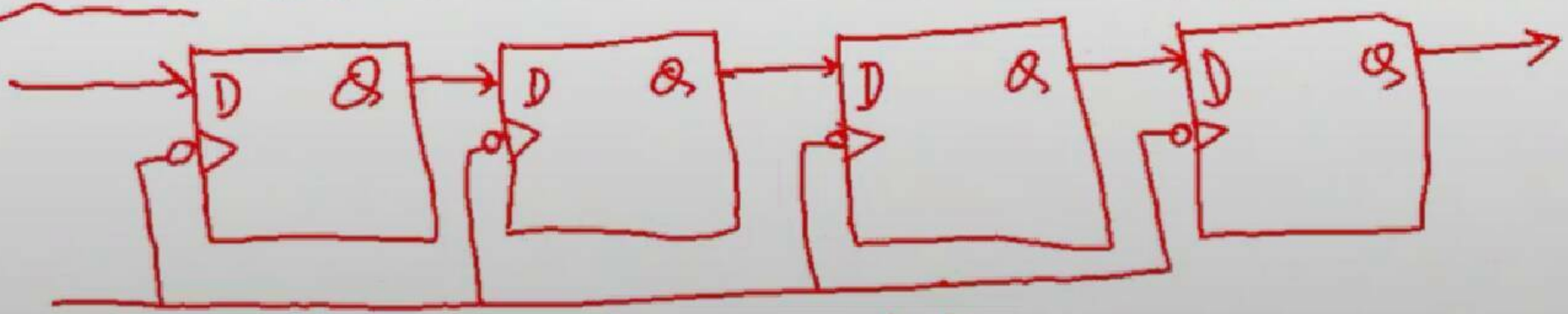
(ii) Output: $O_1, O_2, \cdots O_q$ are the outputs which can take fixed values from the set O.

(iii) States: At any instant the automaton can be in one of the states $q_1, q_2 \cdots q_n$

(iv) State relation: The next state is determined by the present state & present input

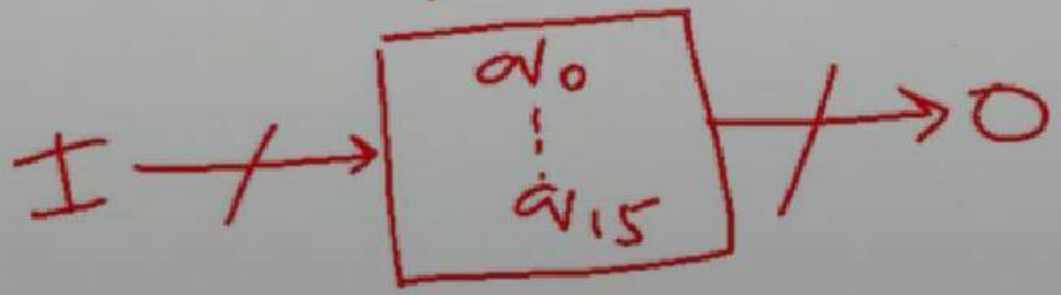
(v) Output relation: The output is related to either state only or both the input & the state

Example: Consider a shift register



There are $2^4 = 16$ states
(0000 → 1111)

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



Definition of Finite Automata

An automata can be defined as five tuples

$Q \rightarrow$ finite set of states

$\Sigma \rightarrow$ finite set of input symbols called alphabet

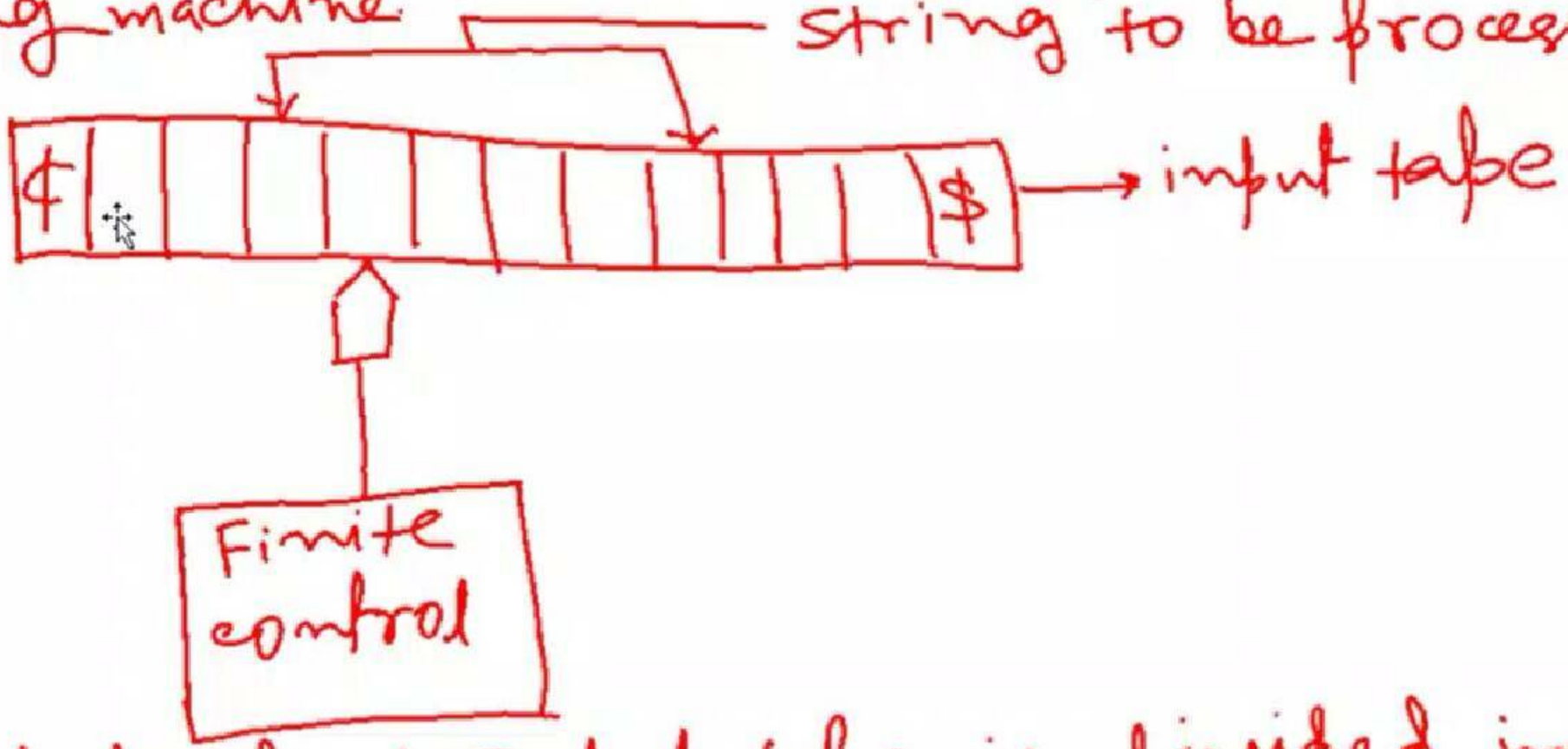
$\delta \rightarrow$ is called transition function

$$\delta : Q \times \Sigma \longrightarrow Q$$

$q_0 \rightarrow$ initial state, where $q_0 \in Q$

$F \subseteq Q$ is the set of final states.

The automaton can be visualized as the following machine



Input tape: Input tape is divided into squares, each square containing a symbol from the alphabet Σ.
¢ → start marker, $ → End marker

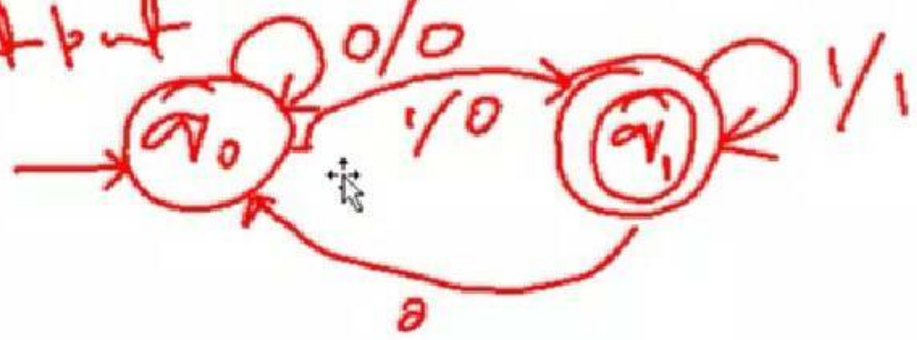
Reading head: Reading head examines one symbol at a time.
(i.e. one square)
It can move either right or left. (We restrict the movement to the right for now).

Finite control: Input to the finite control is a symbol under head and the present state of the machine.
- output of the finite control is
  (a) A motion of the R-head to the next square or a null move.
  (b) The next state is given by $\delta(q,q)$

Transition Systems

A transition graph or transitia system is a visual representation of an automaton.

- A transitia system consists of states and possible transitions shown by directed edges labelled by input/output



- States are represented as a single circle and final states are represented as double circle.
- The δ is represented as

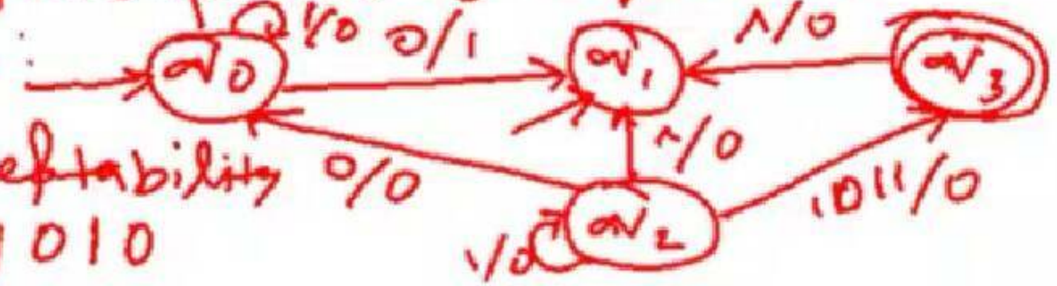
$$Q \times Z^* \times Q$$

i.e (q_1, w, q_2) → means automaton at state q_1 after consuming the string w moves to q_2.

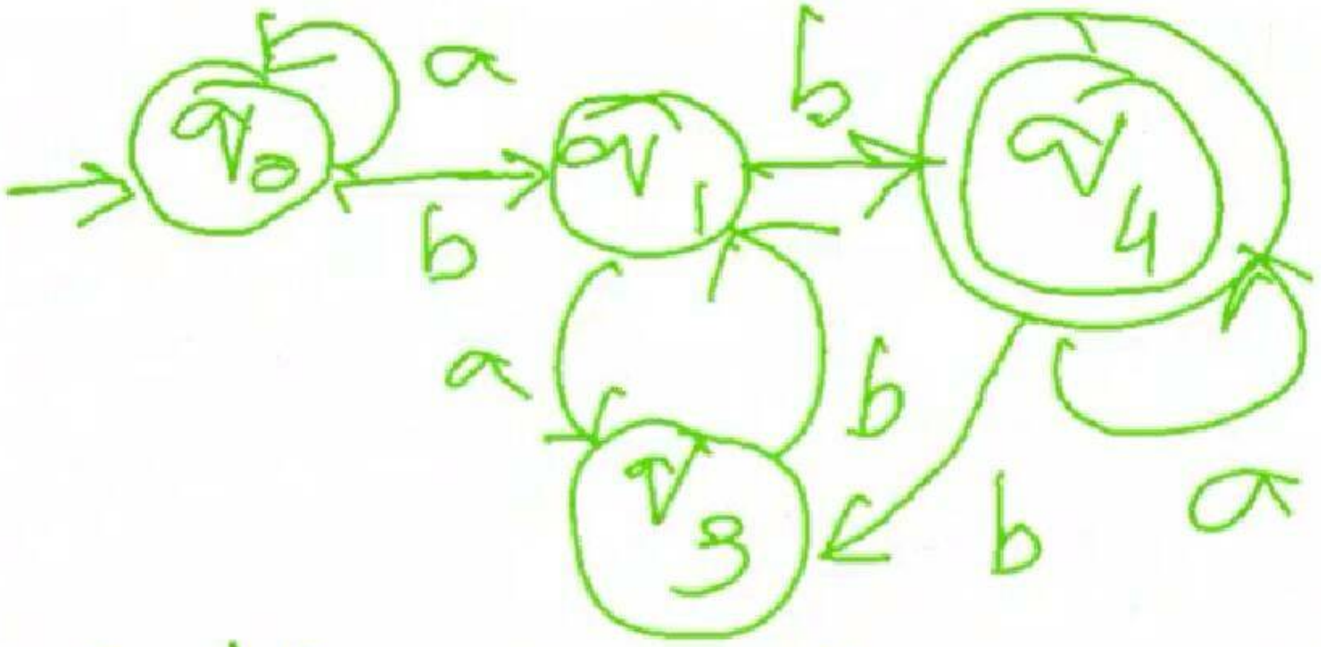
Acceptance of a string

A transition system accepts a string w in Z^* if
(a) there exists a path from initial state to some final state labelled with symbols from w.
(b) The path value obtained by concatenation of all edge labels of the path is equal to w.

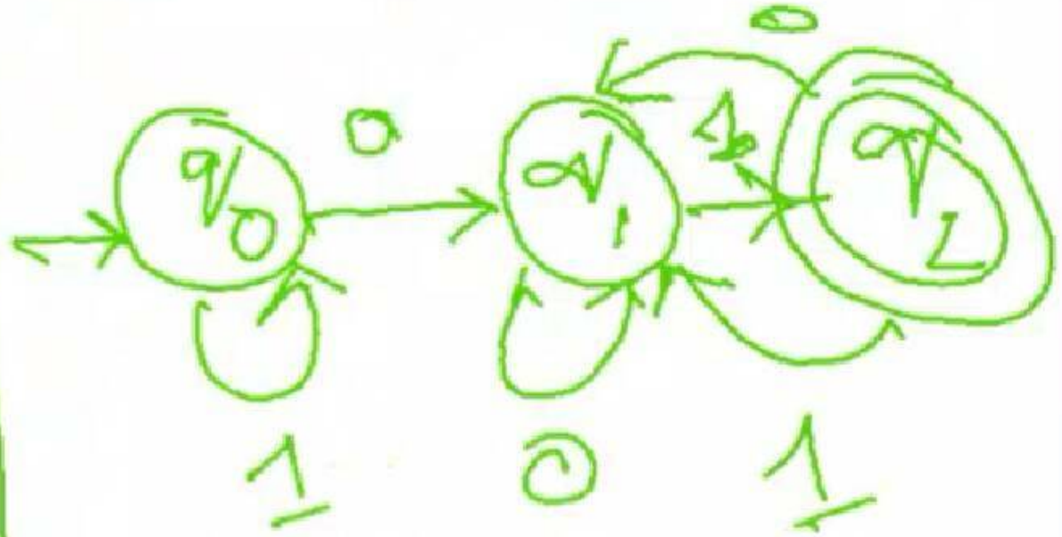
Example:



Determine acceptability of 10101, 111010



Define the language



test for the following

0 1 1 0 1 1

0 1 0 1 0 1 0

1 1 0 1 0 0

1 0 1 1 0 0

properties of transition function

1. $\delta(q, \lambda) = q$, in a finite automaton
2. for all strings w and an input symbol a

$$\delta(q, aw) = \delta(\delta(q, a), w)$$

&

$$\delta(q, wa) = \delta(\delta(q, w), a)$$

first one gives a state after scanning the first symbol of aw.

second property gives a state after scanning the prefix of the string wa.

problem: Show that for a transition function δ and any two strings x & y

$$\delta(q, xy) = \delta(\delta(q, x), y)$$

Proof: let $|y|=1$ and $y=a \in \Sigma$

from second property

L.H.S $\delta(q, xa) = \delta(\delta(q,x), a) =$ R.H.S

Let the rule is true for $|y_1|=n$, where $y=y_1 a$ i.e.

$|y| = n+1$

L.H.S $\delta(q, xy_1 a) = \delta(q, x_1 a)$

$= \delta(q, x_1 a)$ where $x_1 = xy_1$

from property 2

L.H.S. $\delta(\delta(q, x_1), a)$

$= \delta(\delta(q, xy_1), a)$ from induction hypothesis

L.H.S $\delta(\delta(\delta(q,x), y_1), a)$

R.H.S $\delta(\delta(q,x), y) = \delta(\delta(q,x), y_1 a)$

from property 2, $= \delta(\delta(\delta(q,x), y_1), a)$

Problem: prove that if $\delta(q, x) = \delta(q, y)$, then $\delta(q, xz) = \delta(q, yz)$

Sol: $\delta(q, xz) = \delta(\delta(q, x), z)$

$= \delta(\delta(q, y), z)$

$= \delta(q, yz)$

Acceptability of string by a finite automaton

Definition: A string is said to be accepted by a finite automaton $M = (Q, \Sigma, \delta, q_0, F)$ if $\delta(q_0, x) = q$, for some $q \in F$.

This is basically the acceptability of a string by the final state.

Problem: Consider a machine where δ is defined as

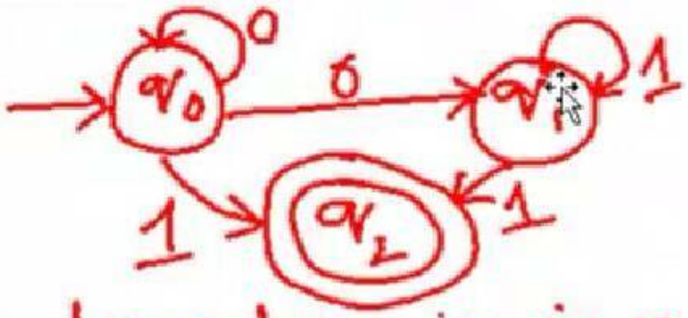
State	input 0	input 1
→ q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

determine whether string 1100 is accepted or not

$F = \{q_0\}$, $\Sigma = \{0, 1\}$

Nondeterministic finite state machine

Let us consider the following transition diagram (transition system)



- If the automaton is in a state $\{q_0\}$ and the input symbol is 0, what will be the next state?
- Thus some moves of the machine cannot be determined uniquely by the input symbol and present state.

Definition

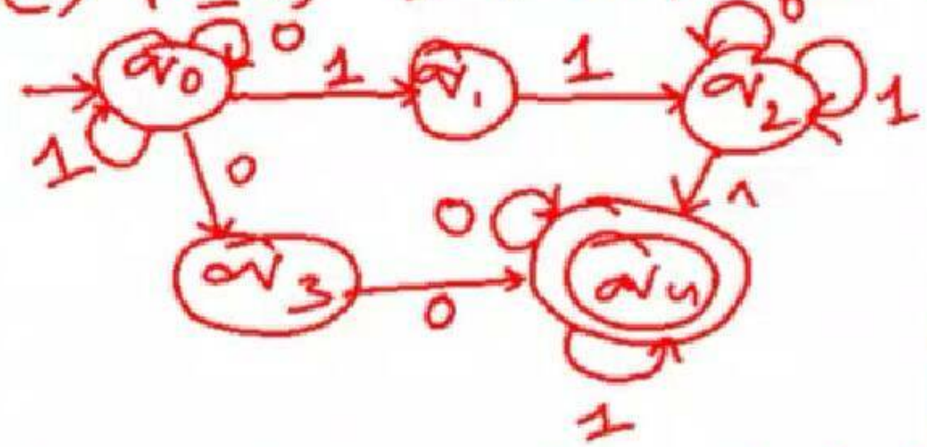
A nondeterministic finite automaton (NDFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$. The only difference is in δ.

(i) Q is a finite non empty set of states.
(ii) Σ is a finite non-empty set of input symbol.
(iii) δ is the transition function defined as

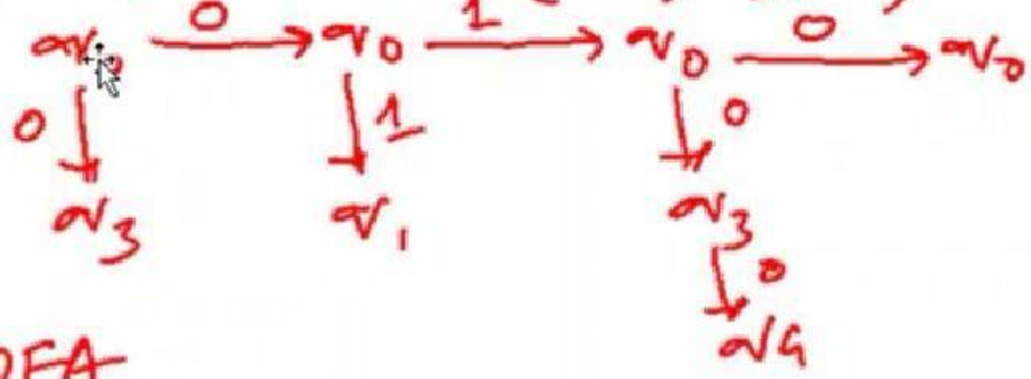
$$\delta : Q \times \Sigma \longrightarrow 2^Q$$

(iv) $q_0 \in Q$ is the initial state.

(v) $F \subseteq Q$ is the final state.



$\delta(q_0, 0100) = \{q_0, q_3, q_4\}$



Equivalence of DFA & NDFA

- A DFA can simulate the behaviour of NDFA by increasing the number of states.
- In other words, a DFA $(Q, \Sigma, \delta', q_0, F)$ can be viewed as an NDFA $(Q, \Sigma, \delta, q_0, F)$ by defining $\delta'(q, a) = \{\delta(q, a)\}$

NDFA is more general machine without being more powerful.

The transition for NDFA

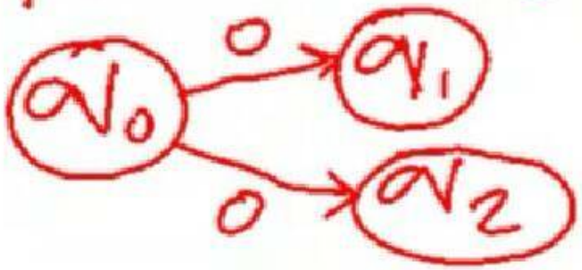
$\delta : \Sigma \times Q \rightarrow 2^Q$

The corresponding DFA

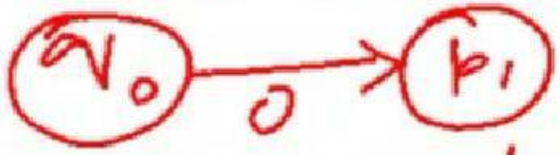
$\delta' : \Sigma \times Q' \rightarrow Q'$

$\therefore Q' \subseteq 2^Q$

If for an NDFA



The the corresponding DFA will have



where $p_1 = [q_1, q_2]$

Theorem: For every NDFA, there exists a DFA which simulates the behaviour of NDFA.

Alternatively, if L be the set of strings accepted by NDFA, then there exists a DFA which also accepts L.

Proof: Let $M = (Q, \Sigma, \delta, q_0, F)$ be an NDFA accepting L.

Let us construct a DFA as follows

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

where

(i) $Q' = 2^Q$ (Any state in Q' is denoted by $(q_1, q_2 \cdots q_j)$ where $q_1, q_2, \cdots q_j \in Q$)

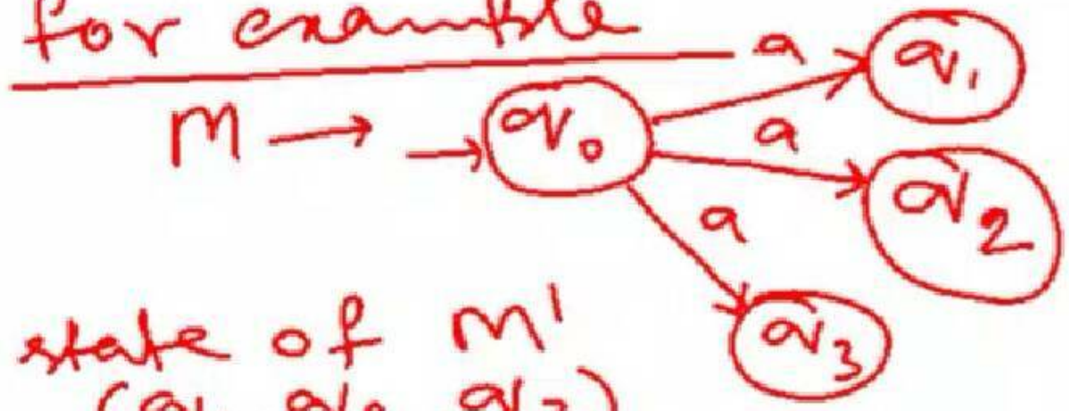
(ii) $q_0' = [q_0]$

(iii) F' is the set of all subsets of Q containing an element of F.

- On applying any input a to M at state q_0, M can reach to state $\delta(q_0, a)$.
- The state of corresponding DFA, M' on input a at

state q_0 is a set of all states that can be reached by $\delta(q_0, a)$ on M.

for example



state of M'
(q_1, q_2, q_3)

- Hence the states of M' are defined as subsets of Q. The M' should also start with the initial state of q_0, Hence $q_0' = [q_0]$
- On processing any string w by M, if M, reaches a final state then $w \in T(M)$
- Hence final state in M' should be any of the final states of M.

(iv) Therefore, δ' can be defined as

$$\delta'([q_1, q_2 \cdots q_j], a) = \delta(q_1, a) \cup \delta(q_2, a) \cdots \cup \delta(q_{j-1})$$

Now let us prove that

$$\delta'(q_0', x) = [q_1, q_2, \cdots q_i] \text{ iff } \delta(q_0, x) = [q_1, q_2, \cdots q_i]$$

– we prove it by induction on $|x|$

when $|x| = 0$, $\delta(q_0, \wedge) = q_0$

and by definition of δ'

$$\delta'(q_0', \wedge) = q_0' = q_0$$

hence $\delta'(q_0', x) = q_i$ if $\delta(q_0, x) = q_i$ for $|x| = 0$

Let the hypothesis is true for all strings y with $|y| \leq m$

Let $\delta(q_0, y) = [p_1, p_2, \cdots p_i]$ & $\delta(\delta(q_0, y), a) = [r_1, r_2 \cdots r_k]$

where $|ya| = m+1$ and $|y| = m$

$$\delta([p_1, p_2 \cdots p_i], a) = [r_1, r_2 \cdots r_k]$$

Now let us see

$\delta'(q_0', ya) = ?$

we have $\delta'(q_0', y) = [p_1, p_2 \cdots p_i] = \delta(q_0, y)$

from hypothesis

$\delta'(q_0', ya) = \delta'(\delta'(q_0', y), a)$

$= \delta'([p_1, p_2 \cdots p_i], a)$

$= \delta'(p_1, a) \cup \delta'(p_2, a) \cdots \cup \delta'(p_i, a)$

$= \delta(p_1, a) \cup \delta(p_2, a) \cdots \cup \delta(p_i, a)$

$= [r_1, r_2 \cdots r_k]$

Hence it is proved for $x = ya$.

Similarly,

if $\delta'(q_0', x) = [q_1, q_2 \cdots q_i]$

then $\delta(q_0, x) = [q_1, q_2 \cdots q_i]$

$\delta'(q_0', y) = [p_1, p_2, \cdots p_i]$

from hypothesis

$$\delta(q_0, y) = [p_1, p_2 \cdots p_i]$$

$$\delta'(\delta'(q_0, y), a) = \delta'([p_1, p_2 \cdots p_i], a) = [r_1, r_2 \cdots r_k] \quad -(1)$$

(1) is possible only if

$$\delta(p_1, a) = r_1, \ \delta(p_2, a) = r_2 \cdots \delta(p_i, a) = r_k$$

Since from definition

$$\delta'([p_1, p_2 \cdots p_i], a) = \delta(p_1, a) \cup \delta(p_2, a) \cdots \cup \delta(p_i, a)$$

Problem: construct a deterministic automaton equivalent to $M = (\{q_0, q_1\}, \{0, 1\}, \delta, q_0, [q_0])$. δ is given as follows

State/Σ	0	1
→ ((q_0))	q_0	q_1
q_1	q_1	q_0, q_1

Sol: For deterministic automaton M_1,

(i) The states are subsets of $\{q_0, q_1\}$ i.e. ϕ, $[q_0]$, $[q_1]$, $[q_0, q_1]$.

(ii) Final states are $[q_0]$ and $[q_0, q_1]$

(iii) $[q_0]$ is the initial state.

(iv) δ_1 for M_1 is defined as,

state/Σ	0	1
ϕ	ϕ	ϕ
q_0	q_0	q_1
q_1	q_1	$[q_0, q_1]$
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_1]$

$\delta_1([q_0, q_1], 0) = \delta(q_0, 0) \cup \delta(q_1, 0) = [q_0, q_1]$

Similarly

$\delta_1([q_0, q_1], 1) = \delta(q_0, 1) \cup \delta(q_1, 1) = [q_1, q_0, q_1] = [q_0, q_1]$

No new states are there, hence we stop here.

When M has n states, the corresponding deterministic automaton has 2^n states. However, we need not construct transition (δ_1) for all 2^n states, but only for those states reachable from $[q_0]$.

state/Σ	0	1
ϕ	ϕ	ϕ
q_0	q_0	q_1
q_1	q_1	$[q_0, q_1]$
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_1]$

$\delta_1([q_0, q_1], 0) = \delta(q_0, 0) \cup \delta(q_1, 0) = [q_0, q_1]$

Similarly

$\delta_1([q_0, q_1], 1) = \delta(q_0, 1) \cup \delta(q_1, 1) = [q_1, q_0, q_1] = [q_0, q_1]$

No new states are there, hence we stop here.

When M has n states, the corresponding deterministic automaton has 2^n states. However, we need not construct transition (δ_1) for all 2^n states, but only for those states reachable from $[q_0]$.

Problem: Find a deterministic automaton (acceptor) equivalent to $M = (\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_2\})$ and δ is defined as

state/Σ	a	b
$\rightarrow q_0$	q_0, q_1	q_2
q_1	q_0	q_1
(q_2)	ϕ	q_0, q_1

Sol: Let the deterministic automaton M' equivalent to M is defined as

$M' = (2^Q, \{a, b\}, \delta', [q_0], F')$

where possible F' is $\{[q_2], [q_0, q_2], [q_1, q_2], [q_0, q_1, q_2]\}$

$\delta'(q_0, a) = \delta(q_0, a) = [q_0, q_1]$

$\delta'(q_0, b) = \delta(q_0, b) = [q_2]$

$\delta'(q_1, a) = \delta(q_1, a) = q_0$

$\delta'(q_1, b) = \delta(q_1, b) = q_1$

$\delta'(q_2, a) = \delta(q_2, a) = \phi$

$\delta'(q_2, b) = \delta(q_2, b) = [q_0, q_1]$

$\delta'([q_0, q_1], a) = \delta(q_0, a) \cup \delta(q_1, a) = [q_0, q_1] \cup [q_0]$

$= [q_0, q_1]$

$\delta'([q_0, q_1], b) = \delta(q_0, b) \cup \delta(q_1, b) = [q_2] \cup [q_1]$

$= [q_1, q_2]$

$\delta'([q_1, q_2], a) = \delta(q_1, a) \cup \delta(q_2, a) = [q_0] \cup \phi = [q_0]$

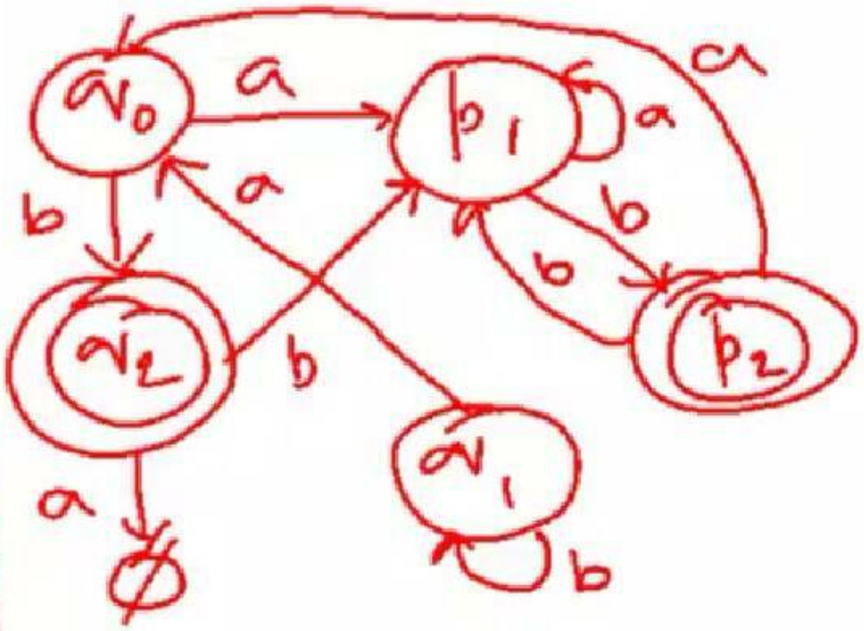
$\delta'([q_1, q_2], b) = \delta(q_1, b) \cup \delta(q_2, b) = [q_1] \cup [q_0, q_1]$

$= [q_0, q_1]$

So the δ' can be defined as

States / Σ	a	b
ϕ	ϕ	ϕ
q_0	$[q_0, q_1]$	q_2
q_1	q_0	q_1
q_2	ϕ	$[q_0, q_1]$
$[q_0, q_1]$	$[q_0, q_1]$	$[q_1, q_2]$
$[q_1, q_2]$	$[q_0]$	$[q_0, q_1]$

Let $[q_0, q_1] = p_1$ &
$[q_1, q_2] = p_2$



Finite automata with output

- The acceptability/rejection these two outputs were possible for the machines we considered so far.
- Now, let us consider the machines which do not have restriction over reachability to final state and output can be chosen from some alphabet
- This output function is dependent either on current state and current input both or current state only.

i.e. if λ is the output function then

either $\lambda(q(t), x(t)) \rightarrow$ (Mealy machine generalized)

or $\lambda(q(t)) \rightarrow$ (Moore machine restricted).

where $q(t)$ is the current state

and $x(t)$ is current input.

Moore Machine

Moore machine is defined using 6-tuples

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

$Q \rightarrow$ finite state

$\Sigma \rightarrow$ input alphabet

$\Delta \rightarrow$ output alphabet

$\delta \rightarrow$ transition function

$$\Sigma \times Q \rightarrow Q$$

$\lambda \rightarrow$ output function

$$\lambda : Q \rightarrow \Delta$$

$q_0 \rightarrow$ initial state

Example

Present state	Next state		output
	$a=0$	$a=1$	
$\rightarrow q_0$	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

for input string 0111 the transition states given by output is

$q_0 \rightarrow$	$q_3 \rightarrow$	$q_0 \rightarrow$	$q_1 \rightarrow$	q_2
0	0	0	1	0

Mealy Machine

The Mealy machine also have 6-tuples, the only difference is in λ defined as

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

Example

Present state	Next state			
	$a=0$		$a=1$	
	state	o/p	state	o/p
$\rightarrow q_1$	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

for the i/p 0011 we have

$q_1 \rightarrow q_3 \rightarrow q_2 \rightarrow q_4 \rightarrow q_3$

o/p 0 1 0 0

Note: In Mealy machine we get the output only on the input to the machine state

- A finite automaton can be converted into a Moore machine by introducing $\Delta = \{a, b\}$ and defining $\lambda(q) = a$, if $q \in F$ and $\lambda(q) = b$ if $q \notin F$
- Moore machine has output string of length n+1 for i/p of length n.
- Mealy has the same o/p length as the i/p length.

for the i/p 0011 we have

$q_1 \rightarrow q_3 \rightarrow q_2 \rightarrow q_4 \rightarrow q_3$

o/p 0 1 0 0

Note: In Mealy machine we get the output only on the input to the machine state

- A finite automaton can be converted into a Moore machine by introducing $\Delta = \{a, b\}$ and defining $\lambda(q) = a$, if $q \in F$ and $\lambda(q) = b$ if $q \notin F$
- Moore machine has output string of length n+1 for i/p of length n.
- Mealy has the same o/p length as the i/p length.

associated with each state q_i in the next state column. The state q_i then split into the number of states equal to the outputs associated with q_i in Mealy machine.

In the given Mealy machine q_1, q_3 are associated outputs 1 and 0 respectively.

q_2, q_4 are associated with 0, 1.

Hence we split q_2 into q_{20}, q_{21} and q_4 into q_{40} and q_{41}.

Hence the table is modified as shown in the next slide.

present state	Next state			
	input $a=0$		input $a=1$	
	state	output	state	output
$\rightarrow q_1$	q_3	0	q_{20}	0
q_{20}	q_1	1	q_{40}	0
q_{21}	q_1	1	q_{40}	0
q_3	q_{21}	1	q_1	1
q_{40}	q_{41}	1	q_3	0
q_{41}	q_{41}	1	q_3	0

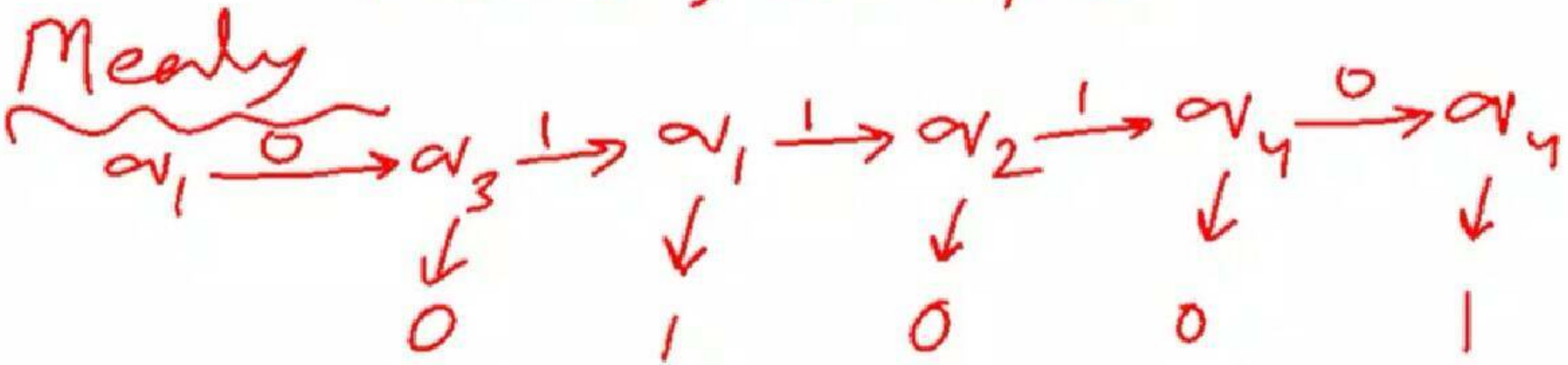
The previous table can be rearranged as

Present state	Next state		output
	input $a=0$	$a=1$	
$\rightarrow q_1$	q_3	q_{20}	1
q_{20}	q_1	q_{40}	0
q_{21}	q_1	q_{40}	1
q_3	q_{21}	q_1	0
q_{40}	q_{41}	q_3	0
q_{41}	q_{41}	q_3	1

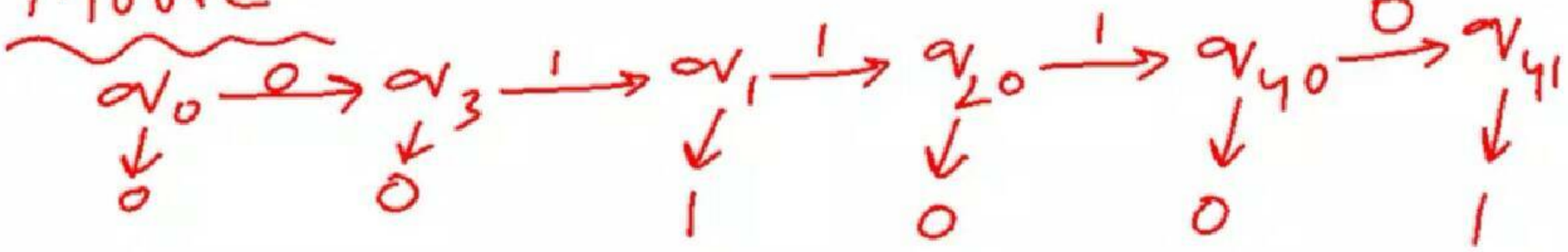
Now this Moore machine accepts a null string as it outputs 1 for $\wedge$ with initial state q_1. In Mealy machine $\wedge$ string is not allowed. Hence, either we can neglect the response for

Example: Show how output is generated for the input 01110 by Mealy and the corresponding Moore machines of previous example.

Mealy



Moore



Procedure to transform Moore into Mealy

A Moore machine M' is equivalent to a Mealy machine M,
if $b\,Z_M(w) = Z_{M'}(w)$,
where $Z(\cdot)$ is the output string function for M and M' for input string w, and b is the output for $\wedge$ string accepted by Moore machine.

Let the Moore machine

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

Construction

We need to define the output function λ' of Mealy machine using output function λ of Moore machine as

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

Where $\delta \rightarrow$ is the transition function of M'.

Problem: Construct an equivalent Mealy machine for Moore machine given below.

P. state	M. state		o/p
	a=0	a=1	
$\rightarrow q_0$	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

Solution: For q_0 in Moore machine and next state q_3 output is

$$\lambda'(q_0, 0) = \lambda(\delta(q_0, 0)) = \lambda(q_3) = 0$$

As q_3 is the state for input 0 at q_0.

Similarly,

$\lambda'(q_0, 1) = \lambda(\delta(q_0, 1)) = \lambda(q_1) = 1$

For present state q_1 in Moore machine

$\lambda'(q_1, 0) = \lambda(\delta(q_1, 0)) = \lambda(q_1) = 1$

$\lambda'(q_1, 1) = \lambda(\delta(q_1, 1)) = \lambda(q_2) = 0$

For present state q_2 in Moore machine

$\lambda'(q_2, 0) = \lambda(\delta(q_2, 0)) = \lambda(q_2) = 0$

$\lambda'(q_2, 1) = \lambda(\delta(q_2, 1)) = \lambda(q_3) = 0$

For q_3 in Moore

$\lambda'(q_3, 0) = \lambda(\delta(q_3, 0)) = \lambda(q_3) = 0$

$\lambda'(q_3, 1) = \lambda(\delta(q_3, 1)) = \lambda(q_0) = 0$

So the transition of the corresponding Mealy machine

P. State	M. State			
	input a=0		input a=1	
	state	o/p	state	o/p
q_0	q_3	0	q_1	1
q_1	q_1	1	q_2	0
q_2	q_2	0	q_3	0
q_3	q_3	0	q_0	0

Q) Let $M=(Q,\Sigma,\delta,q_0,F)$ be a finite automaton. Let R be a relation in Q defined by $q_1 R q_2$, if $\delta(q_1,a)=\delta(q_2,a)$, for some $a\in\Sigma$. Is R an equivalent relation.

Q) Construct an NDFA accepting {ab,ba} and use it to find a deterministic automaton accepting the same string.

Q) The transition table for an NDFA is in the next slide. construct the equivalent DFA

state	input		
	0	1	2
→ q_0	q_1, q_4	q_4	q_2, q_3
q_1		q_4	
q_2			
(q_3)			q_4
q_4			

State minimization in automaton

The objective is to construct an automaton M' with minimum number of states from the given automaton M.

Definition: Two states are called equivalent i.e q_1 and q_2 are $q_1 \equiv q_2$, if either both $\delta(q_1, x)$ & $\delta(q_2, x)$ are final states, or both are non final states, $\forall x \in \Sigma^*$.

Note: It is very difficult to check for all $x \in \Sigma^*$ whether $\delta(q_1, x)$ & $\delta(q_2, x)$ are final or non final states.

Modified definition

Two states q_1 and q_2 are k equivalent ($k \geq 0$) if both $\delta(q_1, x)$ and $\delta(q_2, x)$ are either non final or final states, $\forall x \in \Sigma^*$ such that $|x| \leq k$.

Base case: Every final or non final states are 0-equivalent, i.e. any two final or non final states are 0-equivalent.

Properties of k-equivalence relations

1. These k-equivalence relations are equivalence relations i.e. reflexive, symmetric, & transitive.

2. These k-equivalence relations create partition over Q. These partitions can be denoted as Π or Π_k where Π_k denotes partitions of k-equivalence classes.

3. If q_1 & q_2 are k equivalent for all $k \geq 0$, then they are equivalent.

4. If q_1 & q_2 are $(k+1)$ equivalent then they are k-equivalent.

5. $\Pi_n = \Pi_{n+1}$, $\exists n$ denotes that there can be no further partitions.
(TTn denotes the set of equivalence classes under n-equivalence)