Formal Language & Automata Theory

Prof. Sankhadeep Chatterjee

Regular Expressions

- *Regular expressions* are an algebraic way to describe languages.

- They describe exactly the regular languages.

- If E is a regular expression, then L(E) is the language it defines.

- We'll describe RE's and their languages recursively.

The Operators of Regular Expressions

- The Union of two languages L and M, denoted $L \cup M$, is the set of strings that are in either L or M, or both.
  - For example, if $L = \{001, 10, 111\}$ and $M = \{\epsilon, 001\}$ then $L \cup M = \{\epsilon, 001,10, 111\}$

- The Concatenation of languages L and M is the set of strings that can be formed by taking any string in L and concatenating it with any string in M.
  - For example, if $L = \{001, 10, 111\}$ and $M = \{\epsilon, 001\}$ then $L.M = \{001, 10, 111, 001001, 10001, 111001\}$

The Operators of Regular Expressions

- The Closure (or star or Kleene closure) of a language L is denoted L* represents the set of those strings that can be formed by taking any number of strings from L.
- If L = {0, 1}, then L* denotes all strings of 0's and 1's.
- Note: Formally, $L^* = \bigcup_{i \geq 0} L^i$ which is an infinite set
- Can you give an example of a language whose closure is not infinite?
  - $\phi^* = \{\epsilon\}$ as $\phi^0 = \{\epsilon\}$ and ϕ^i is empty for $i \geq 1$

Building Regular Expressions

- Basis 1: If a is any symbol, then **a** is a RE, and L(**a**) = {a}.
  - Note: {a} is the language containing one string, and that string is of length 1.
- Basis 2: ε is a RE, and L(ε) = {ε}.
- Basis 3: ∅ is a RE, and L(∅) = ∅.

Building Regular Expressions

- Induction 1: If E_1 and E_2 are regular expressions, then E_1+E_2 is a regular expression, and $L(E_1+E_2) = L(E_1) \cup L(E_2)$.
- Induction 2: If E_1 and E_2 are regular expressions, then E_1E_2 is a regular expression, and $L(E_1E_2) = L(E_1)L(E_2)$.
  - *Concatenation* : the set of strings wx such that w is in $L(E_1)$ and x is in $L(E_2)$.
- Induction 3: If E is a RE, then E* is a RE, and $L(E^*) = (L(E))^*$.
  - *Closure*, or "Kleene closure" = set of strings $w_1w_2 \ldots w_n$, for some $n \geq 0$, where each w_i is in $L(E)$.
  - Note: when n=0, the string is ε.

Precedence of Operators

- Parentheses may be used wherever needed to influence the grouping of operators.

- Order of precedence is * (highest), then concatenation, then + (lowest).

- Examples:
  - L(**01**+**0**) = {01, 0}.
  - L(**0**(**1**+**0**)) = {01, 00}.

Regular expressions

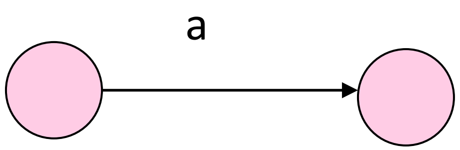
- Identities & Annihilators
- ∅ is the identity for +.
  - R + ∅ = R.
- ε is the identity for concatenation.
  - εR = Rε = R.
- ∅ is the annihilator for concatenation.
  - ∅R = R∅ = ∅.
- Examples of Language of Regular Expressions:
  - L(**01**) = {01}.
  - L(**0***) = {ε, 0, 00, 000,… }.
  - L((**0**+**10**)*(ε+**1**)) = all strings of 0's and 1's without two consecutive 1's.

Examples

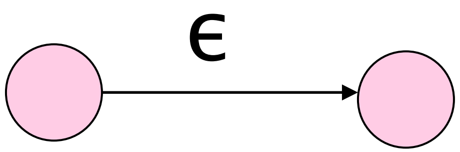
- L = {w|length of w = 2} over $\Sigma = \{a, b\}$
  - $(a+b)^2$
- L = {w|length of w ≥ 2} over $\Sigma = \{a, b\}$
  - $(a+b)^2(a+b)^*$
- L = {w|length of w is even} over $\Sigma = \{a, b\}$
  - $((a+b).(a+b))^*$
- L = {w|no. of a's in w = 2} over $\Sigma = \{a, b\}$
  - $b^*ab^*ab^*$
- L = {w|no. of a's in w ≥ 2} over $\Sigma = \{a, b\}$
  - $b^*ab^*a(a+b)^*$
- L = {w|no. of a's in w is even} over $\Sigma = \{a, b\}$
  - $(b^*ab^*ab^*)^*+b^*$

Regular Expression to ϵ-NFA

- Symbol **a**:

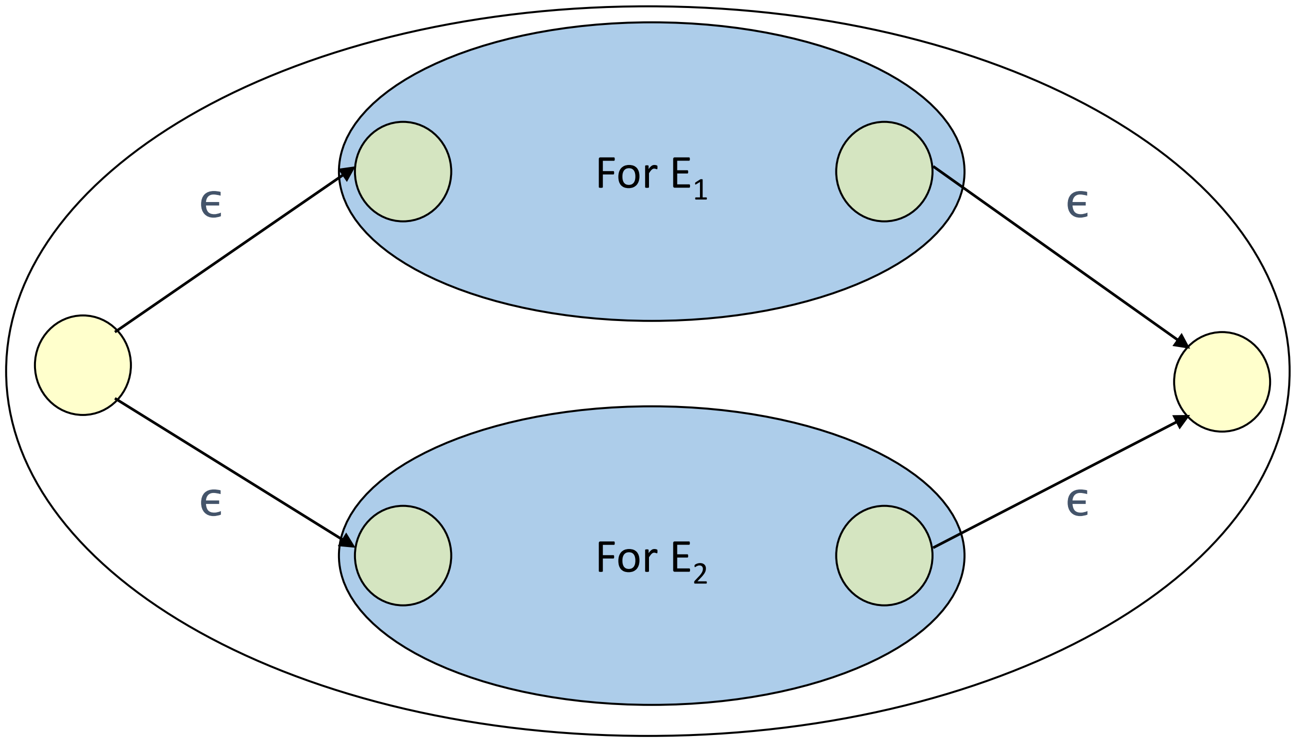


- ε:



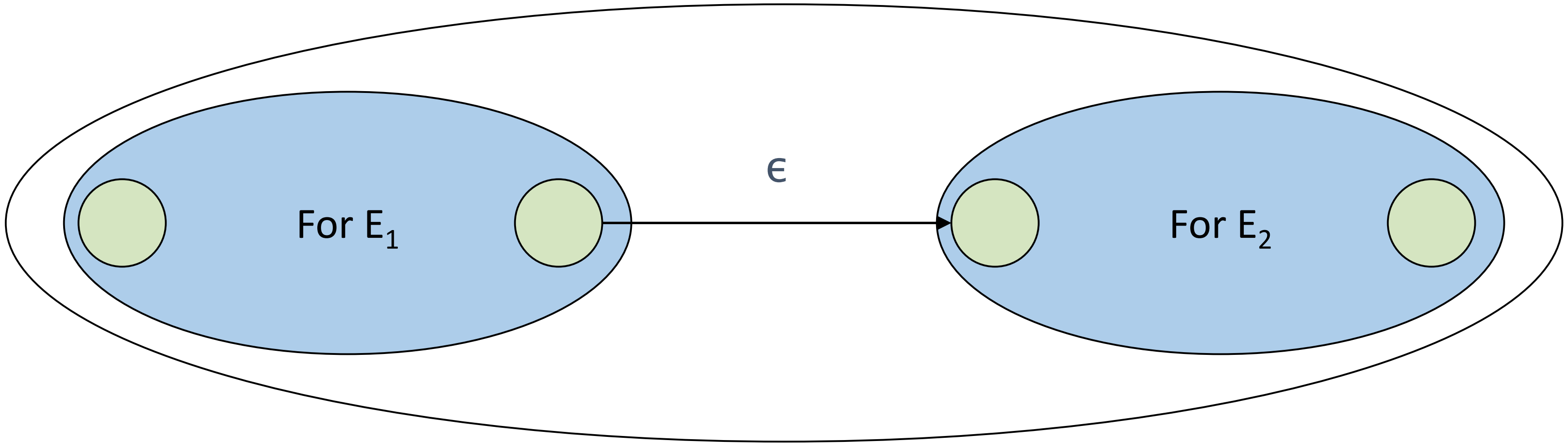
- ∅:

RE to ϵ-NFA: Induction 1 – Union



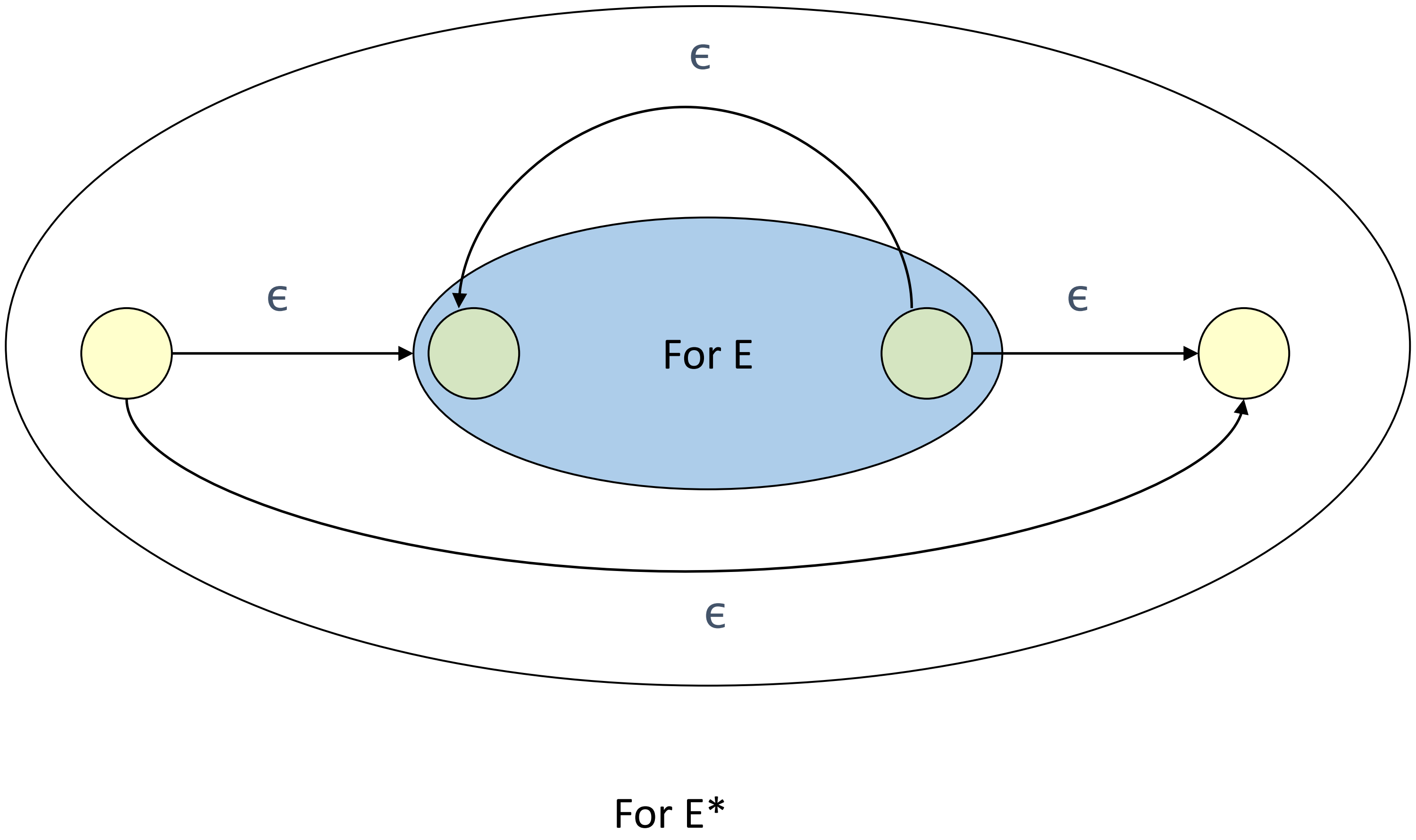
For $E_1 \cup E_2$

RE to ϵ-NFA: Induction 2 – Concatenation



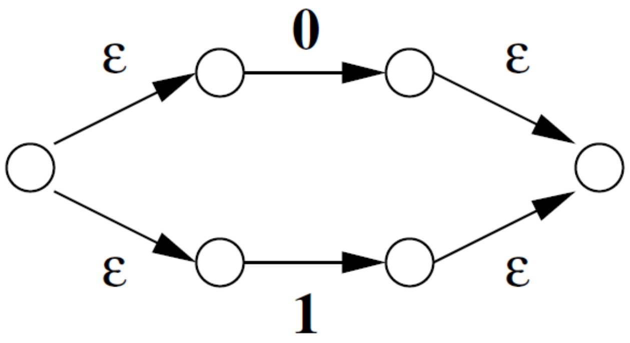
For E_1E_2

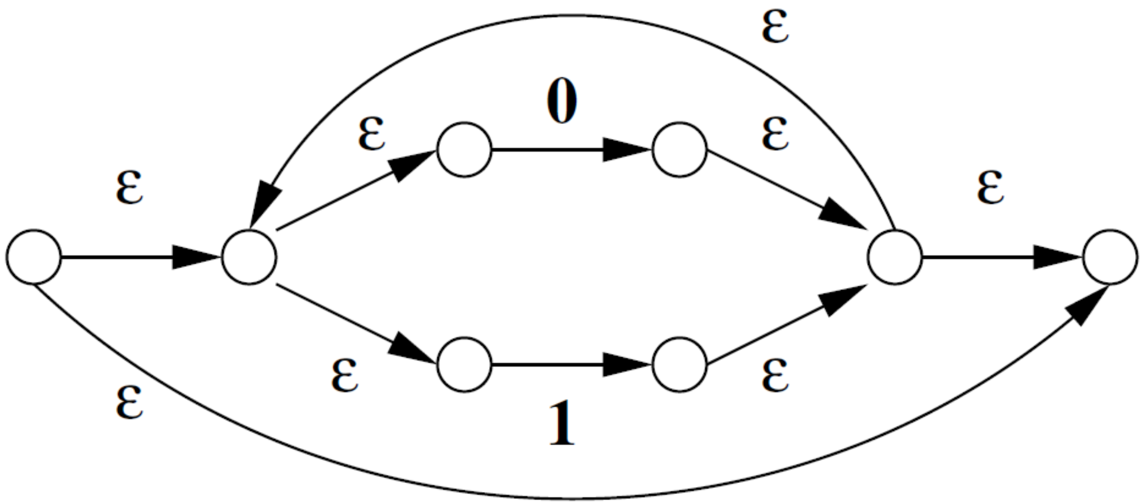
RE to ϵ-NFA: Induction 3 – Closure



RE to ϵ-NFA: Example

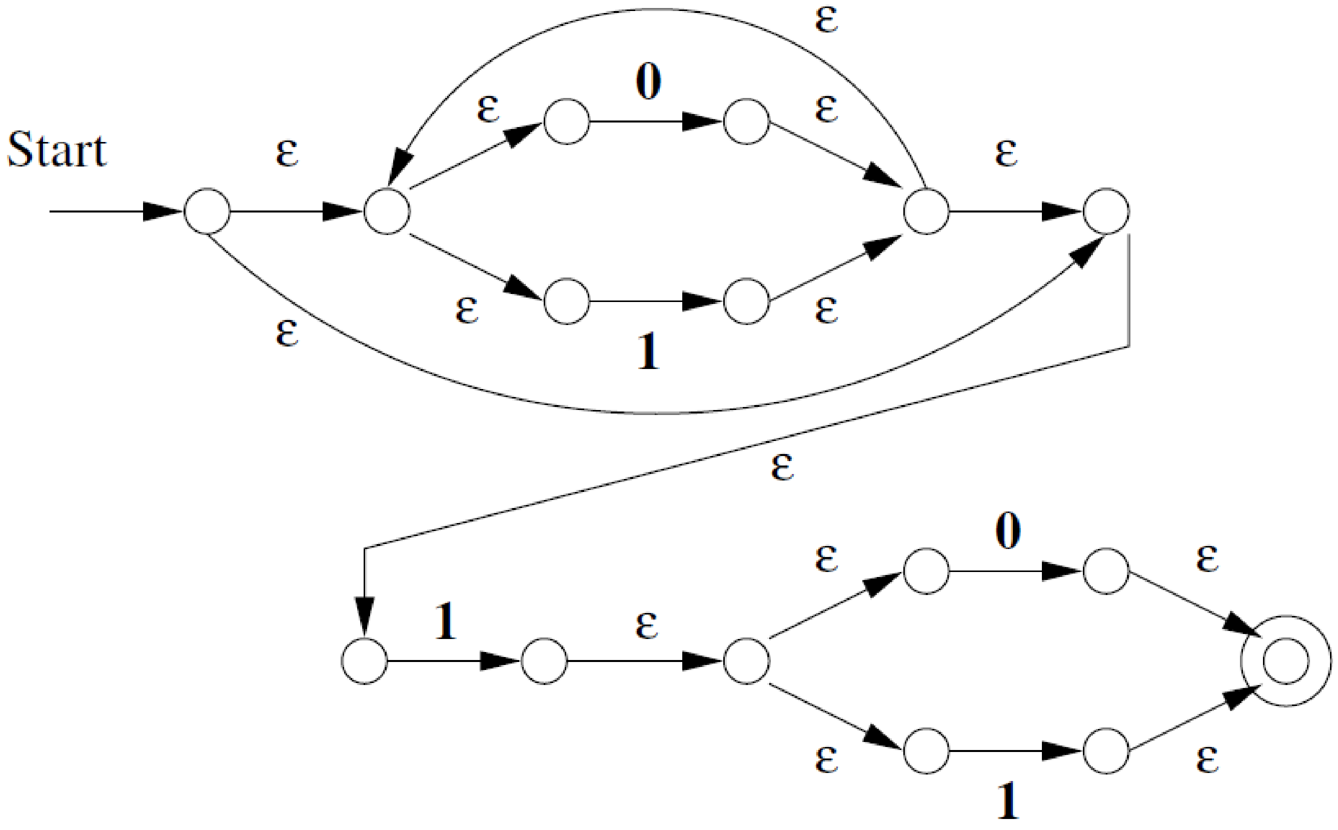
- Convert the regular expression (0+1)*1(0+1)* to an ϵ-NFA





RE to ϵ-NFA: Example

- Convert the regular expression (0+1)*1(0+1)* to an ϵ-NFA



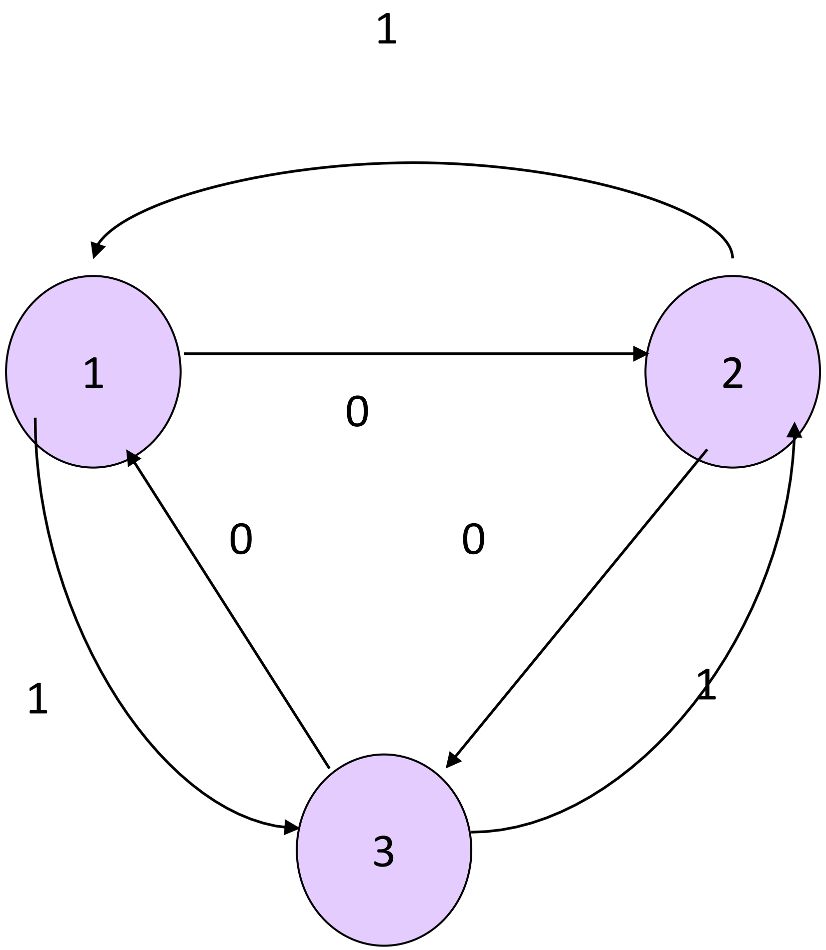
DFA to Regular Expression

- States of the DFA are assumed to be 1,2,…,n.
- We construct RE's for the labels of restricted sets of paths.
  - Basis: single arcs or no arc at all.
  - Induction: paths that are allowed to traverse next state in order.

DFA to Regular Expression

- A k-path is a path through the graph of the DFA that goes though no state numbered higher than k.
- Endpoints are not restricted; they can be any state.

DFA to Regular Expression



0-paths from 2 to 3:
RE for labels = **0**.

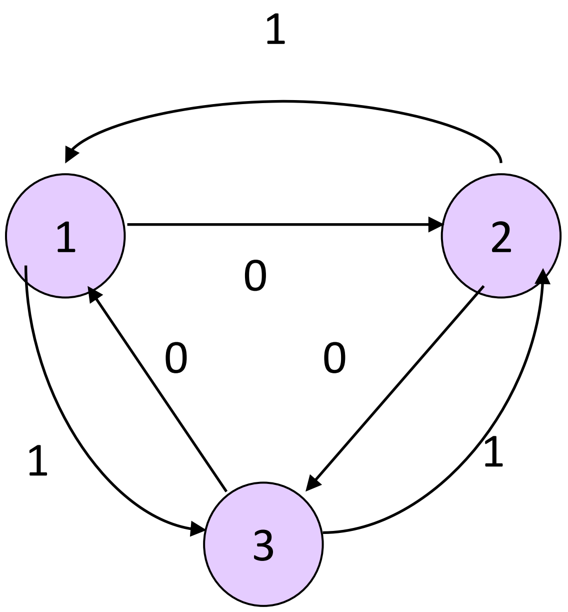
1-paths from 2 to 3:
RE for labels = **0**+**11**.

2-paths from 2 to 3:
RE for labels =
(**10**)***0**+**1**(**01**)***1**

3-paths from 2 to 3:
RE for labels = ??

k-Path Induction

- Let R_{ij}^{k} be the regular expression for the set of labels of k-paths from state i to state j.
- Basis: k=0. R_{ij}^{0} = sum of labels of arc from i to j.
  - $\varnothing$ if no such arc.
  - But add ϵ if i=j.
- Example
- $R_{12}^{0} = \mathbf{0}$.
- $R_{11}^{0} = \varnothing + \epsilon = \epsilon$.



k-Path Induction

- A k-path from i to j either:
    1. Never goes through state k, or
    2. Goes through k one or more times.

$$R_{ij}^{k} = R_{ij}^{k-1} + R_{ik}^{k-1}(R_{kk}^{k-1})^{*} R_{kj}^{k-1}.$$

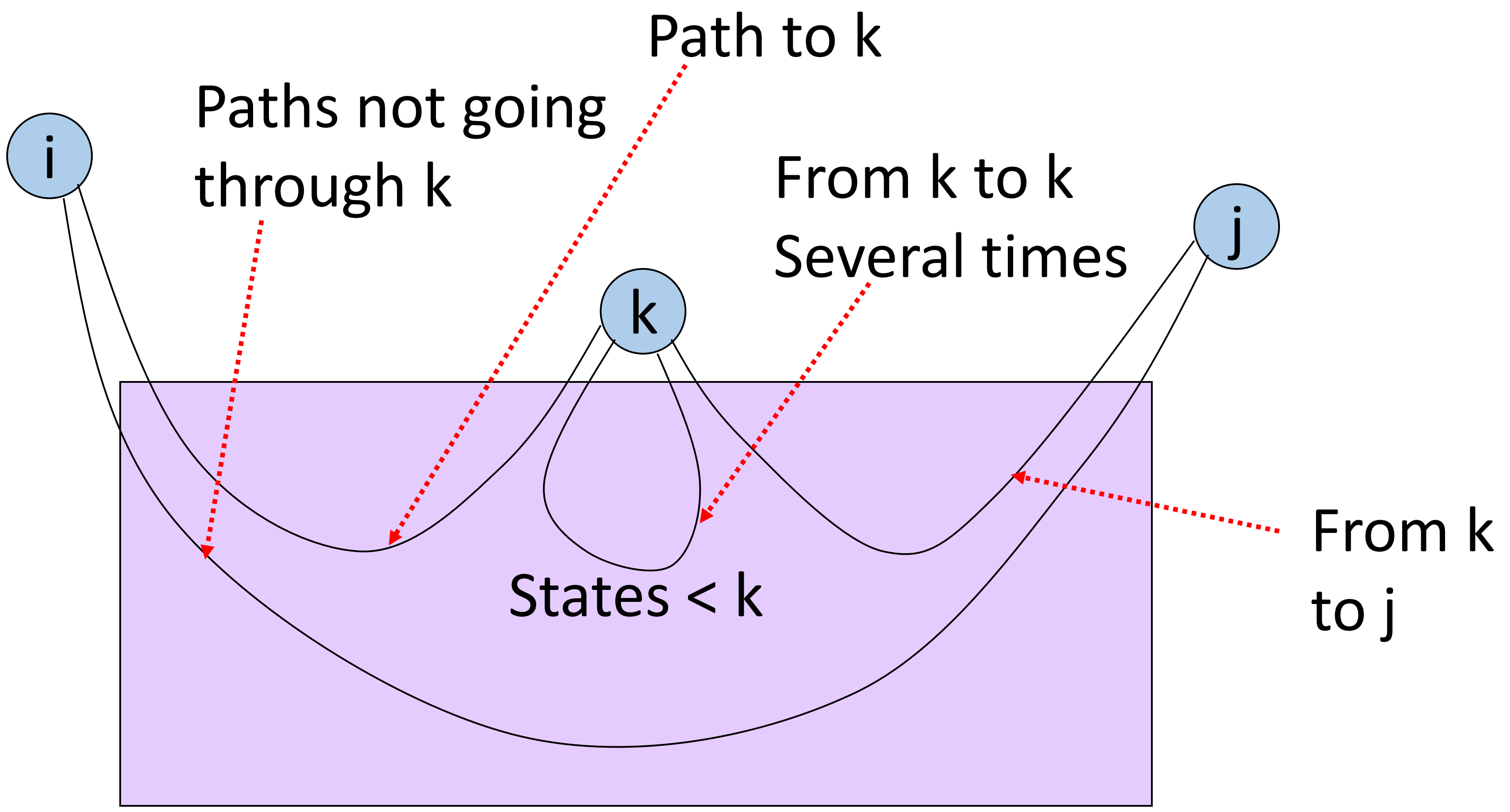
Doesn’t go through k

Goes from i to k the first time

Zero or more times from k to k

Then, from k to j

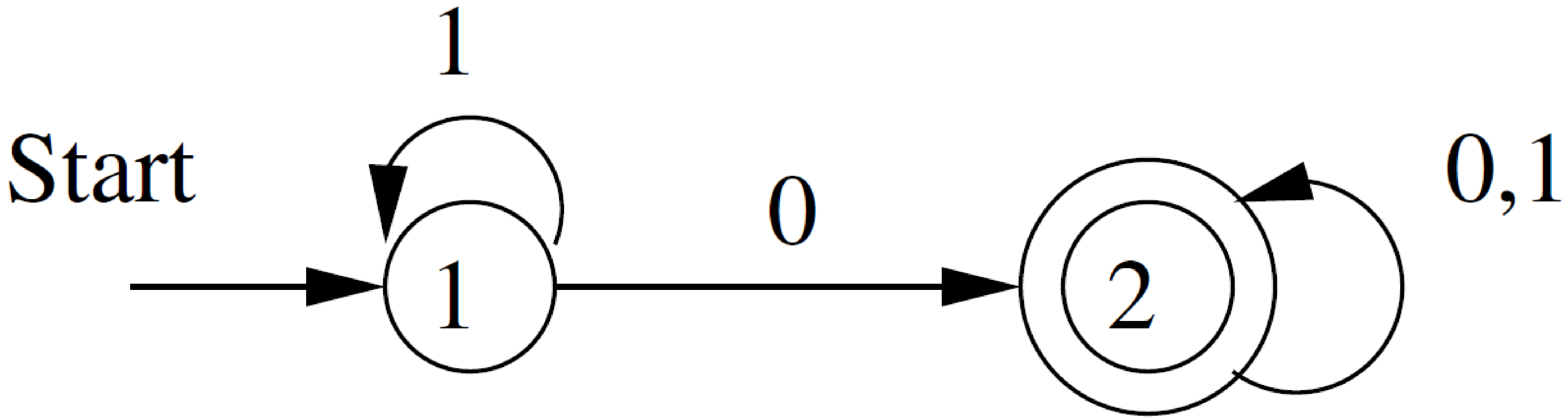
k-Path Induction

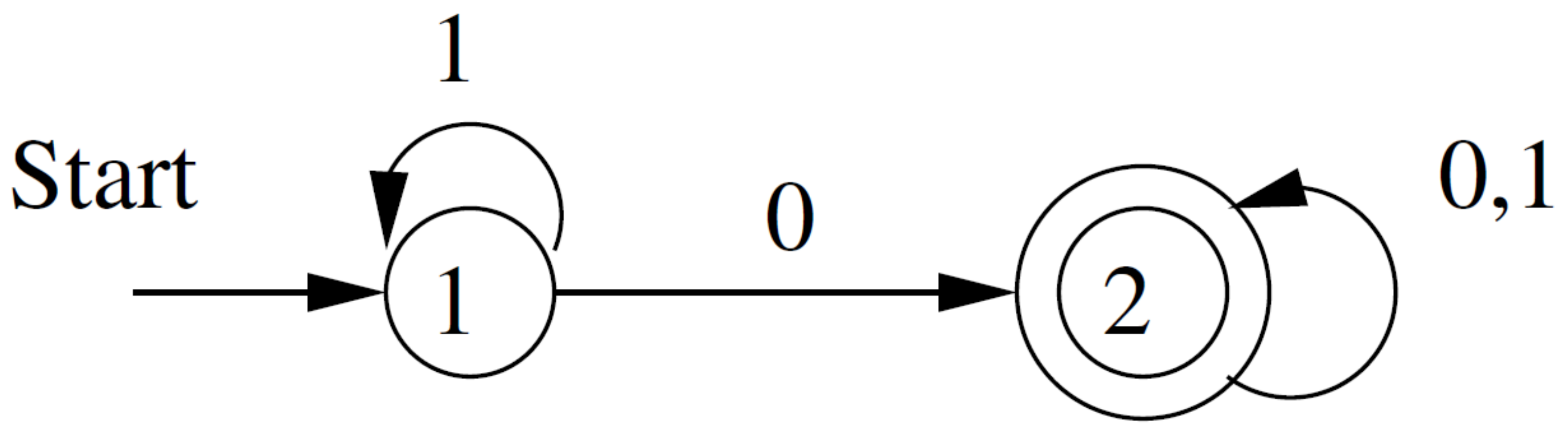


Final Step: DFA to Regular Expression

- The RE with the same language as the DFA is the sum (union) of R_{ij}^{n}, where:
  1. n is the number of states; i.e., paths are unconstrained.
  2. i is the start state.
  3. j is one of the final states.

Example: DFA to Regular Expression





$R_{11}^{(0)}$	$\epsilon + \mathbf{1}$
$R_{12}^{(0)}$	$\mathbf{0}$
$R_{21}^{(0)}$	$\emptyset$
$R_{22}^{(0)}$	$(\epsilon + \mathbf{0} + \mathbf{1})$

	By direct substitution	Simplified
$R_{11}^{(1)}$	$\epsilon + \mathbf{1} + (\epsilon + \mathbf{1})(\epsilon + \mathbf{1})^*(\epsilon + \mathbf{1})$	$\mathbf{1}^*$
$R_{12}^{(1)}$	$\mathbf{0} + (\epsilon + \mathbf{1})(\epsilon + \mathbf{1})^*\mathbf{0}$	$\mathbf{1}^*\mathbf{0}$
$R_{21}^{(1)}$	$\emptyset + \emptyset(\epsilon + \mathbf{1})^*(\epsilon + \mathbf{1})$	$\emptyset$
$R_{22}^{(1)}$	$\epsilon + \mathbf{0} + \mathbf{1} + \emptyset(\epsilon + \mathbf{1})^*\mathbf{0}$	$\epsilon + \mathbf{0} + \mathbf{1}$

$R_{11}^{(0)}$	$\epsilon+\mathbf{1}$
$R_{12}^{(0)}$	$\mathbf{0}$
$R_{21}^{(0)}$	$\emptyset$
$R_{22}^{(0)}$	$(\epsilon+\mathbf{0}+\mathbf{1})$

	By direct substitution	Simplified
$R_{11}^{(1)}$	$\epsilon+\mathbf{1}+(\epsilon+\mathbf{1})(\epsilon+\mathbf{1})^*(\epsilon+\mathbf{1})$	$\mathbf{1}^*$
$R_{12}^{(1)}$	$\mathbf{0}+(\epsilon+\mathbf{1})(\epsilon+\mathbf{1})^*\mathbf{0}$	$\mathbf{1}^*\mathbf{0}$
$R_{21}^{(1)}$	$\emptyset+\emptyset(\epsilon+\mathbf{1})^*(\epsilon+\mathbf{1})$	$\emptyset$
$R_{22}^{(1)}$	$\epsilon+\mathbf{0}+\mathbf{1}+\emptyset(\epsilon+\mathbf{1})^*\mathbf{0}$	$\epsilon+\mathbf{0}+\mathbf{1}$

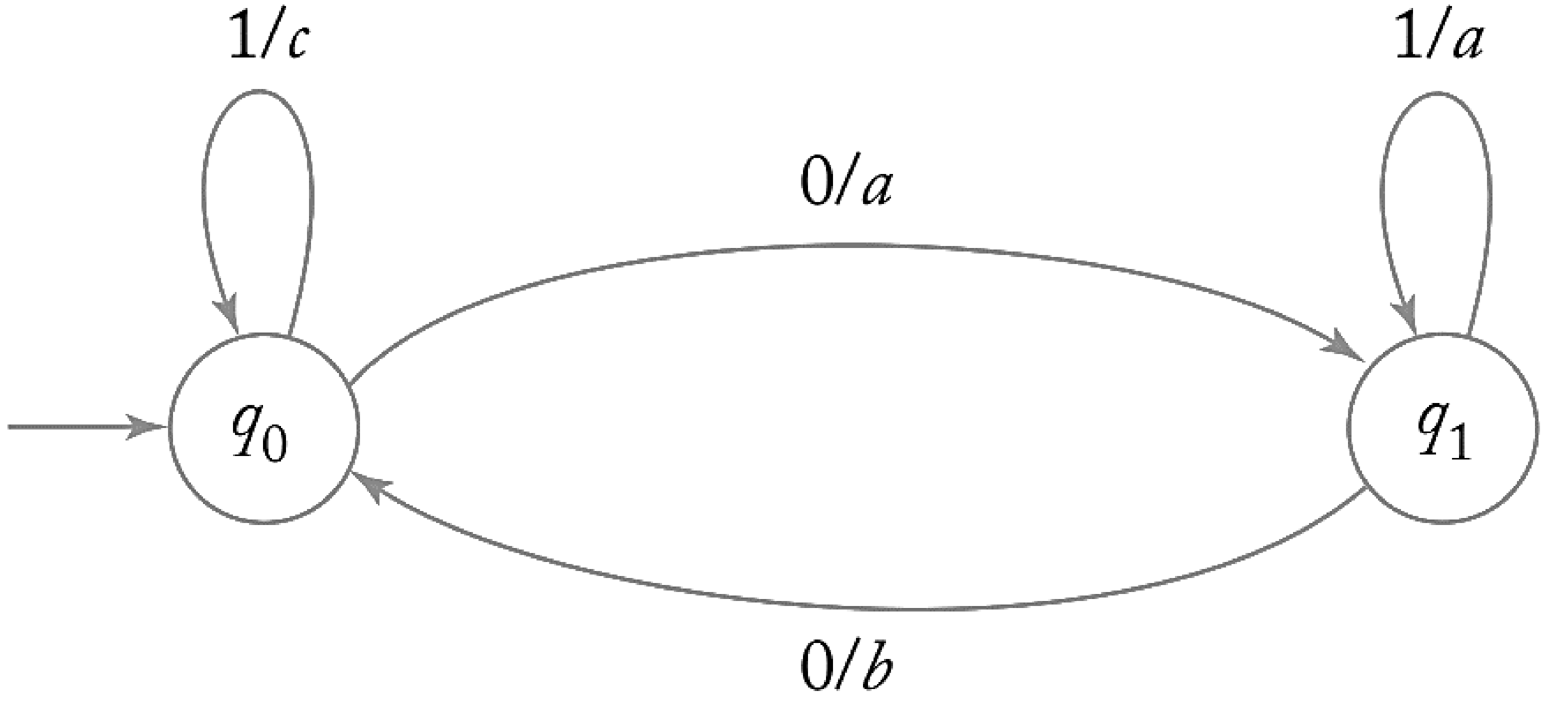
	By direct substitution	Simplified
$R_{11}^{(2)}$	$\mathbf{1}^*+\mathbf{1}^*\mathbf{0}(\epsilon+\mathbf{0}+\mathbf{1})^*\emptyset$	$\mathbf{1}^*$
$R_{12}^{(2)}$	$\mathbf{1}^*\mathbf{0}+\mathbf{1}^*\mathbf{0}(\epsilon+\mathbf{0}+\mathbf{1})^*(\epsilon+\mathbf{0}+\mathbf{1})$	$\mathbf{1}^*\mathbf{0}(\mathbf{0}+\mathbf{1})^*$
$R_{21}^{(2)}$	$\emptyset+(\epsilon+\mathbf{0}+\mathbf{1})(\epsilon+\mathbf{0}+\mathbf{1})^*\emptyset$	$\emptyset$
$R_{22}^{(2)}$	$\epsilon+\mathbf{0}+\mathbf{1}+(\epsilon+\mathbf{0}+\mathbf{1})(\epsilon+\mathbf{0}+\mathbf{1})^*(\epsilon+\mathbf{0}+\mathbf{1})$	$(\mathbf{0}+\mathbf{1})^*$

Mealy Machine

- In a Mealy machine, the output produced by each transition depends on the internal state prior to the transition and the input symbol used in the transition.
- A Mealy machine is defined by the six tuple $M = (Q, \Sigma, \Gamma, \delta, \theta, q0)$
- where
  - Q is a finite set of internal states,
  - Σ is the input alphabet,
  - Γ is the output alphabet,
  - $\delta : Q \times \Sigma \rightarrow Q$ is the transition function,
  - $\theta : Q \times \Sigma \rightarrow \Gamma$ is the output function,
  - $q0 \in Q$ is the initial state of M.

Mealy Machine

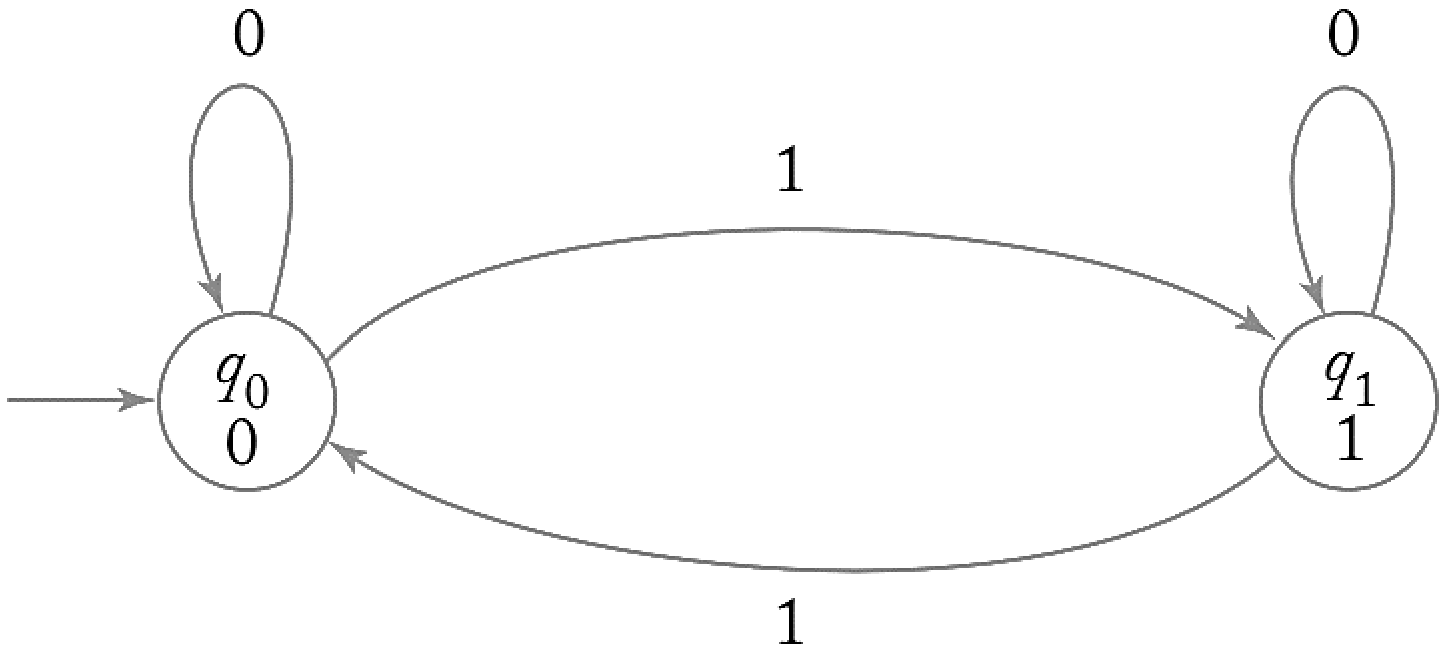
- The mealy machine with Q = {q0, q1}, Σ= {0, 1}, Γ= {a, b, c}, initial state q0, and
- δ(q0, 0) = q1,
- δ(q0, 1) = q0,
- δ(q1, 0) = q0,
- δ(q1, 1) = q1,
- θ(q0, 0) = a,
- θ(q0, 1) = c,
- θ(q1, 0) = b,
- θ(q1, 1) = a



Moore Machine

- In a Moore machine every state is associated with an element of the output alphabet
- Whenever the state is entered, this output symbol is printed.
- A Moore machine is defined by the six tuple $M = (Q, \Sigma, \Gamma, \delta, \theta, q0)$.
  - Q is a finite set of internal states,
  - Σ is the input alphabet,
  - Γ is the output alphabet,
  - $\delta : Q \times \Sigma \to Q$ is the transition function,
  - $\theta : Q \to \Gamma$ is the output function,
  - $q0 \in Q$ is the initial state.

Moore Machine



Mealy to Moore Conversion

For different output associated with states, split them into separate states.

Present State	Next State (input = 0)	Output	Next State (input = 1)	Output
q1	q3	0	q2	0
q2	q1	1	q4	0
q3	q2	1	q1	1
q4	q4	1	q3	0

Present State	Next State (input = 0)	Output	Next State (input = 1)	Output

Mealy to Moore Conversion

For different output associated with states, split them into separate states.

Present State	Next State input = 0	Output	Next State input = 1	Output
q1	q3	0	q2	0
q2	q1	1	q4	0
q3	q2	1	q1	1
q4	q4	1	q3	0

Present State	Next State input = 0	Output	Next State input = 1	Output
q1	q3	0	q20	0
q20	q1	1	q40	0
q21	q1	1	q40	0
q3	q21	1	q1	1
q40	q41	1	q3	0
q41	q41	1	q3	0

Mealy to Moore Conversion

Merge the output columns to obtain the Moore machine. Output depends on present state only.

Present State	Next State	Output	Next State	Output
	input = 0		input = 1	
q1	q3	0	q20	0
q20	q1	1	q40	0
q21	q1	1	q40	0
q3	q21	1	q1	1
q40	q41	1	q3	0
q41	q41	1	q3	0

Present State	Next State	Next State	Output
	input = 0	input = 1	
q1	q3	q20	1
q20	q1	q40	0
q21	q1	q40	1
q3	q21	q1	0
q40	q41	q3	0
q41	q41	q3	1

Moore to Mealy machine

Separate output columns for each input value. Output is decided based on present state only.

Present State	Next State		Output
	0	1	
q0	q3	q1	0
q1	q1	q2	1
q2	q2	q3	0
q3	q3	q0	0

Present State	Next State	Output	Next State	Output
	input = 0		input = 1	
q0				
q1				
q2				
q3				

Moore to Mealy machine

Separate output columns for each input value. Output is decided based on present state only.

Present State	Next State		Output
	0	1	
q0	q3	q1	0
q1	q1	q2	1
q2	q2	q3	0
q3	q3	q0	0

Present State	Next State	Output	Next State	Output
	input = 0		input = 1	
q0	q3	0	q1	1
q1	q1	1	q2	0
q2	q2	0	q3	0
q3	q3	0	q0	0