West 7: Lecture Notes

Minimization of FA

- $M = (Q, \Sigma, \delta, q, F)$
- $q_1 \equiv q_2$ if $\hat{\delta}(q_1, x)$, $\hat{\delta}(q_2, x)$ both $\in F$ $\forall x \in \Sigma^*$ or both $\notin F$
- not possible to check for all $x \in \Sigma^*$ as Σ^* has infinite number of elements
- k-equivalent

  q_1, q_2 are k-equivalent $(k \geq 0)$ if $\hat{\delta}(q_1, x)$, $\hat{\delta}(q_2, x)$ both $\in F$ or both $\notin F$ | for all strings x of length $\leq k$

  $\forall x \in \Sigma^*$ with $|x| \leq k$

Properties

equivalence → partition of Q → π

k-equivalence → partition of Q → π_k → both equivalence relation → induced partion of Q

π_k → elements in "π_k" are k-equivalence classes

- q_1, q_2 k-equivalent $\forall k \geq 0 \Rightarrow q_1, q_2$ are equivalent
- q_1, q_2 $(k+1)$-equivalent $\Rightarrow$ they are k-equivalent
- $\pi_n = \pi_{n+1}$ for some n, where π_n is the set of equivalence classes under n-equivalence.

Theorem

Two states q_1, q_2 are $(K+1)$-equivalent if

i. they are K-equivalent

ii. $\delta(q_1, a)$, $\delta(q_2, a)$ are also K-equivalent for every $a \in \Sigma$

Proof:

We prove this by contradiction.

Let q_1, q_2 are not $(K+1)$ equivalent

$\Rightarrow \exists\ w = aw_1 \in \Sigma^*$, $|w| = K+1$, $|w| = K$ such that

$$\hat{\delta}(q_1, w) \in F, \text{ but } \hat{\delta}(q_2, w) \notin F$$

i.e. $\hat{\delta}(\underbrace{\delta(q_1, a)}_{=q_3}, w_1) \in F$ but $\hat{\delta}(\underbrace{\delta(q_2, a)}_{=q_4}, w_1) \notin F$

$\Rightarrow q_3 = \delta(q_1, a)$, $q_4 = \delta(q_2, a)$ are not K-equivalent as w_1 is a string of length K.

which is a contradiction.

Construction of Minimum Automata (DFA)

1. Construction of π_0 → by definition of 0-equivalence

   $\pi_0 = \{ \theta_1^0, \theta_2^0 \}$ where

   θ_1^0 = the set of accepting states of DFA $M = (Q, \Sigma, \delta, q_0, F)$

   $\theta_2^0 = Q - \theta_1^0$

2. Construction of π_{k+1} from π_k

   Let $\theta_i^k \in \pi_k$

   - $q_1, q_2 \in \theta_i^k$ are $(k+1)$-equivalent if

     $\delta(q_1, a)$, $\delta(q_2, a)$ are in the same equivalence class of π_k for every $a \in \Sigma$

   - if so, θ_i^k is further divided into $(k+1)$-equivalence classes.
   - Repeat this for every $\theta_i^k \in \pi_k$ to get all the elements of π_{k+1}

3. Construct π_n for $n = 1, 2, \ldots$ until $\pi_n = \pi_{n+1}$

4. Construction of minimum automata

   - State table — states are equivalence classes obtained in step 3, i.e. the elements of π_n

     replace q by its equivalence class $[q]$

   - number of equivalence classes $\leq |Q|$

     Q → set of the states of F

- $[q_1] = \{q_1, q_2, \ldots q_k\}$ → an equivalence class
- let q_1 is reached while processing the string $w_1 w_2 \in L(M)$ with $\hat{\delta}(q_0, w_1) = q_1$

  Then $\hat{\delta}(q_1, w_2) \in F$

  $\Rightarrow \hat{\delta}(q_i, w_2) \in F$ for $i = 2, \ldots, k$

  as $q_2, \ldots q_k \in [q_1]$

i.e. q_i is reached on processing $w \in L(M)$, $i = 1, 2, \ldots k$ iff q_1 is reached on processing w.

i.e. q_1 of $[q_1] = \{q_1, \ldots, q_k\}$ can play the role of $q_2 \ldots q_k$

Example (8-states)

δ	0	1
q_0	q_1	q_5
q_1	q_6	q_2
q_2	q_0	q_2
q_3	q_2	q_6
q_4	q_7	q_5
q_5	q_2	q_6
q_6	q_6	q_4
q_7	q_6	q_2

8 states DFA

↓

5 states minimum DFA

- $\pi_0 = \{\{q_2\}, \{q_0, q_1, q_3, q_4, q_5, q_6, q_7\}\}$ - 0-equivalence
- $\pi_1 = \{\{q_2\}, \{q_0, q_4, q_6\}, \{q_1, q_7\}, \{q_3, q_5\}\}$

  → 1-equivalence

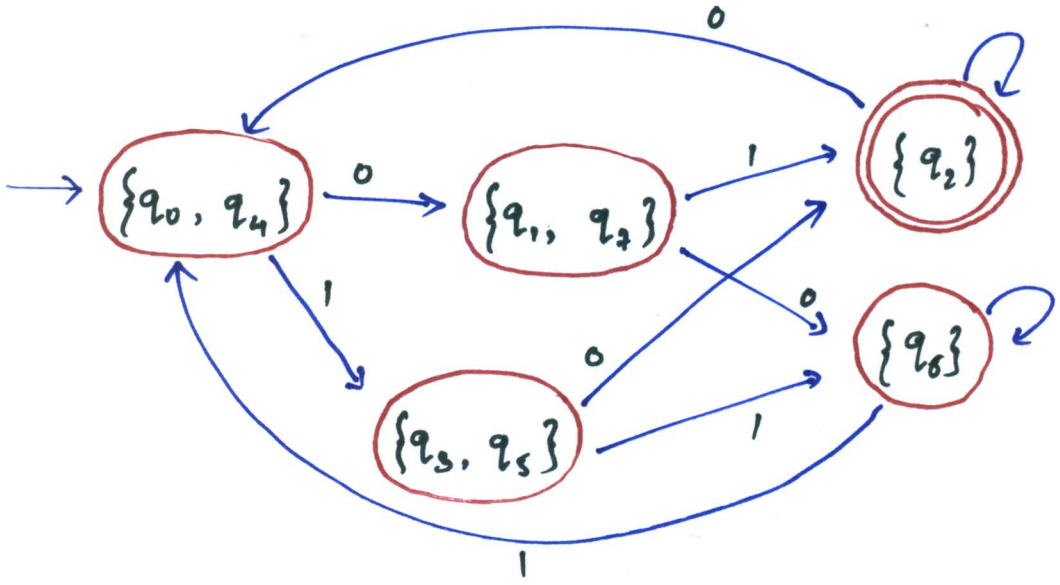
$\pi_2 = \{\{q_2\}, \{q_0, q_4\}, \{q_1, q_3\}, \{q_3, q_5\}, \{q_6\}\}$

→ 2-equivalence

$\pi_3 = \{\{q_2\}, \{q_6\}, \{q_0, q_4\}, \{q_1, q_3\}, \{q_3, q_5\}\}$

$= \pi_2$ STOP.

Minimum DFA (5-states)



The minimization process consists of two stages

1. Get rid of inaccessible states; that is state for which there exists no string $x \in \Sigma^*$ such that $\hat{\delta}(q_0, x) = q$, q_0 being the start state

   (DFS may be used to find such inaccessible states)

2. Collapse "equivalent" states

   (Removing inaccessible states doesnot change the set of accepted states)

   - Collapsing relation $\equiv$ defined over Q by

$$p \equiv q \overset{\text{def}^n}{\iff} \forall x \in \Sigma^* \left(\hat{\delta}(p,x) \in F \iff \hat{\delta}(q,x) \in F \right)$$

   p, q are indistinguishable

   - $\equiv$ is an equivalence relation on that induces a partition of into disjoint equivalence classes

$$[p] \overset{\text{def}^n}{=} \{ q \in Q \mid q \equiv p \}$$

   - every element $p \in Q$ is contained in exactly one equivalence class $[p]$ and

$$p \equiv q \iff [p] = [q]$$

Quotient Automation Construction from DFA M

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

  using collapsing relation $\equiv$ on Q, define DFA

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

  where $Q' = \{ [p] \mid p \in Q \}$

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

  $q_0' = [q_0]$

  $F' = \{ [p] \mid p \in F \}$

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

  This is easy to establish.

  For any $x \in \Sigma^*$

  $x \in L(M')$

  $\Leftrightarrow \hat{\delta}'(q_0', x) \in F'$

  $\Leftrightarrow \hat{\delta}'([q_0], x) \in F'$

  $\Leftrightarrow [\hat{\delta}(q_0, x)] \in F'$

  $\Leftrightarrow \hat{\delta}(q_0, x) \in F$

  $\Leftrightarrow x \in L(M)$

- Quotient automation M' cannot be further collapsed
- All states in an equivalence class are indistinguishable
- Two states from <u>two different</u> equivalence classes are distinguishable

Minimization Algorithm

(Computing the collapsing relation $\approx$ for a given DFA M with no inaccessible states)

Table filling algorithm:

1. Write down a table of all pairs $\{p, q\}$, initially unmarked
2. Mark $\{p, q\}$ if $p \in F$ and $q \notin F$ and vice-versa
3. Repeat the following until no more changes occur: if there exists an unmarked pair $\{p, q\}$ such that $\{\delta(p, a), \delta(q, a)\}$ is marked for some $a \in \Sigma$, then mark $\{p, q\}$
4. When done, $p \approx q$ iff $\{p, q\}$ is not marked

Running time $\rightarrow O(n^4)$

Note:

- if $\{p, q\}$ is marked in Step 2, then p, q are surely not equivalent

$p \approx q$ (unmarked pair)

$\overset{\text{def}^n}{\iff} \forall x \in \Sigma^* \left(\hat{\delta}(p, x) \in F \iff \hat{\delta}(q, x) \in F \right)$

Note:

Test equivalence of regular languages.

Two DFA are equivalent if their start states are equivalent using the above table filling algorithm.

Example: Minimize the FA below

		a	b
→	0	1	2
*	1	3	4
*	2	4	3
	3	5	5
	4	5	5
*	5	5	5

Step 1:

0
- 1
- - 2
- - - 3
- - - - 4
- - - - - 5

Step 2:

0
✓ 1
✓ - 2
- ✓ ✓ 3
- ✓ ✓ - 4
✓ - - ✓ ✓ 5

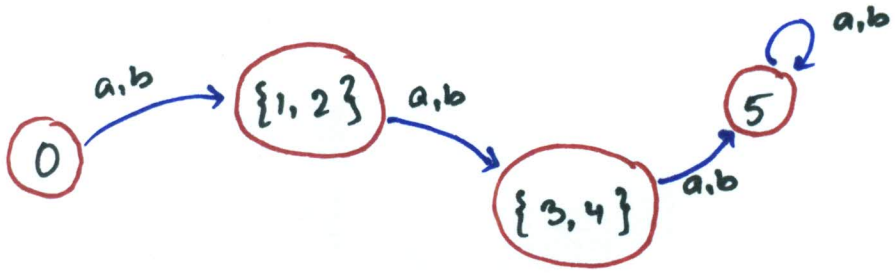
Step 3:

0
✓ 1
✓ - 2
- ✓ ✓ 3
- ✓ ✓ - 4
✓ ✓ ✓ ✓ ✓ 5

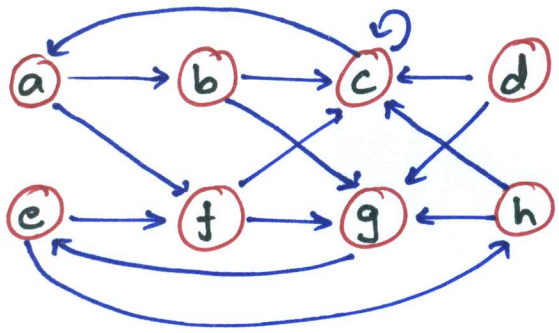
Step 4:

0
✓ 1
✓ - 2
✓ ✓ ✓ 3
✓ ✓ ✓ - 4
✓ ✓ ✓ ✓ ✓ 5

$1 \approx 2, \quad 3 \approx 4$



Example:



δ	0	1
→ a	b	f
b	g	c
* c	a	c
e	h	f
f	c	g
g	g	e
h	g	c

1.

a
- b
- - c
- - - d inaccessible state
- - - e e
- - - - - f
- - - - - - g
- - - - - - - h

2.

a
- b
✓ ✓ c
- - ✓ e
- - ✓ - f
- - ✓ - - g
- - ✓ - - - h

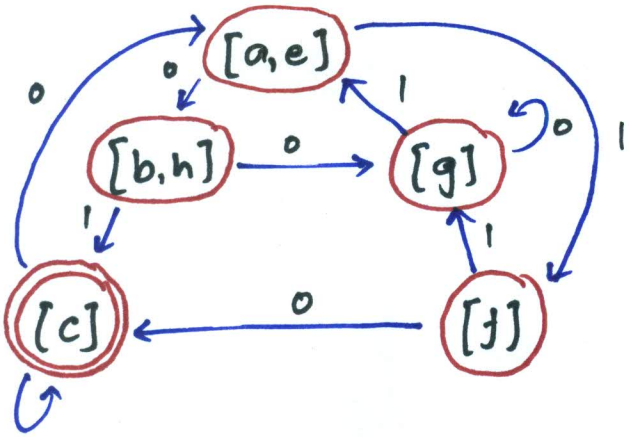
3.

a
✓ b
✓ ✓ c
- ✓ ✓ e
✓ ✓ ✓ ✓ f
- ✓ ✓ ✓ ✓ g
✓ - ✓ ✓ ✓ ✓ h

4.

a
✓ b
✓ ✓ c
- ✓ ✓ e
✓ ✓ ✓ ✓ f
✓ ✓ ✓ ✓ ✓ g
✓ - ✓ ✓ ✓ ✓ h

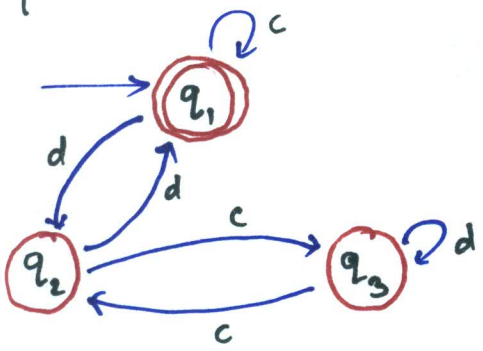
∴ $a \equiv e$, $b \equiv h$

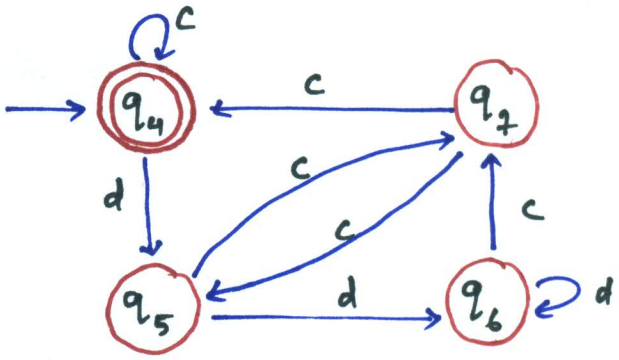


- Marked pair = distinguishable
- Unmarked pair = equivalent

Example

Show that the DFA M_1 and M_2 defined below are not equivalent





We have

δ	c	d
$\rightarrow^* q_1$	q_1	q_2
q_2	q_3	q_1
q_3	q_2	q_3
$\rightarrow^* q_4$	q_4	q_5
q_5	q_7	q_6
q_6	q_2	q_6
q_7	q_4	q_5

Alternatively

	c	d
(q_1, q_4)	(q_1, q_4)	(q_2, q_5)
(q_2, q_5)	(q_3, q_7)	(q_1, q_6)

accepting state

non-accepting state

M_1, M_2 are not equivalent.

Comparison Method

q_1

$\not\simeq$ q_2

$\not\simeq$ $\not\simeq$ q_3

$\not\simeq$ $\not\simeq$ $\not\simeq$ q_4

$\not\simeq$ $\not\simeq$ $\not\simeq$ $\not\simeq$ q_5

$\not\simeq$ $\not\simeq$ $\not\simeq$ $\not\simeq$ – q_6

$\not\simeq$ $\not\simeq$ $\not\simeq$ $\not\simeq$ $\not\simeq$ $\not\simeq$ q_7

$q_5 \approx q_6$, $q_1 \not\simeq q_4 \Rightarrow$ given DFA's are not equivalent

Example

	0	1
a	b	f
b	g	c
c	a	c
d	c	g
e	h	f
f	c	g
g	g	e
h	g	c

Find the minimum state FA equivalent to the following transition table.

OR

Find the automation obtained by collapsing equivalent states and removing inaccessible states.

Step 1:

a
- b
✓ ✓ c
- - ✓ d
- - ✓ - e
- - ✓ - - f
- - ✓ - - - g
- - ✓ - - - - h

Step 2:

a
✓ b
✓ ✓ c
✓ ✓ ✓ d
- ✓ ✓ ✓ e
✓ ✓ ✓ - ✓ f
- ✓ ✓ ✓ ✓ ✓ g
✓ - ✓ ✓ ✓ ✓ - h

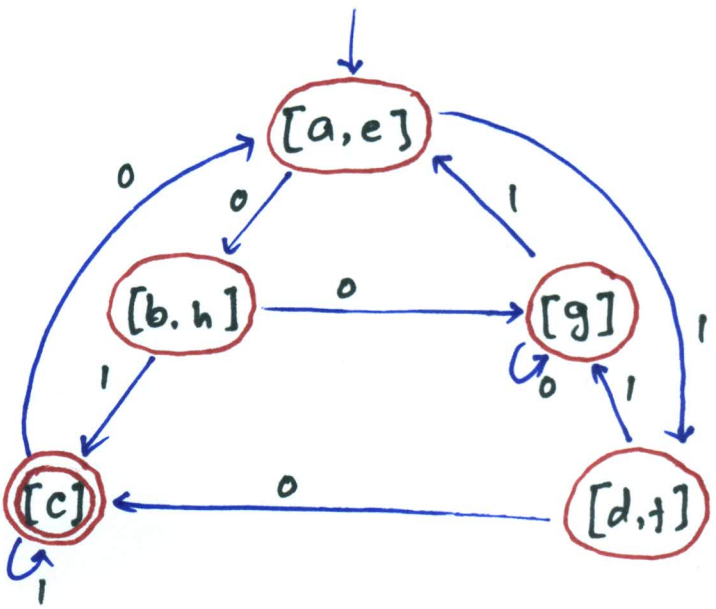
Step 3:

a
✓ b
✓ ✓ c
✓ ✓ ✓ d
- ✓ ✓ ✓ e
✓ ✓ ✓ - ✓ f
✓ ✓ ✓ ✓ ✓ ✓ g
✓ - ✓ ✓ ✓ ✓ ✓ h

$a \approx e$

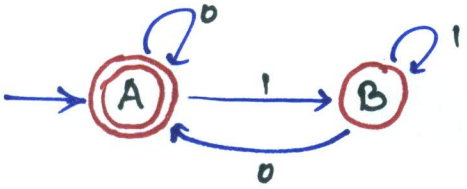
$d \approx f$

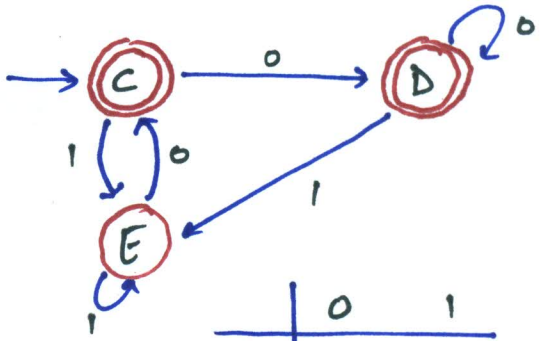
$b \approx h$



Example:

Check the equivalence of the given DFA's





	0	1
A	A	B
B	A	B
C	D	E
D	D	E
E	C	E

A
✓ B
– ✓ C
– ✓ – D
✓ – ✓ ✓ E

$A \approx C$, $A \approx D$, $C \approx D$, $B \approx E$

∴ the two DFA's are equivalent

Note 1: If $\{p,q\}$ is marked in Step 2, then p,q are surely not equivalent. Take $x = \varepsilon$ in the definition of $\approx$

Note 2: We may have to look at the same pair $\{p,q\}$ many times in Step 3, since any change in the table may suddenly allow $\{p,q\}$ to be marked.

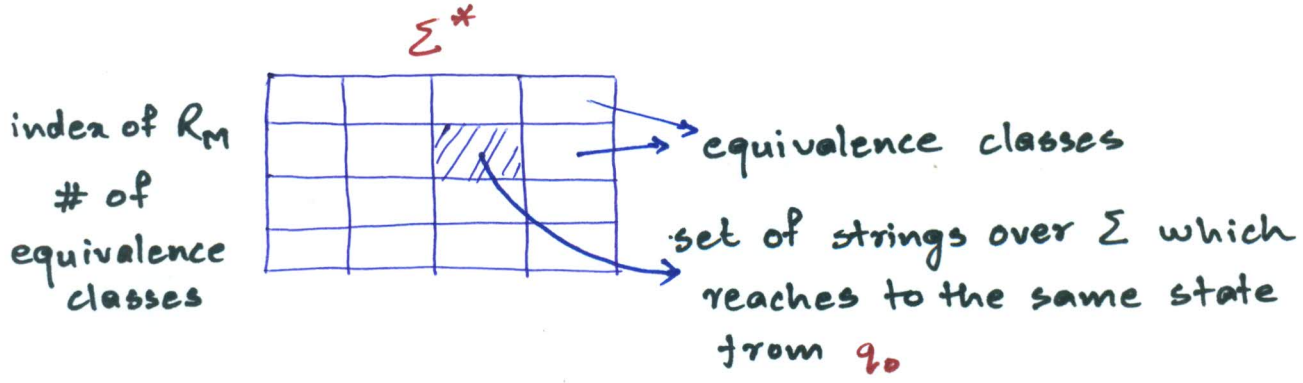
We stop only after we have marked all or we make an etire pass through the table with no new marks.

The Myhill-Nerode Theorem and Minimization of FA

- $M = (Q, \Sigma, \delta, q_0, F)$ a DFA
- Define a relation R_M as follows

  For $x, y \in \Sigma^*$, $x R_M y$ iff $\hat{\delta}(q_0, x) = \hat{\delta}(q_0, y)$

- R_M : equivalence class on Σ^* induced by DFA M



Claim: $x R_M y \Rightarrow xz R_M yz \quad \forall z \in \Sigma^*$

(Hint: $\hat{\delta}(q_0, xz) = \hat{\delta}(\hat{\delta}(q_0, x), z) = \hat{\delta}(\hat{\delta}(q_0, y), z) = \hat{\delta}(q_0, yz)$)

Right invariant relation: An equivalence relation R is right invariant if $xRy \Rightarrow xz\, R\, yz \quad \forall z$.

- R_M defined above is right invariant.

The Myhill-Nerode Theorem

The following three statements are equivalent

i. The set $L \subseteq \Sigma^*$ is accepted by some finite automata

ii. L is the union of some of the equivalence classes of a right invariant equivalence relation of finite index

iii. Let equivalence relation R_L be defined by

"$x R_L y$ iff $\forall z \in \Sigma^*$, xz is in L exactly when yz is in L"

Then R_L is of finite index

Proof:

(1) ⇒ (2)

Assume L is accepted by some DFA $M = (Q, \Sigma, \delta, q_0, F)$

Define (right invariant equivalence) relation R_M as:

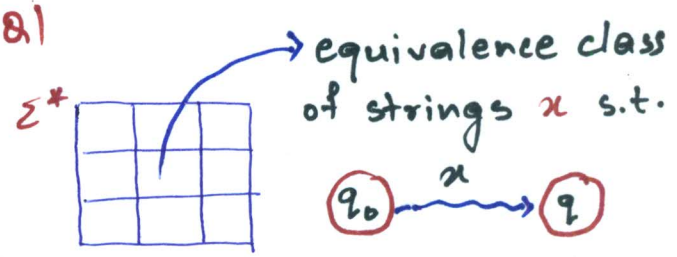
$x R_M y$ iff $\hat{\delta}(q_0, x) = \hat{\delta}(q_0, y)$

Index of R_M is finite $\leq |Q|$

L is the union of these equivalence classes that include string x such that

$\hat{\delta}(q_0, x) \in F$

i.e. the equivalence classes corresponding to final states.



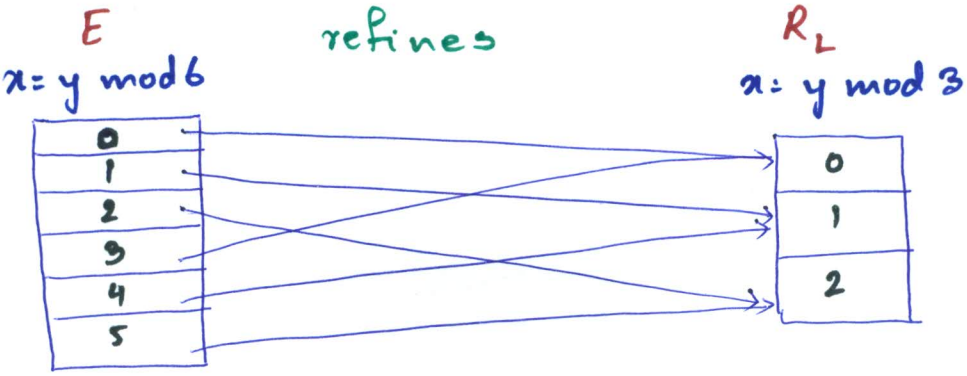
- corresponds to a state $q \in Q$
- $x \in L$ if $q \in F$

$\Rightarrow L = \cup \{$ equivalence classes of strings x corr. to $\delta(q_0, x) = q \in F$

(2) ⇒ (3)

claim: Any equivalence relation E satisfying (2) is a refinement of R_L

i.e. every equivalence class of is entirely contained in some equivalence class of R_L



Thus index of R_L cannot be greater than the index of E and so is finite.

Assume $x E y$

Since E is right invariant, for each $z \in \Sigma^*$, $xz \, E \, yz$

$\Rightarrow xz \in L \Rightarrow yz \in L$

$\Rightarrow x R_L y$

Hence equivalence class of x in E is the equivalence class of x in R_L

$\Rightarrow$ each equivalence class of E is contained within some equivalence class of R_L

Hence the result.

(3) $\Rightarrow$ (1)

Let R_L be an equivalence relation defined by

$x R_L y$ iff $\forall z \in \Sigma^*$

xz is in L exactly when yz is in L

and let R_L has finite index

Claim: R_L is right invariant.

Let $x R_L y$ and $w \in \Sigma^*$ be any string

We must prove that $xw \, R_L \, yw$, i.e. prove that $\forall z \in \Sigma^*$, xwz is in L exactly when ywz is in L

$x R_L y \Rightarrow$ for any $v \in \Sigma^*$, xv is in L exactly when yv is in L

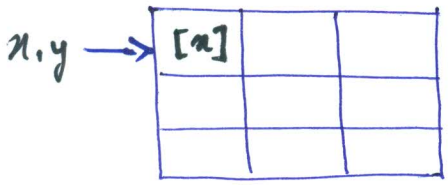
Take $v = wz$

$\Rightarrow xwz \in L$ exactly when $ywz \in L$

$\Rightarrow xw \, R_L \, yw$.

This proves the claim

- Consider the equivalence classes induced by the right invariance relation R_L



Q' : finite set of equivalence classes of R_L

$[x]$ → the class of Q' contains the string x

Construction of FA that accepts L

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

$$q_0' = [\varepsilon]$$

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

$$\delta'([x], a) = [xa]$$

δ' is well defined as R_L is right invariant

$$\delta'([x], a) = [ya], \quad y \in [x]$$

But $xR_Ly \Rightarrow xz \in L$ exactly when $yz \in L$ for any $z \in \Sigma^*$

Take $z = az'$

Then we get $xaz' \in L$ exactly when $yaz' \in L$ for any $z' \in \Sigma^*$

$\Rightarrow xaR_Lya$

$\Rightarrow [xa] = [ya]$

The FA M accepts L since

$\hat{\delta}'(q_0', x) = [\varepsilon x] = [x]$ and then

$x \in L(M')$ iff $[x] \in F$

i.e. iff $x \in L$.

- $L \rightarrow$ a regular language accepted by DFA M

Construction of unique minimal DFA for L using Myhill-Nerode Relation R_L for L on Σ^*

$$x R_M y \Leftrightarrow \hat{\delta}(q_0, x) = \hat{\delta}(q_0, y)$$; q_0 - start state of M.

Definition:

An equivalence relation $\equiv$ for L is said to be a Myhill-Nerode Relation for L if it satisfies

1. right congruence

   for any $x, y \in \Sigma^*$ and $a \in \Sigma$

   $$x \equiv y \Rightarrow xa \equiv ya$$

2. refines L

   for any

   $$x \equiv y \Rightarrow (x \in L \Leftrightarrow y \in L)$$

   ($\hat{\delta}(q_0, x) = \hat{\delta}(q_0, y)$ implies either both x and y are accepted or both are rejected)

3. Finite index

   It has only finitely many equivalence classes

   ($\exists$ exactly one equivalence class $\{x \in \Sigma^* \mid \hat{\delta}(q_0, x) = q\}$ corresponding to each state q of M)

- A natural one to one correspondence (upto isomorphism of automatas) between

  - DFA M for L with input alphabet Σ and with no inaccessible states

  and - Myhill Nerode relations for L on Σ^*

Define DFA $M = (Q', \Sigma, \delta', q_0', F')$ where

↓ Minimal if constructed using the equivalence relation R_L ↓ R_L is the coarsest possible Myhill-Nerode relation for L

$$Q' = \{ [x] \mid x \in \Sigma^* \}$$

$$q_0' = [\varepsilon]$$

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

$$\delta'([x], a) = [xa]$$

Refinement between equivalence relations

→ is a partial order

i.e. i. reflexive ii. transitive iii. antisymmetric

→ A refines B implies A is fined and B is coarser

Example (Myhill-Nerode Relation)

1. R_M : $x R_M y \Leftrightarrow \hat{\delta}(q_0, x) = \hat{\delta}(q_0, y)$

2. R_L : $x R_L y \Leftrightarrow \forall z \in \Sigma^* \ (xz \in L \Leftrightarrow yz \in L)$

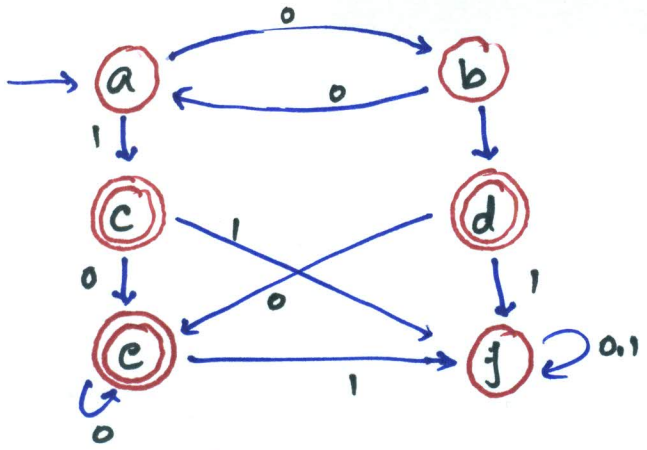
   provided L is a regular set.

R_M refines R_L and R_L has smaller index than R_M

Example:

Let L be the language 0^*10^*

L is accepted by the following DFA



Equivalence relation R_M

$x R_M y$ iff $\hat{\delta}(q_0, x) = \hat{\delta}(q_0, y)$

- As all states of M are reachable from $q_0 = a$ there are six equivalence classes induced by R_M

$C_a = (00)^*$ $\quad$ $C_d = (00)^*01$

$C_b = (00)^*0$ $\quad$ $C_e = 0^*100^*$

$C_c = (00)^*1$ $\quad$ $C_f = 0^*10^*1(0+1)$

- $L = C_c \cup C_d \cup C_e \rightarrow \delta(a,x) = e \;\; \forall x \in C_e$

  $\delta(a,x) = c \;\; \forall x \in C_c$ $\quad$ $\delta(a,x) = d \;\; \forall x \in C_d$

Note: $C_a, C_b, \ldots, C_f$: equivalence classes of strings

Equivalence relation R_L: (less index than R_M)

$x R_L y$ iff $\forall z \in \Sigma^*$ ($xz \in L$ exactly when $yz \in L$)

- R_M is a refinement of R_L, i.e. index of R_L < index of R_M

- $L = 0^*10^*$

  $x R_L y$ iff either

  1. x and y have no 1's

  2. x and y have one 1

  or 3. x and y have more than one 1

Example:

a. Let $x = 010$, $y = 1000$

Then $xz \in L$ iff $z \in 0^*$

$yz \in L$ also, under the same condition.

b. Let $x = 01$, $y = 00$

Then $\exists\ z = 0$, such that $xz = 010 \in L$ but $yz = 000 \notin L$

Equivalence classes of R_L

$C_1 = 0^*$

$C_2 = 0^*10^*$

$C_3 = 0^*10^*1(0+1)^*$

- L consists of only one of these classes namely C_2

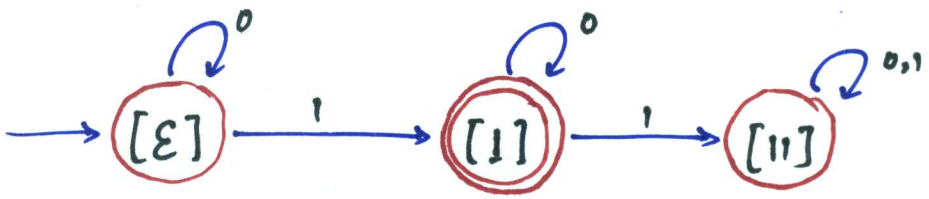
R_M refines R_L

C_a	C_b	C_c
C_d	C_e	C_f

$L = C_c \cup C_e \cup C_d$

C_1	C_2	C_3

$L = C_2$



$C_1 = [\varepsilon]$ no 1's

$C_2 = [1]$ one 1's

$C_3 = [11]$ two 1's

Myhill-Nerode Theorem

Let $L \subseteq \Sigma^*$. The following statements are equivalent

a. L is regular

b. $\exists$ a Myhill-Nerode relation for R

c. the relation R_L defined by

$$x R_L y \iff \forall z \in \Sigma^* \left(xz \in L \iff yz \in L\right)$$

is of finite index

An application

The Myhill-Nerode theorem can be used to determine whether a set R is regular or non-regular by determining the number of equivalence classes induced by R_L

Example:

Consider the set $L = \{a^n b^n \mid n \geq 0\}$

if $k \neq n$, then $a^k b^k \in L$, but $a^m b^k \notin L$

$\Rightarrow a^k R_L a^m$ does not hold

$\Rightarrow a^k, a^m$ an indifferent equivalence classes of R_L

Therefore, there are infinitely many equivalence classes of R_L, atleast one for each a^k, $k \geq 0$

$\therefore$ By Myhill-Nerode Theorem, L is not regular

Infact, one can show that the equivalence classes of R_L are exactly

$$G_k = \{a^k\}, \quad k \geq 0$$

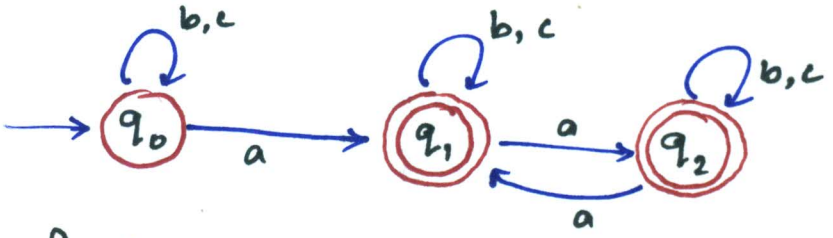
$$H_k = \{a^{n+k} b^n \mid 1 \leq n\}, \quad k \geq 0$$

$$F_k : \quad \Sigma^* \cup (G_k \cup H_k) = \Sigma^* - \{a^m b^m \mid 0 \leq n \leq m\}$$

- For strings in G_k all and only the strings in $\{a^n b^{n+k} \mid n>0\}$ can be appended to obtain a string in L
- For strings in H_k only the string b^k can be appended to obtain a string in L
- No string can be appended to a string in f_k to obtain a string in L

Example:

DFA M:



- M accepts set of strings having atleast one a
- $\Sigma^* = \{a, b, c\}^*$ is partitioned by R_M in 3 equivalence classes:

$C_0 = \{b, c\}^*$

C_1 = set of strings over Σ^* having an odd no. of a's

C_2 = set of strings over Σ^* having an even no. of a's

- $L = L(M) = C_1 \cup C_2$

Equivalence classes of R_L

- C_0, C_1 cannot be merged since

  if $x \in C_0, y \in C_1$ then $xb \notin L$ but $yb \in L$

- C_0, C_2 cannot be merged since

  $x \in C_0, y \in C_2$ then $xb \notin L$ but $yb \in L$

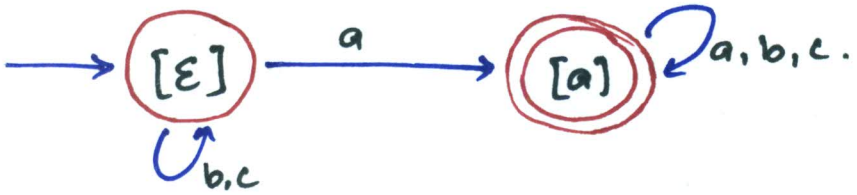
- C_1, C_2 can be merged since

  if $x \in C_1, y \in C_2$ then

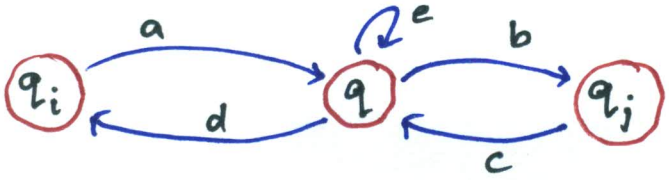
  $xz \in L$ exactly when $yz \in L$ $\forall z \in \Sigma^* = \{a,b,c\}^*$

$C_0 = [\varepsilon]$ strings with no $\underline{a}$

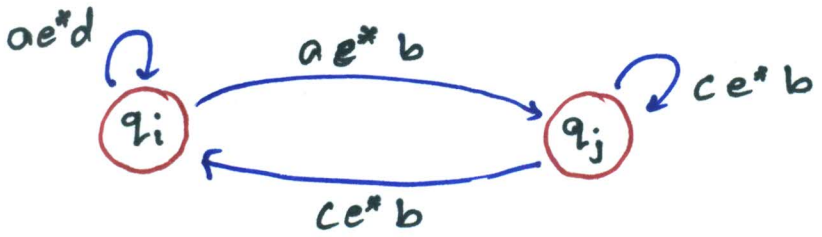
$C_1 \cup C_2 = [a]$ strings with atleast one $\underline{a}$



State Removal Method



Desired pattern for state removal (q to be removed)



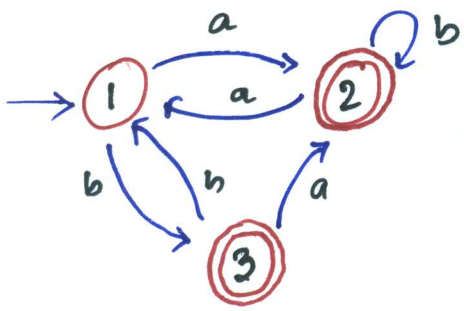
Algorithm (Linz)

Step 1: If the state (start) is an accepting state or has transition in, add a new - non-accepting start state and add an ε- transition between the new start state and the former start state.

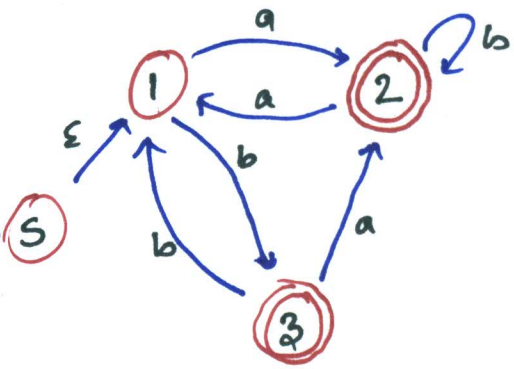
Step 2: If there is more than one accepting state, or if the single accepting state has transition out, add a new accepting state. Make all other states non-accepting, and add an ε-transition from each former accepting state to new accepting state.

Step 3: For each non-accepting state in turn, eliminate the state and update transitions according to the procedure above.

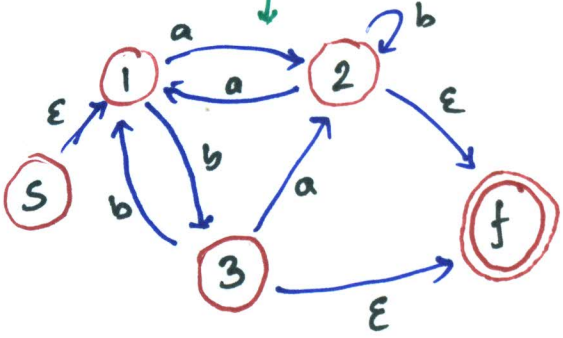
Illustration



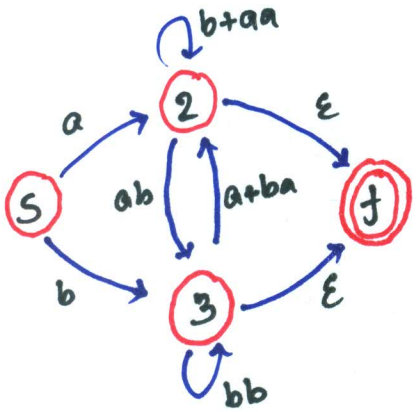
step 1



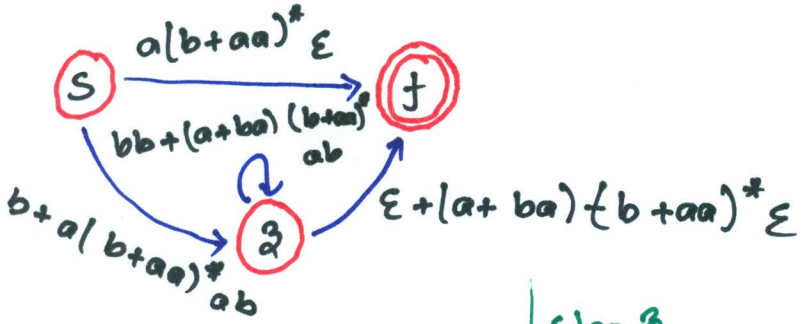
step 2



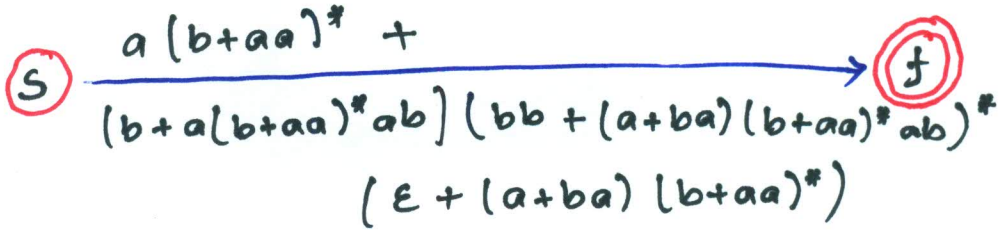
step 3
remove 1



step 3
Remove 2



step 3
Remove 3



Two way FA

DFA: a control unit that reads a tape moving one square right at each move

NFA: nondeterminism is added to the model many copies of the control unit to exist and scan the tape simultaneously

ε-NFA: Change the state without reading the input symbol or moving the tape head

2-way FA: tape head moves left as well as right

- These generalizations does not increase the power of FA $\longrightarrow$ all accepts only regular sets.

2-DFA (two-way deterministic finite automata)

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

$$\delta : Q \times \Sigma \longrightarrow Q \times \{L, R\}$$

i) $(q) \xrightarrow{a} (p)$ $\qquad \delta(q, a) = (p, R)$

ii) $(p) \xleftarrow{a} (q)$ $\qquad \delta(q, a) = (p, L)$

In (i) state q, scanning input symbol a, the 2-DFA enters state p and moving its head right one square. Similarly in (ii).

Extended transition function: $\hat{\delta} : Q \times \Sigma^*$

Concept is insufficient for 2-DFA as it can move left

Instantaneous Description (ID) of a 2-DFA

- $M = (Q, \Sigma, \delta, q_0, F)$ → 2-DFA
- ID $\in \Sigma^* Q \Sigma^*$
- relation on IDs → $\vdash_M$

  $I_1 \vdash_M I_2$ iff M can go from the instantaneous description I_1 to I_2 in one move

When M is understood, we say $I_1 \vdash I_2$

ID wqx, $w, x \in \Sigma^*$, $q \in Q$

i. wx is the input string

ii. q is the current state

iii. the input head is scanning the first symbol of x

- If $x = \varepsilon$, then the input head has moved off the right end of the input.

Relation $\vdash_M$

1. $a_1 a_2 \cdots a_{i-1}\, q\, a_i \cdots a_n \vdash a_1 a_2 \cdots a_{i-1} a_i\, p\, a_{i+1} \cdots a_n$

   if $\delta(q, a_i) = (p, R)$

2. $a_1 a_2 \cdots a_{i-2} a_{i-1}\, q\, a_i \cdots a_n \vdash a_1 a_2 \cdots a_{i-2}\, p\, a_{i-1} a_i \cdots a_n$

   if $\delta(q, a_i) = (p, L)$ and $i > 1$

- $i > 1$ prevents any action in the event that the tape head would move off the left end of the tape.
- no move is possible if $i = n+1$, i.e. if the tape head has moved off the right end.

$\vdash^*$ (reflexive, transitive closure of $\vdash$)

i.e. $I \vdash^* I$ $\forall$ ID's $I \in \Sigma^* Q \Sigma$

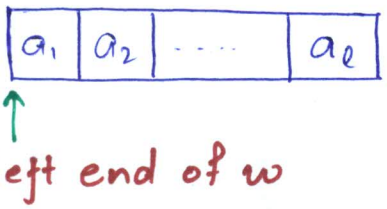
$I_1 \vdash^* I_k$ whenever $I_1 \vdash I_2 \vdash \cdots \vdash I_k$

for some $I_2, \ldots I_{k-1}$

We define

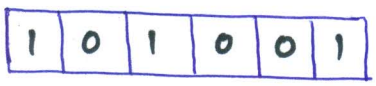
$$L(M) = \{ w \mid q_0 w \vdash^* wp \text{ for some } p \in F \}$$

input tape containing $w = a_1 a_2 \cdots a_\ell$



Example:

	0	1
q_0	(q_0, R)	(q_1, R)
q_1	(q_1, R)	(q_2, L)
q_2	(q_0, R)	(q_2, L)



$q_0 101001 \vdash^* 101001 \triangleright$ when $\triangleright \in F$

$q_0 101001 \vdash 1q_0 01001$

$\vdash 10q_1 1001$

$\vdash 1q_2 0100 1$

$\vdash 10q_0 1001$

$\vdash 101q_1 001$

$\vdash 1010q_1 01$

$\vdash 10100q_1 1$

$\vdash 1010q_2 01$

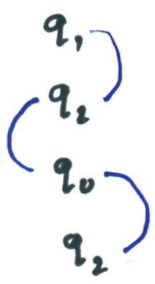
$\vdash 10100q_0 1$

$\vdash 101001q_1$

Crossing Sequence:

List of states below each boundary between squares

- A crossing sequence $q_1, q_2, \dots q_k$ is valid if it is of odd length, and no two odd and no two even-numbered elements are identical.



Not valid forming loop (as 2-DFA is deterministic) and could never fall off the right end of the tape.

- In the crossing sequence $q_1, q_2, \dots, q_k$

  $q_1, q_3, \dots$ → right moves

  $q_2, q_4, \dots$ → left moves

input accepted → all crossing sequences of odd length

# of crossing sequence → finite

Theorem:

If L is accepted by a 2-DFA, then L is a regular set.

Proof:

2-DFA M $\xrightarrow{\text{construct}}$ NFA M' having valid crossing sequences of M as its state

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

Finite automata with output

- FA → output $\in$ {accept, reject}

Other alphabet for output of FA

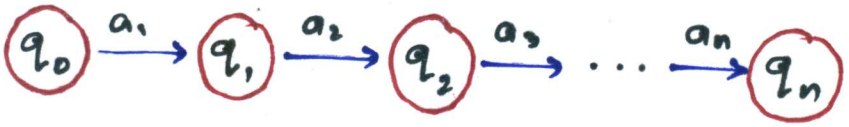
Moore machine
(output associated with the state)

Mealy machine
(output associated with the transition)

Moore Machines

- six tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$
  - Δ, λ ↓ output alphabet

$\lambda : Q \to \Delta$ giving output associated with each state.



$$\delta(q_{i-1}, a_i) = q_i \quad , \ 1 \le i \le n, \ n \ge 0$$

input: $a_1 a_2 \ldots a_n$

output: $\lambda(q_0)\lambda(q_1)\ldots\lambda(q_n)$ (length $n+1$)

input : ε

output: $\lambda(q_0)$

DFA is a special case of Moore machine

output alphabet $\Delta = \{0, 1\}$

q is an accepting state iff $\lambda(q) = 1$

Example : A Moore machine

	$\delta(q,0)$ $a=0$	$\delta(q,1)$ $a=1$	output $\lambda(q)$
q_0	q_3	q_1	0
q_1	q_1	q_2	1
q_2	q_2	q_3	0
q_3	q_3	q_0	0

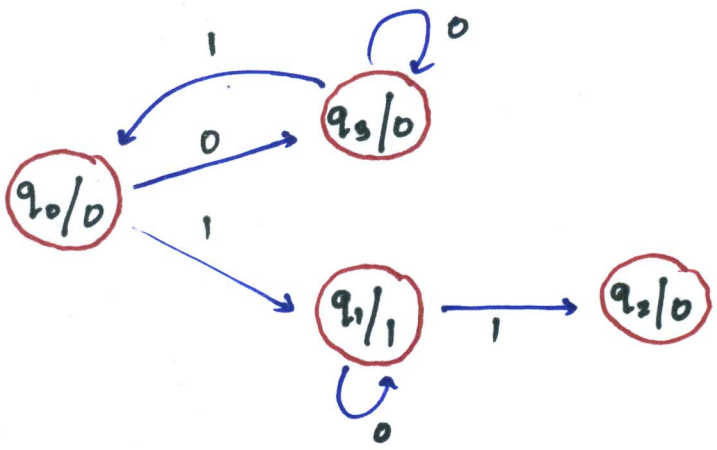
input: 0111 (length 4)

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

output:

$$\lambda(q_0)\,\lambda(q_3)\,\lambda(q_0)\,\lambda(q_1)\,\lambda(q_2) = 00010$$

(length 5)



Mealy Machine

- six tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$

$$\lambda : Q \times \Sigma \rightarrow \Delta$$

$$\delta(q_{i-1}, a_i) = q_i \qquad 1 \le i \le n, \quad n \ge 0$$

input: $a_1 a_2 \ldots a_n$

output: $\lambda(q_0, a_1)\ \lambda(q_1, a_2) \ldots \lambda(q_{n-1}, a_n)$

where $\delta(q_{i-1}, a_i) = q_i$

input: ε

output: ε

Example

	$a = 0$		$a = 1$	
	state $\delta(q,a)$	output $\lambda(q,a)$	state $\delta(q,a)$	output $\lambda(q,a)$
q_1	q_3	0	q_2	0
q_2	q_1	1	q_4	0
q_3	q_2	1	q_1	1
q_4	q_4	1	q_3	0

input: 0011

$$q_1 \xrightarrow{0/0} q_3 \xrightarrow{0/1} q_2 \xrightarrow{1/0} q_4 \xrightarrow{1/0} q_3$$

output:

$$\lambda(q_1, 0)\ \lambda(q_3, 0)\ \lambda(q_2, 1) \cdot \lambda(q_4, 1) = 0100$$

Equivalence of Moore and Mealy Machines

- $T_M(w)$ - output produced by M on input w when M is a Machine
- M - Mealy machine M' - Moore machine

$$T_M(w) \neq T_{M'}(w) \text{ as } |T_M(w)| = |T_{M'}(w)| - 1$$

- Neglect the response of a Moore machine M' on ε (i.e. discarding $\lambda(q_0)$ from output string $\lambda(q_0)\,\lambda(q_1)\cdots\lambda(q_k)$ for input string $a_1 \ldots a_k$ where $q_0 \xrightarrow{a_1} q_1 \longrightarrow \cdots \xrightarrow{a_k} q_k$)
- M, M' are equivalent if $\forall$ inputs $w \in \Sigma^*$

$$bT_M(w) = T_{M'}(w)$$

where b is the output of M' for its initial state q_0.

Theorem (Transforming a Moore Machine into a Mealy Machine)

If $M_1 = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ is a Moore machine, then there is a Mealy machine M_2 equivalent to M_1

Proof:

Given Moore machine $M_1 = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ construct a Mealy machine $M_2 = (Q, \Sigma, \Delta, \delta, \lambda', q_0)$ where

$$\lambda'(q, a) = \lambda(\delta(q, a))$$

Example: Given Moore machine

M_1

$\delta(q,a)$	$a=0$	$a=1$	$\lambda(q)$
q_0	q_3	q_1	0
q_1	q_1	q_2	1
q_2	q_2	q_3	0
q_3	q_3	q_0	0

Equivalent Mealy Machine

$\delta(q,a)$	$a=0$	$\lambda'(q,0)$	$a=1$	$\lambda'(q,1)$
q_0	q_3 $\xrightarrow{\lambda}$	0	q_1	1
q_1	q_1 $\xrightarrow{\lambda}$	1	q_2	0
q_2	q_2 $\xrightarrow{\lambda}$	0	q_3	0
q_3	q_3 $\xrightarrow{\lambda}$	0	q_0	0

input: 0111 M_1: $q_0 \xrightarrow{0} q_3 \xrightarrow{1} q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_2$

M_2: $q_0 \xrightarrow{0} q_3 \xrightarrow{1} q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_2$

output: $\lambda(q_3)\lambda(q_0)\lambda(q_1)\lambda(q_2) = 0010$

- M_1, M_2 enter the same sequence of states on the same input, and with each transition M_2 emits the output $(\lambda'(q_i,a))$ that M associates with the state entered $(\delta(q_i,a))$

M_2: $\lambda'(q_i,a) = \lambda(\delta(q_i,a))$

$M_1 \rightarrow \lambda(q_j)$

Theorem (Transforming a Mealy machine into a Moore machine)

If $M_1 = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ is a Mealy machine, then there is a Moore machine M_2 equivalent to M_1

Proof:

Given Mealy Machine

$$M_1 = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$$

construct a Moore machine

$$M_2 = (Q \times \Delta, \Sigma, \Delta, \delta', \lambda', \{q_0, b_0\})$$

where $b_0 \rightarrow$ an arbitrarily selected element of Δ

- $\delta'(\{q, b\}, a) = \{\delta(q,a), \lambda(q,a)\} \in Q \times \Delta$

  $b \in \Delta, a \in \Sigma, q \in Q, \lambda: Q \times \Sigma \rightarrow \Delta$

- $\lambda'(\{q, b\}) = b$

Example:

Given Mealy Machine M_1

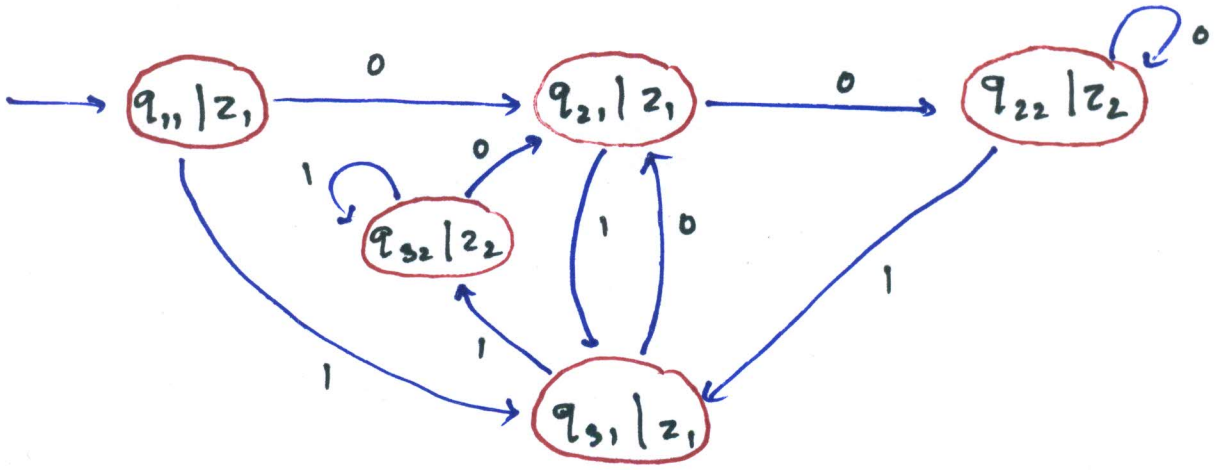
$\delta(q,a)$	$a=0$	$\lambda(q,0)$	$a=1$	$\lambda(q,1)$
q_1	q_2	z_1	q_3	z_1
q_2	q_2	z_1	q_3	z_1
q_3	q_2	z_1	q_3	z_2

Equivalent Moore Machine (M_2)

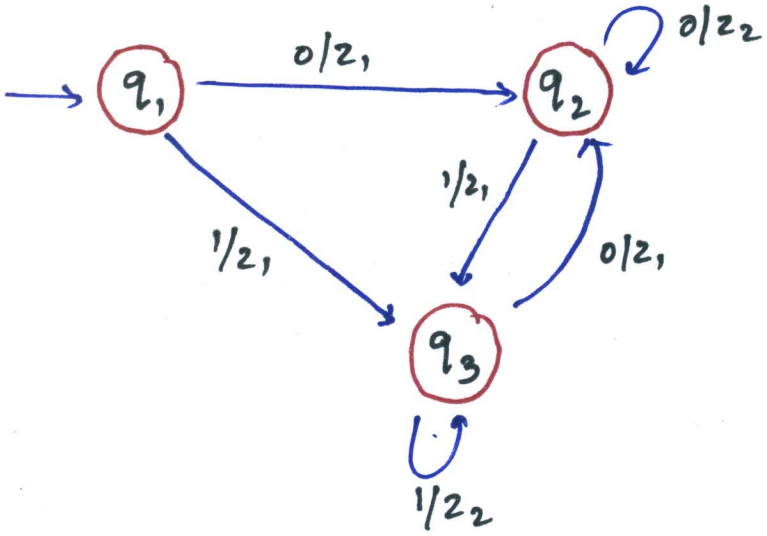
$\delta'(q,a)$	$a=0$	$a=1$	output λ'
$\{q_1, z_1\}$	$\{q_2, z_1\}$	$\{q_3, z_1\}$	z_1
$\{q_3, z_1\}$	$\{q_2, z_1\}$	$\{q_3, z_2\}$	z_1
$\{q_2, z_1\}$	$\{q_2, z_2\}$	$\{q_3, z_1\}$	z_1
$\{q_3, z_2\}$	$\{q_2, z_1\}$	$\{q_3, z_2\}$	z_2
$\{q_2, z_2\}$	$\{q_2, z_2\}$	$\{q_3, z_1\}$	z_2

- q_1 – not splitted
- q_2, q_3 – splitted

- $\{q_i, z_j\} \rightarrow q_{ij}$



Moore machine
constructed from the
Mealy machine given below

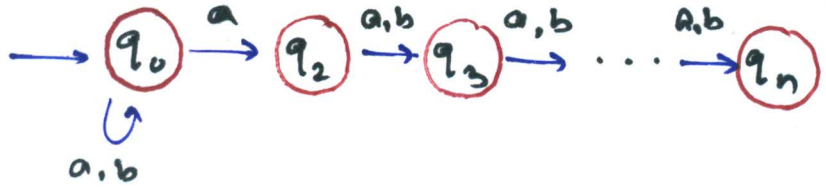


Mealy machine

Example 1:

$L_n = (a+b)^* a(a+b)^{n-1}$ – n^{th} symbol from the right is 'a'.

NFA (n+1 states)



Equivalent DFA → 2^n states

2-DFA (n+states)

First scan the input from the left to right to reach the right end

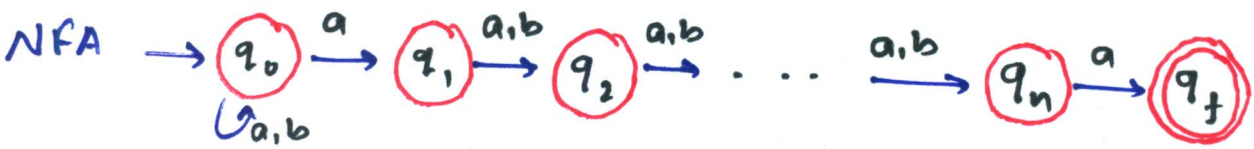
Then scan right to left n positions and finally checking whether or not the reached input cell contains the symbol a

Note: Possibility of moving the input head in both directions can dramatically reduce the size of DFA

In this case, one reversal is enough to reduce an automaton of exponential size in n to an automaton of linear size

Example: $L = (a+b)^* a\,(a+b)^{n-1} a\,(a+b)^n$

NFA



Equivalent DFA: $2^n + 1$ states

2-DFA: $2n+$ states

Example:

Given Mealy Machine (M_1)

$\delta(q,a)$	$a=0$	$\lambda(q,0)$	$a=1$	$\lambda(q,1)$
q_1	q_2	z_1	q_3	z_1
q_2	q_2	z_2	q_3	z_1
q_3	q_2	z_1	q_3	z_2

Equivalent Moore machine (M_2)

$\delta'(q,a)$	$a=0$	$a=1$	output λ'
$\{q_1, z_1\}$	$\{q_2, z_1\}$	$\{q_3, z_1\}$	z_1
$\{q_3, z_1\}$	$\{q_2, z_1\}$		
$\{q_2, z_1\}$	$\{q_2, z_2\}$		
$\{q_2, z_1\}$	{		
$\{q_3, z_2\}$			
$\{q_2, z_2\}$			