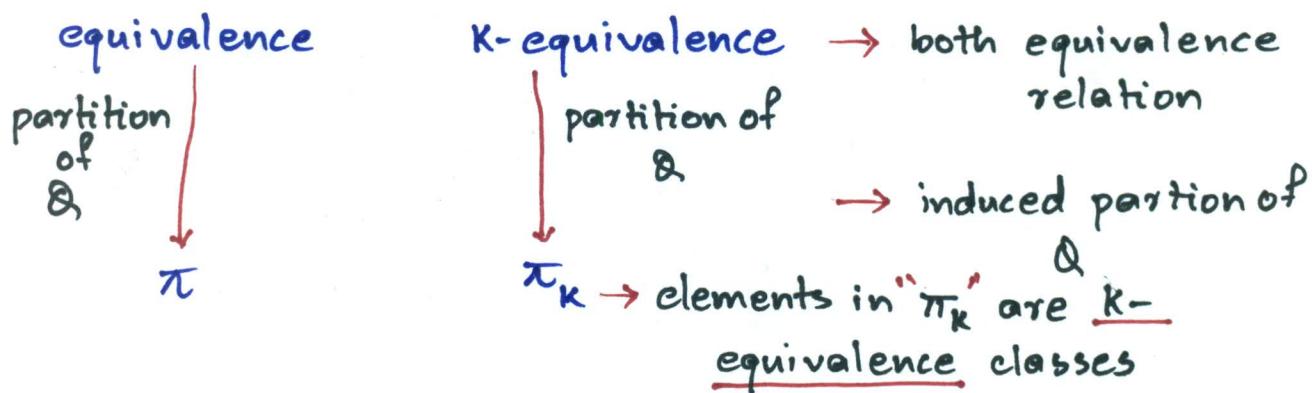


West 7: Lecture Notes

Minimization of FA

- $M = (\mathcal{Q}, \Sigma, \delta, q_0, F)$
- $q_1 \equiv q_2$ if $\hat{\delta}(q_1, x), \hat{\delta}(q_2, x)$ both $\in F$ or $x \in \Sigma^*$
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 ($k > 0$)
if $\hat{\delta}(q_1, x), \hat{\delta}(q_2, x)$ both $\in F$ | for all strings x
or both $\notin F$ | of length $\leq k$
 $\forall x \in \Sigma^*$ with $|x| \leq k$

Properties



- q_1, q_2 k -equivalent $\nabla k > 0 \Rightarrow q_1, q_2$ 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

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

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

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

$\Rightarrow q_3 = \delta(q_1, a), q_4 = \delta(q_2, a)$ 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 $\pi_0 \rightarrow$ by definition of 0-equivalence

$$\pi_0 = \{ Q_1^0, Q_2^0 \} \text{ where}$$

$Q_i^0 = \text{the set of accepting states of DFA}$
 $M = (Q, \Sigma, S, q_0, F)$

$$Q_2^0 = Q - Q_1^0$$

2. Construction of π_{k+1} from π_k

Let $Q_i^k \in \pi_k$

- $q_1, q_2 \in Q_i^k$ 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, Q_i^k is further divided into $(k+1)$ -equivalence classes.

- Repeat this for every $Q_i^k \in \pi_k$ to get all the elements of π_{k+1}

3. Construct π_n for $n=1, 2, \dots$ 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 $\leq |Q|$

$Q \rightarrow$ set of the states of F

- $[q_1] = \{q_1, q_2, \dots, q_k\} \rightarrow$ 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 $i = 2, \dots, k$
as $q_2, \dots, q_k \in [q_1]$

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

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

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

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

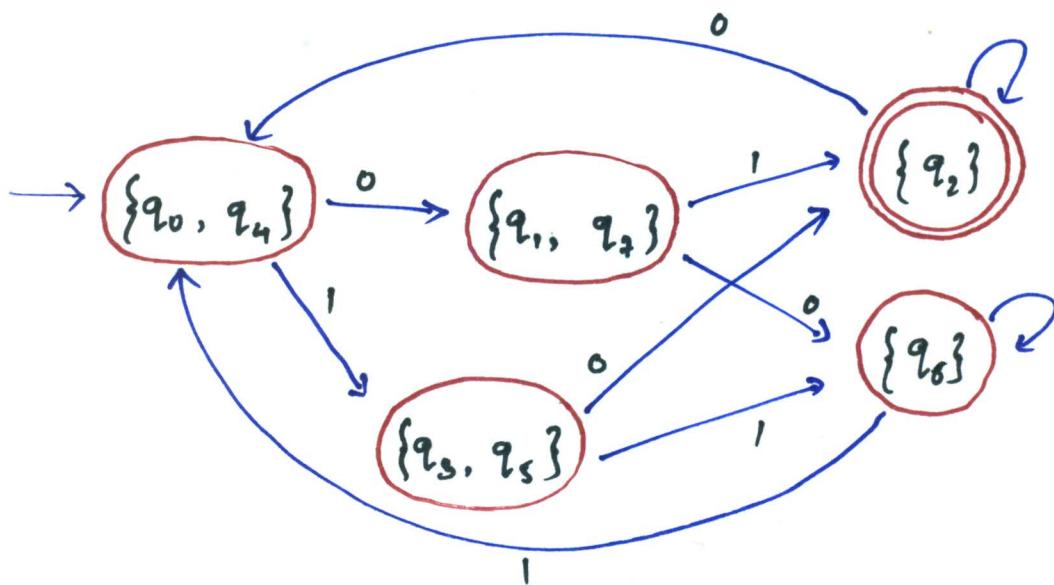
\rightarrow 2-equivalence

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

= π_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

$$\hat{\delta}(q_0, x) = q, \quad q_0 \text{ being the start state}$$

(DFS may be used to find such inaccessible states)

2. Collapse "equivalent" states

(Removing inaccessible states does not change the set of accepted states)

- Collapsing relation \equiv defined over Δ by

$$p \equiv q \stackrel{\text{defn}}{\iff} \forall x \in \Sigma^* (\hat{\delta}(p, x) \in F \iff \hat{\delta}(q, x) \in F)$$

p, q are indistinguishable

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

$$[p] \stackrel{\text{defn}}{=} \{q \in Q \mid q \equiv p\}$$

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

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

Quotient Automaton Construction from DFA M

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

using collapsing relation \equiv on Δ , define DFA

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

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

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

$$q_0' = [q_0]$$

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

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

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 automaton M' cannot be further collapsed
- All states in an equivalence class are indistinguishable
- Two states from two different 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 $\{\delta(p, q), \delta(q, a)\}$ 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

$$\begin{aligned} p &\approx q && (\text{unmarked pair}) \\ \xrightarrow{\text{defn}} & \forall x \in \Sigma^* \left(\hat{\delta}(p, x) \in F \Leftrightarrow \hat{\delta}(q, x) \in F \right) \end{aligned}$$

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

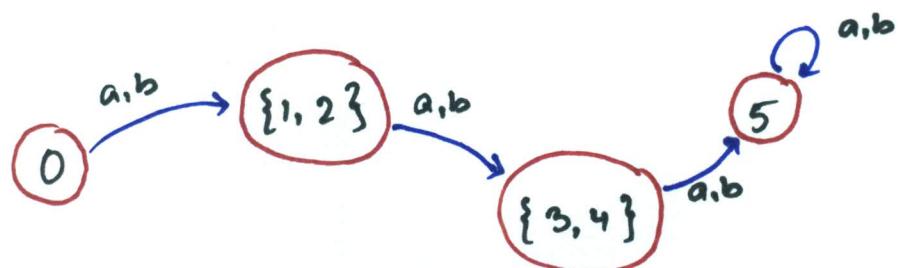
Step 3:

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

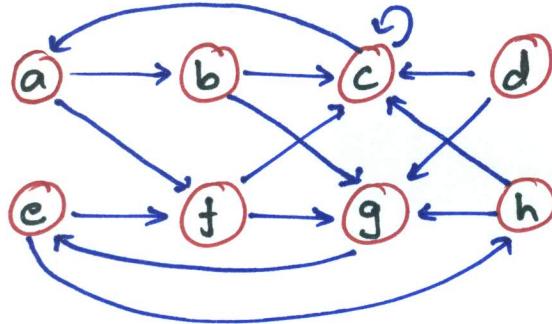
Step 4:

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

$1 \approx 2, 3 \approx 4$

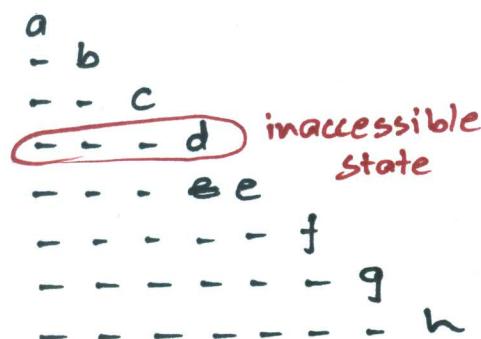


Example:

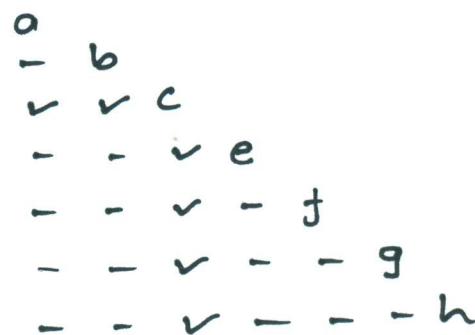


<u>S</u>	<u>O</u>	<u>I</u>
a	b	f
b	g	c
x c	a h	c
e	c	f
f	g	q
g	g	e
h	g	c

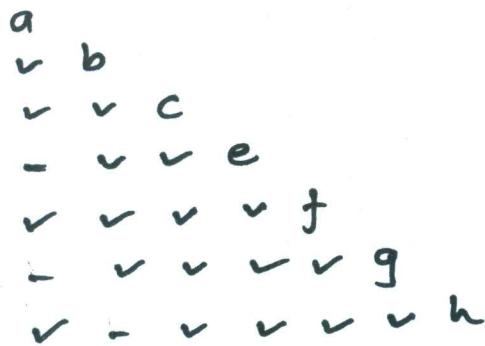
1.



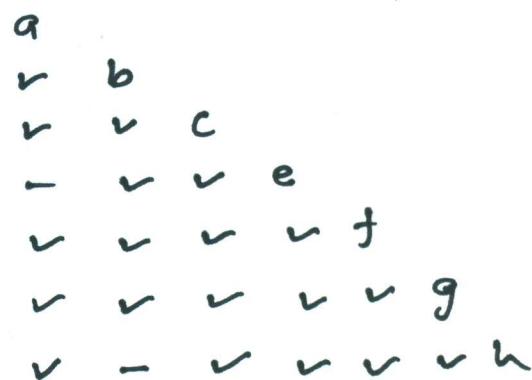
2



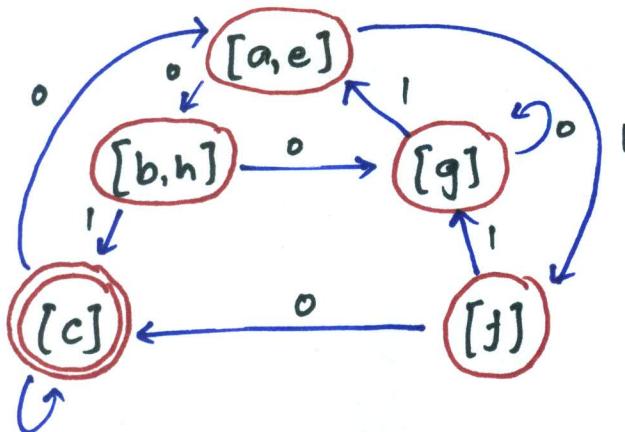
3.



4.



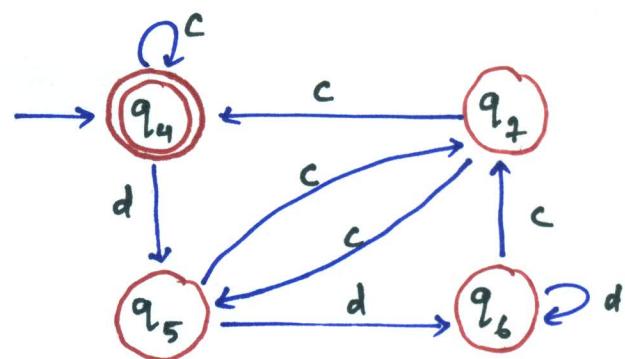
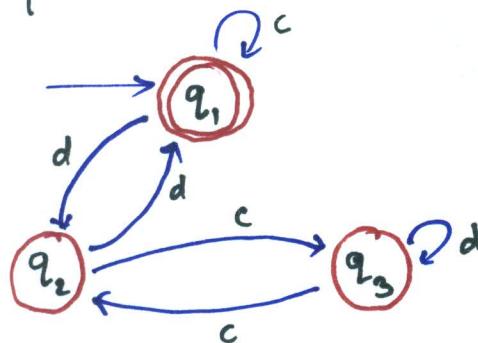
$$\therefore a \in e, b \in h$$



- 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
$\xrightarrow{*} q_1$	q_1	q_2
q_2	q_3	q_1
q_3	q_2	q_3
$\xrightarrow{*} q_4$	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

$$\begin{array}{l}
 q_1 \\
 \swarrow \quad \searrow \\
 q_2 \\
 \swarrow \quad \searrow \\
 q_3 \\
 \swarrow \quad \searrow \quad \swarrow \quad \searrow \\
 q_4 \\
 \swarrow \quad \searrow \quad \swarrow \quad \searrow \\
 q_5 \\
 \swarrow \quad \searrow \quad \swarrow \quad \searrow \quad \swarrow \quad \searrow \\
 q_6 \\
 \swarrow \quad \searrow \quad \swarrow \quad \searrow \quad \swarrow \quad \searrow \\
 q_7
 \end{array}$$

$q_5 \approx q_6$, $q_1 \not\approx q_4 \Rightarrow$ 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					
v	v	c					
-	-	v	d				
-	-	v	-	e			
-	-	v	-	-	f		
-	-	v	-	-	-	g	
-	-	v	-	-	-	-	h

Step 2:

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

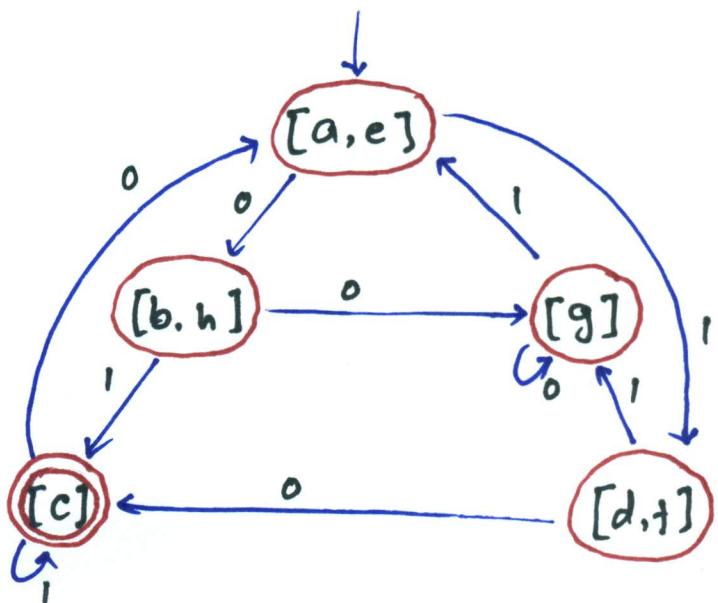
Step 3:

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

$a \approx e$

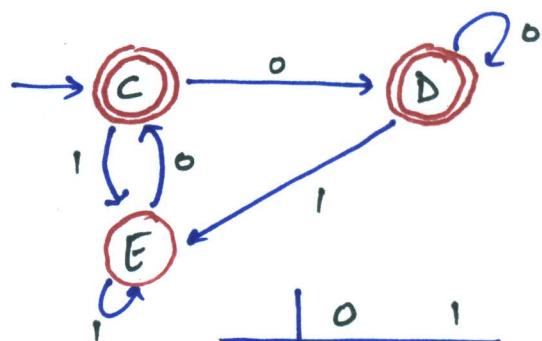
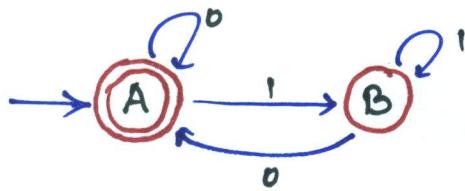
$d \approx f$

$b \approx h$



Example:

Check the equivalence of the given DFA's



A		
✓	B	
-	-	C
-	✓	-
✓	-	✓
		✓
		=
		E

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

$$\underline{A \approx C}, \quad \underline{A \approx D}, \quad C \approx D, \quad B \approx E$$

∴ the two DFA's are equivalent

Note 1: If $\{p, q\}$ is marked in Step 2, then p, q are surely not equivalent. Take $x \approx \epsilon$ 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 entire pass through the table with no new marks.

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 \rightarrow all accepts only regular sets.

2-DFA (two-way deterministic finite automata)

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

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

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

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

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

- $M = (Q, \Sigma, \delta, q_0, F) \rightarrow 2\text{-DFA}$

- $ID \in \Sigma^* Q \Sigma^*$

- relation on ID's $\rightarrow \overleftarrow{t_M}$

$I_1 \overleftarrow{t_M} I_2$ iff M can go from the instantaneous description I_1 to I_2 in one move

When M is understood, we say $I_1 \overleftarrow{t} I_2$

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

Relation \vdash_M

1. $a_1 a_2 \dots a_{i-1} q a_i \dots a_n \vdash_M a_1 a_2 \dots a_{i-1} a_i p a_{i+1} \dots a_n$

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

2. $a_1 a_2 \dots a_{i-2} a_{i-1} q a_i \dots a_n \vdash_M a_1 a_2 \dots a_{i-2} p a_{i-1} a_i \dots a_n$

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. $I \vdash^* I \quad \vee \quad ID's \quad I \in \Sigma^* Q \Sigma$

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

for some I_2, \dots, I_{k-1}

We define

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

input tape containing $w = a_1 a_2 \dots a_l$

a_1	a_2	\dots	a_l
-------	-------	---------	-------

↑
left end of w

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)

1 0 1 0 0 1

$q_0 101001 \xrightarrow{*} 101001P$

when $p \in F$

$q_0 101001 \xrightarrow{} 1q_0 001$
 $\xrightarrow{} 10q_1 001$
 $\xrightarrow{} 1q_2 01001$
 $\xrightarrow{} 10q_0 1001$
 $\xrightarrow{} 101q_1 001$
 $\xrightarrow{} 1010q_2 01$
 $\xrightarrow{} 10100q_1 1$
 $\xrightarrow{} 1010q_2 01$
 $\xrightarrow{} 10100q_0 1$
 $\xrightarrow{} 101001q_1$

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.

q_1
 q_2
 q_3
 q_4

q_1
 q_2
 q_3
 q_4

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

$q_1, q_3, \dots \rightarrow$ right moves

$q_2, q_4, \dots \rightarrow$ left moves

input accepted \rightarrow all crossing sequences of odd length

of crossing sequence \rightarrow finite

Theorem:

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

Proof:

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

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

Finite automata with output

- FA \rightarrow output $\in \{\text{accept, reject}\}$

Other alphabet for output of FA

Moore machine
(output associated
with the state)

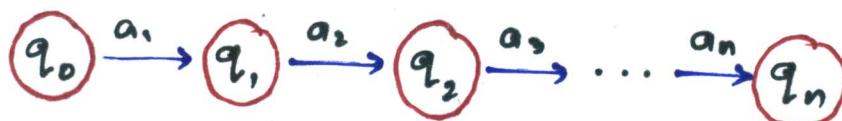
Mealy machine
(output associated with
the transition)

Moore Machines

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

↓ ↓
output alphabet

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



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

input: a, a_2, \dots, a_n

output: $\underbrace{\lambda(q_0), \lambda(q_1), \dots, \lambda(q_n)}$

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 $\underline{\lambda(q)} = 1$

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

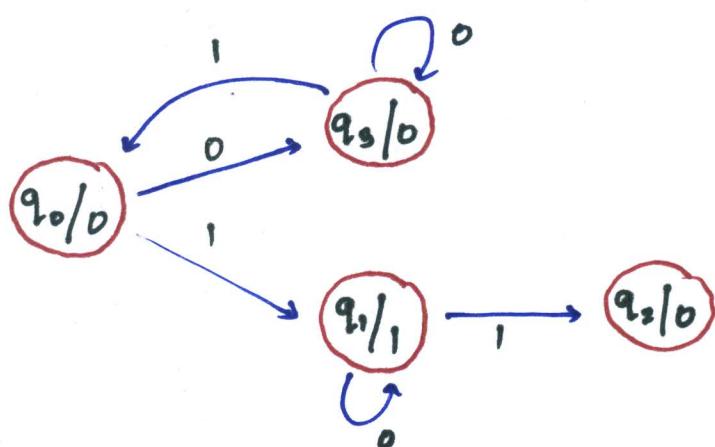
input: 0111 (length 4)



output:

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

(length 5)



Mealy Machine

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

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



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

input: a_1, a_2, \dots, a_n

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

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



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
 $T_M(w) \neq T_{M'}(w)$ 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
 $\lambda(q_0) \lambda(q_1) \dots \lambda(q_k)$ for input string $a_1 \dots a_k$
where $q_0 \xrightarrow{a_1} q_1 \rightarrow \dots \xrightarrow{a_k} q_k$)
- 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 $M_1 = (\Delta, \Sigma, \Delta, \delta, \lambda, q_0)$ is a Moore machine, then there is a Mealy machine M_2 equivalent to M_1 ,

Proof:

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

where

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

Example: Given Moore machine

$\delta(q, a)$	$a=0$	$a=1$	$\lambda(q)$
M_1	q_3	q_1	0
	q_1	q_2	1
	q_2	q_3	0
	q_3	q_0	0

Equivalent Mealy Machine

$\delta(q, a)$	$a=0$	$\lambda'(q, 0)$	$a=1$	$\lambda'(q, 1)$
M_1	q_3	$\xrightarrow{0}$	0	1
	q_1	$\xrightarrow{1}$	1	0
	q_2	$\xrightarrow{0}$	0	0
	q_3	$\xrightarrow{0}$	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

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

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

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

Example:

Given Mealy Machine M,

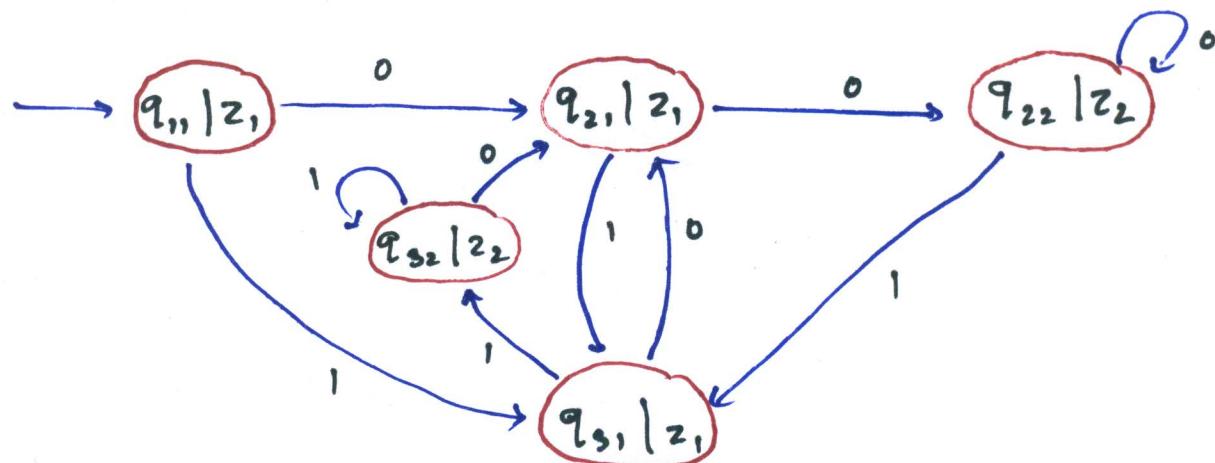
$\delta(q, a)$	$a=0$	$\lambda(q, 0)$	$a=1$	$\lambda(q, 1)$
q_1	q_2	z_1	q_3	z_1
q_2	q_2	z_1	q_3	z_1
q_3	q_2	z_1	q_3	z_2

Equivalent Moore Machine (M_2)

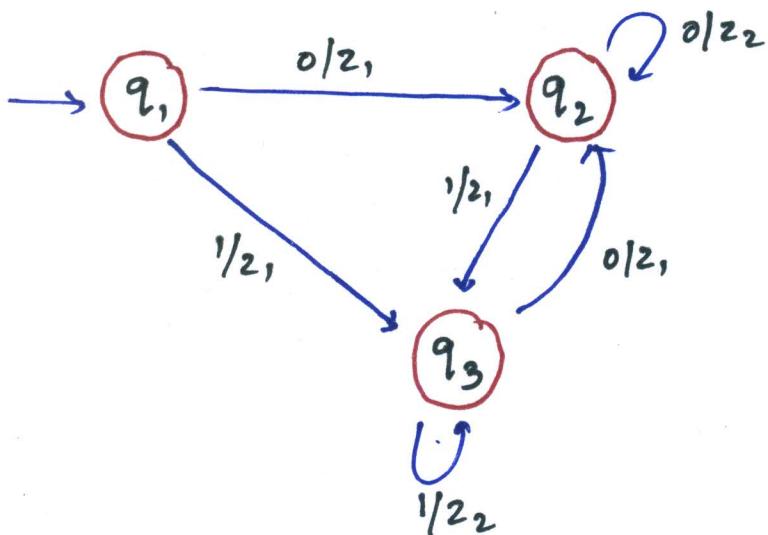
$\delta'(q, a)$	$a=0$	$a=1$	output λ'
$\{q_1, z_1\}$	$\{q_2, z_1\}$	$\{q_3, z_1\}$	z_1
$\{q_3, z_1\}$	$\{q_2, z_1\}$	$\{q_3, z_2\}$	z_1
$\{q_2, z_1\}$	$\{q_2, z_2\}$	$\{q_3, z_1\}$	z_1
$\{q_3, z_2\}$	$\{q_2, z_1\}$	$\{q_3, z_2\}$	z_2
$\{q_2, z_2\}$	$\{q_2, z_2\}$	$\{q_3, z_1\}$	z_2

- q_1 - not splitted
- q_2, q_3 - splitted

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



Moore machine
constructed from the
Mealy machine given below

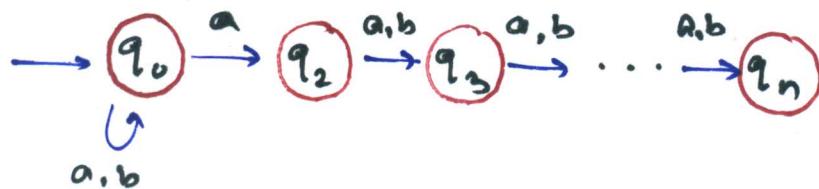


Mealy machine

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 $\rightarrow 2^n$ states

2- DFA ($n+1$ 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: $L = (a+b)^* a (a+b)^{n-1} a (a+b)^n$



Equivalent DFA: $2^n + 1$ states

2- DFA: $2n +$ states

Example:

Given Mealy Machine (M_1)

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

Equivalent Moore machine (M_2)

output

- 2 -

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