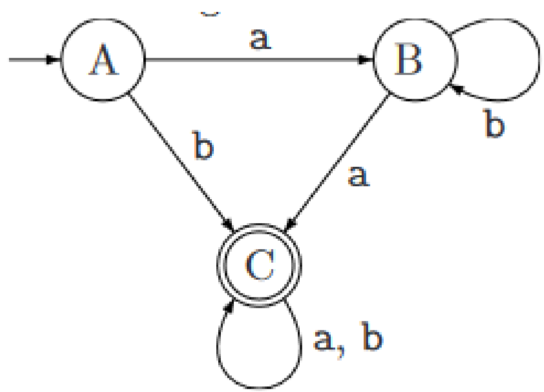
- By repeatedly applying substitution and Arden's Theorem, we can express v_i in terms of α_{ij}'s
- The set of strings recognized by the system is found by taking the 'union' of all v_i's corresponding to the final states.
- Example:



Find the regular expression equivalent to the given NFSA

Regular Expressions: cont'd

- Exercise: Find the regular expressions equivalent to the following NFSAs



Minimization of DFSA

- A DFSA with possible minimum states is called a minimal DFSA.
- Given M, DFSA that generates a language L, then we denote its minimal DFSA by M_L.

Construction of Minimal DFSA

Let M = (Q, ∑, t, q_o, F) be DFSA such that L = L(M)

To construct M_L, we proceed as follows:

I. Find all disjoint indistinguishable states of M

Define a number of relations D_o, D_1, …

- Given states q and q', q is distinguishable from q', by a string of length 0, denoted by

qDoq' iff either:

q∈F and q' not ∈F OR

q not ∈F and q'∈F

- If i>0, q is distinguishable from q' by a string of length ≤ i, denoted by

q D_i q' iff

qD_{i-1}q' OR

if there exists a∈∑ such that t(q, a)D_{i-1}t(q', a)

Minimization of DFSA: cont'd

II. $M_L = (Q_L, \Sigma, t_L, q_{oL}, F_L)$ where

Q_L= indistinguishable states

q_{oL}= indistinguishable states containing q_o

F_L = Final states of M

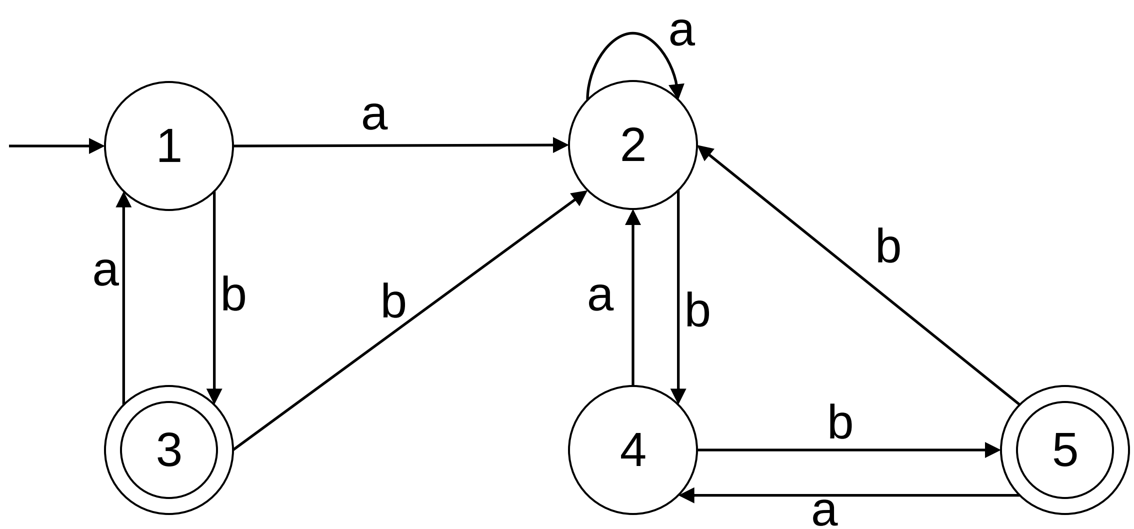
t_L is given by the following:

Let $Q' \subseteq Q$, where Q' is indistinguishable states

tL(Q', a) = Q'', indistinguishable states containing t(q, a), $q \in Q'$

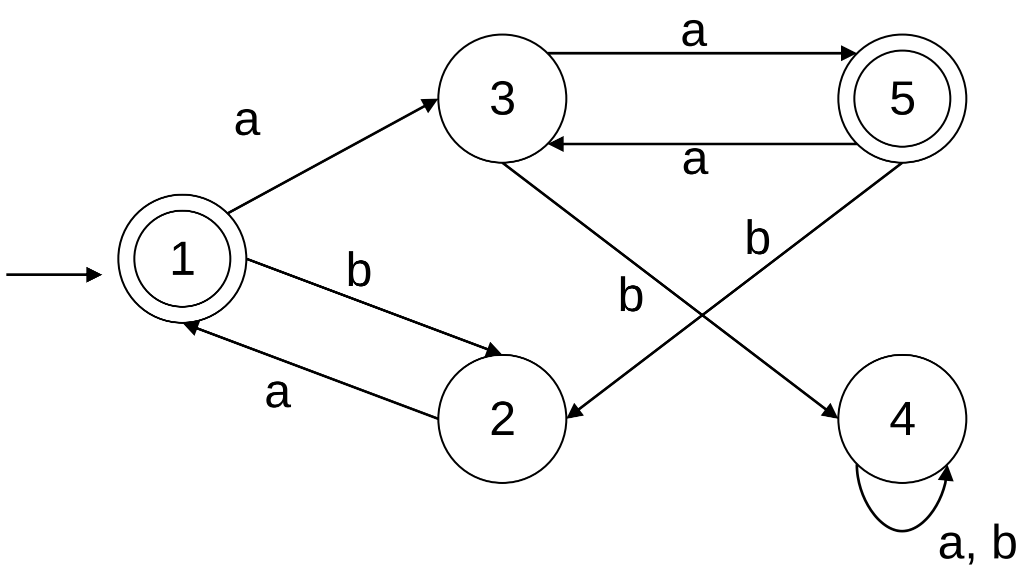
Minimization of DFSA: cont'd

- Illustration: Construct a minimal DFSA for the following DFSA



Minimization of DFSA: cont'd

- <u>Exercise</u>: Minimize the following DFSA



Identification of non-regular languages

- Lemma (Pumping lemma for regular languages)

  Let L be a regular language and w Є L. Then there exist substrings x, y, and z of w with:

  a. $w = xyz$, $|w| \geq m$
  b. $|xy| \leq m$, for some $m \in Z^+$
  c. $y \neq \lambda$

  such that $xy^nz \in L$ for $n \geq 0$

Non-regular languages: cont'd

- Example: Show that the following languages are not regular.

1. $L = \{a^n b^n \mid n \geq 0\}$
2. $L = \{a^n : n \text{ is prime}\}$
3. $L = \{ww^r \mid w \sum^*\}, \sum=\{a, b\}$