UNIT-II

NONDETERMINISTIC FINITE AUTOMATA WITH ε TRANSITIONS:

- ✓ In the automata theory, a nondeterministic finite automaton (NFA) or nondeterministic finite state machine is a finite state machine where from each state and a given input symbol the automaton may jump into several possible next states.
- ✓ This distinguishes it from the deterministic finite automaton (DFA), where the next possible state is uniquely determined.
- ✓ Although the DFA and NFA have distinct definitions, a NFA can be translated to equivalent DFA using power set construction, i.e., the constructed DFA and the NFA recognize the same formal language.
- ✓ Both types of automata recognize only regular languages. NFAs are sometimes studied by the name subshifts of finite type. NFAs have been generalized multiple ways, e.g., nondeterministic finite automaton with ε-moves, pushdown automaton.
- ✓ In the automata theory, a nondeterministic finite automaton with ε-moves (NFA-ε) is an extension of nondeterministic finite automaton (NFA), which allows a transformation to a new state without consuming or reading any input symbols.
- ✓ The transitions without consuming an input symbol are called ε-transitions. In the state diagrams, they are usually labeled with the Greek letter ε.

SIGNIFICANCE

- ✓ ε-transitions provide a convenient way of modeling the systems whose current states are not precisely known.
- ✓ ε-transitions do not add any extra capacity of recognizing formal languages. NFA-ε's and NFA's recognize same class of formal languages, namely regular languages.
- ✓ NFA-ε's are defined because certain properties can be more easily proved on them as compared to NFA. Since a NFA-ε can always be transformed into a NFA, the properties are also true for NFAs.

Use of ε-transitions

- ✓ To build an NFA that recognizes a set of keywords, the strategy can be simplified further if we allow the ε-transitions.
- ✓ For example consider the NFA recognizing the keywords **state** and **regular**, which is implemented with ε-transitions as in the figure below:

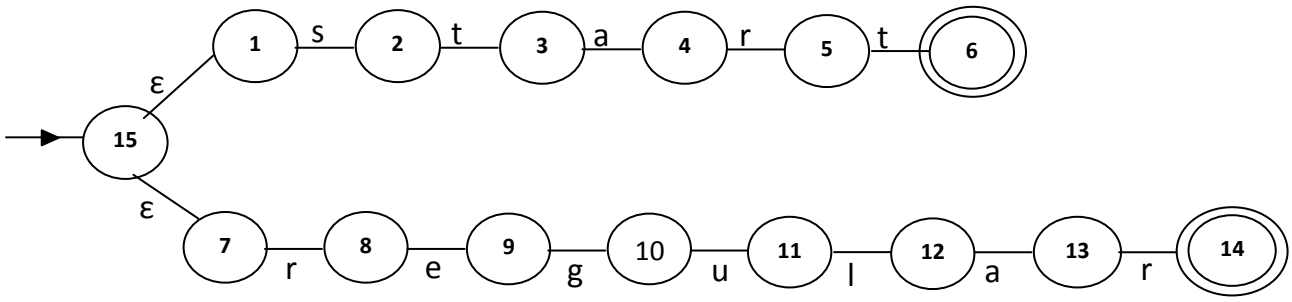


Figure 1: using ε-transitions to help recognize keywords

- ✓ In general we construct a complete sequence of states for each keyword, as if it were the only word the automaton needed to recognize.
- ✓ Then we add a new start state (state 15 in figure:1) with ε-transitions to the start-states of the automata for each of the keywords.

Formal Notation for an ε-NFA

- ✓ We may represent an ε-NFA exactly as we do an NFA, with one exception: the transition function must include information about transitions on ε.
- ✓ Formally we represent an ε-NFA A by A = (Q, Σ, Δ, q_0,F), where all components have their same interpretations as for an NFA, except that Δ is now a function that takes as arguments:
  1. A state in Q, and
  2. A number of Σ ∪ {ε}, i.e. either an input or the symbol ε. We require that ε, the symbol for the empty string, cannot be a member of the alphabet Σ, so no confusion results.
- ✓ In the above ε-NFA A
  1. Q → Finite set of states
  2. Σ → input Alphabet
  3. Δ → Transition function (Δ : Q × (Σ ∪{ε}) → P(Q))
  4. q_0 → start state in Q
  5. F → a set of final states F ⊆ Q.
- ✓ Consider the NFA as in the figure 1.1:

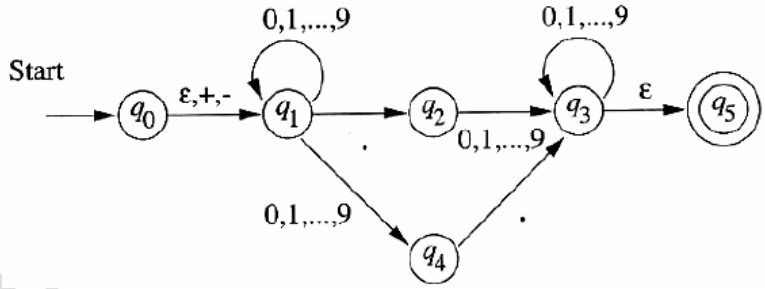


Figure 1. 1: ε-NFA accepting decimal numbers

- ✓ The ε-NFA in the above diagram is represented formally as

$$E = (\{q_0, q_1, \ldots, q_5\}, \{., +, -, 0, 1, \ldots, 9\}, \Delta, q_0, \{q_5\})$$

- ✓ Where Δ is defined by the transition table given in Figure 1.2: given below.

	ε	+, -	.	0, 1, . . . , 9
→q_0	{ q_1}	{ q_1}	ϕ	ϕ
q_1	ϕ	ϕ	{ q_2}	{ q_1, q_4}
q_2	ϕ	ϕ	ϕ	{ q_3}
q_3	{ q_5}	ϕ	ϕ	{ q_3}
q_4	ϕ	ϕ	{ q_3}	ϕ
*q_5	ϕ	ϕ	ϕ	ϕ

Figure 1. 2: Transition table for figure 1.1

- ✓ In the formal representation of NFA, P(Q) denotes the power set of Q and ε denotes empty string.

Equivalence between NFA with and without ε transitions:

- ✓ It can be shown that ordinary NFA and NFA-ε are equivalent, in that, given either one, one can construct the other, which recognizes the same language.

Example:

- ✓ Let *M* be a NFA-ε, with a binary alphabet, that determines if the input contains an even number of 0s or an even number of 1s.
- ✓ Note that 0 occurrences is an even number of occurrences as well. In formal notation, let $M = (\{s_0, s_1, s_2, s_3, s_4\}, \{0, 1\}, \Delta, s_0, \{s_1, s_3\})$ where the transition relation Δ can be defined by the state transition table given below:

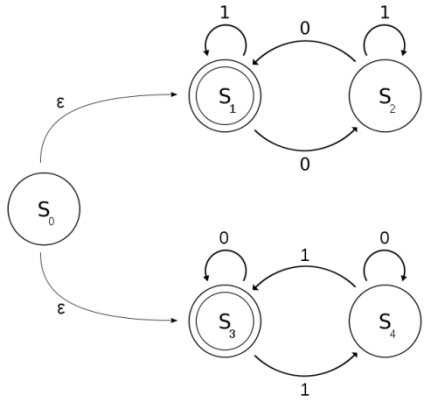


figure 1. 3 State Diagram M

	0	1	ε
S_0	ϕ	ϕ	$\{S_1,S_3\}$
S_1	$\{S_2\}$	$\{S_1\}$	ϕ
S_2	$\{S_1\}$	$\{S_2\}$	ϕ
S_3	$\{S_3\}$	$\{S_4\}$	ϕ
S_4	$\{S_4\}$	$\{S_3\}$	ϕ

Table 1: State Transition Table for M.

- ✓ M can be viewed as the union of two DFAs: one with states {S1, S2} and the other with states {S3, S4}.
- ✓ The language of M can be described by the regular language given by this regular expression (1*(01*01*)*) ∪ (0*(10*10*)*). We define M using ε-moves but M can be defined without using ε-moves.

Epsilon – Closure

- ✓ ε-closure of a state q is the set of states that can be reached from q along a path in which all arcs are labeled with ε.
- ✓ If q is in ε-closure then it is denoted as ε-closure(q). If p is in ε-closure(q) and there is an ε transition from p to r, then r is in ε-closure(q).
- ✓ NFA's are said to be closed under the following operations:
  1. Union
  2. Intersection
  3. Concatenation
  4. Negation
  5. Kleene closure

Kleene star or Kleene operator or Kleene closure is a unary operation, performed either on sets of strings or on sets of symbols or characters. The application of the Kleene star to a set **V** is written as **V***.

It is widely used for regular expressions to characterise certain automaton where it means zero or more.

ACCEPTANCE OF LANGUAGES

- ✓ Like DFA the transition function on an NFA (A) is uniquely determined by A. According to the formal definition of NFA, it is a 5-tuple consisting of A = (Q, Σ, Δ, q_0, F).
- ✓ Let us consider a string **w** over an Σ.
- ✓ **w** is accepted by A if there is an accept state q Є F such that q is reachable from a start state under input string **w** i.e. q Є Δ(q_0,**w**)
- ✓ The language that is accepted by A is denoted by L(A), is the set of all the strings accepted by A. i.e. L(A) = {**w** | **w** Є Σ*}
- ✓ Let M denotes DFA and N denotes NFA. Then Two finite automata M and N are said to be equivalent if L(M) = L(N).
- ✓ Under such definition every DFA i.e. M =(Q, Σ, δ, q_0, F) is equivalent to an NFA i.e. N = (Q, Σ, Δ, q_0,F) where Δ (q, a) = {δ(q, a)} for every state q and input a.

CONVERSIONS AND EQUIVALENCES

Equivalence of Deterministic and Nondeterministic Finite Automata

- ✓ Deterministic and nondeterministic finite automata recognize the same class of languages. Such equivalence is both surprising and useful.
- ✓ It is surprising because NFAs appear to have more power than DFAs, so we might expect that NFAs recognize more languages.
- ✓ It is useful because describing an NFA for a given language sometimes is much easier than describing a DFA for that language.
- ✓ It is also a surprising fact that every language that can be described by NFA can also be described by some DFA.
- ✓ Moreover the DFA in practice has about as many states as the NFA, although it often has more transitions.
- ✓ In the worst case however the smallest DFA can have 2^n states while the smallest NFA for the same language has only n states.
- ✓ If a language is recognized by an NFA then we must show the existence of a DFA that also recognizes it i.e. every NFA has an equivalent DFA.

✓ NFA to DFA CONVERSIONS

✓ Let N = (Q, Σ, δ, q_0, F) be the NFA recognizing some language A. Now we construct DFA M = (Q',Σ', δ', q_0',F') recognizing A.

✓ Before doing the full construction let's first consider the easier case wherein N has no ε arrows.

1. Q' = P(Q).
   - Every state of M is a set of states of N.
2. For R Є Q' and *a* Є Σ let δ'(R, *a*) = {q Є Q| q Є δ(r, *a*) for some r Є R}.
   - If R is a state of M, it is also a set of states of N. when M reads a symbol *a* in state R, it shows where *a* takes each state in R.
   - Because each state may go to a set of states, we take the union of all these sets. Another way to write this expression is :

$$\delta'(R, a) = \bigcup_{r \in R} \delta(r, a).$$

3. q_0' = {q_0}
   - M starts in the state corresponding to the collection containing just the start state of N.
4. F' = {R Є Q' | R contains an accept state of N}.
   - The machine M accepts if one of the possible states that N could be in at this point is an accept state.

✓ Now we need to consider the ε arrows. To do so we set up an extra bit of notation.

✓ For any state R of M we define E(R) to be the collection of states that can be reached from R by going only along ε arrows, including the members of R themselves.

✓ Formally for R ⊆ Q let

E(R) = {q | q can be reached from R by travelling along 0 or more ε arrows}.

✓ Now we modify the transition function of M to place additional fingers on all states that can be reached by going along ε arrows for every step.

✓ Replacing δ(r, *a*) by E(δ(r, *a*)) achieves this effect. Thus

δ'(R, *a*) = {q Є Q | q Є E(δ(r, *a*)) for some r Є R}.

✓ Additionally we need to modify the start state of M to move the fingers initially to all possible states that can be reached from the start state of N along the ε arrows. i.e. changing q_0' to be E({q_0}) achieves this effect.

Example:

✓ Let us consider the following NFA N_4 as below:

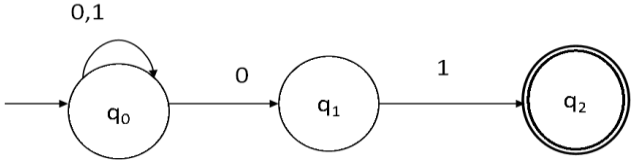


FIGURE 1. 4 NFA ACCEPTING STRINGS END WITH 01

- ✓ Given the NFA $N_4 = (Q, \{a, b\}, \delta, 1, \{1\})$, the set of states Q is {1, 2, 3} as shown in the figure 1.4.
- ✓ To construct DFA D that is equivalent to N_4, we first determine D's states.
- ✓ N_4, has three states {1, 2, 3} so we construct D with eight states one for each subset, one for each subset of N_4's states.
- ✓ We label each of D's states with the corresponding subset. Thus D's state set is

$$\{\phi, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$$

- ✓ Next determine the start and accept states of D. The start state is E({1}), the set of states that are reachable from 1 by travelling along ε arrow plus 1 itself.
- ✓ An ε arrow goes from 1 to 3, so E({1}) = {1, 3}.
- ✓ The new accept states are those containing N_4's accept state; thus {[1}, [1, }, [1, 3}, {1, 2, 3}}.
- ✓ Finally we determine D's transition function where each of D's states goes to one place on input a and one place on input b.
- ✓ The transition table for the NFA N_4 is as follows:

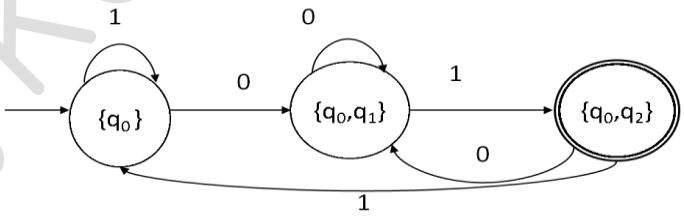
	0	1
→q_0	{q_0, q_1}	q_0
q_1	ϕ	q_2
*q_2	ϕ	ϕ

- ✓ Now change the transition table for NFA into DFA as given below:

	0	1
→{q_0}	{q_0, q_1}	{q_0}
{q_0, q_1}	{q_0, q_1}	{q_0, q_2}
{q_0, q_2}	{q_0, q_1}	{q_0}

$\{q_0, q_1\} : \{q_0, q_1\} \cup \{\phi\} = \{q_0, q_1\}$

- ✓ Now draw the DFA for the above transition table.



MINIMIZATION OF FINITE STATE MACHINE

- ✓ In this section we construct an automaton with minimum number of states equivalent to a given automaton M.
- ✓ As our interest lies only in strings accepted by M, what really matters is whether a state is a final state or not. We define some relations in Q.

- Two states q_1 and q_2 are equivalent (denoted by $q_1 \equiv q_2$) if both $\delta(q_1, x)$ and $\delta(q_2, x)$ are final states or both of them are non final states for all $x \in \Sigma^*$.
- 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 final states or both non final states for all strings x of length k or less.

✓ In particular any two final states are 0-equivalent and any two non final states are also 0-equivalent.

CONSTRUCTION OF MINIMUM AUTOMATON

✓ Step -1: (Construction of π_o). By definition of 0-equivalence, $\pi_o = [Q_1^0, Q_2^0]$ where Q_1^0 is the set of all final states and $Q_2^0 = Q - Q_1^0$.

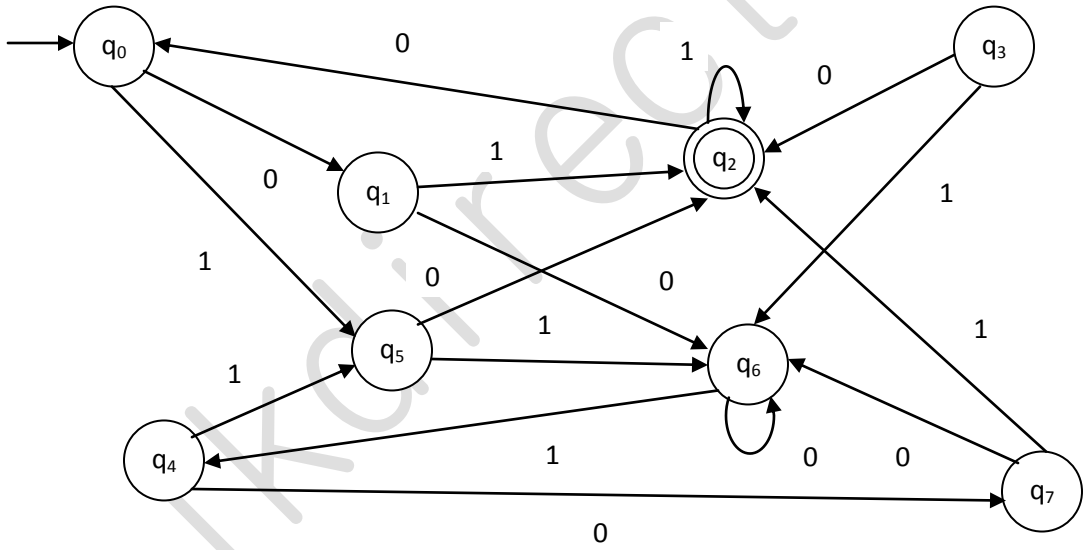
✓ Step -2: Construction of π_{k+1} from π_k.

✓ Step -3: Construct π_n for n= 1, 2, … until π_{n+1}.

✓ Step -4: Construction of minimum automaton.

Example:

✓ Construct a minimum state automaton equivalent to the finite automaton given in figure below:



Solution:

✓ It will be easier if we construct the transition table given in table below:

	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

- ✓ By applying step -1, we get

$$Q_1^0 = F = \{q_2\}, \quad Q_2^0 = Q - Q_1^0$$

- ✓ So,

$$\pi_0 = \{\{q_2\}, \{q_0, q_1, q_3, q_4, q_5, q_6, q_7\}\}$$

- ✓ $\{q_2\}$ in π_0 cannot be further partitioned. So **Q_1'** = $\{q_2\}$. Consider q_0 and $q_1 \in Q_2^0$.
- ✓ The entries under 0-column corresponding to q_0 and q_1 are q_1 and q_6, they lie in Q_2^0. The entries under 1-column are q_5 and q_2.$q_2 \in Q_1^0$ and $q_5 \in Q_2^0$. Therefore q_0 and q_1 are not 1-equivalent.
- ✓ Similarly q_0 is not 1-equivalent to q_3, q_5, q_7.
- ✓ Now consider q_0 and q_4. The entries under 0-column are q_1 and q_7. Both are in Q_2^0. The entries under 1-column are q_5, q_5. So q_4 and q_0 are 1-equivalent. Similarly q_0 is 1-equivalent to q_6.
- ✓ $\{q_0, q_4, q_6\}$ is a subset in π_1 . So **Q_2'** = $\{q_0, q_4, q_6\}$.
- ✓ Repeat the construction by considering q_1 and any one the states q_3, q_5, q_7. q_1 is not 1-equivalent to q_3 or q_5 but 1-equivalent to q_7. Hence **Q_3'** = $\{q_1, q_7\}$.
- ✓ The elements left over in Q_2^0 are q_3 and q_5. By considering the entries under 0-column and 1-column we see that q_3 and q_5 are 1-equivalent.
- ✓ So Q_4' = $\{q_3,q_5\}$. Therefore,

$$\pi_1 = \{\{q_2\}, \{q_0, q_4, q_6\}, \{q_1, q_7\}, \{q_3, q_5\}\}$$

- ✓ $\{q_2\}$ is also in π_2 as it cannot be partitioned further. Now the entries under 0-column corresponding to q_0 and q_4 are q_1 and q_7, and these lie in the same equivalence class in π_1.
- ✓ The entries under 1-column are q_5, q_5. So q_0 and q_4 are 2-equivalent but q_0 and q_6 are not 2-equivalent. Hence $\{q_0, q_4, q_6\}$ is partitioned into $\{q_0, q_4\}$ and $\{q_6\}$.
- ✓ q_1 and q_7 are 2-equivalent. q_3 and q_5 are also 2-equivalent. Thus, π_2 = $\{\{q_2\}, \{q_0, q_4\}, \{q_6\}, \{q_1, q_7\}, \{q_3, q_5\}\}$. And also q_0 and q_4 are 3-equivalent; q_1 and q_7 are 3-equivalent. Also q_3 and q_5 are 3-equivalent. Therefore,

$$\pi_3 = \{\{q_2\}, \{q_0, q_4\}, \{q_6\}, \{q_1, q_7\}, \{q_3, q_5\}\}$$

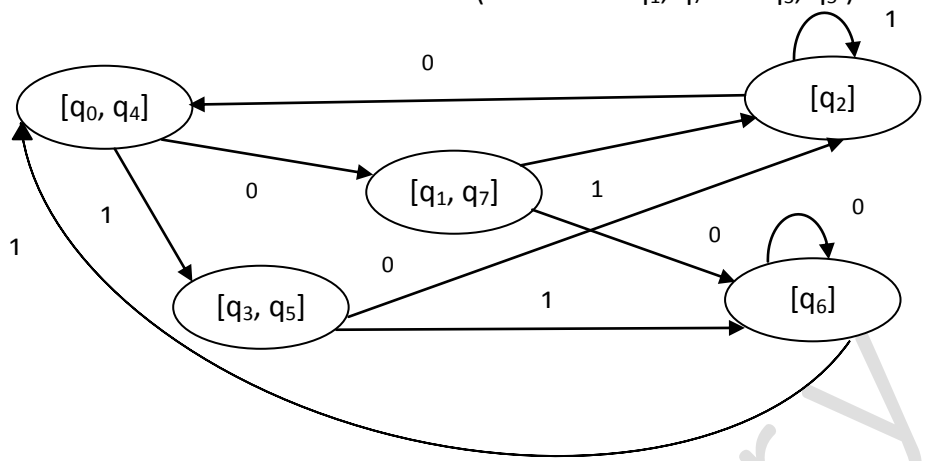
- ✓ As $\pi_2 = \pi_3$, π_2 gives the equivalence classes, the minimum state automaton is

$$M' = (Q', \{0, 1\}, \delta', q_0', F')$$

- ✓ Where Q' = $\{[q_2], [q_0, q_4], [q_6], [q_1, q_7], [q_3, q_5]\}$, q_0'(initial state) = $[q_0,q_4]$, F' = $[q_2]$ and δ' is given by the table as below: (transition table of minimum state automaton)

State / Σ	0	1
$[q_0, q_4]$	$[q_1, q_7]$	$[q_3,q_5]$
$[q_1, q_7]$	$[q_6]$	$[q_2]$
$[q_2]$	$[q_0, q_4]$	$[q_2]$
$[q_3,q_5]$	$[q_2]$	$[q_6]$
$[q_6]$	$[q_6]$	$[q_0, q_4]$

- ✓ The transition diagram for the minimum state automaton is given in figure below. The states q_0 and q_4 are identified and treated as one state. (So also are q_1, q_7 and q_3, q_5.)



- ✓ If there is a transition from q_i to q_j with label *a*, then there is a transition from $[q_i]$ to $[q_j]$ with the same label in the diagram for minimum state automaton.
- ✓ Symbolically, if $\delta(q_i, a) = q_j$, then $\delta'([q_i], a) = [q_j]$.

TESTING THE EQUIVALENCE OF STATES

- ✓ When two distinct states *p* and *q* can be replaced by a single state that behaves like p and q, we say that states *p* and *q* are *equivalent* if:
  - ➢ For all input strings w, $\delta^{\wedge}(p, w)$ is an accepting state if and only if $\delta^{\wedge}(q, w)$ is an accepting state.
- ✓ For example consider the DFA given below with equivalent states:

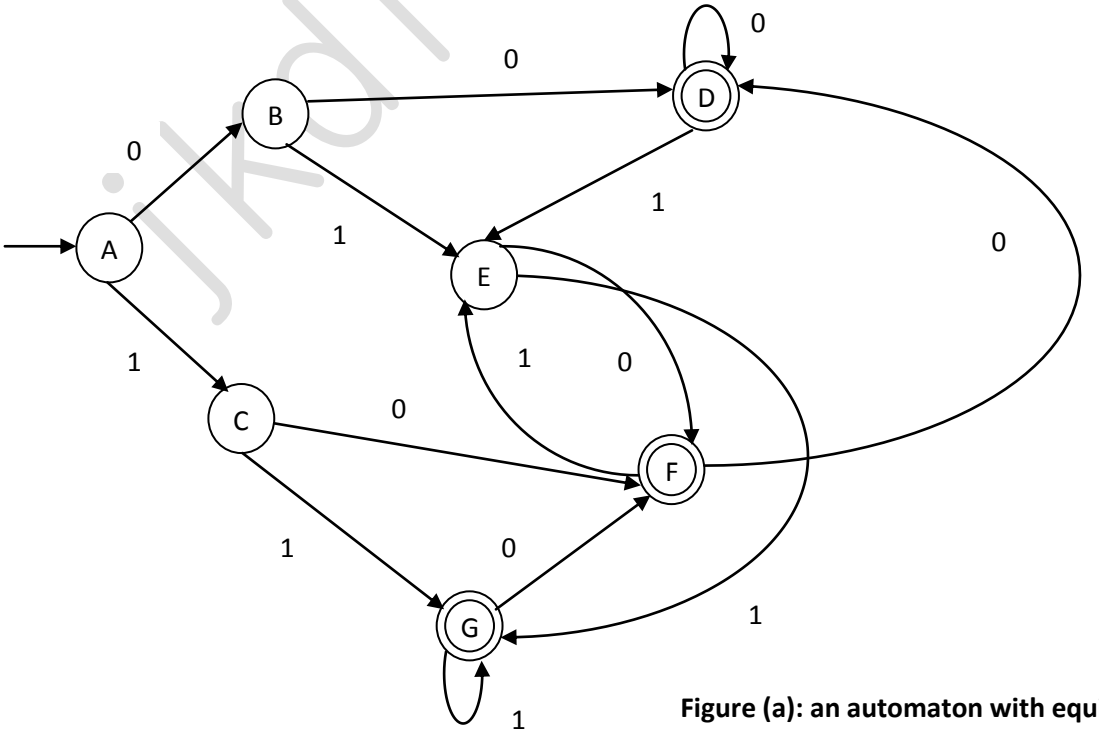


Figure (a): an automaton with equivalent states

- ✓ Basis: if *p* is accepting state and *q* is nonaccepting state, then the pair {*p*, *q*} is distinguishable.
- ✓ Induction: Let *p* and *q* be states such that for some input symbol *a*, r = δ(p, *a*) and s = δ(q, *a*) are a pair of states know to be distinguishable. Then {*p*, *q*} is a pair of distinguishable states.
- ✓ Executing the table filling algorithm on the DFA given in **Figure (a)** we get the table in **Table (a)** as given below:

B	**X**					
C	**X**	**X**				
D	**X**	**X**	**X**			
E	**X**	**X**		**X**		
F	**X**	**X**	**X**		**X**	
G	**X**	**X**	**X**	**X**	**X**	**X**
	A	**B**	**C**	**D**	**E**	**F**

Table (a): Table of state equivalences

- ✓ Initially the entire table is blank. Now since states D, F and G are the only accepting states they cannot be equivalent to A, B, C and E.
- ✓ Accordingly we fill the corresponding squares with x. now we can use these distinguishable pairs to find others.
- ✓ For example since {E, G} is distinguishable and states B and E go to E and G on input 1 we find that {B, E} is also distinguishable.

FINITE AUTOMATA WITH OUTPUTS

- ✓ The finite automata which we considered earlier have binary output i.e. they accept the string or do not accept the string.
- ✓ This acceptability was decided on the basis of reachability of the final state by the initial state. Now we remove this restriction and consider the model where the outputs can be chosen from some other alphabet.
- ✓ The value of the output function Z{t} is :
  - ➢ In the most general case it is a function of the present state q(t) and the present input x(t) i.e.

$$Z(t) = \lambda(q(t), x(t))$$

  - ➢ Where λ is called the output function. This generalized model is called Mealy Machine.
  - ➢ If the output function Z(t) depends only on the present state and is independent of the current input, the output function may be written as:

$$Z(t) = \lambda(q(t))$$

  - ➢ This restricted model is called Moore Machine. It is more convenient to use Moore machine in automata theory.

✓ **Definition**:-

Moore Machine

✓ It is a six tuple $(Q, \Sigma, \Delta, \lambda, \delta, q_0)$, where
- Q is a finite set of states;
- Σ is the input alphabet
- Δ is the output alphabet
- δ is the transition function Σ x Q → Q.
- λ is the output function mapping Q → Δ and
- q_0 is the initial state

Mealy Machine

✓ A mealy machine is a six tuple $(Q, \Sigma, \Delta, \lambda, \delta, q_0)$, where all the symbols except λ have the same meaning as in the Moore machine.

✓ λ is the output function mapping Σ x Q → Δ

✓ For example table given below shows the **Moore machine**. The initial state is q_0 marked with an arrow.

Present State	Next state δ		Output
	a = 0	a = 1	
→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 the input string 0111, the transition of states is given by q_0 →q_3 → q_0 → q_1 →q_2. The output string is 00010.

✓ *For the input string ε the output is $\lambda(q_0) = 0$.*

✓ The following table describes the **Mealy machine**.

Present State	Next State			
	a = 0		a = 1	
	State	Output	State	Output
→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 input string 0011, the transition of states is given by q_1 → q_3 → q_2 → q_4 → q_3 and the output string is 0100.

- ✓ *In the case of mealy machine we get an output only on the application of an input symbol. So for the input string ε, the output is only ε.*
- ✓ For Moore machine if the input string is of length *n*, the output string is of length *n+1*, because the first output is $\lambda(q_0)$ for all the output strings.
- ✓ In the case of Mealy machine if the input string is of length *n*, the output string is also of the same length *n*.

PROCEDURE FOR TRANSFORMING A MEALY MACHINE INTO A MOORE MACHINE

- ✓ Consider the Mealy machine described by the transition table given the table below:

Present State	Next State			
	input a = 0		input a = 1	
	State	Output	State	Output
→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

Solution

- ✓ At the first stage develop the procedure so that both the machines accept exactly the same set of input sequences.
- ✓ We look into next state column for any state (say q_i) and determine the number of different outputs associated with q_i in that column.
- ✓ Now we split into several different states, the number of such states being equal to the number of different outputs associated with q_i.
- ✓ For example in this problem ***q_1*** is associated with one output ***1*** and q_2 is associated with two outputs 0 and 1. Similarly q_3 and q_4 are associated with the outputs 1 and 0, 1.
- ✓ Now we split q_2 into q_{20} and q_{21}. Similarly q_4 is split into q_{40} and q_{41}. Now the above table is reconstructed for the new states as given below:

Present State	Next State			
	input a = 0		input a = 1	
	State	Output	State	Output
→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 pair of states and outputs in the next state column can be rearranged as given in the table below:

Present State	Next State		Output
	a = 0	a = 1	
→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

- ✓ The above table gives the Moore machine. Here we observe that the initial sate q_1 is associated with output 1.
- ✓ This means that with input ε we get an output of 1, if the machine starts at state q1.
- ✓ That this Moore machine accepts a zero-length sequence (null sequence) which is not accepted by the Mealy machine.
- ✓ To overcome this situation, either we must neglect the response of a Moore machine to input ε or we must add a new starting state q_0, whose state transitions are identical with those of q_1 but whose output is 0.
- ✓ So the above table is transformed in to :

Present State	Next State		Output
	a = 0	a = 1	
→q_0	q_3	q_{20}	0
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

PROCEDURE FOR TRANSFORMING A MOORE MACHINE INTO CORRESPONDING MEALY MACHINE

- ✓ We modify the acceptability of input string by a Moore machine by neglecting the response of the Moore machine to input ε.
- ✓ Construction of Moore to Mealy
  - ➢ We have to define the output function λ' for Mealy machine as a function of present state and input symbol. We define λ' by

    $\lambda'(q, a) = \lambda(\delta(q, a))$ for all states q and input symbols a.
  - ➢ The transition function is the same as that of the given Moore machine.

- ✓ For example now consider the Moore machine described by the transition table given the table below:

Present State	Next state δ		Output
	a = 0	a = 1	
→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

- ✓ We must follow the reverse procedure of converting Mealy machine into Moore machine.
- ✓ In the case of Moore machine for every input symbol we form the pair consisting of the next state and the corresponding output and reconstruct the table for Mealy machine.
- ✓ For example the states q_3 and q_1 in the next state column should be associated with outputs 0 and 1 respectively.
- ✓ The transition table for Mealy machine is given as below:

Present State	Next State			
	input a = 0		input a = 1	
	State	Output	State	Output
→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

Note: we can reduce the number of states in any model by considering states with identical transitions. If two states are identical (i.e. the rows corresponding to these two states are identical), then we can delete one of them.

Something Useful (for exam point of view)

✓ Consider the transition table given below:

States	Inputs	
	0	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

✓ Solution: (**Acceptability of a given string**)

$\delta(q_0, 110101) = \delta(q_1, 10101)$

$= \delta(q_0, 0101)$

$= \delta(q_2, 101)$

$= \delta(q_3, 01)$

$= \delta(q_1, 1)$

$= \delta(q_0, \varepsilon)$ $= q_0$

Hence,

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

✓ The symbol down arrow indicates the current input symbol being processed by the machine.