Module-2

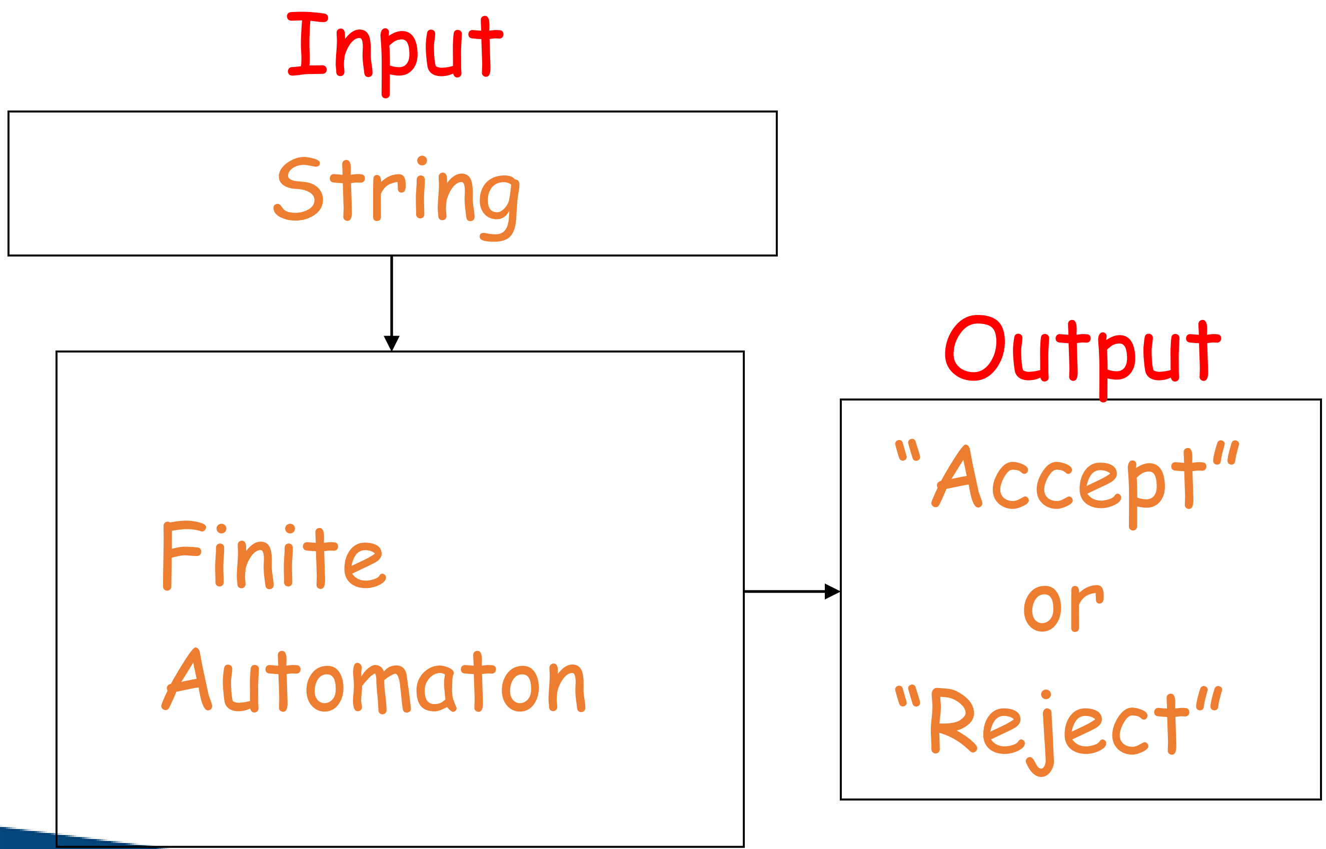
Finite Automata

Language Recognizers : An example of Finite Automata

An automaton is an abstract model of a digital computer.

Finite Automata(FA) is the simplest machine to recognize patterns.

Finite Automaton

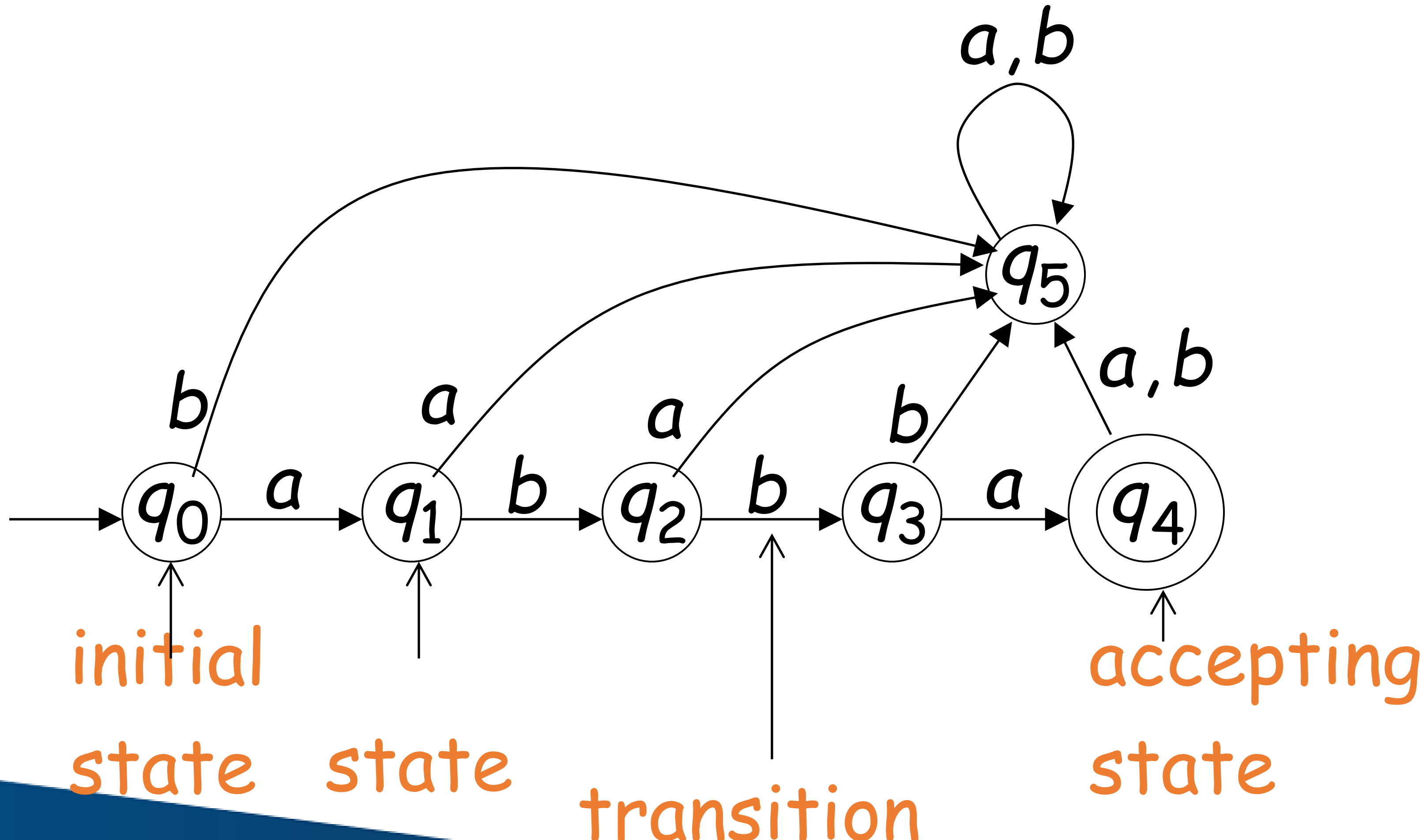


Representation of Finite Automata

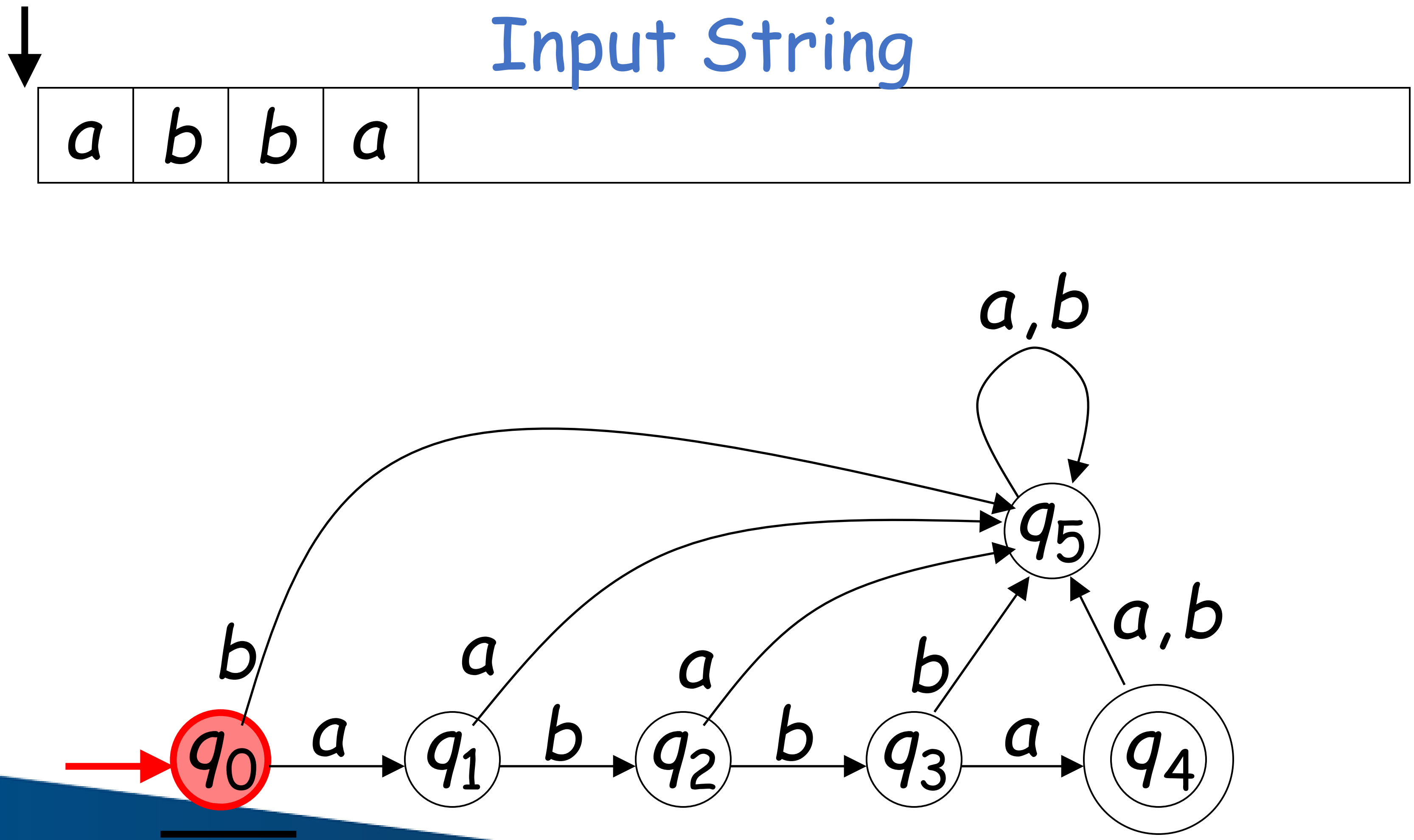
Finite Automata is represented by –

1. Transition Graph
2. Transition Table
3. Regular Expression

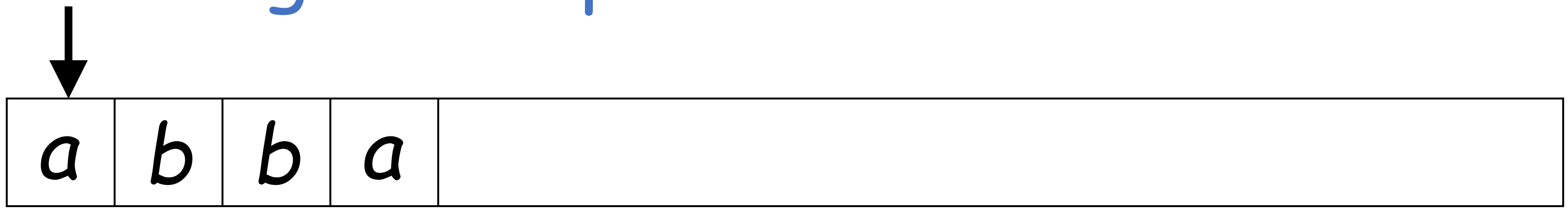
Transition Graph

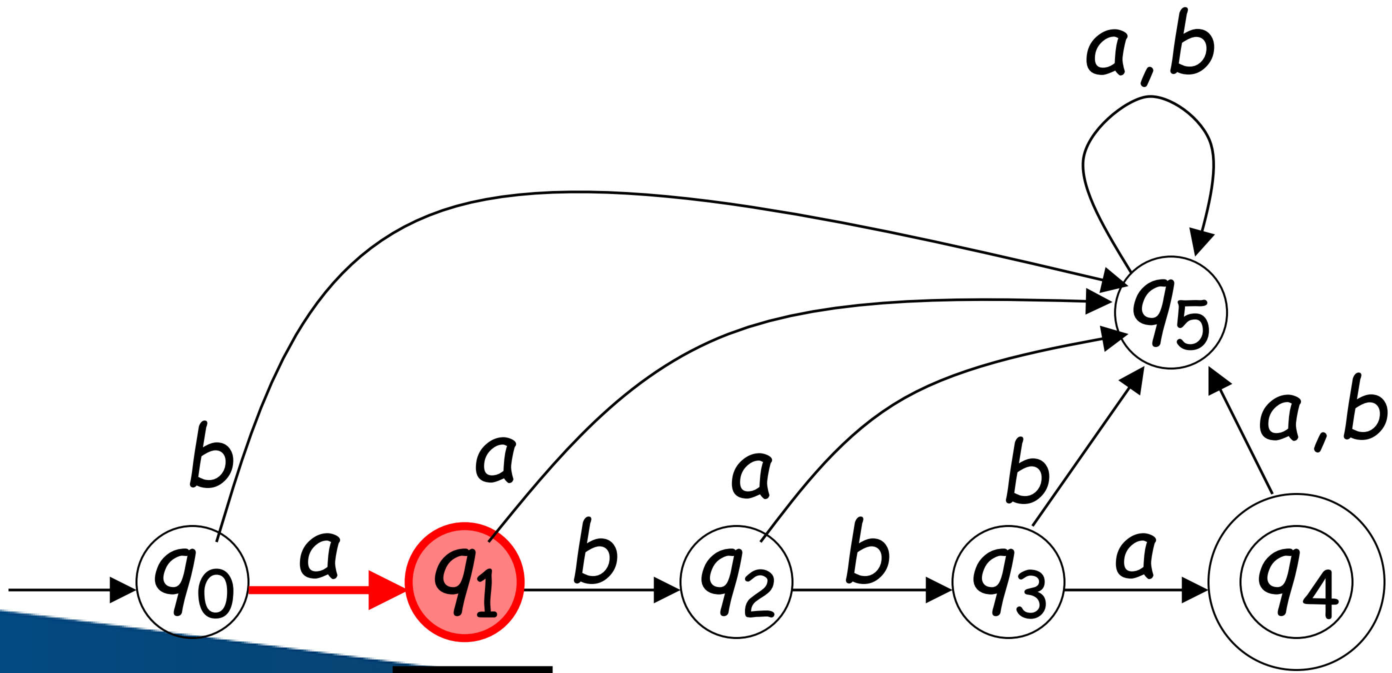


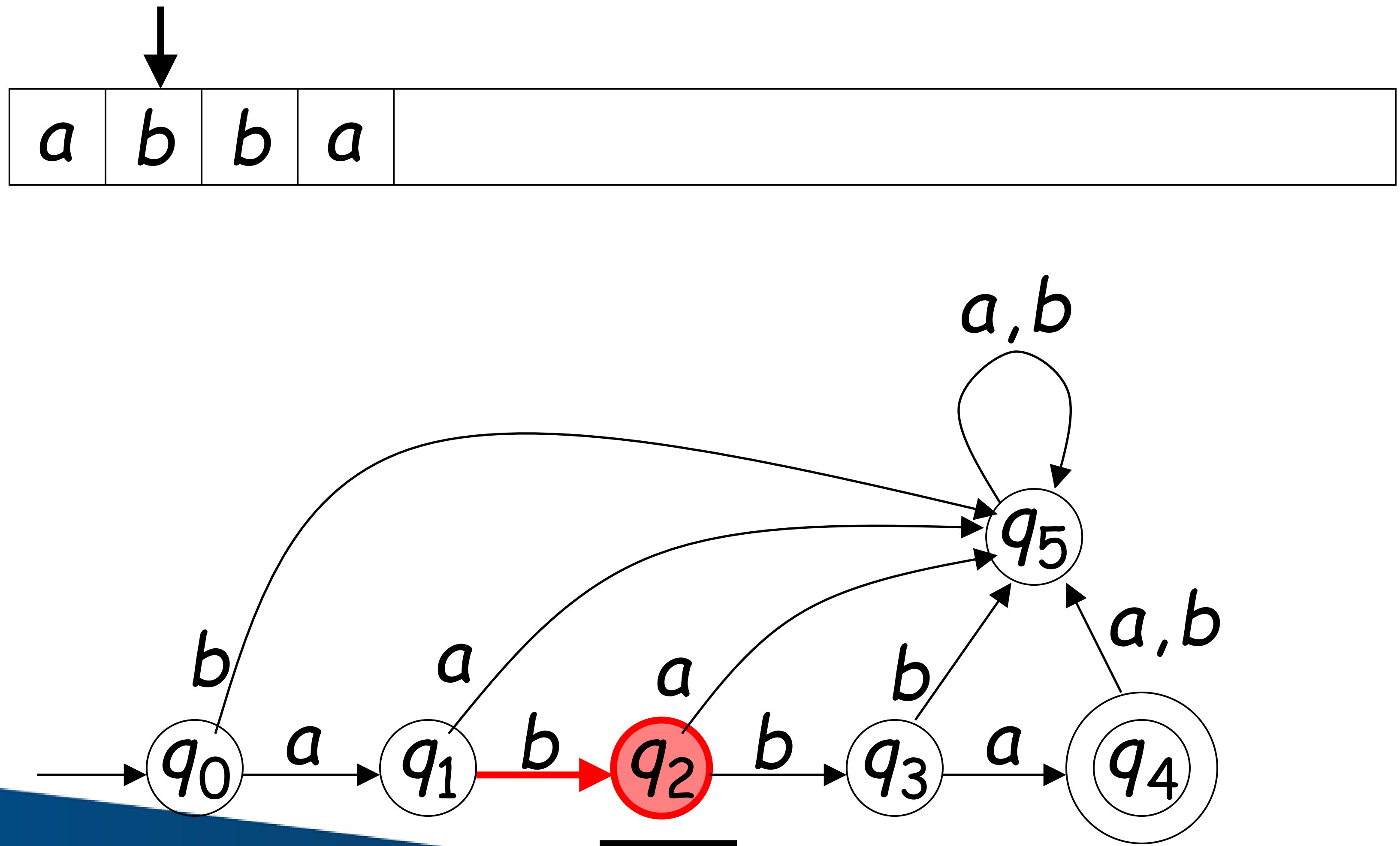
Initial Configuration

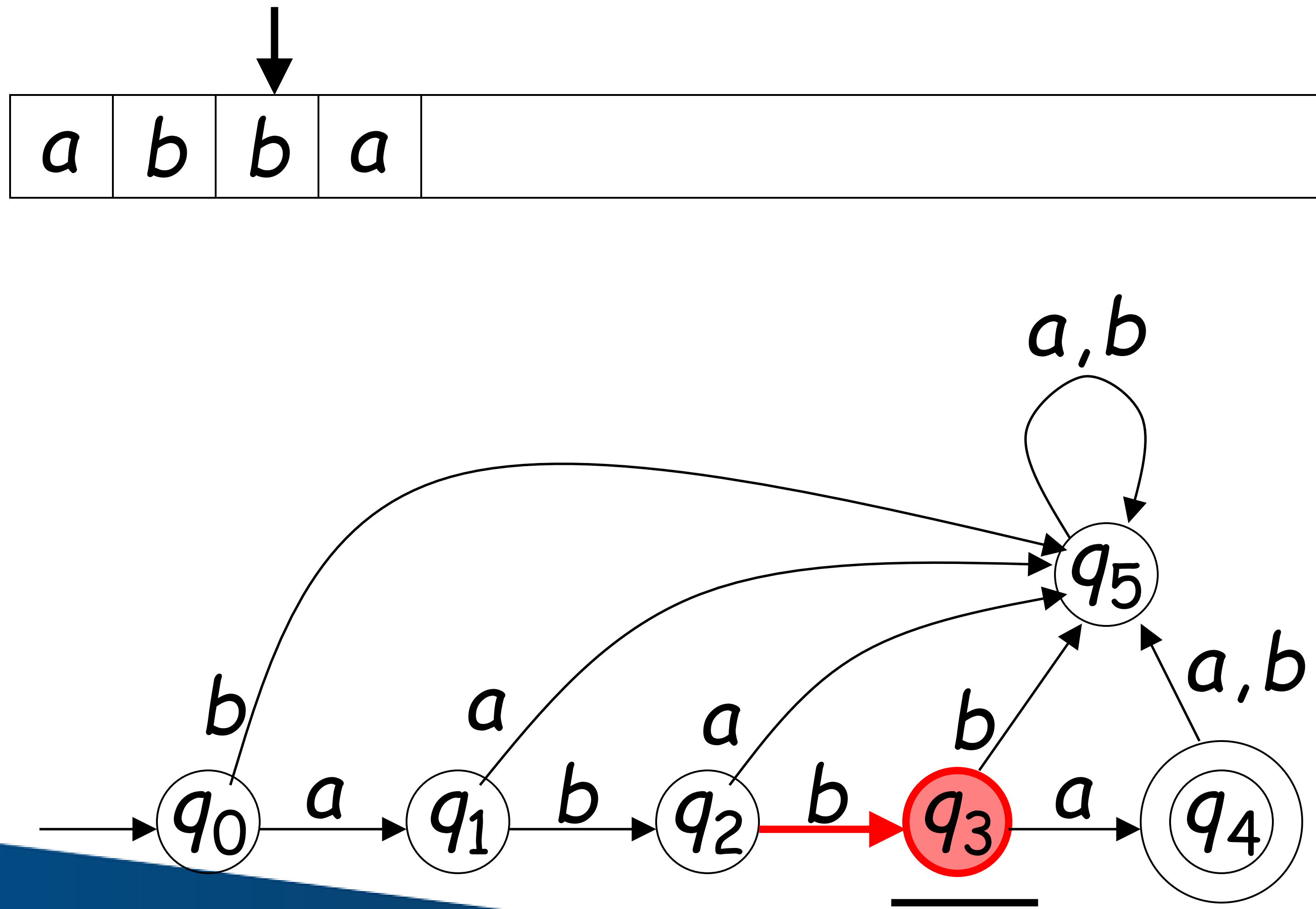


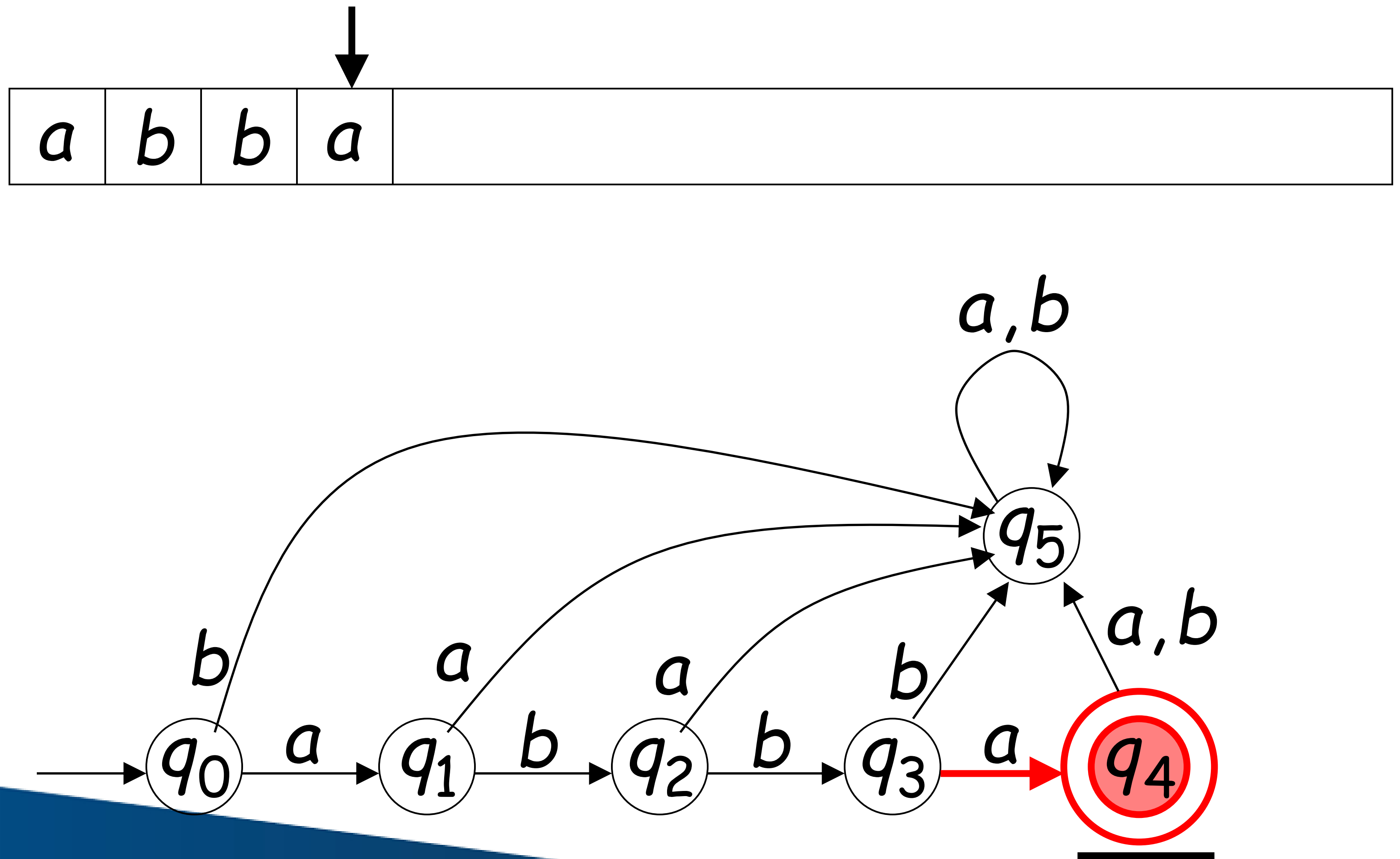
Reading the Input



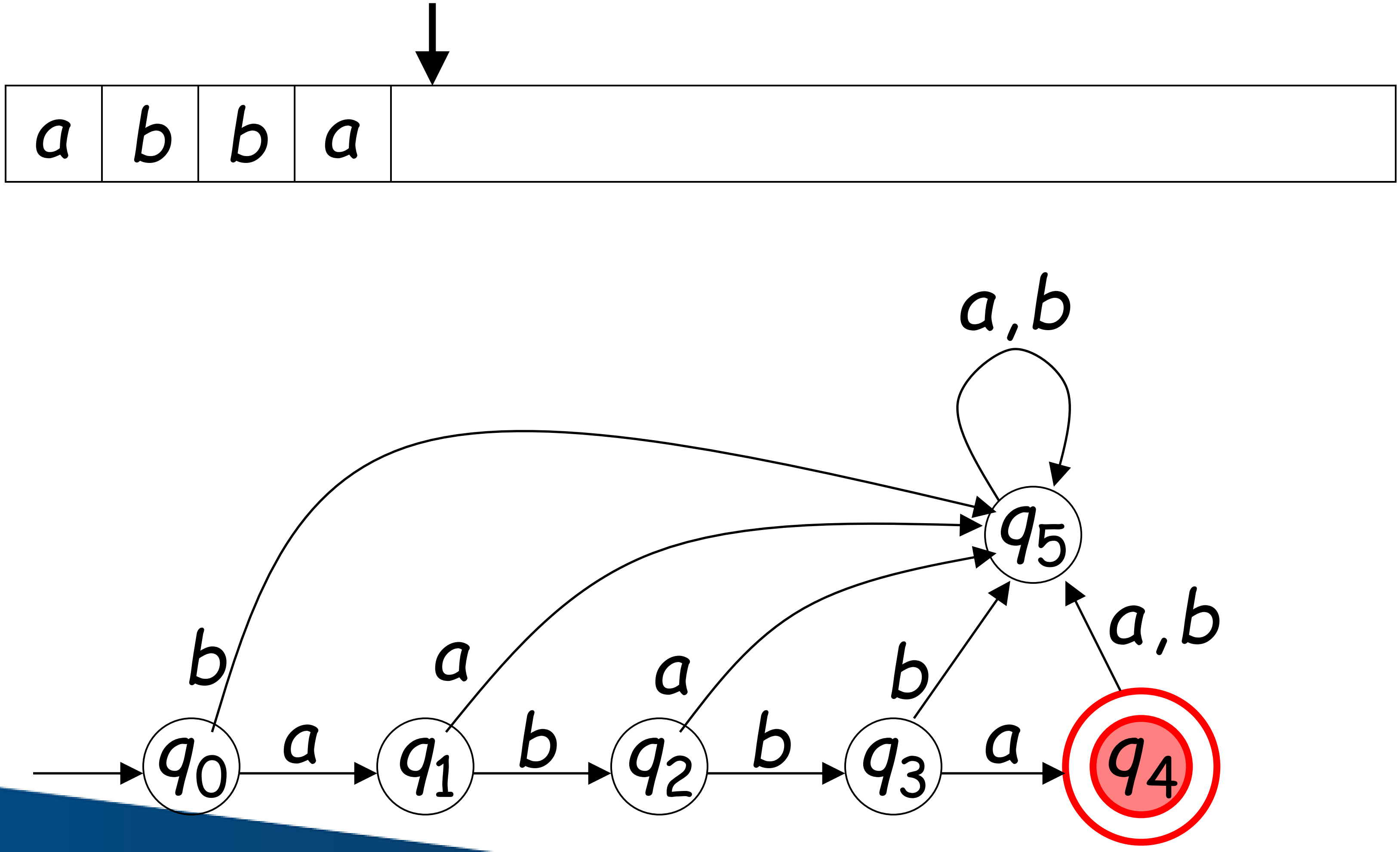



a b b a
a, b
b
a
a
b
a
b
b
a
a, b
q_0
q_1
q_2
q_3
q_4
q_5


a
b
b
a
a,b
b
a
a
b
a,b
q_0
q_1
q_2
q_3
q_4
q_5
a
b
b
a


a
b
b
a
a,b
q5
b
a
a
b
a,b
q0
a
q1
b
q2
b
q3
a
q4

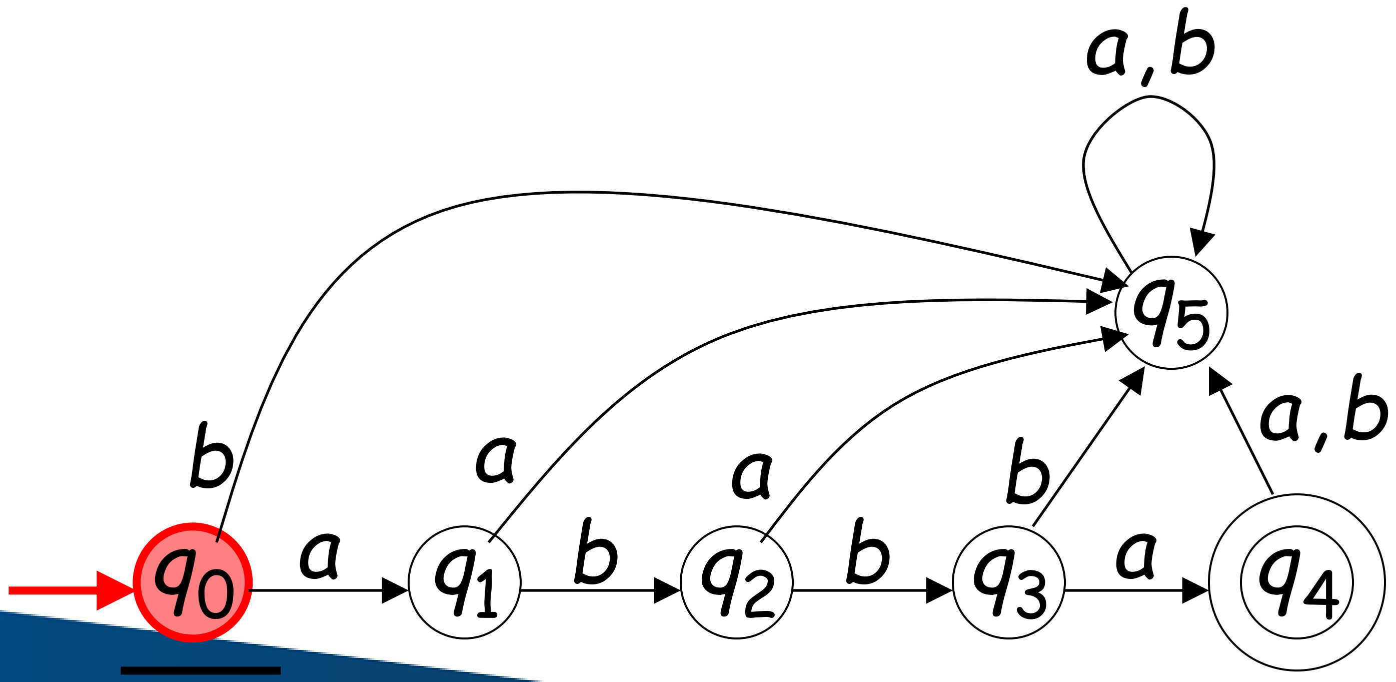
Input finished

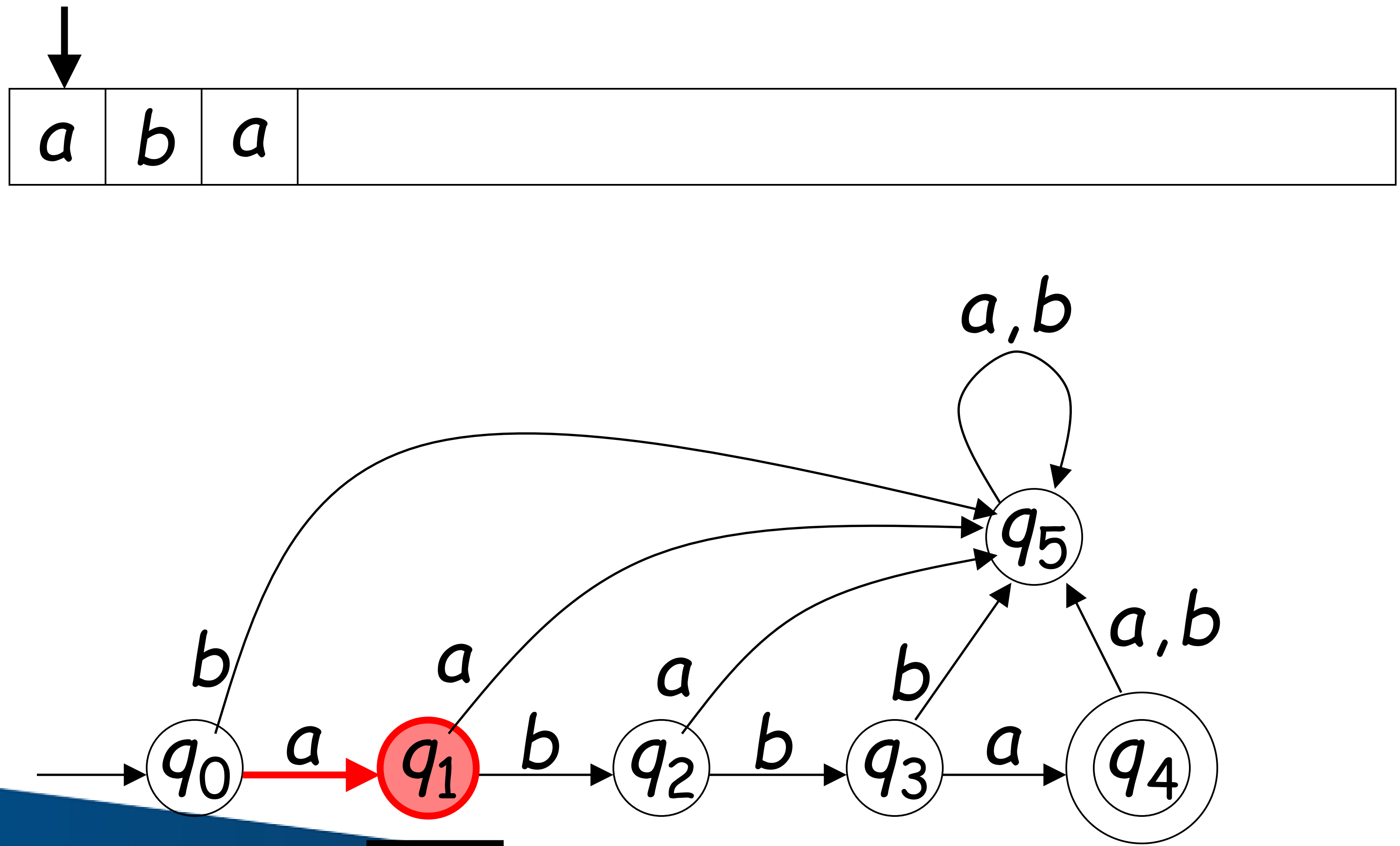


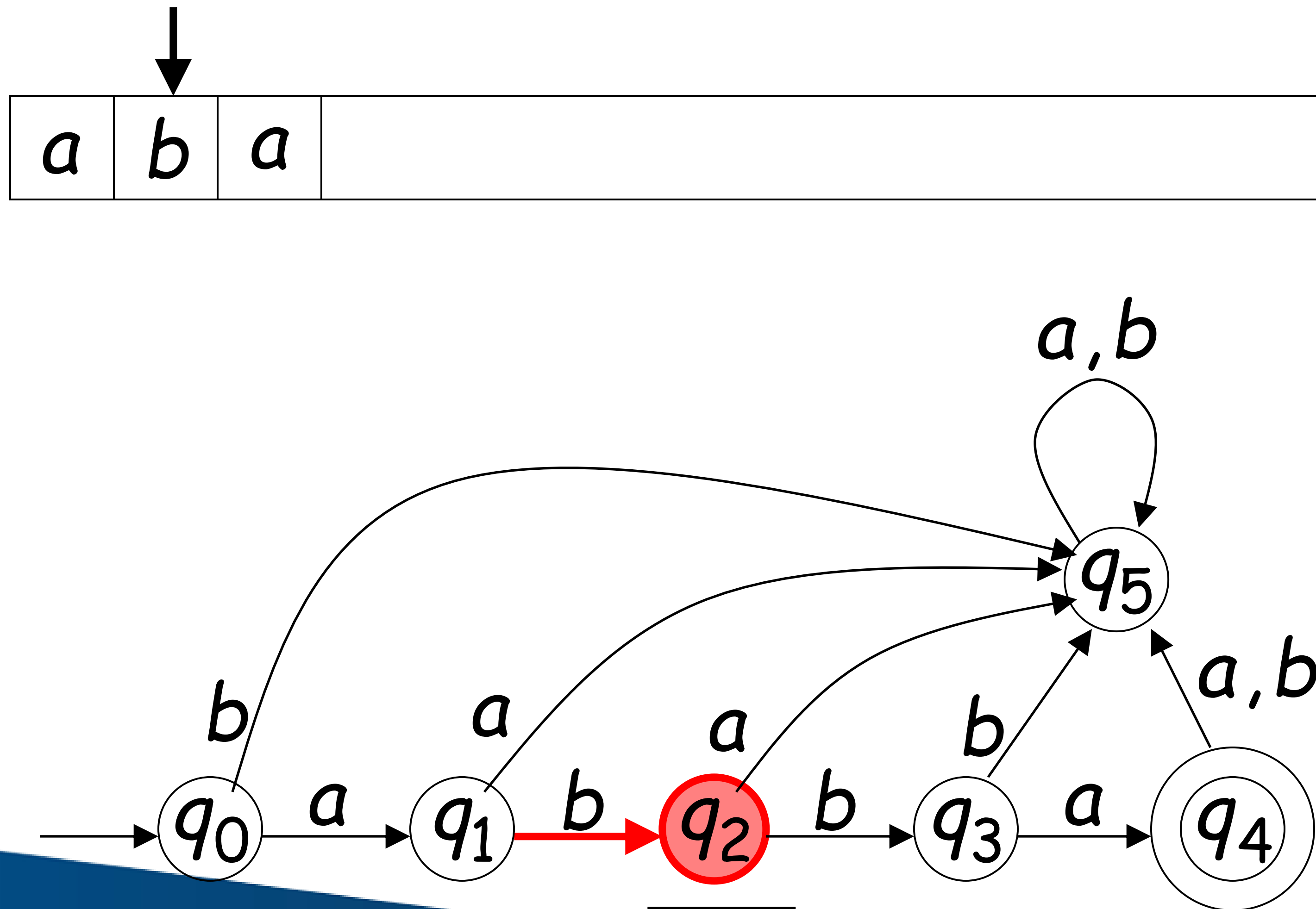
accept

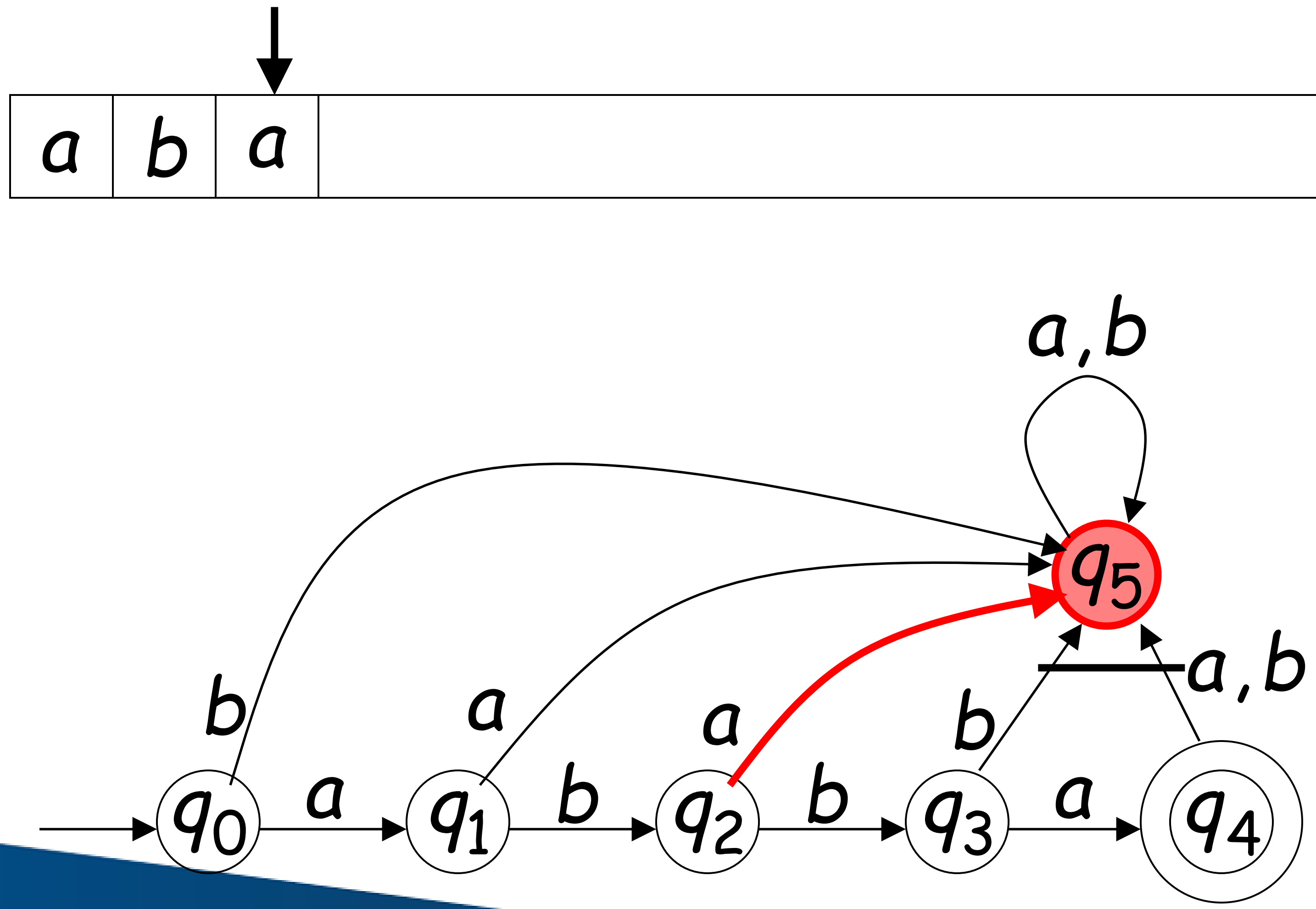
Rejection

a	b	a	



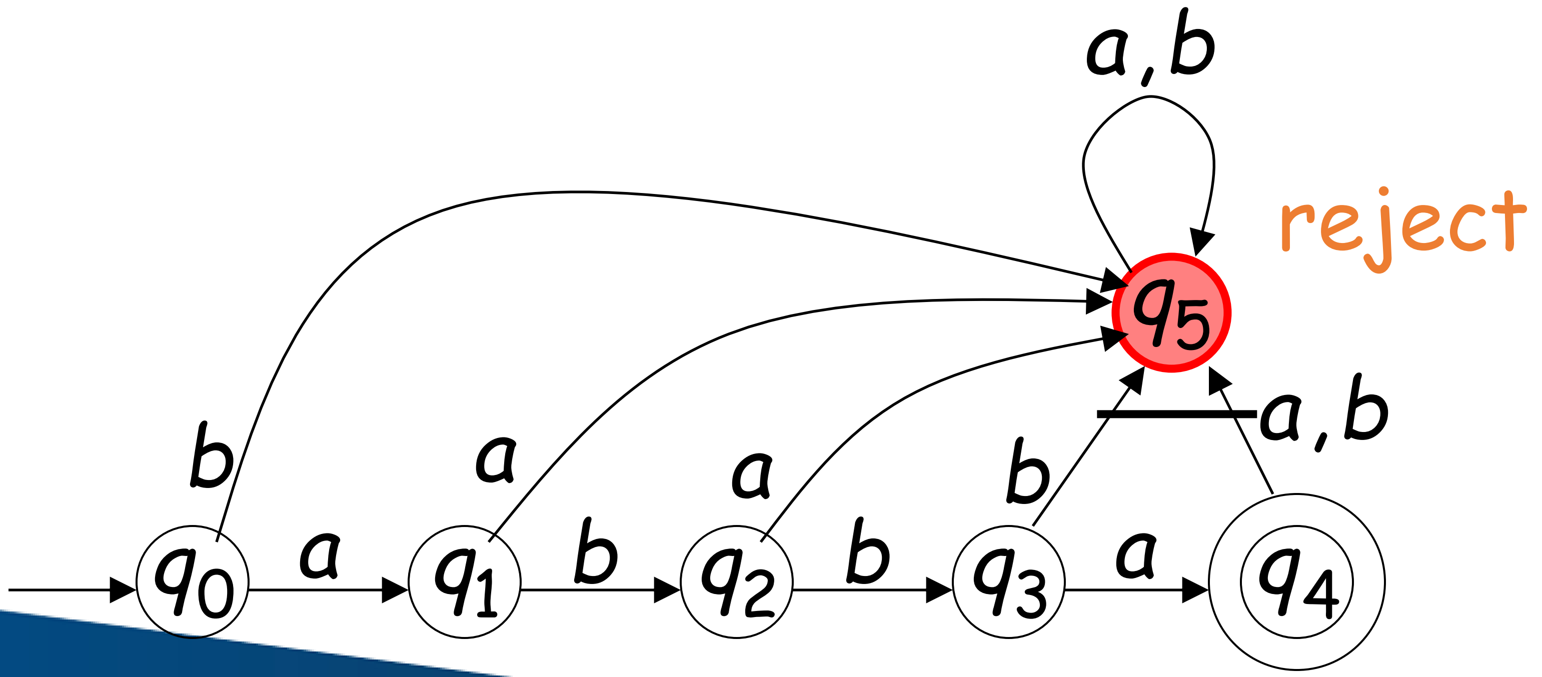

a
b
a
a,b
q5
b
a
a
b
a,b
q0
a
q1
b
q2
b
q3
a
q4


a b a
a,b
b
a
a
b
a
a,b
q0 a q1 b q2 b q3 a q4
q5


a
b
a
a,b
q5
a,b
b
a
a
b
q0
a
q1
b
q2
b
q3
a
q4

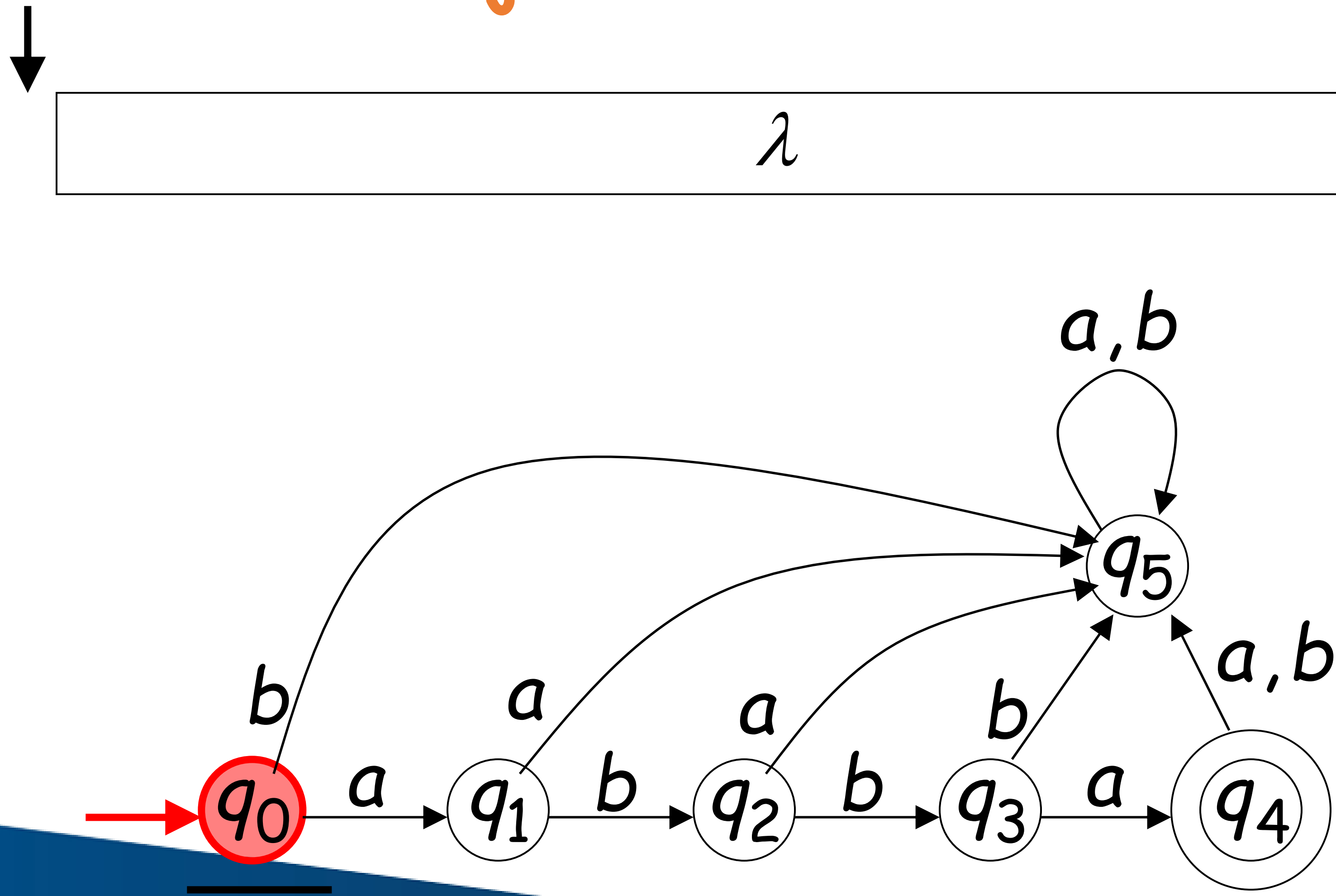
Input finished

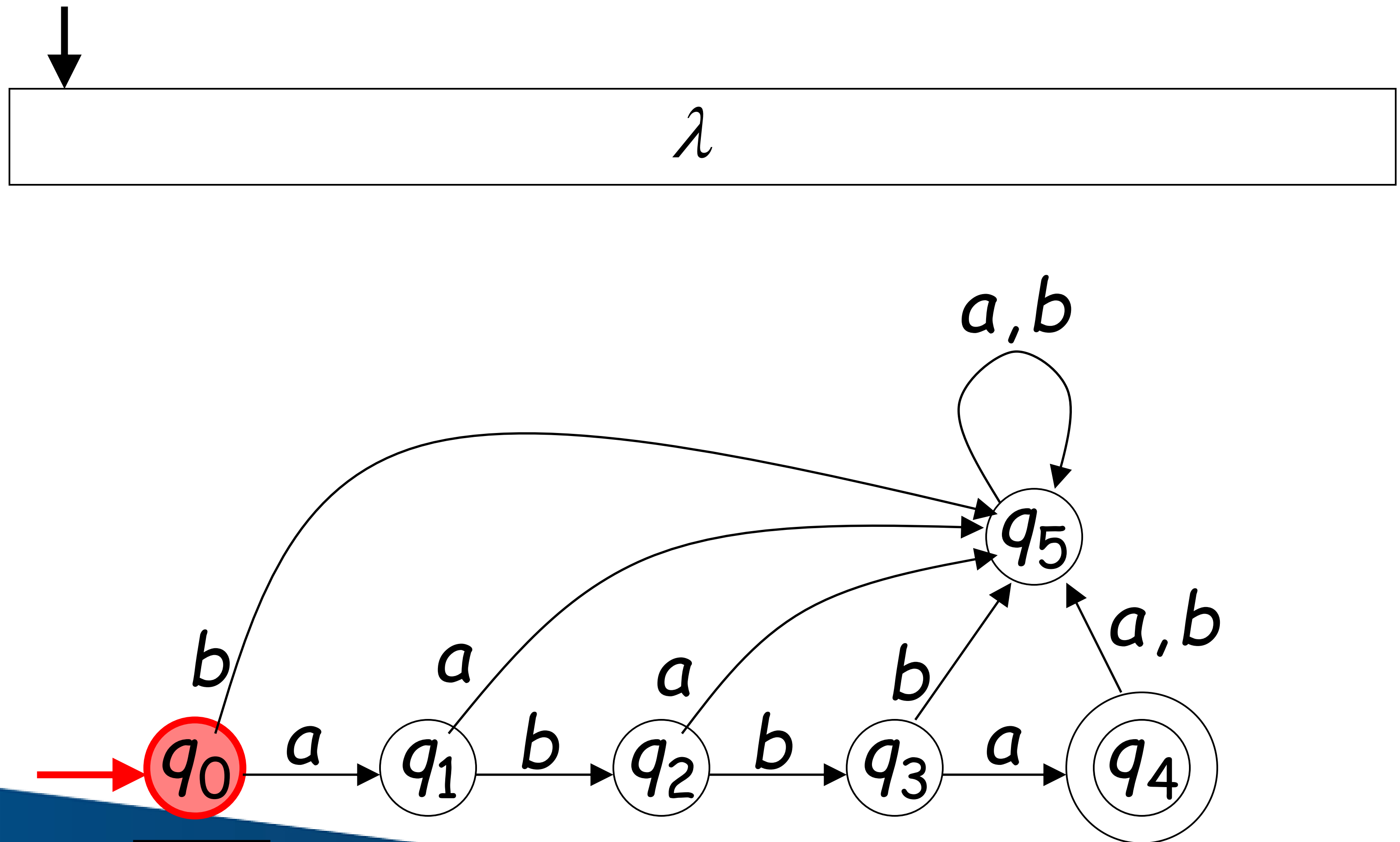
a	b	a	



Another Rejection

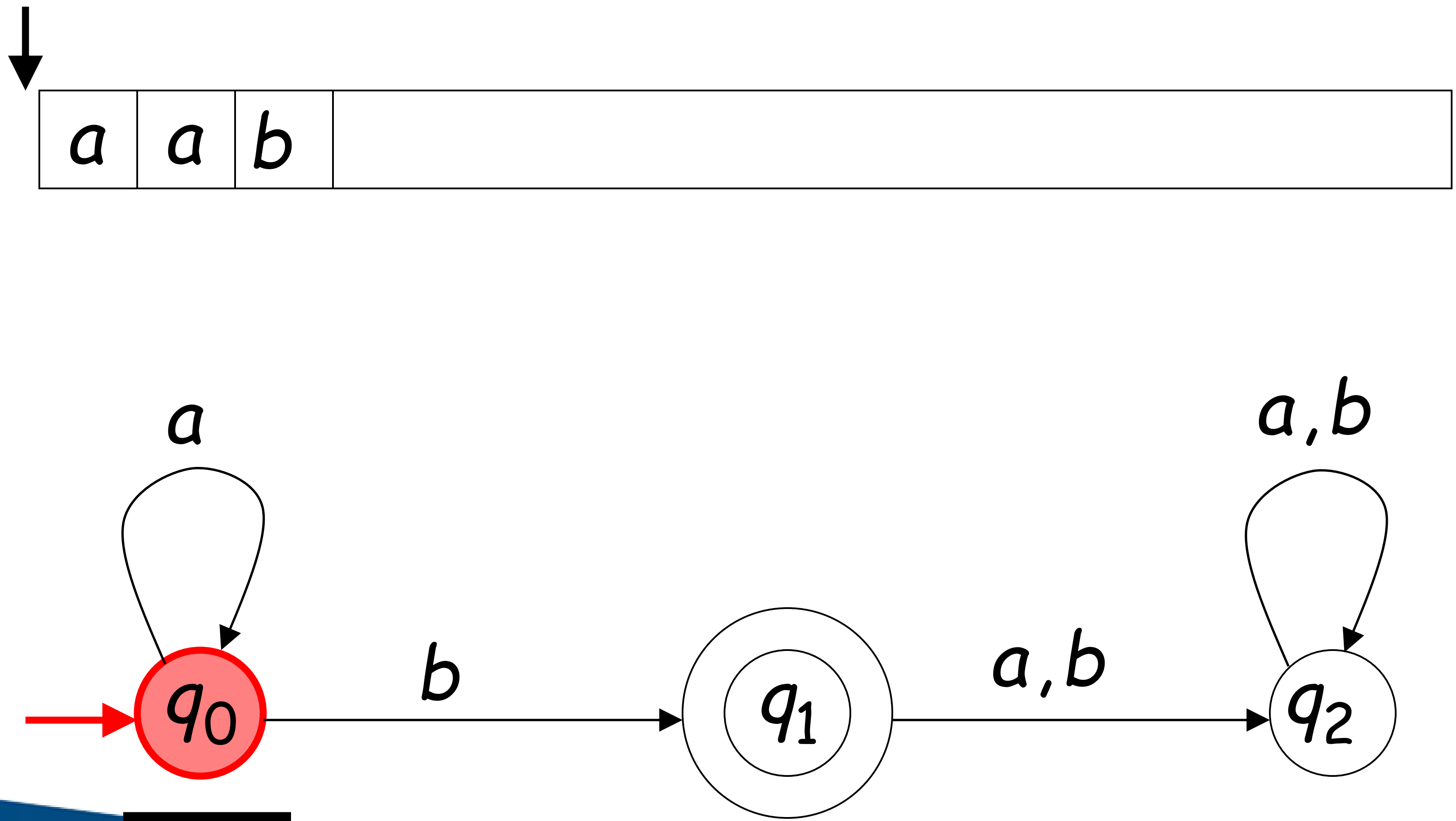
λ

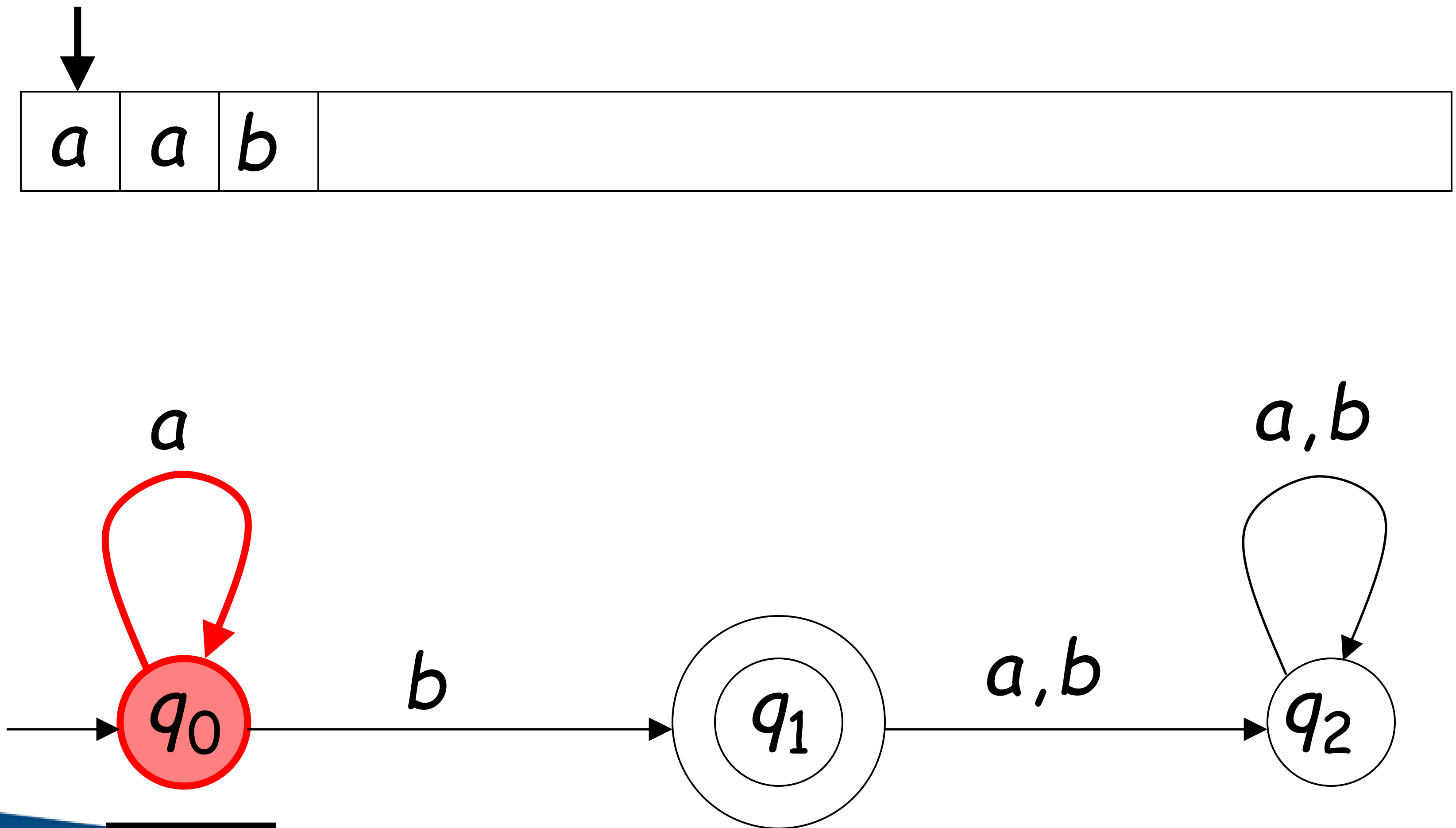


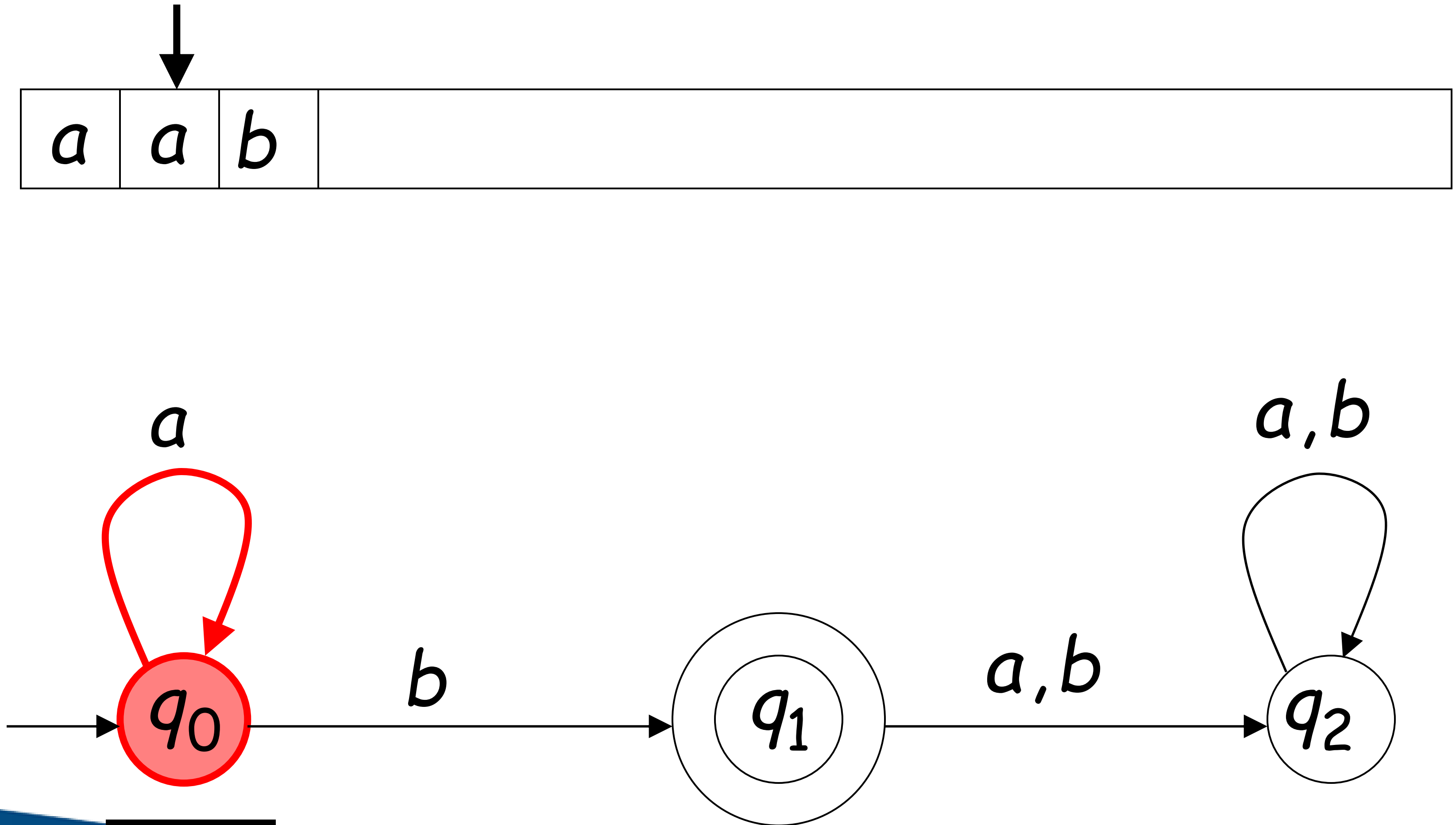


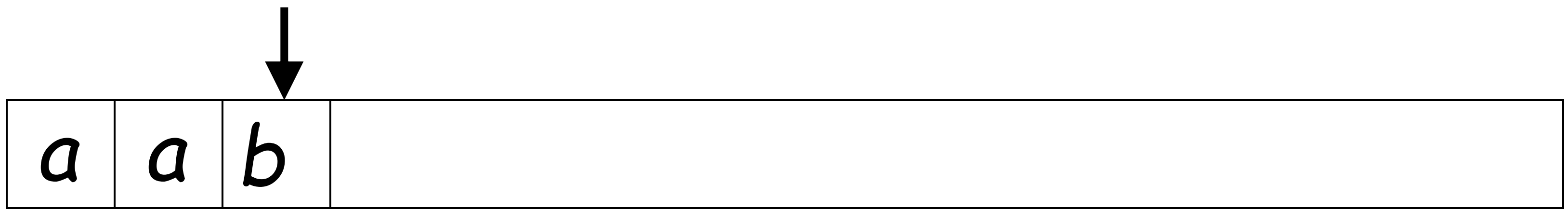
reject

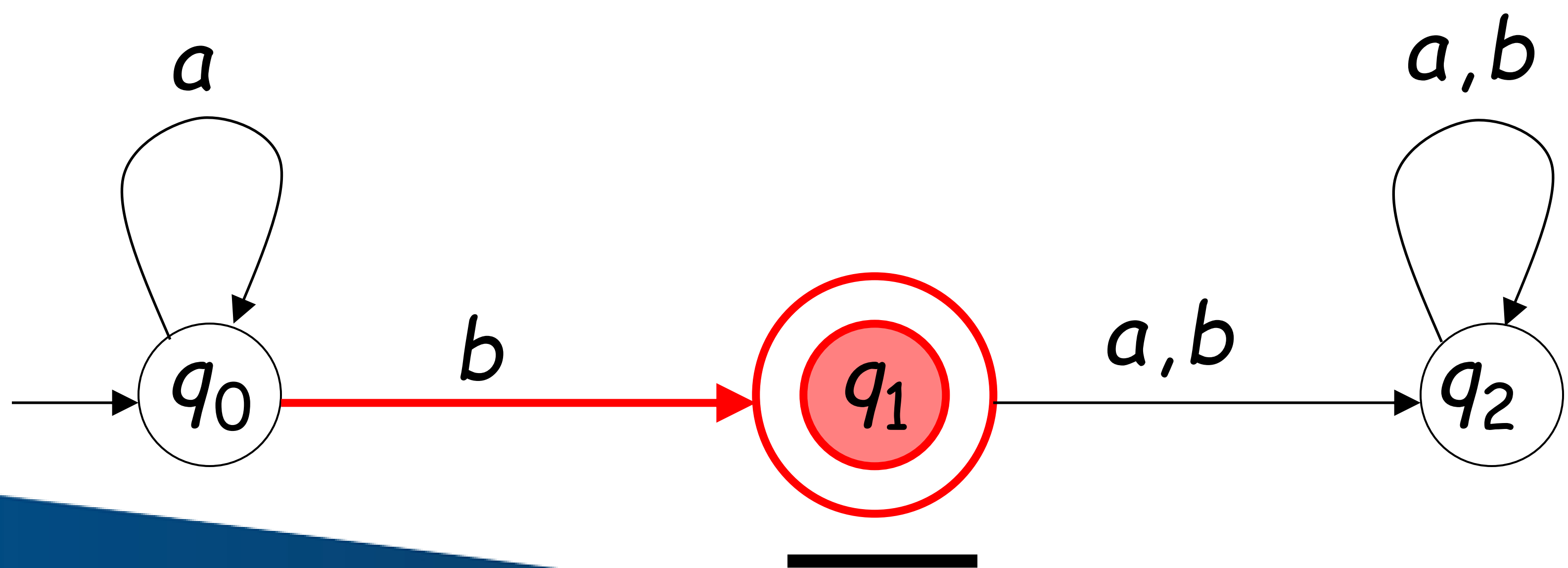
Another Example



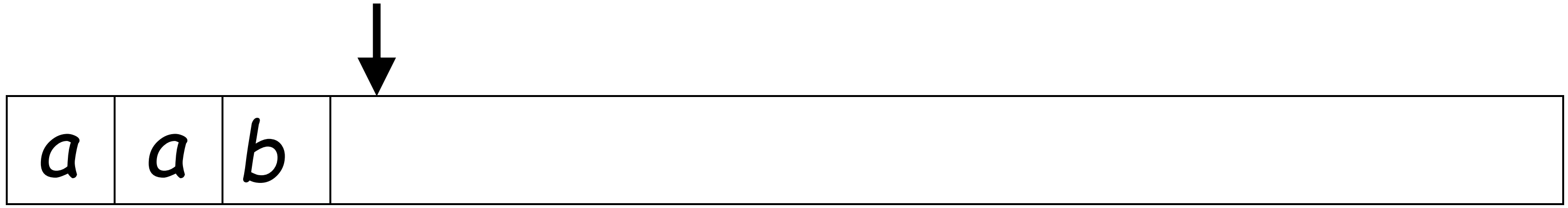

a a b
a
a,b
b
a,b
q_0
q_1
q_2

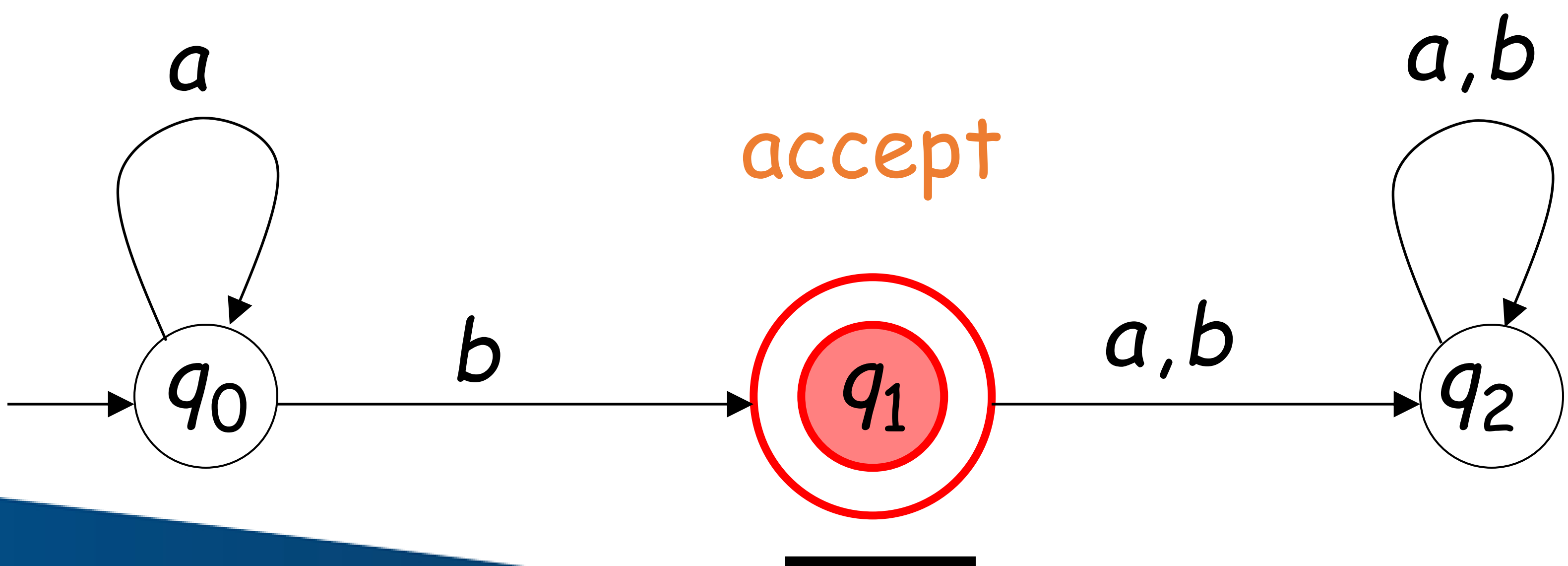

a a b
a
b
a,b
a,b
q_0
q_1
q_2


a
a
b

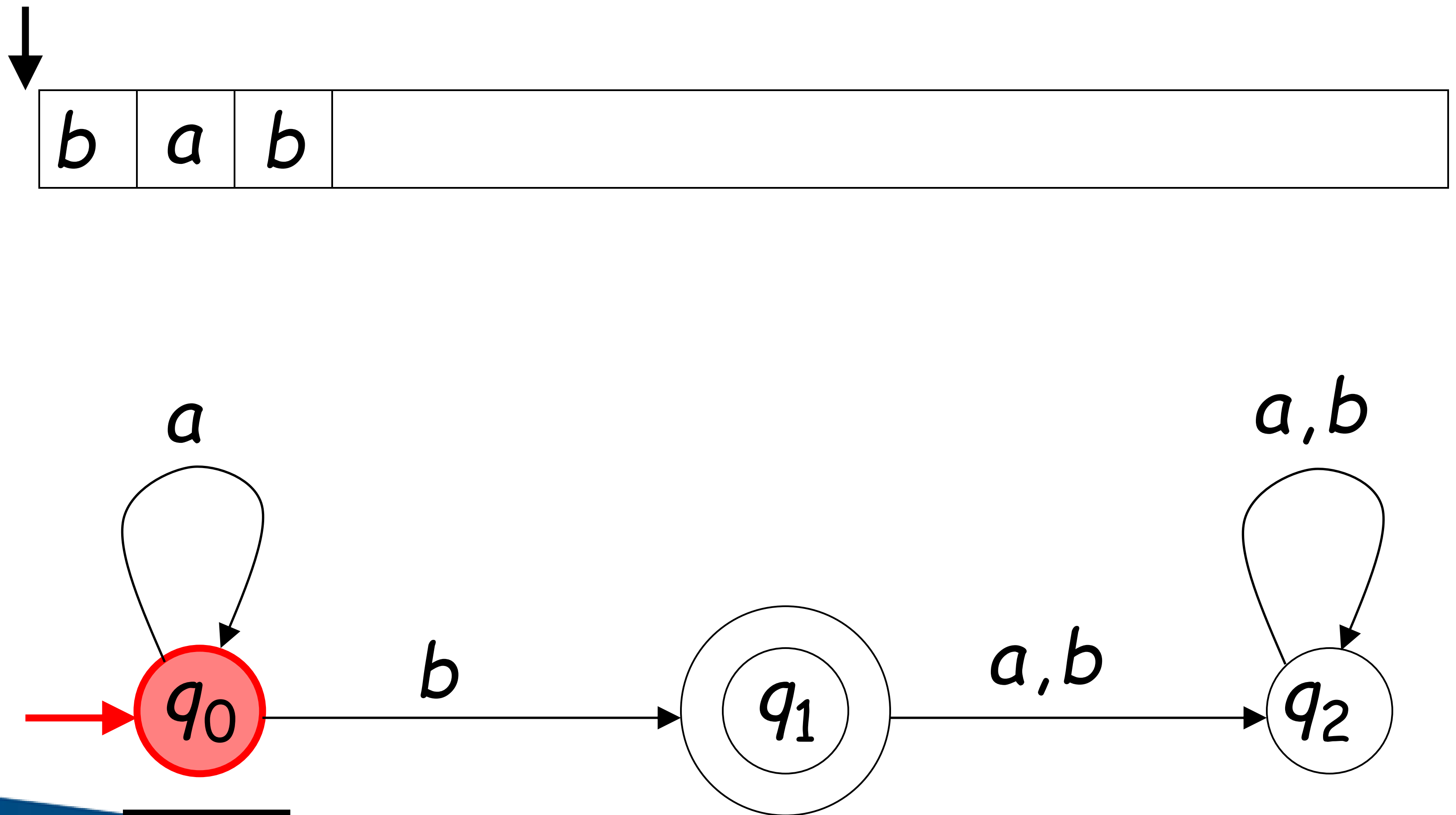

a
a,b
b
a,b
q_0
q_1
q_2

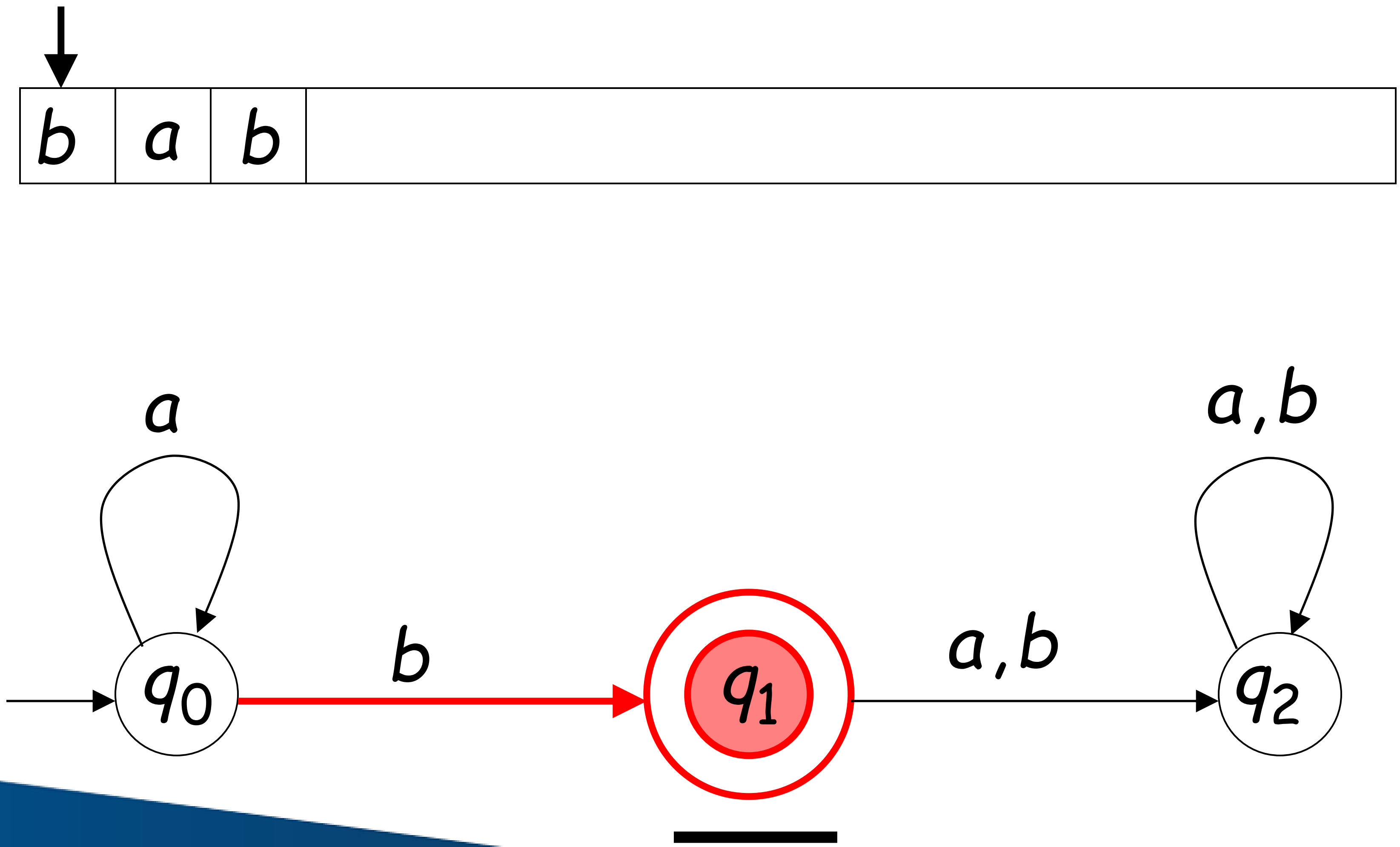
Input finished

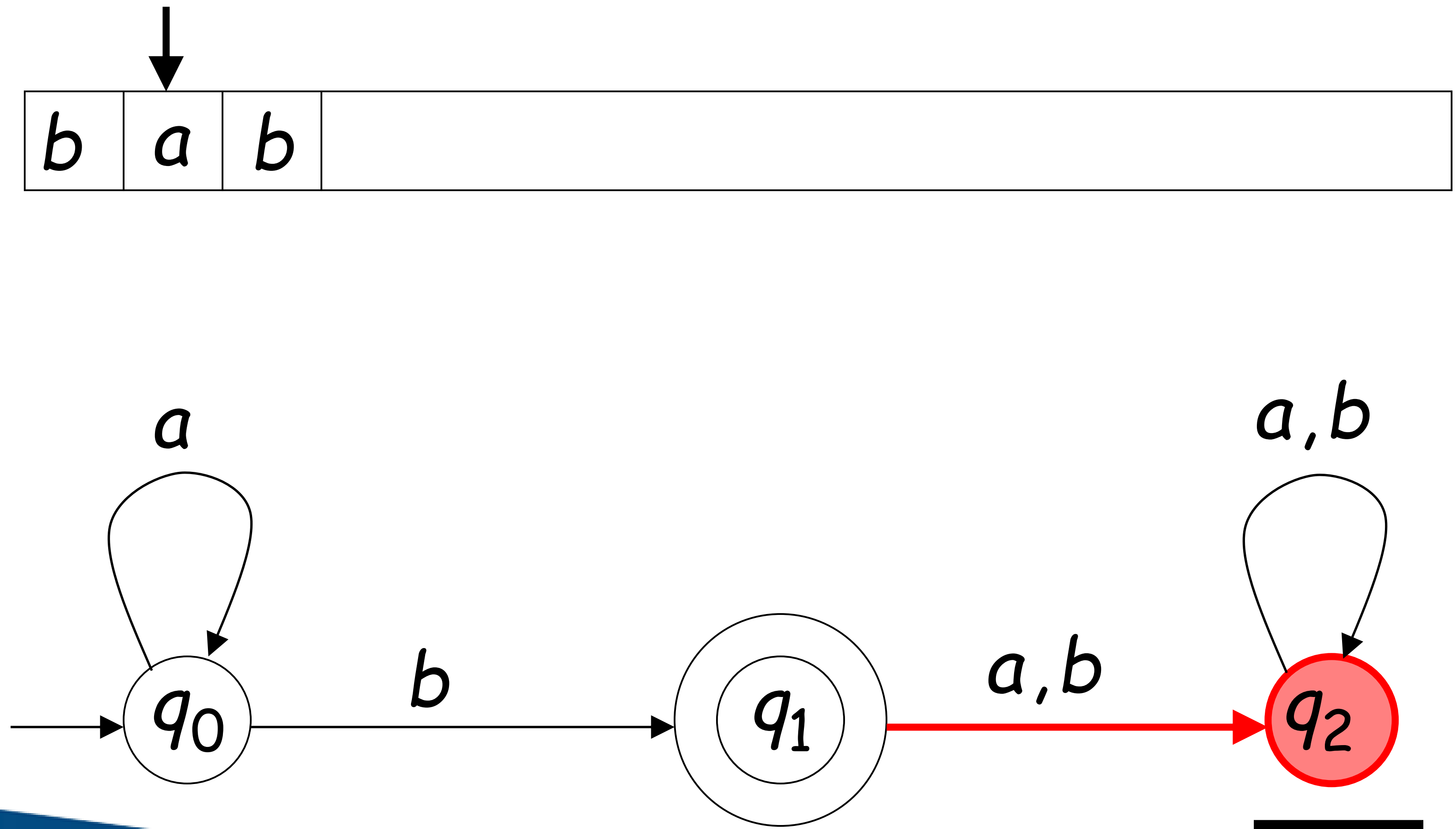


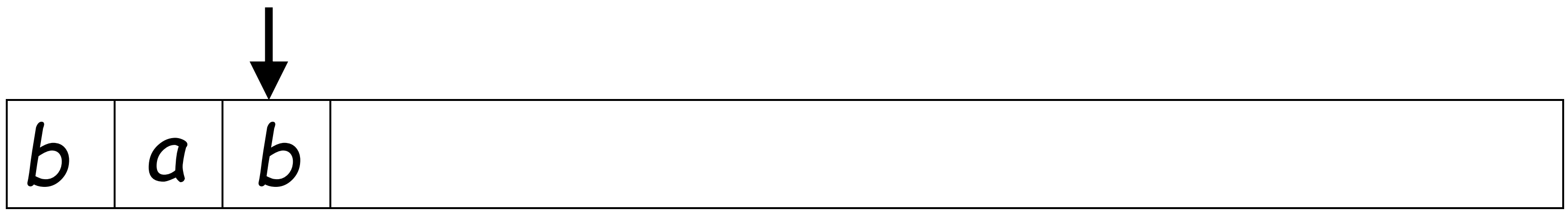


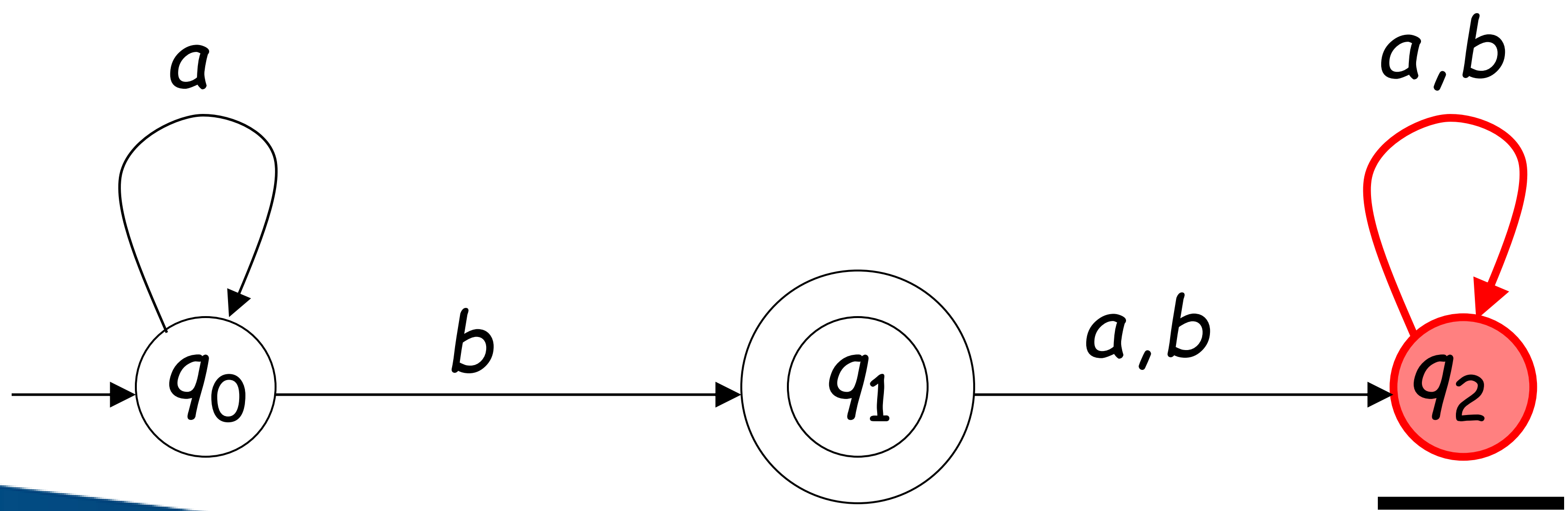
Rejection Example



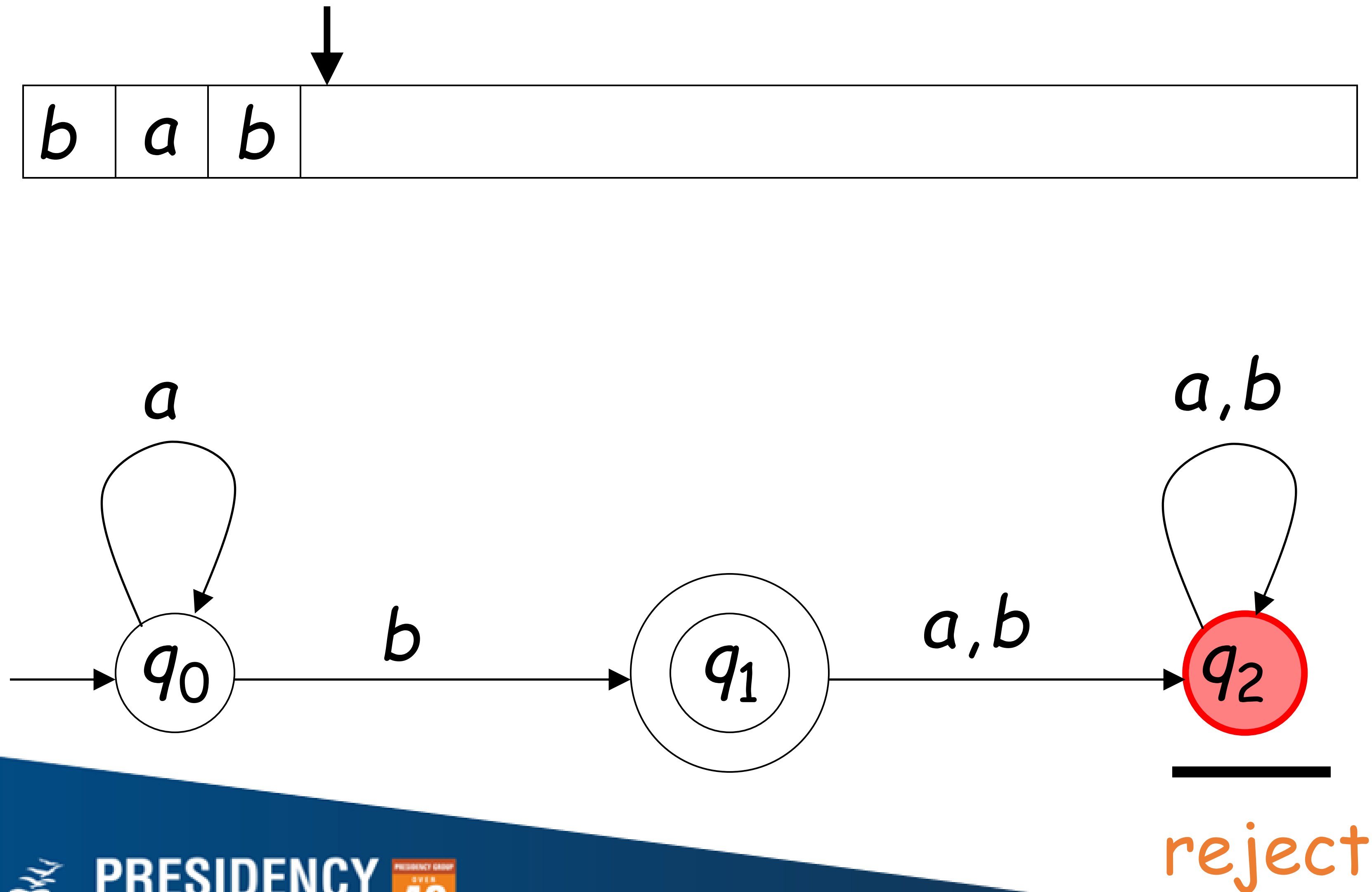

b
a
b
a
a,b
b
a,b
q_0
q_1
q_2


b a b
a
a,b
b
a,b
q_0
q_1
q_2


b
a
b


a
b
a,b
a,b
q_0
q_1
q_2

Input finished



Languages Accepted by FAs

Definition:

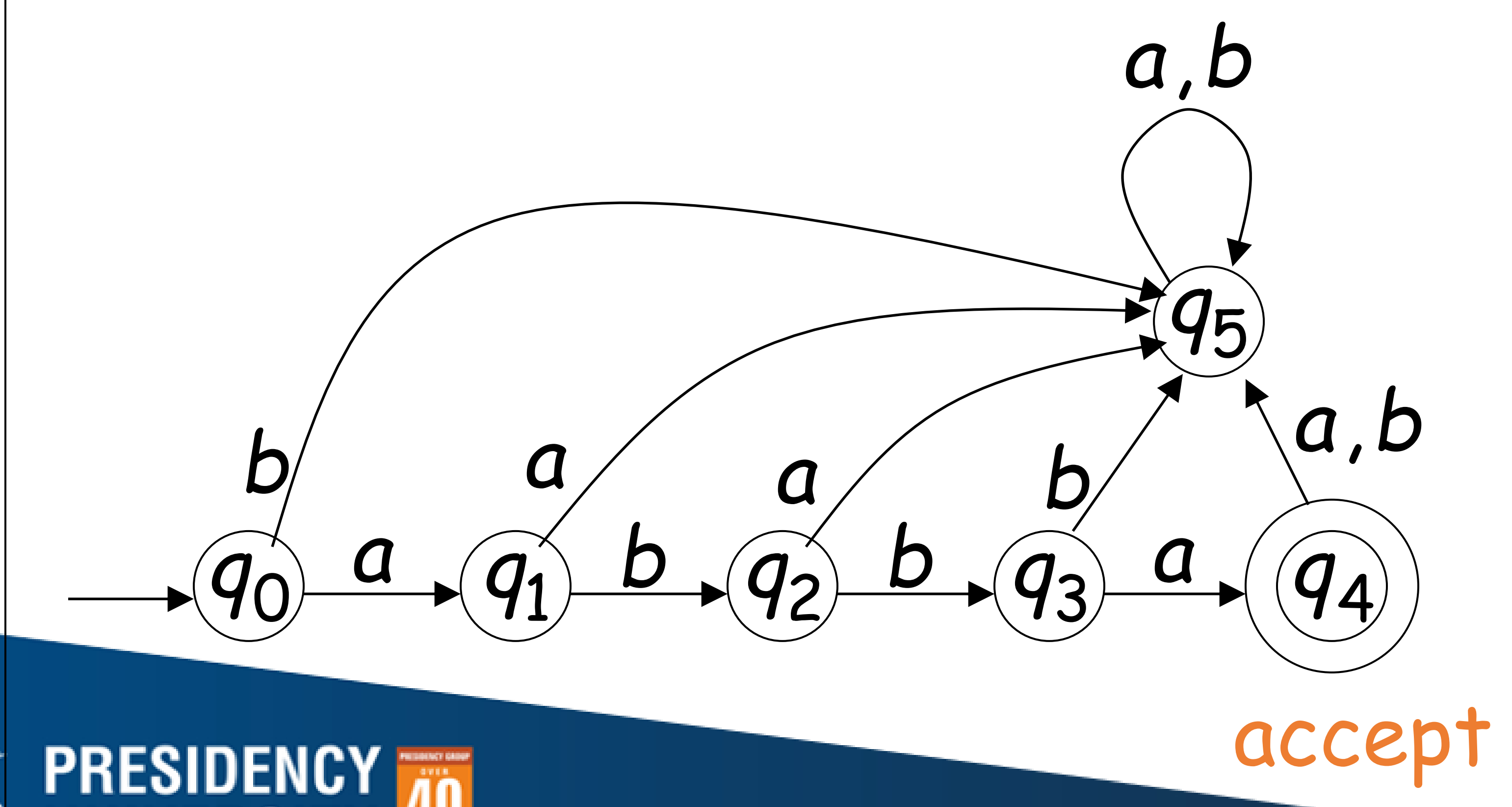
The language $L(M)$ contains all input strings accepted by M

$L(M)$ = { strings that bring M to an accepting state}

Example

$$L(M) = \{abba\}$$

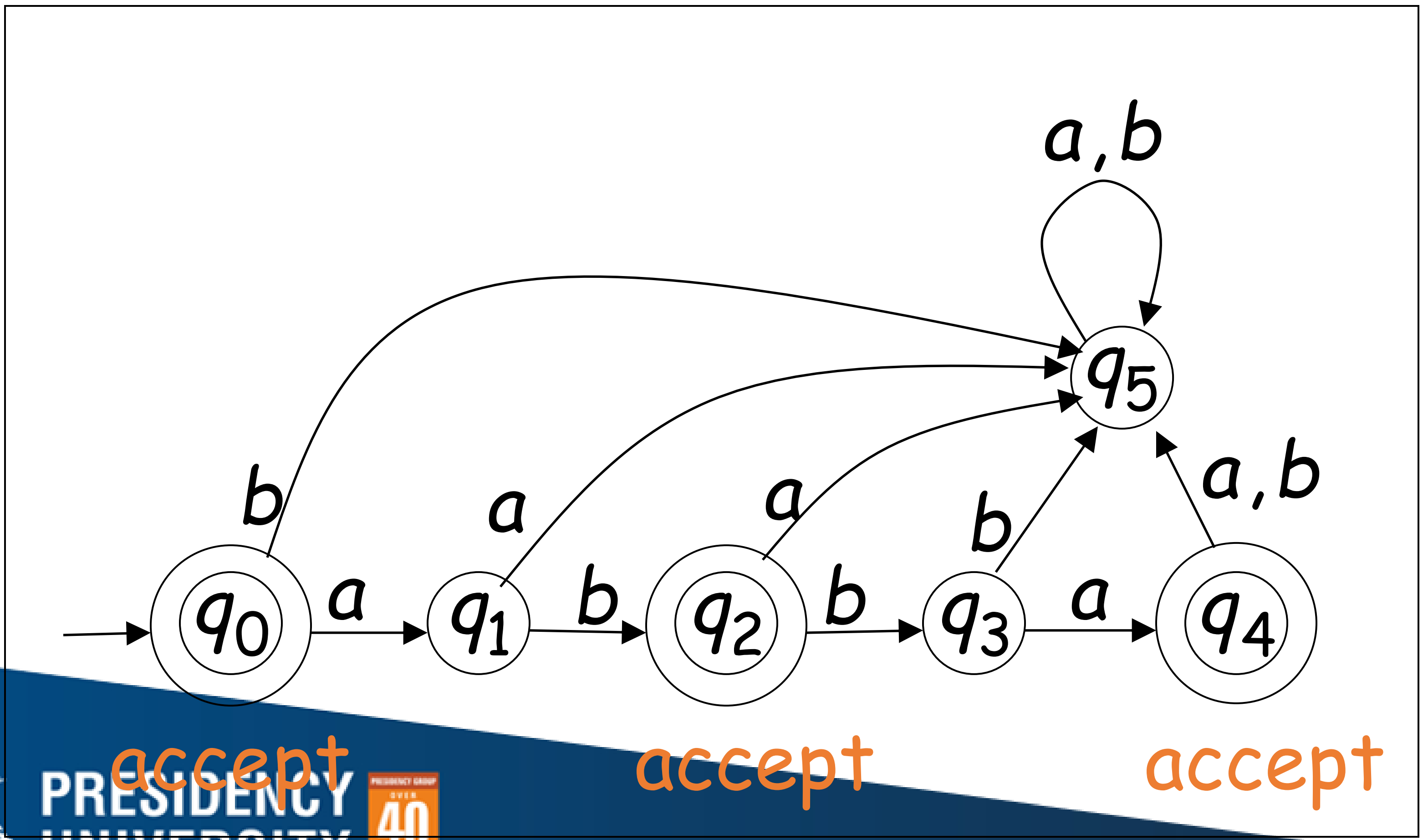
M



Example

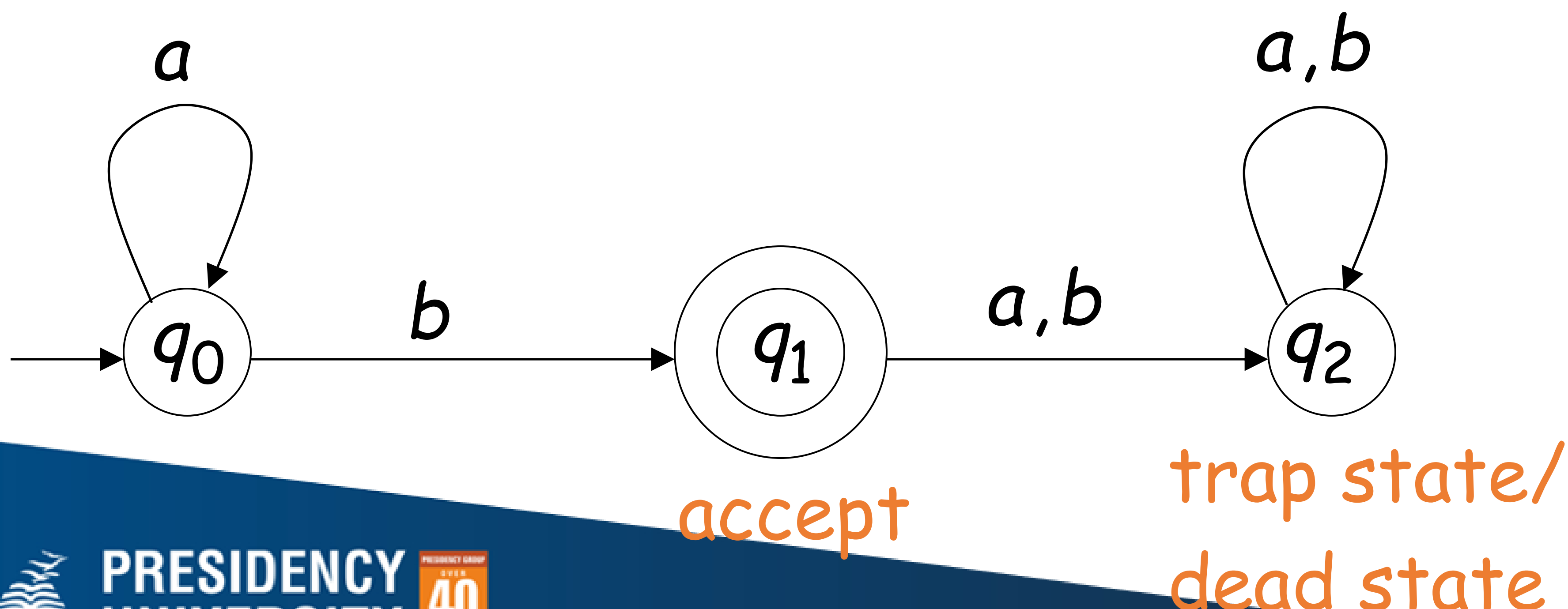
$$L(M) = \{\lambda, ab, abba\}$$

M



Example

$$L(M) = \{a^n b : n \geq 0\}$$



Formal Definition

- Finite Automaton (FA)

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

Q : set of states

Σ : input alphabet

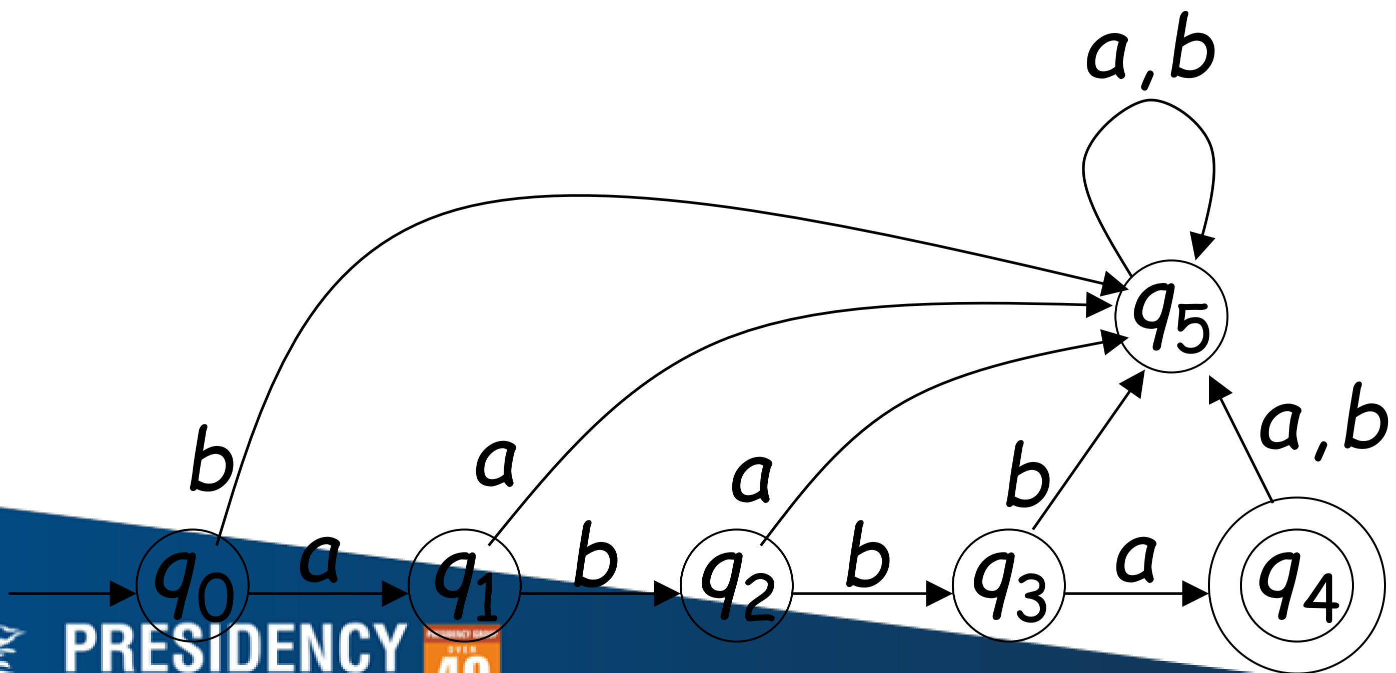
δ : transition function

q_0 : initial state

F : set of accepting states

Input Alphabet Σ

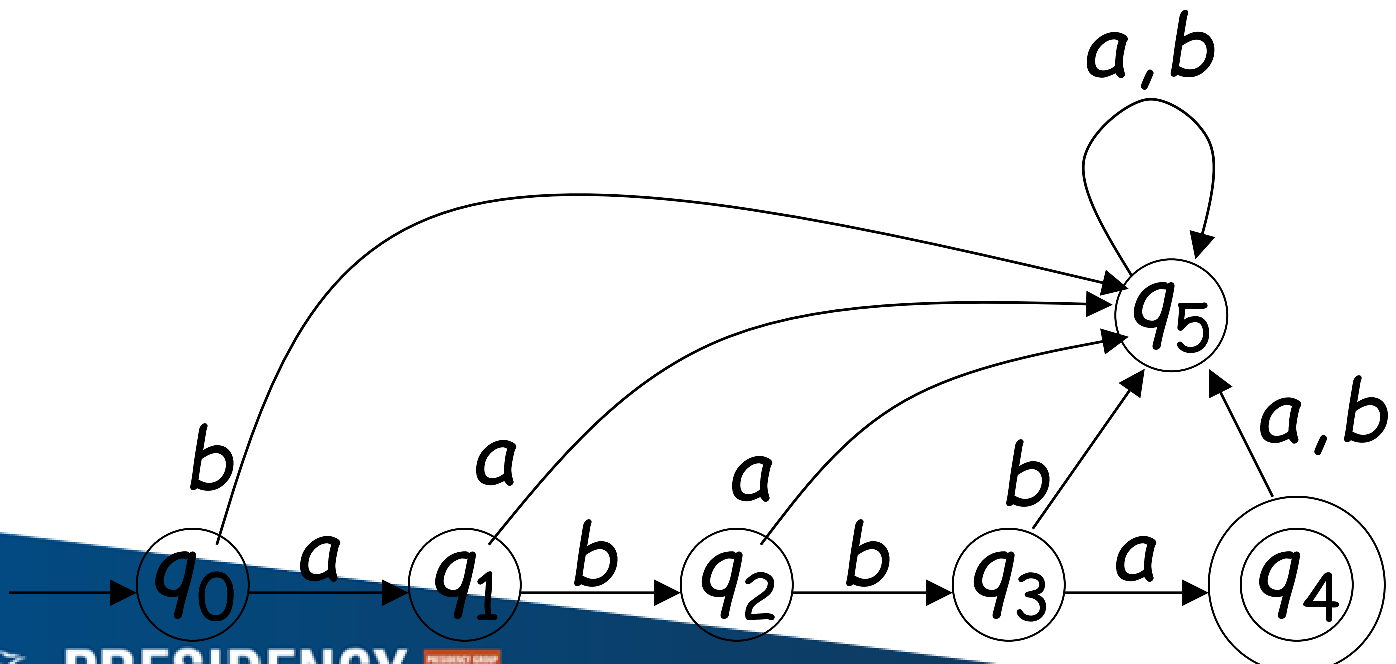
- $\Sigma = \{a, b\}$



PRESIDENCY UNIVERSITY
GAIN MORE KNOWLEDGE REACH GREATER HEIGHTS
Private University Estd. in Karnataka State by Act No. 41 of 2013
OVER 40 YEARS OF ACADEMIC WISDOM

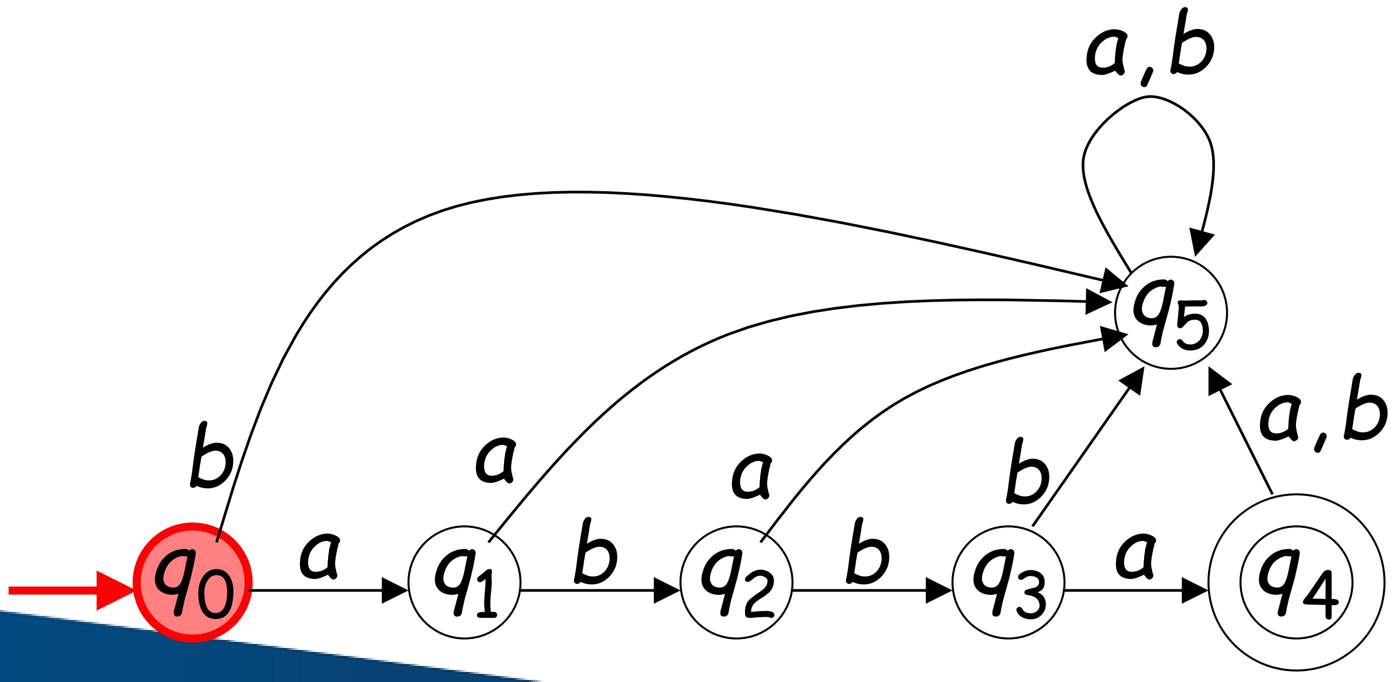
Set of States Q

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



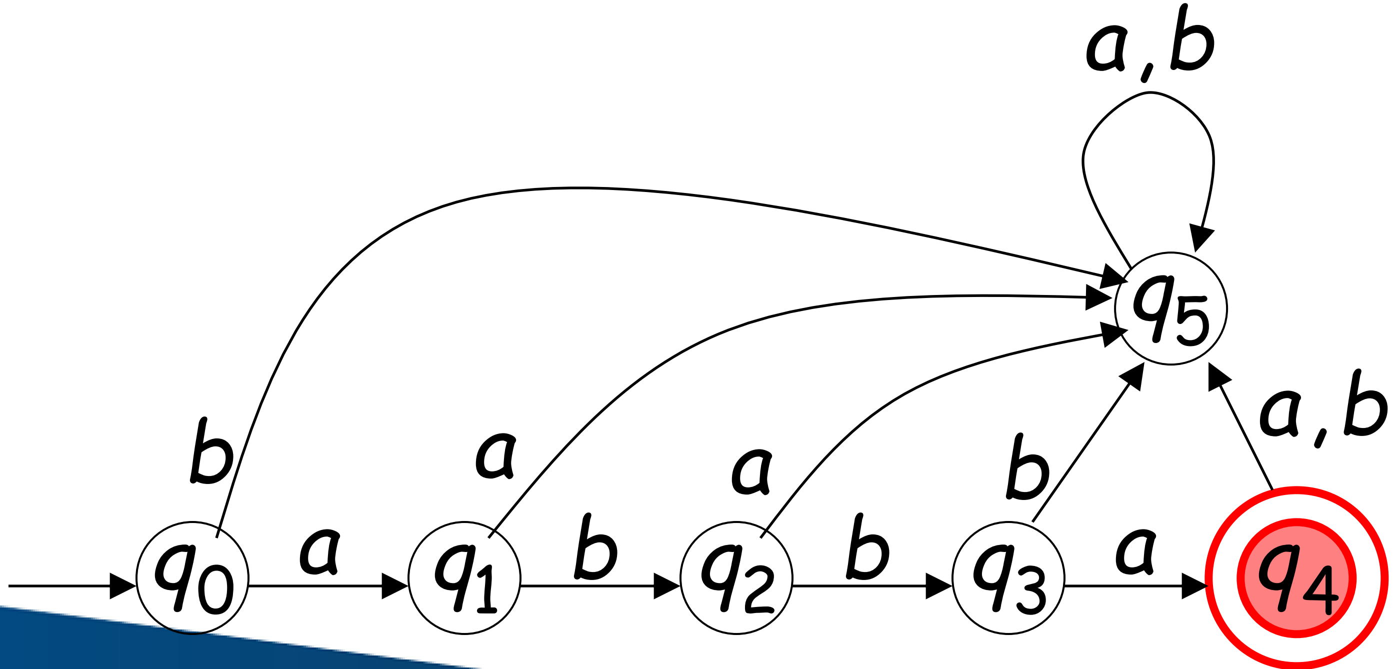
Initial State q_0

•



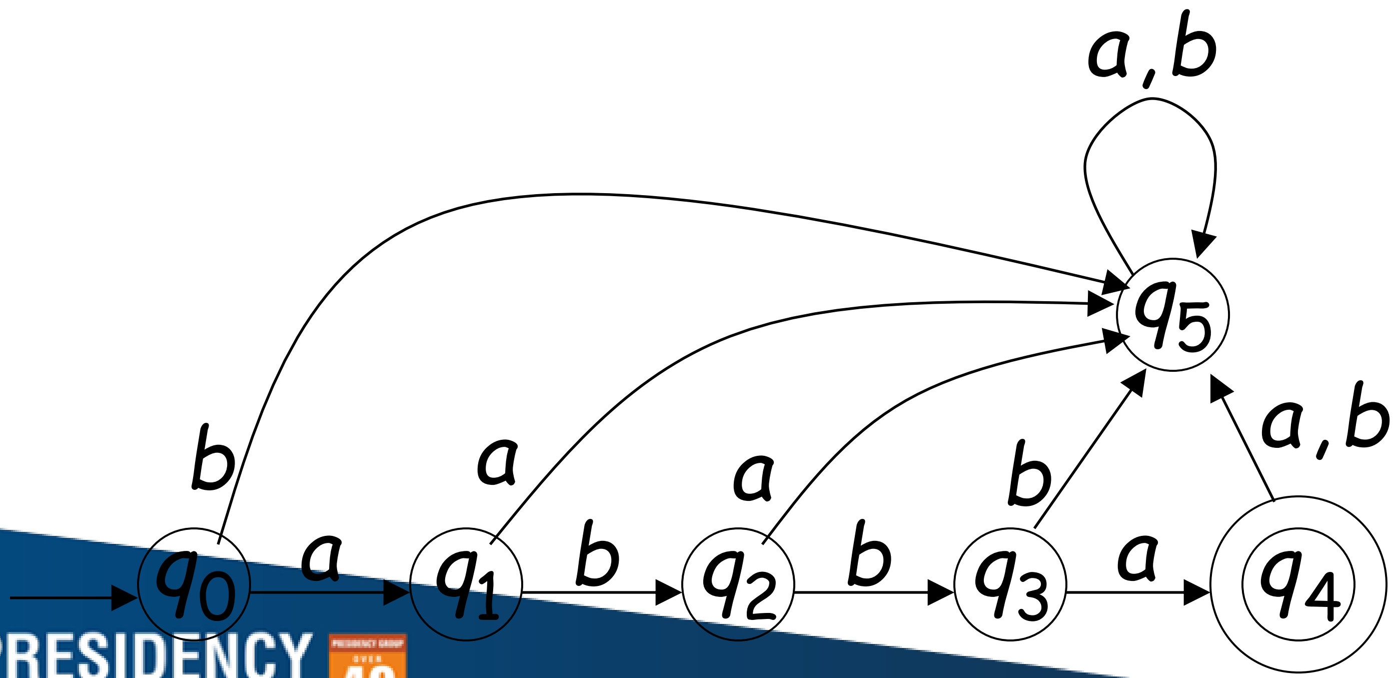
Set of Accepting States F

- $F = \{q_4\}$

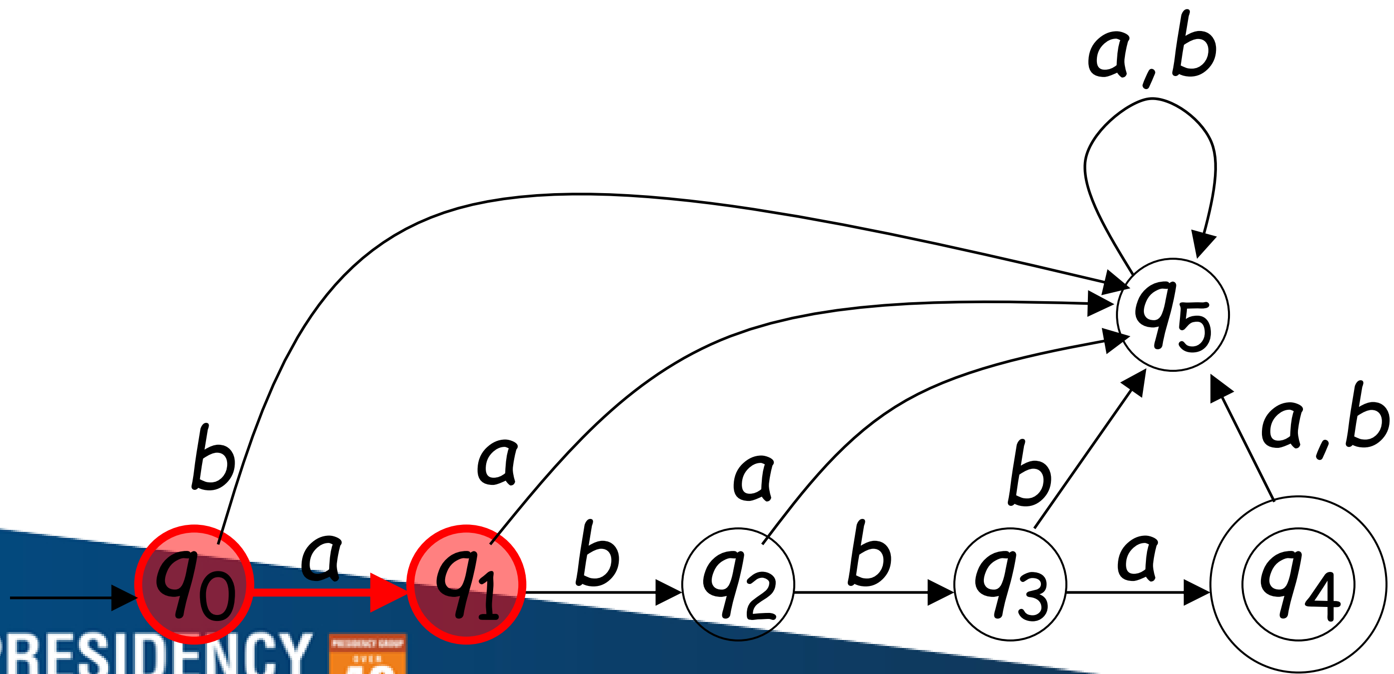


Transition Function δ

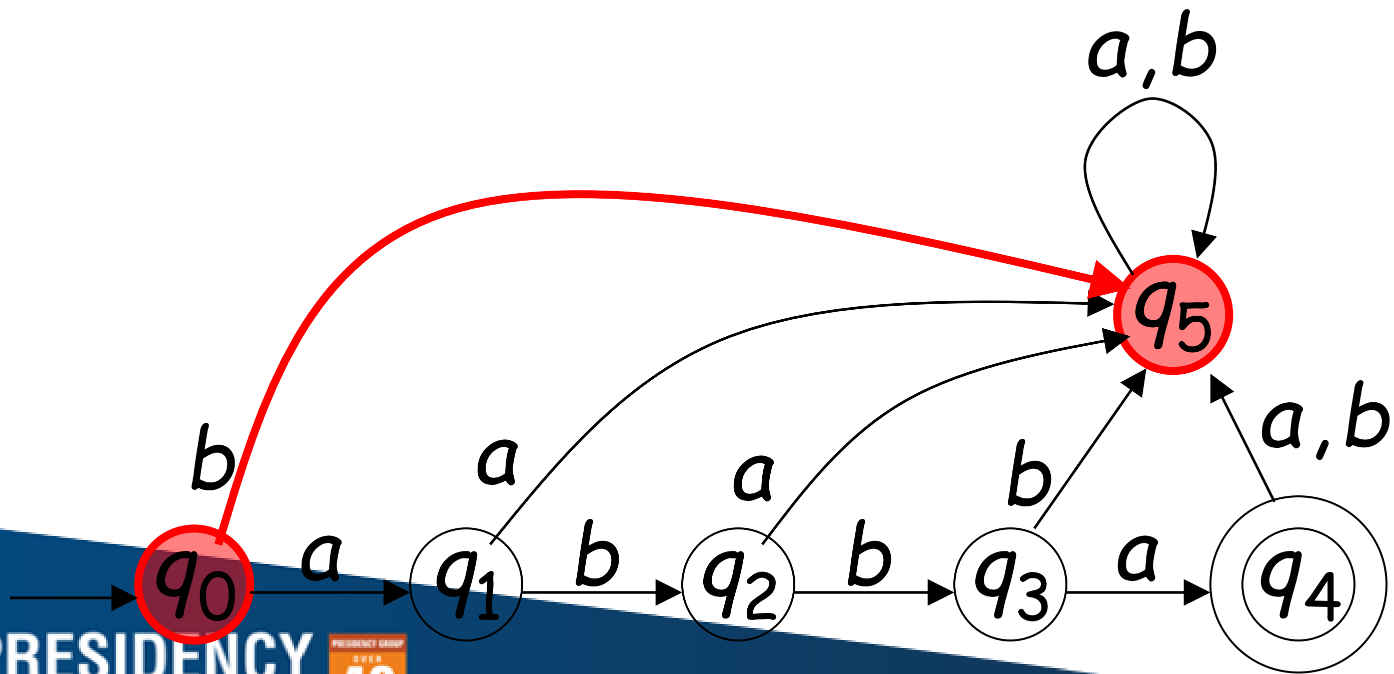
- $$\delta : Q \times \Sigma \rightarrow Q$$



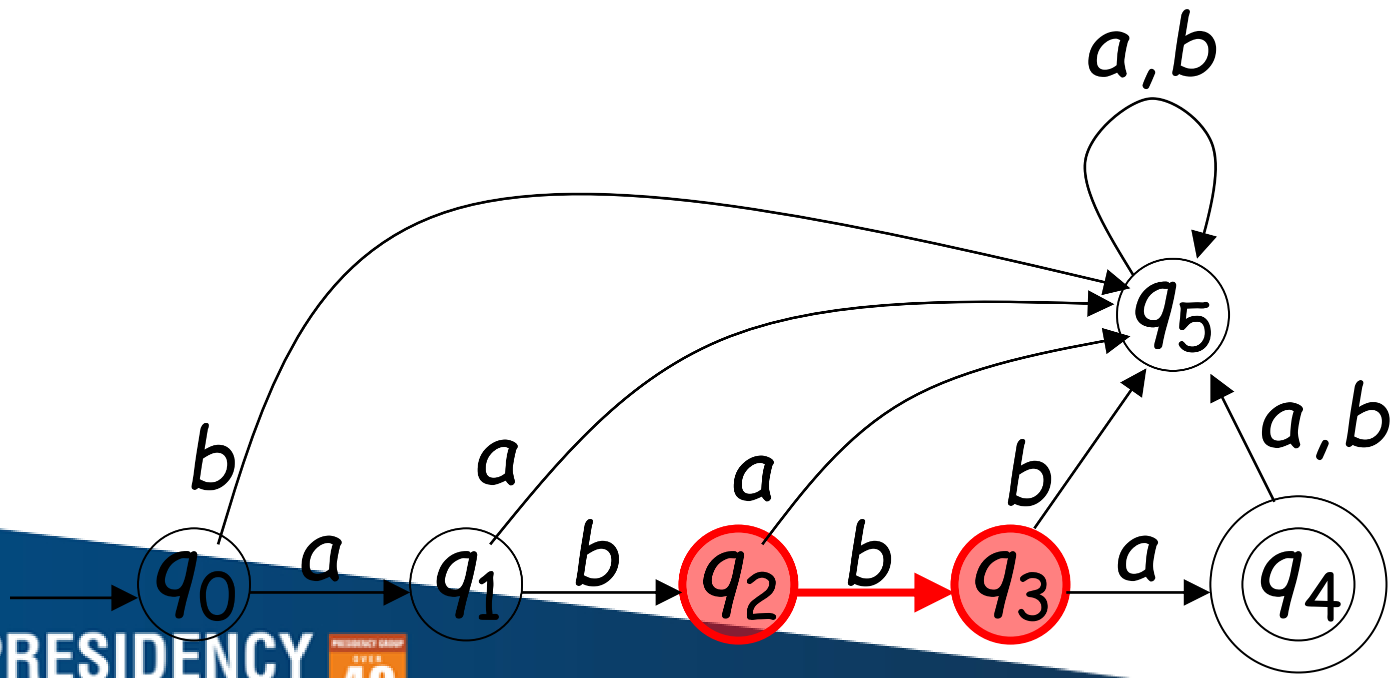
$$\delta(q_0, a) = q_1$$



$$\delta(q_0, b) = q_5$$

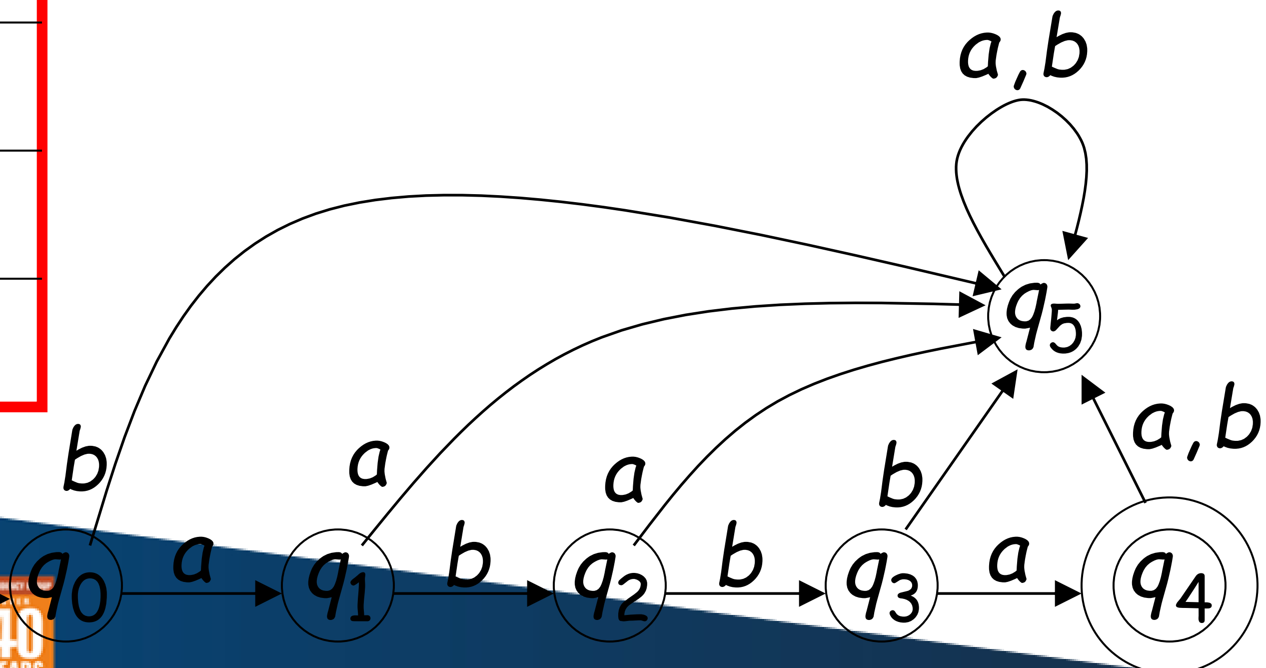


$$\delta(q_2, b) = q_3$$



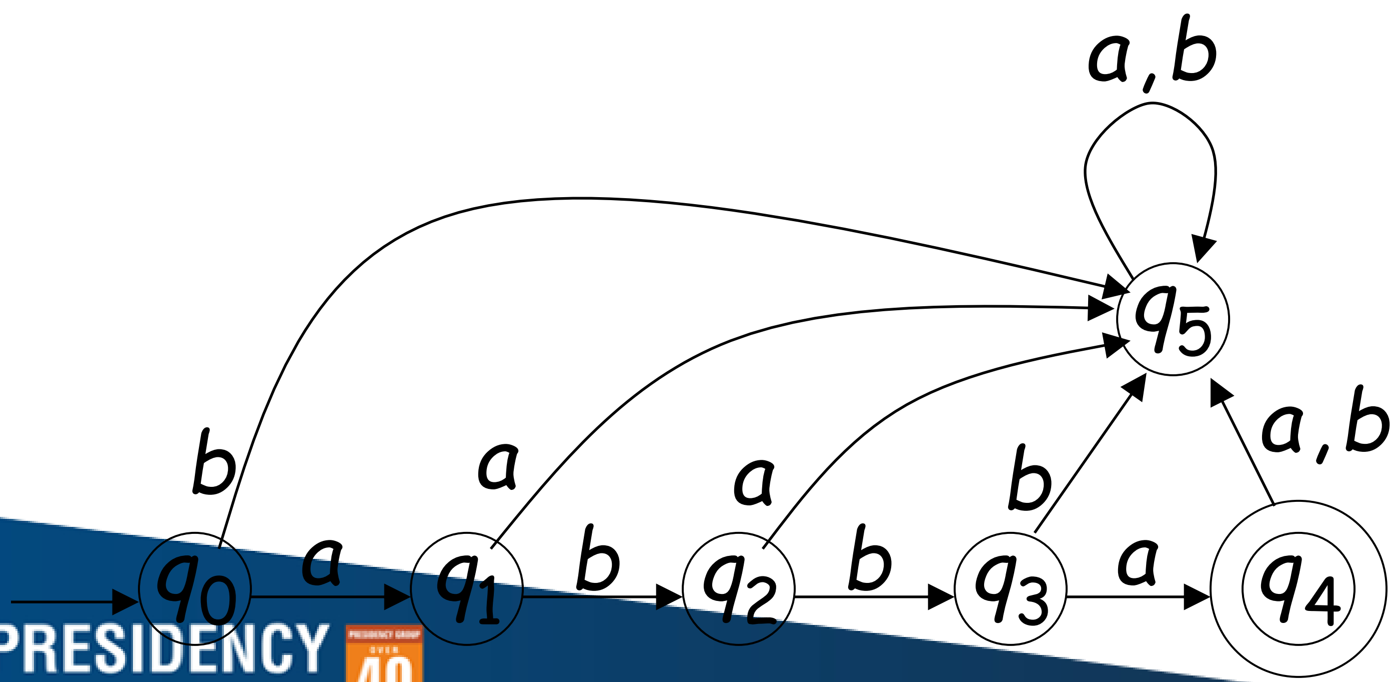
Transition Table δ

δ	a	b
$\rightarrow q_0$	q_1	q_5
q_1	q_5	q_2
q_2	q_5	q_3
q_3	q_4	q_5
q_4	q_5	q_5
q_5 *	q_5	q_5

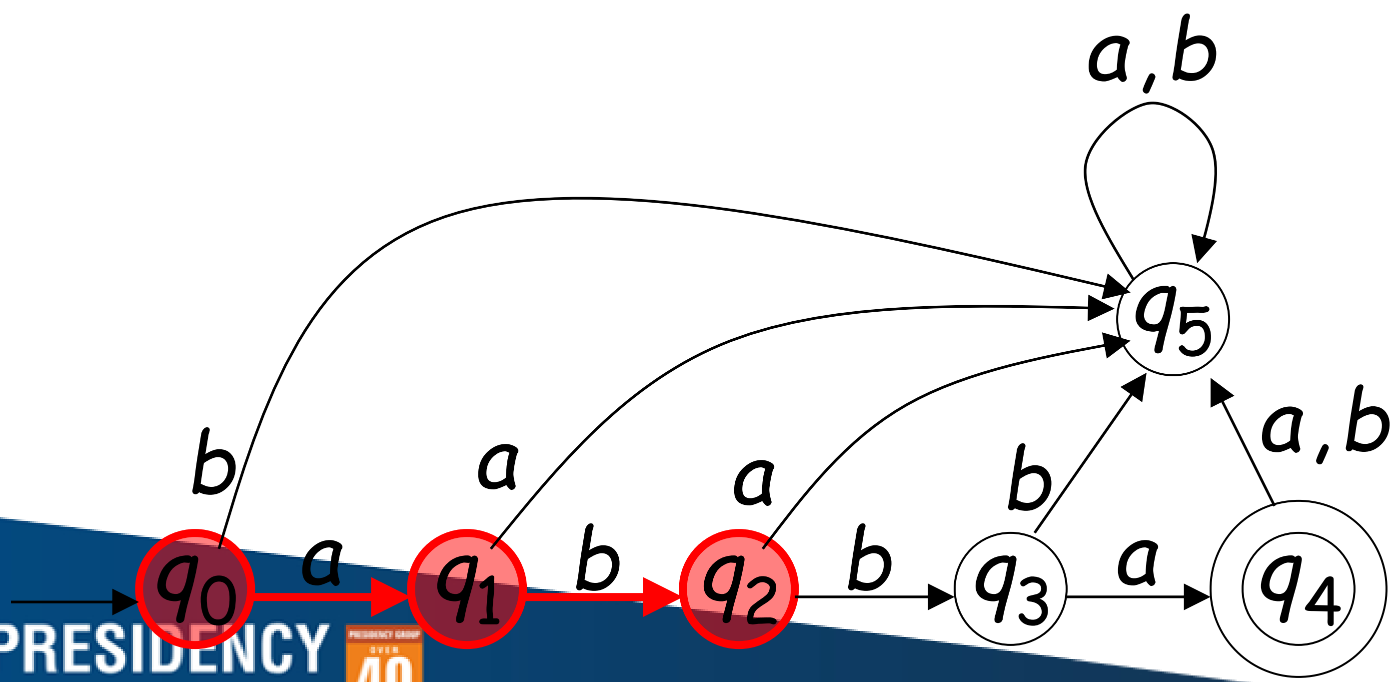


Extended Transition Function δ^*

- $$\delta^* : Q \times \Sigma^* \to Q$$

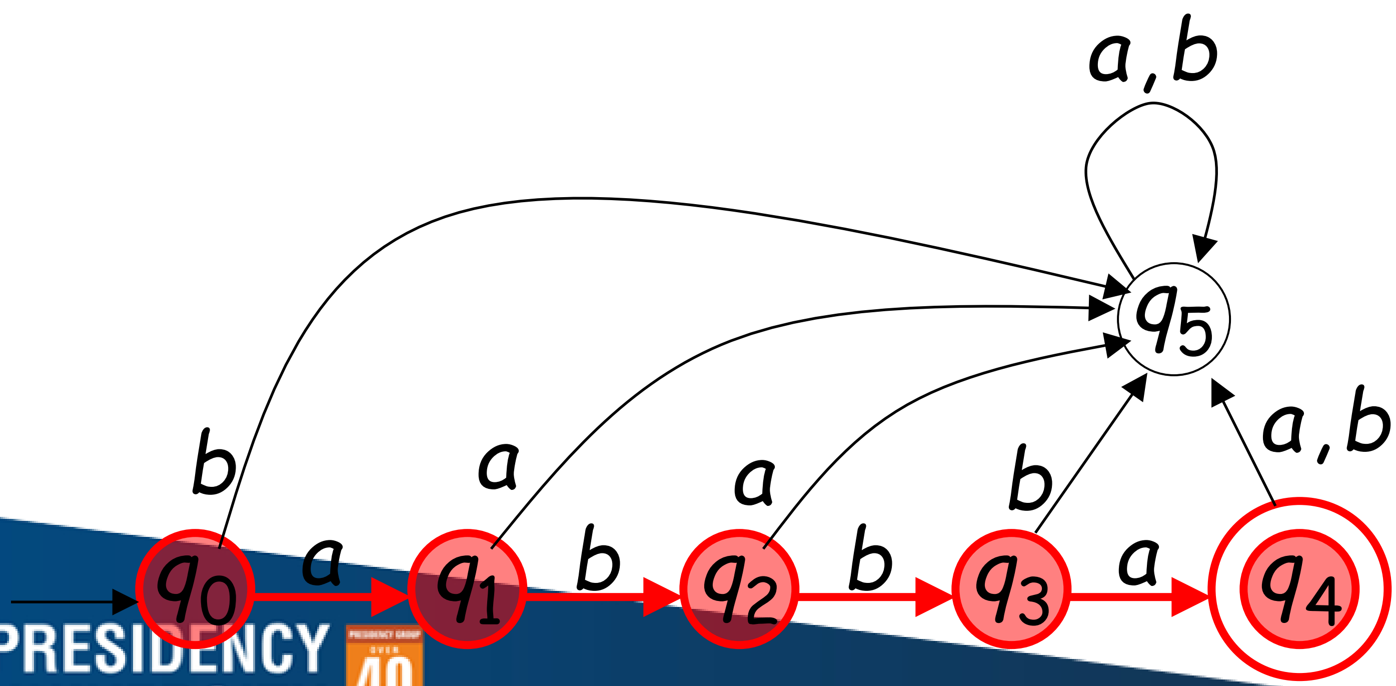


$$\delta^*(q_0, ab) = q_2$$

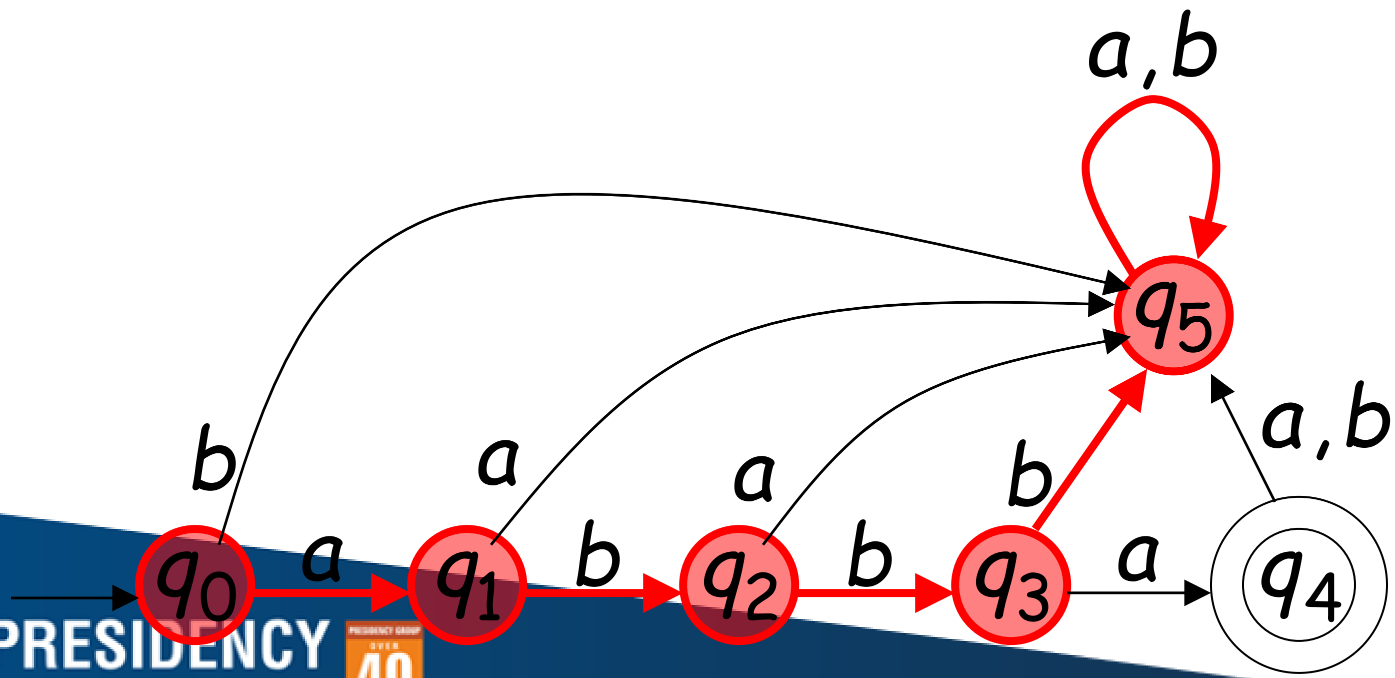


PRESIDENCY UNIVERSITY

$$\delta^*(q_0, abba) = q_4$$

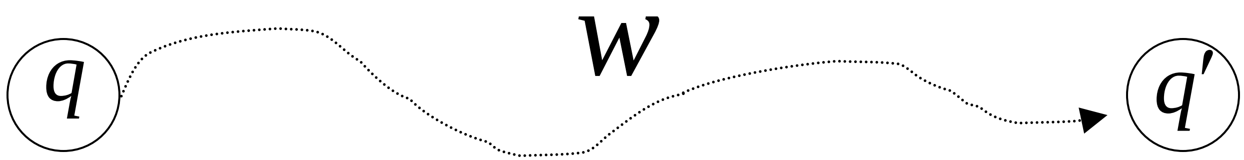


$$\delta^*(q_0, abbbaa) = q_5$$

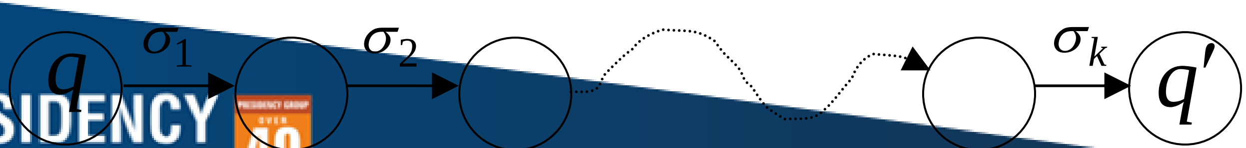


Observation: if there is a walk from q to q' with label w then

$$\delta^*(q, w) = q'$$

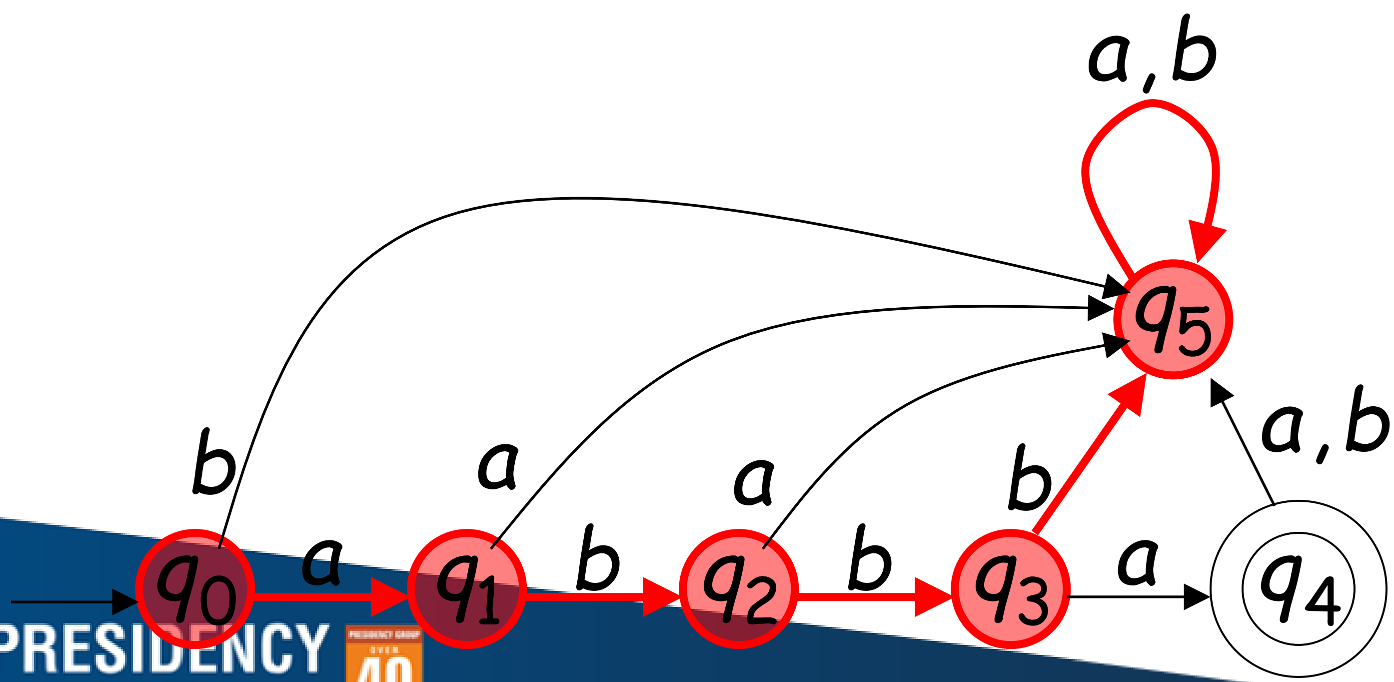


$$w = \sigma_1 \sigma_2 \cdots \sigma_k$$



Example: There is a walk from q_0 to q_5 with label $abbbaa$

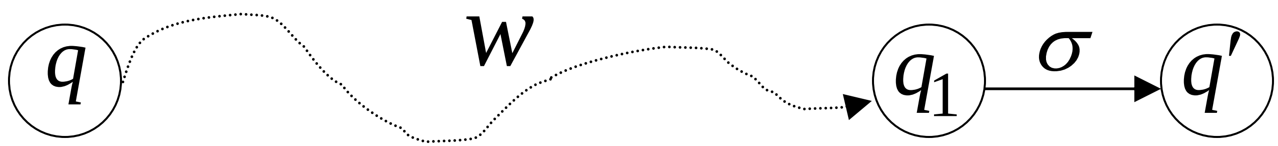
$$\delta^*(q_0, abbbaa) = q_5$$



Recursive Definition

$$\delta^*(q,\lambda)=q$$

$$\delta^*(q,w\sigma)=\delta(\delta^*(q,w),\sigma)$$



$$\delta^*(q,w\sigma)=q'$$

$$\delta(q_1,\sigma)=q'$$

$$\Rightarrow \delta^*(q,w\sigma)=\delta(q_1,\sigma)$$

$$\delta^*(q,w)=q_1$$

$$\Rightarrow \delta^*(q,w\sigma)=\delta(\delta^*(q,w),\sigma)$$

$$\delta^*(q_0, ab) =$$

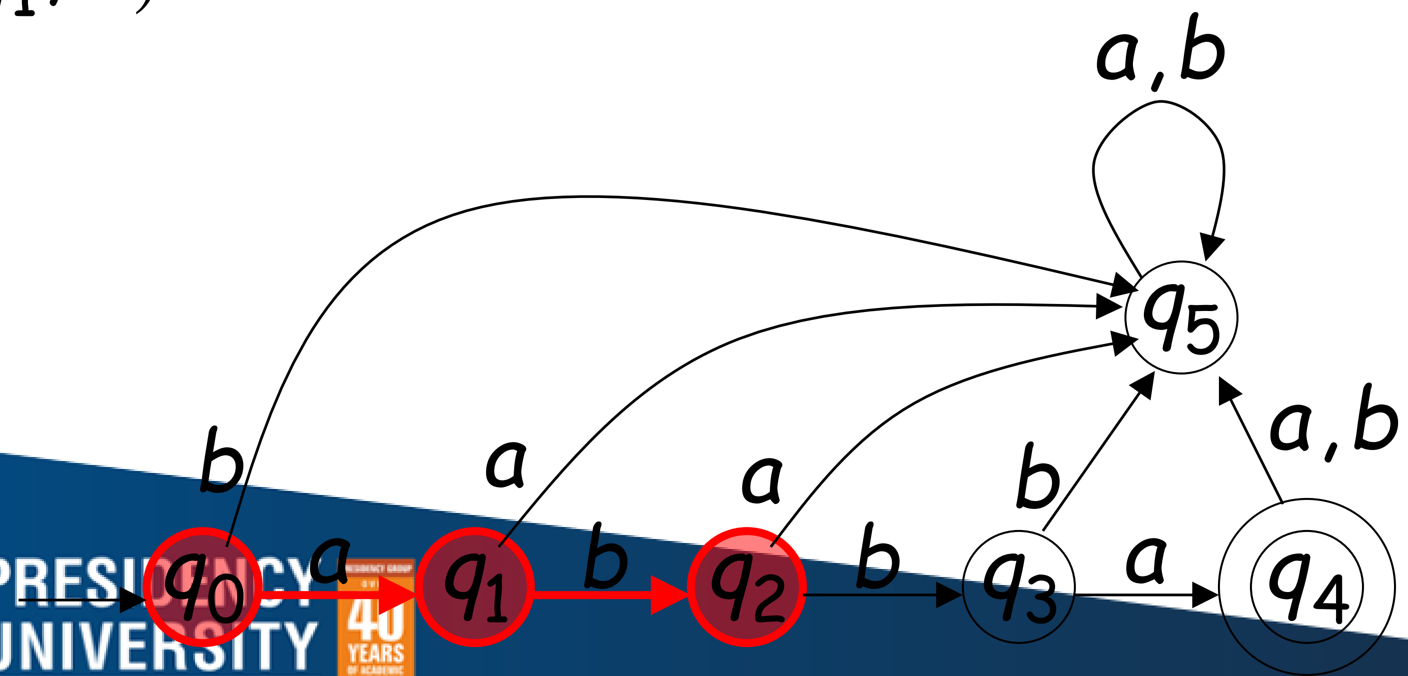
$$\delta(\delta^*(q_0, a), b) =$$

$$\delta(\delta(\delta^*(q_0, \lambda), a), b) =$$

$$\delta(\delta(q_0, a), b) =$$

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

$$q_2$$



Check the string acceptance of abba

- $\delta^*(qo,abba)= \delta(\delta^*(q0,abb)\ , a)$
- $\quad = \delta(\delta\ (\delta^*(q0,ab),b)\ , a)$
- $\quad = \delta(\delta\ (\delta\ (\delta^*(q0,a),b),b)\ , a)$
- $\quad = \delta(\delta\ (\delta\ (\delta\ (\delta^*(q0,\lambda),a),b),b)\ , a)$
- $\quad = \delta(\delta\ (\delta\ (\delta\ (q0,a),b),b)\ , a)$
- $\quad =\delta(\delta\ (\delta\ (q1,b),b)\ , a)$
- $\quad =\delta(\delta\ (q2,b)\ , a)$
- $\quad =\delta(q3, a)$
- $\quad =q4\ \in\ F$
- String abba is accepted as q4 is a final state

Language Accepted by FAs

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

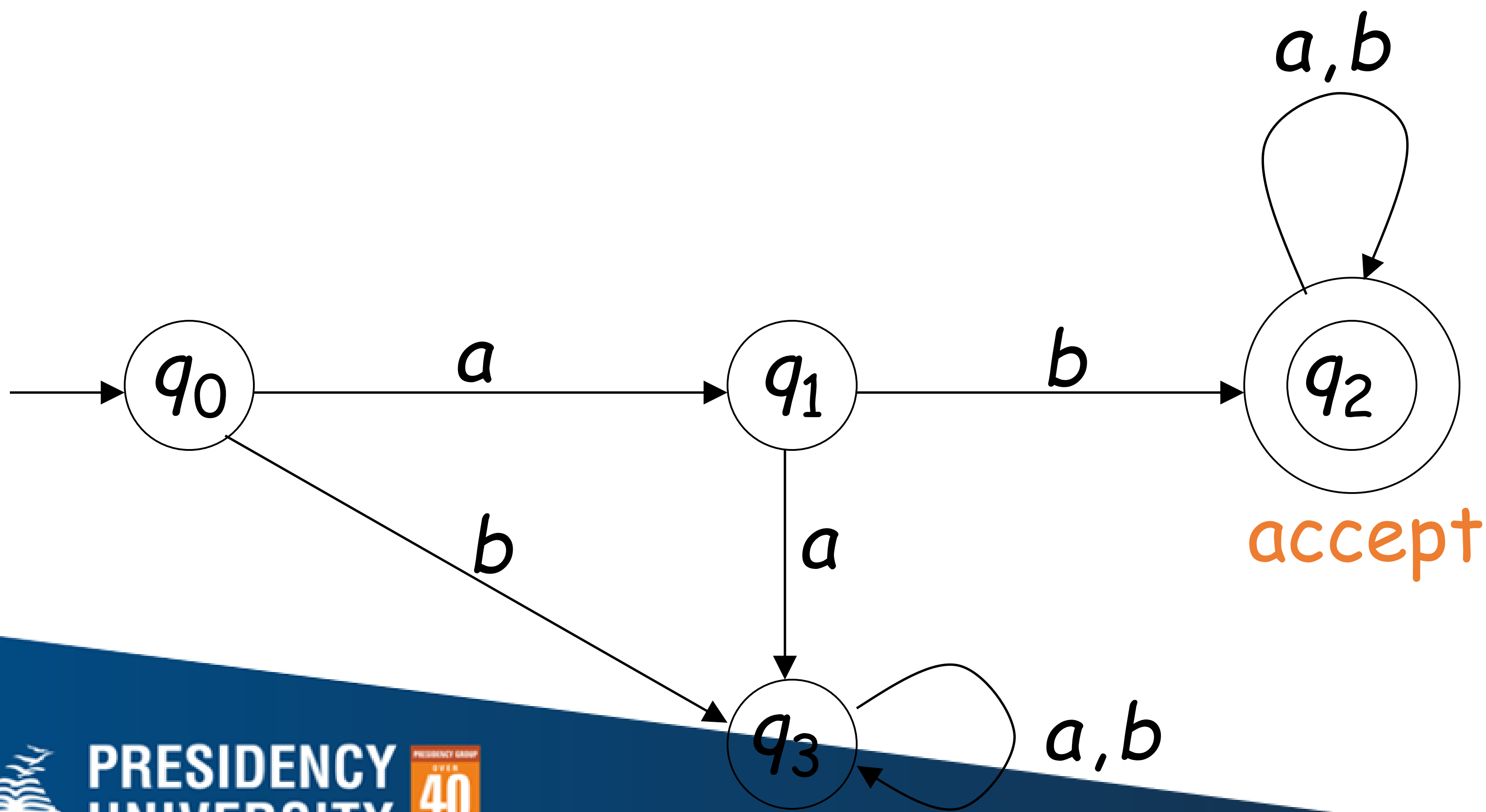
- Language accepted by : M

- $$L(M) = \{ w \in \Sigma^* : \delta^*(q_0, w) \in F \}$$

$q' \in F$

Example

- $L(M) =$ { all strings with prefix ab }

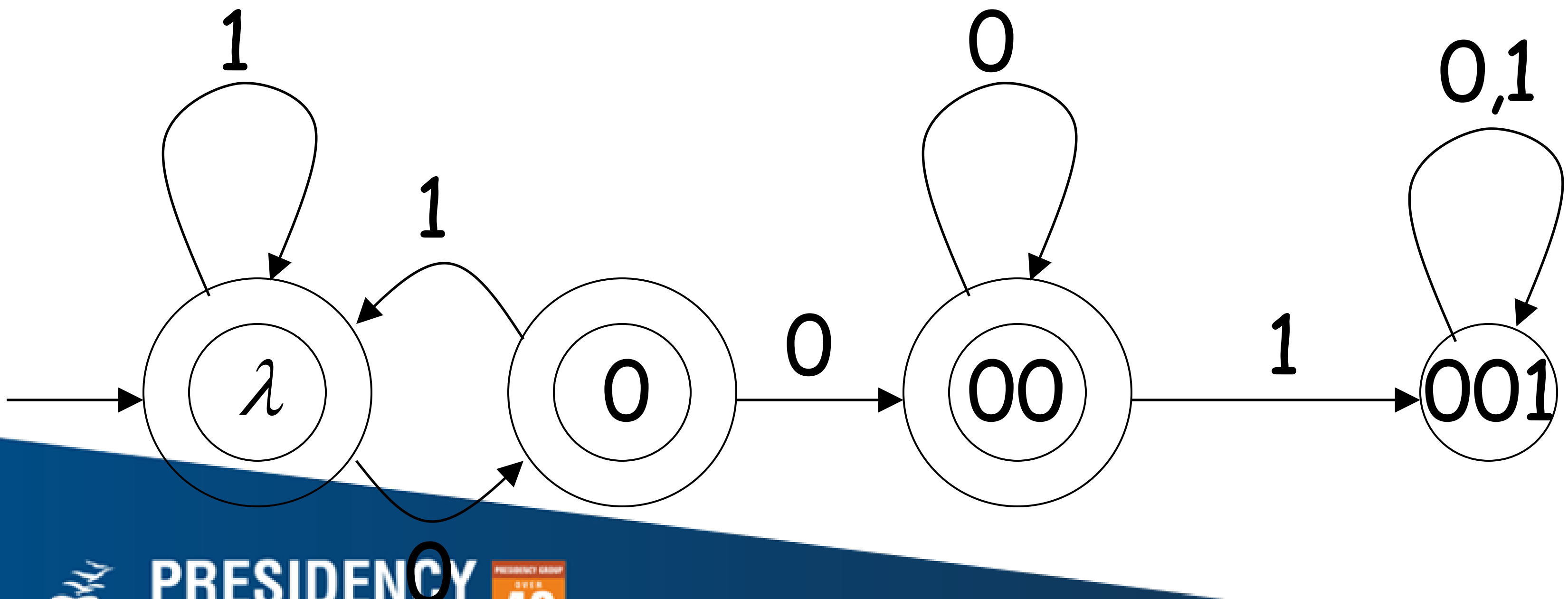


PRESIDENCY UNIVERSITY
GAIN MORE KNOWLEDGE REACH GREATER HEIGHTS
Private University Estd. in Karnataka State by Act No. 41 of 2013
PRESIDENCY GROUP OVER 40 YEARS OF ACADEMIC WISDOM

L=strings with substring ‘101’ over {0, 1}

Example

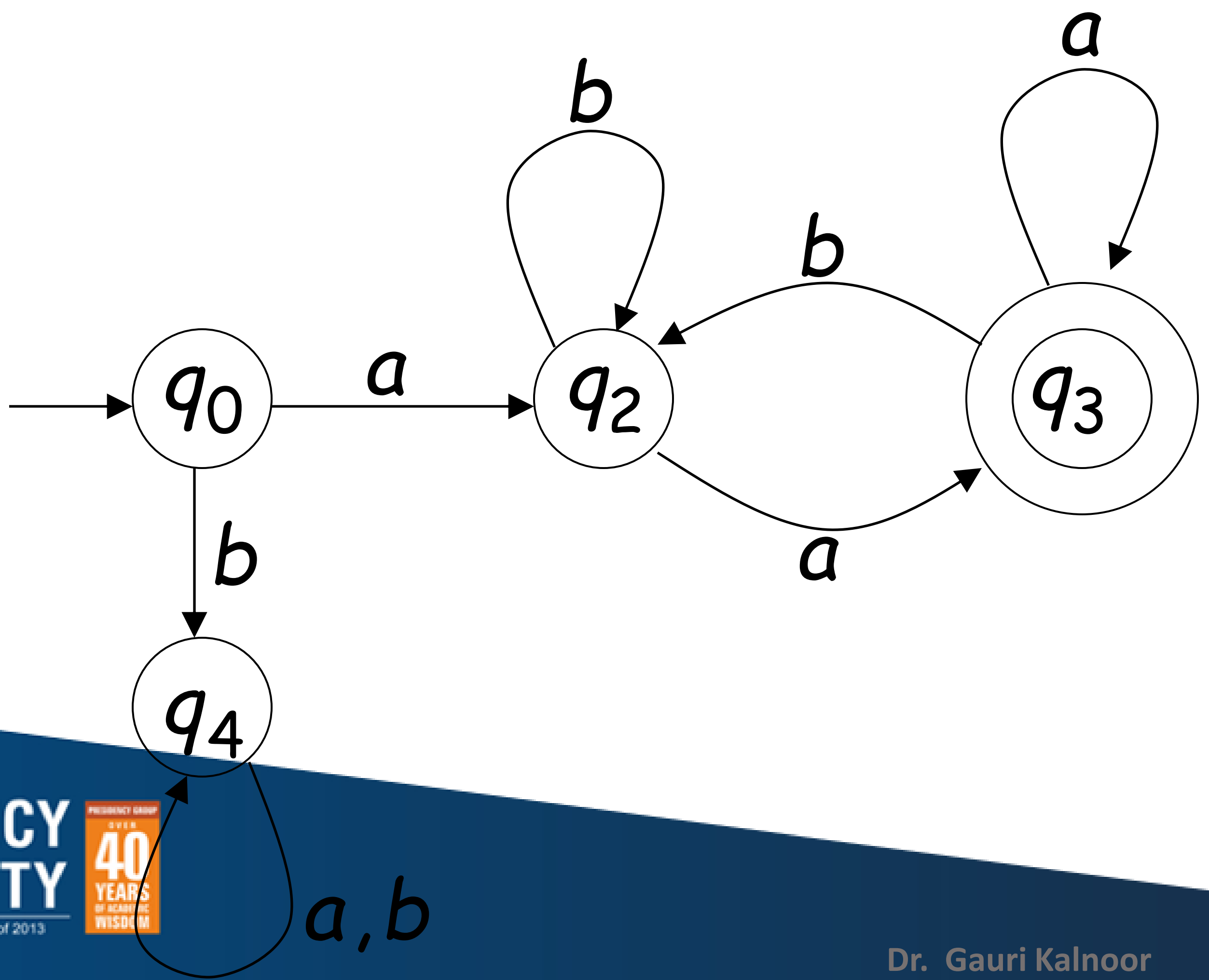
- $L(M) = \{$ all strings without substring 001 $\}$



PRESIDENCY UNIVERSITY
Private University Estd. in Karnataka State by Act No. 41 of 2013

Example

$$L(M) = \{awa : w \in \{a,b\}^*\}$$

- 

Regular Languages

- Definition:
- A language L is regular if there is
- FA M such that $L = L(M)$

- Observation:
- All languages accepted by FAs
- form the family of regular languages
  -

Examples of regular languages:

$\{abba\}$ $\{\lambda, ab, abba\}$

$\{awa : w \in \{a,b\}^*\}$ $\{a^n b : n \geq 0\}$

{ all strings with prefix ab }

{ all strings without substring 001 }

There exist automata that accept these Languages

There exist languages which are not Regular:

Example: $L = \{a^n b^n : n \geq 0\}$

There is no FA that accepts such a language

(we will prove this later in the class)

Deterministic Finite Automata

- Every State should have transition over every input symbol
- There should be only One next state for each transition
- DFA is defined as

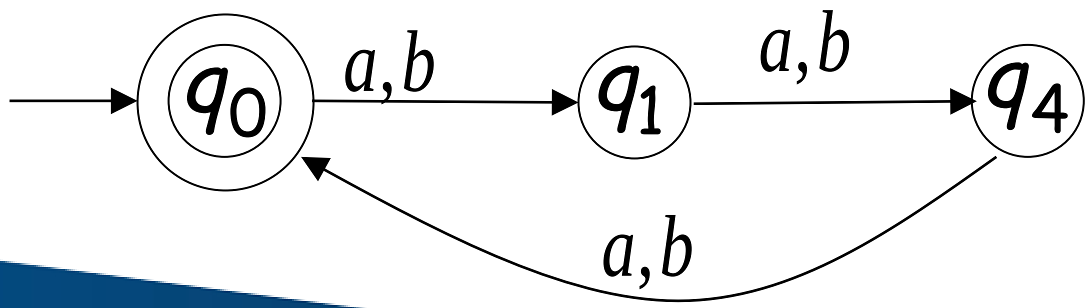
$$M = (Q, \Sigma, \delta, q_0, F)$$

DFA Example

Design DFA that accepts language L= { w : |w| mod 3 = 0 } over ∑ = {a, b}

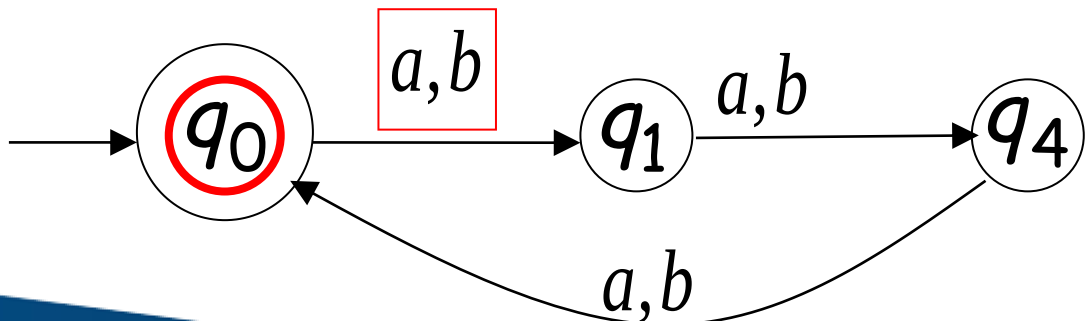
Solution

- **Strings accepted= {ϵ, aaa, bbb, aba, aab, bab, aaabbb, ababab, ...}**
- **Strings rejected= {a, b, ab, ba, abab, baba, bbaa, aaabb, …}**
- **Transition Diagram**



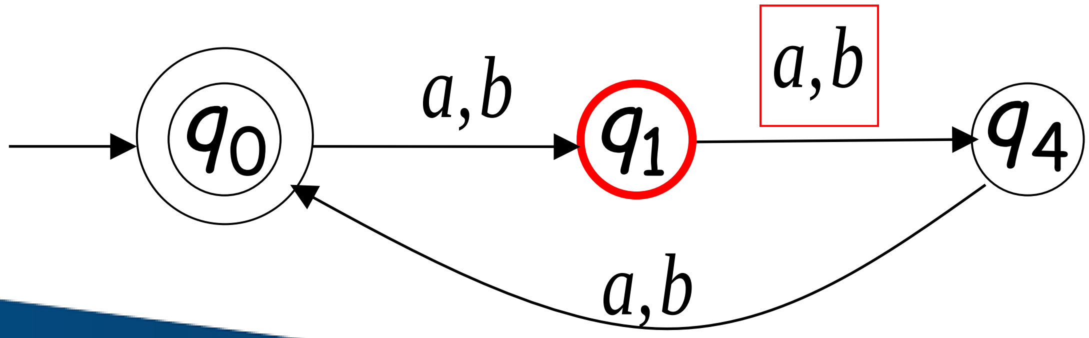
DFA Example

- Consider Sample String : **a a b**



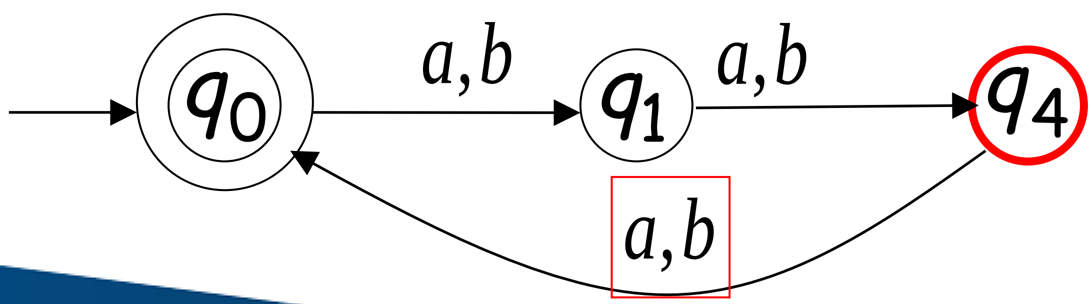
DFA Example

- Consider Sample String : **a a b**



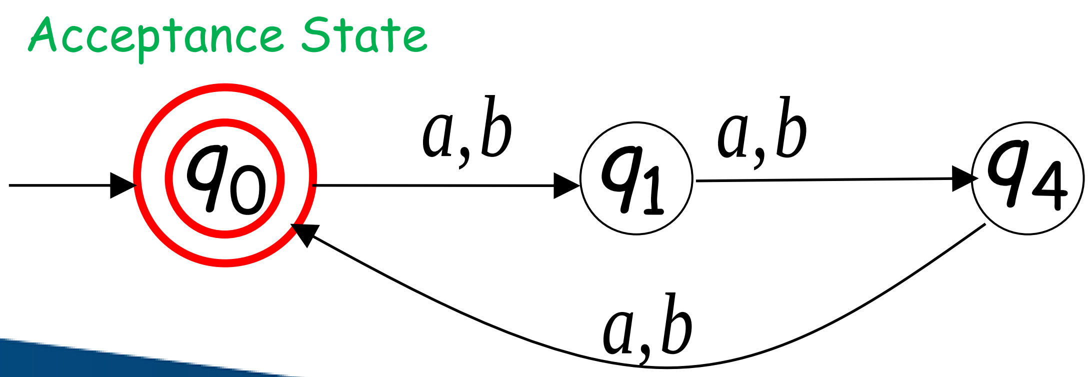
DFA Example

- Consider Sample String : **a a b**



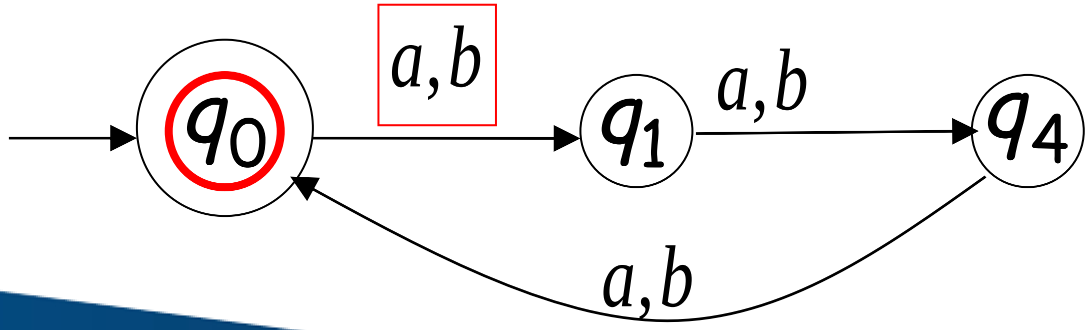
DFA Example

- ➢Consider Sample String : **a a b**
- ➢ String is accepted



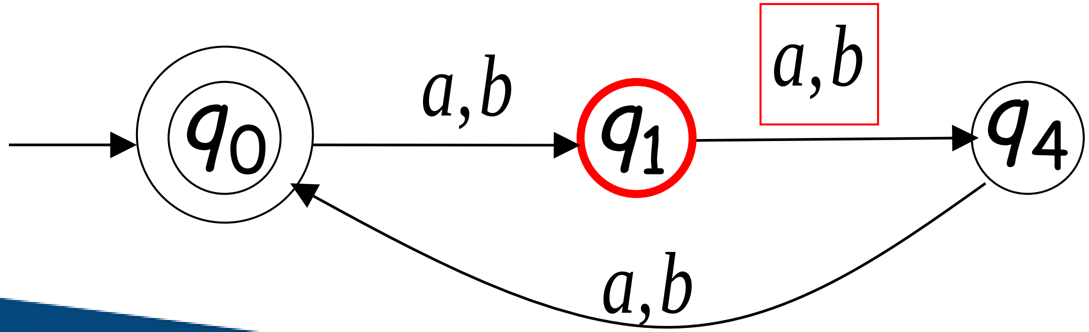
DFA Example

- Consider Another Sample String : **b a b a**



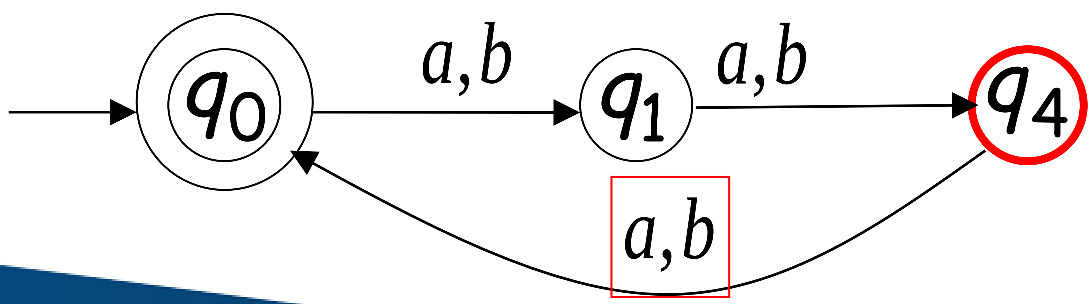
DFA Example

- Consider Another Sample String : **b a b a**



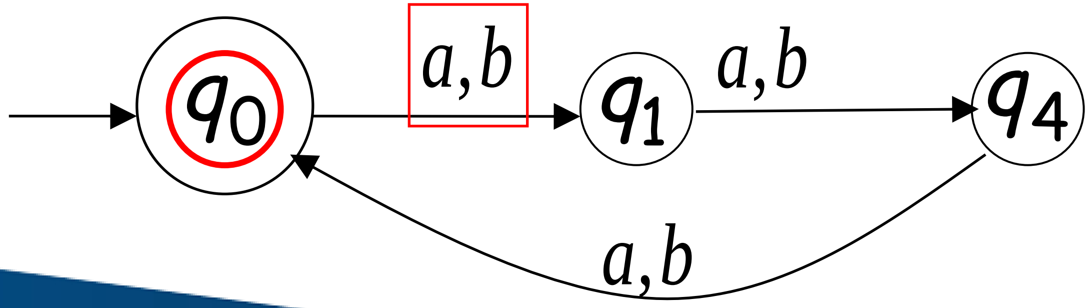
DFA Example

- **Consider Another Sample String :** **b a b a**



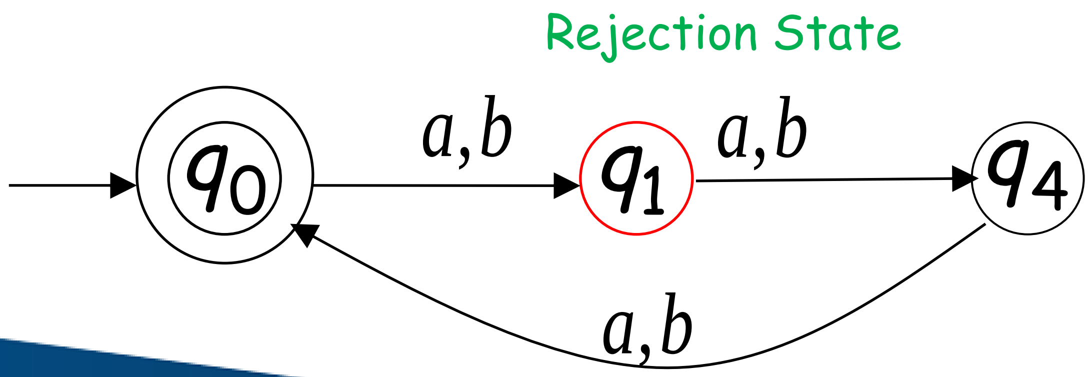
DFA Example

- **Consider Another Sample String :** **b a b a**



DFA Example

- ➢Consider Another Sample String : **b a b a**
- ➢ String is rejected



DFA Example

- DFA Tuples M=(Q, Σ, δ, q0, F)
- Q= {q0, q1, q4}
- Σ= {a, b}
- δ = Transition function represented by Transition table
- q0= Initial State
- F= {q0} → Acceptance State

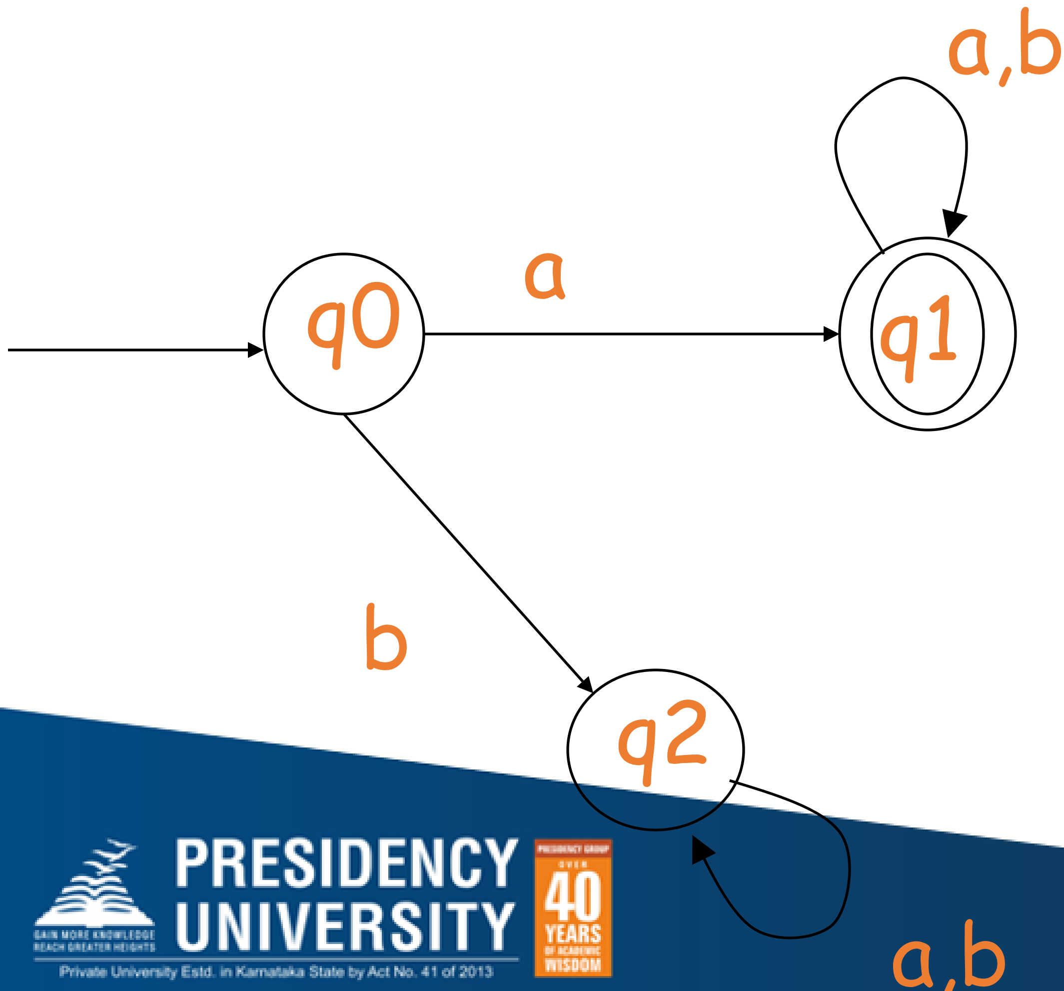
DFA Example

➢Transition Table

Q \ Σ	a	b
→*q0	q1	q1
q1	q4	q4
q4	q0*	q0*

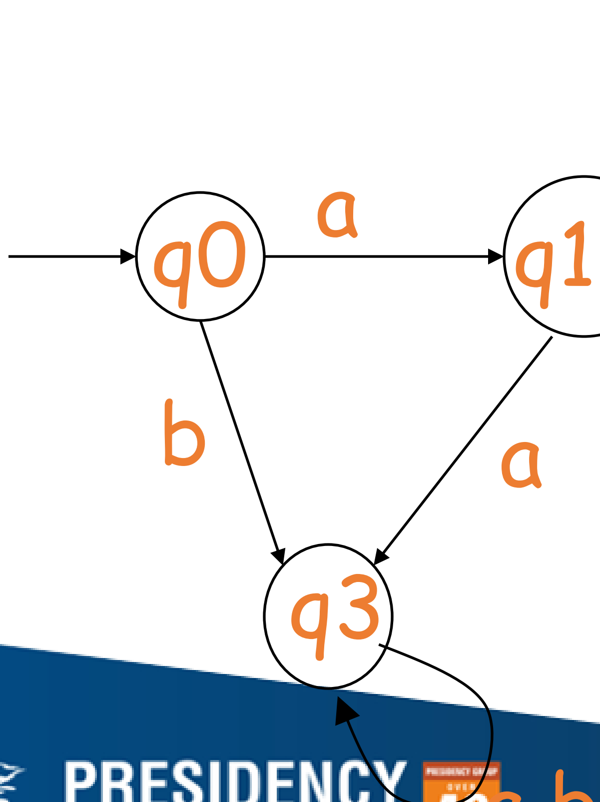
DFA - Starts with a , $\sum$={a,b}

- L= { a,aa,ab,aaa,aab,aba,….}



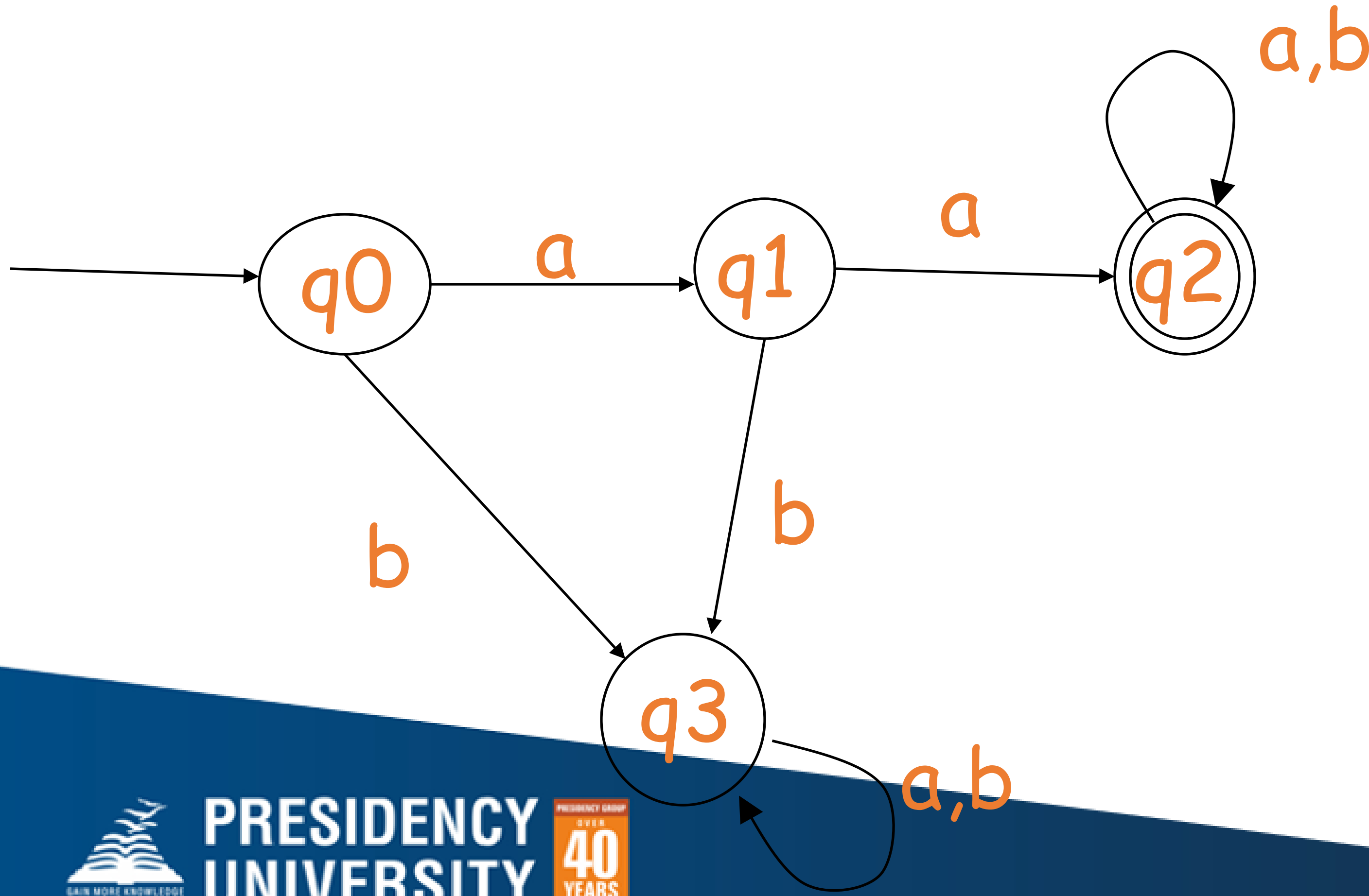
DFA - Starts with ab , ∑={a,b}

- L= { ab,aba,abb,….}



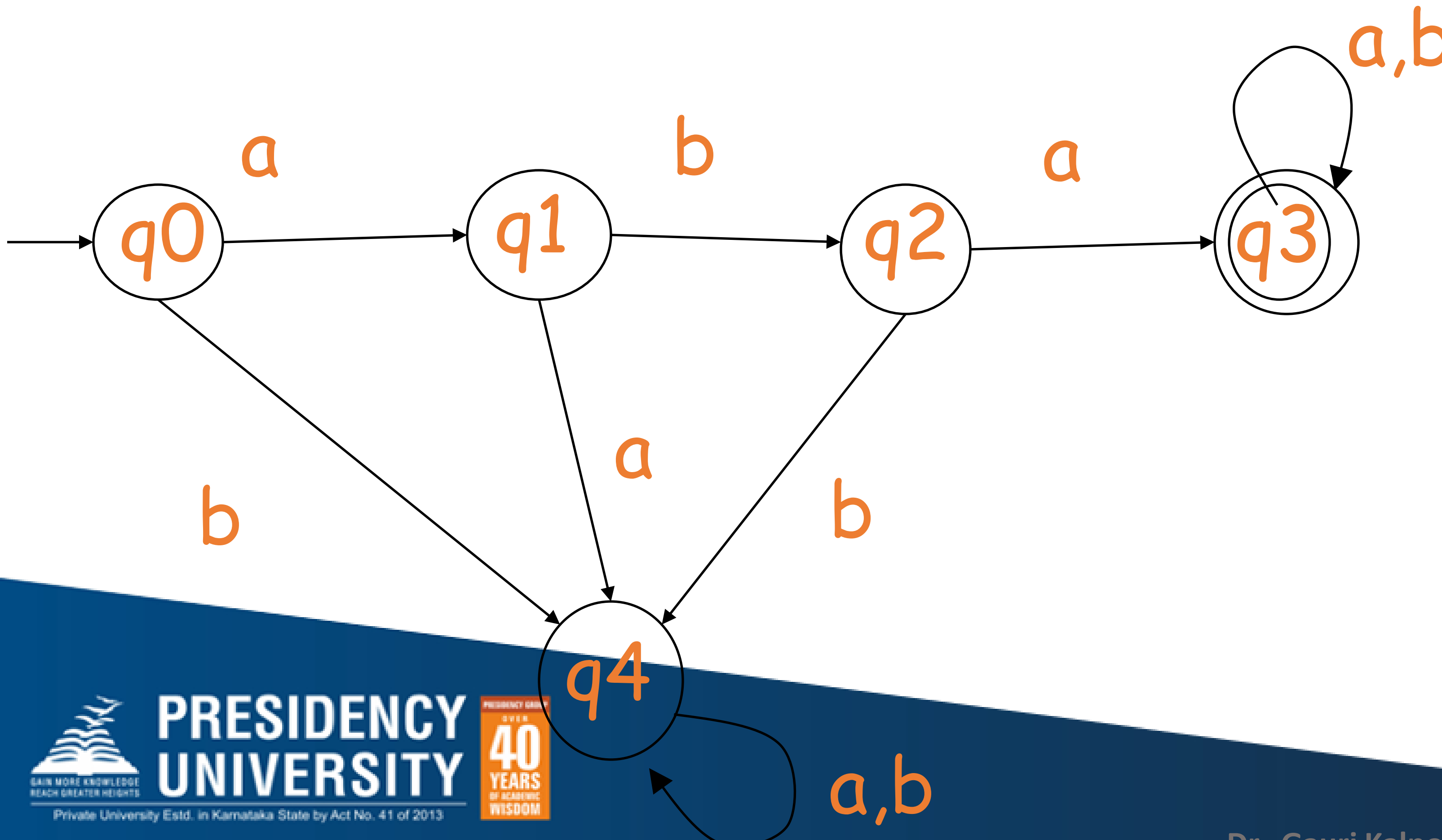
DFA - Starts with aa , ∑={a,b}

- L = { aa,aab,aaba,….}



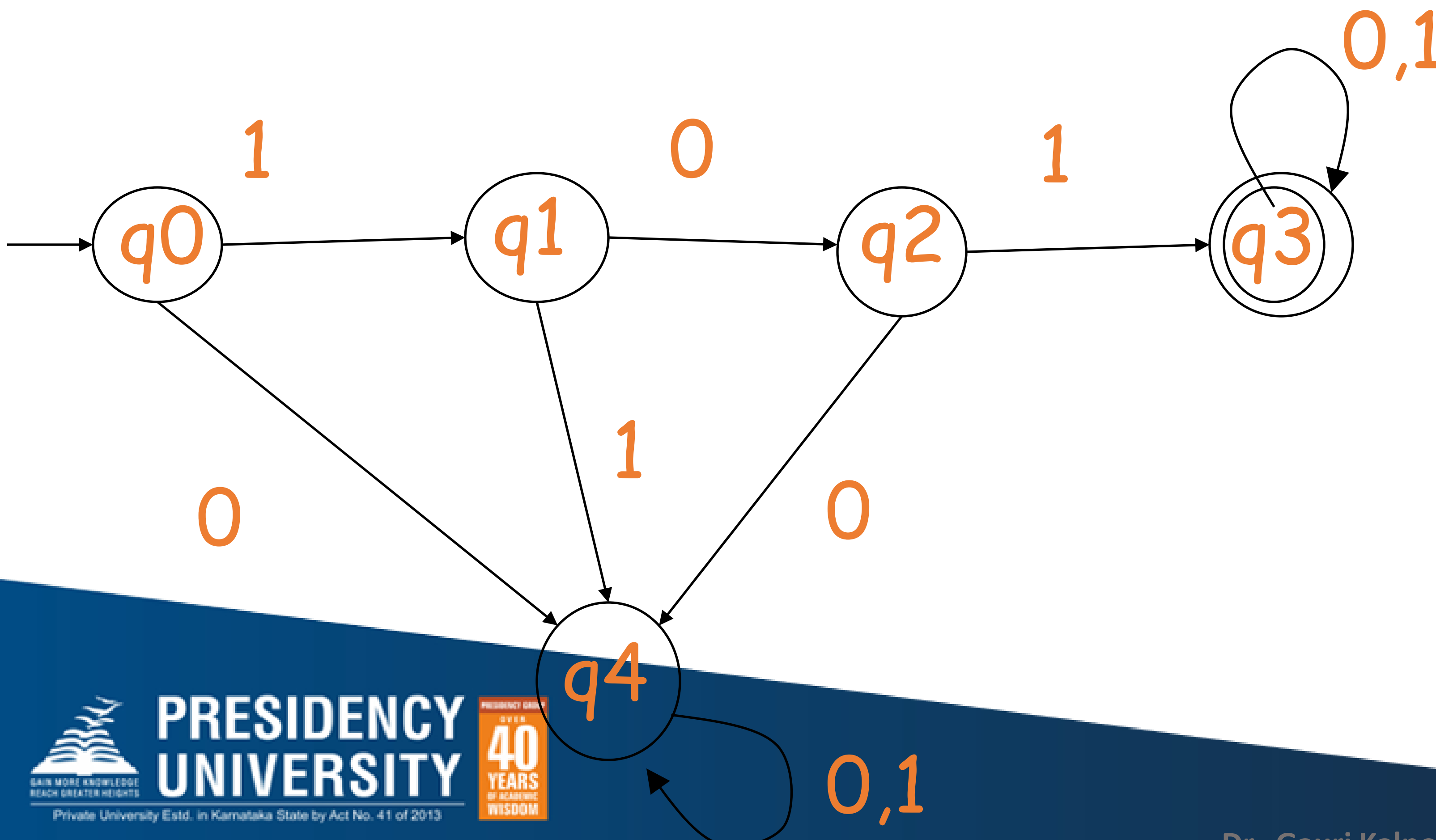
DFA - Starts with aba , ∑={a,b}

- L={ aba,abaa,abab,abaaa,...}



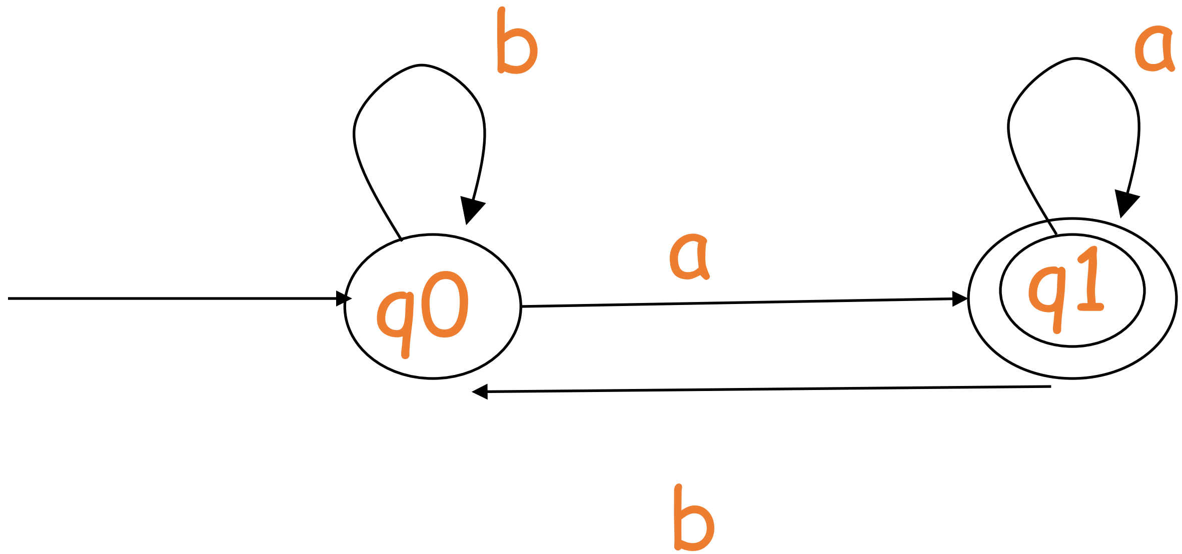
DFA - Starts with 101 , ∑={0,1}

- L={ 101,1010,1011,101101,…}



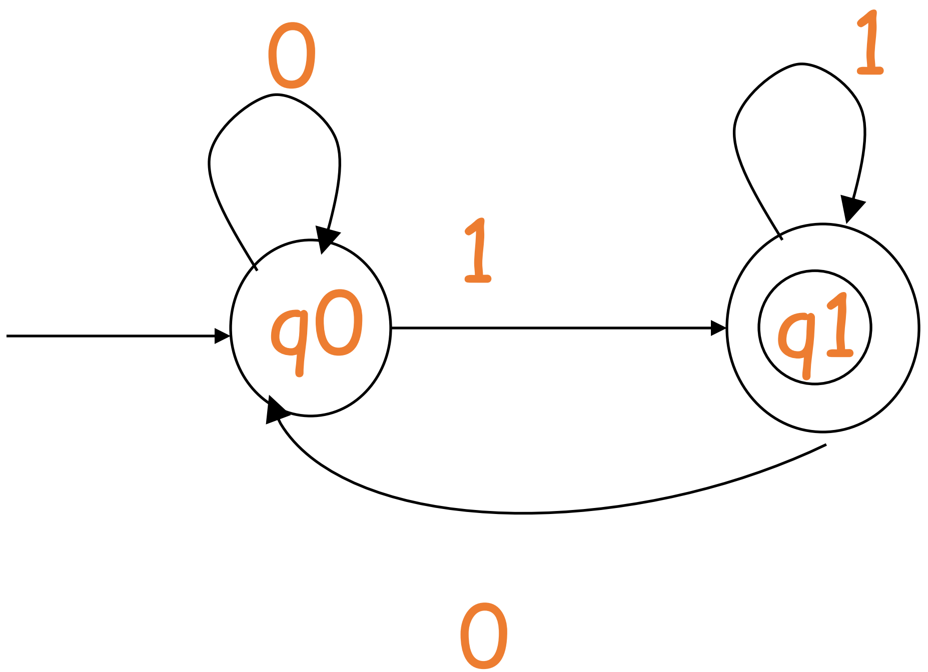
Ends with a

- L={a,aa,ba,aaa,aba,…}



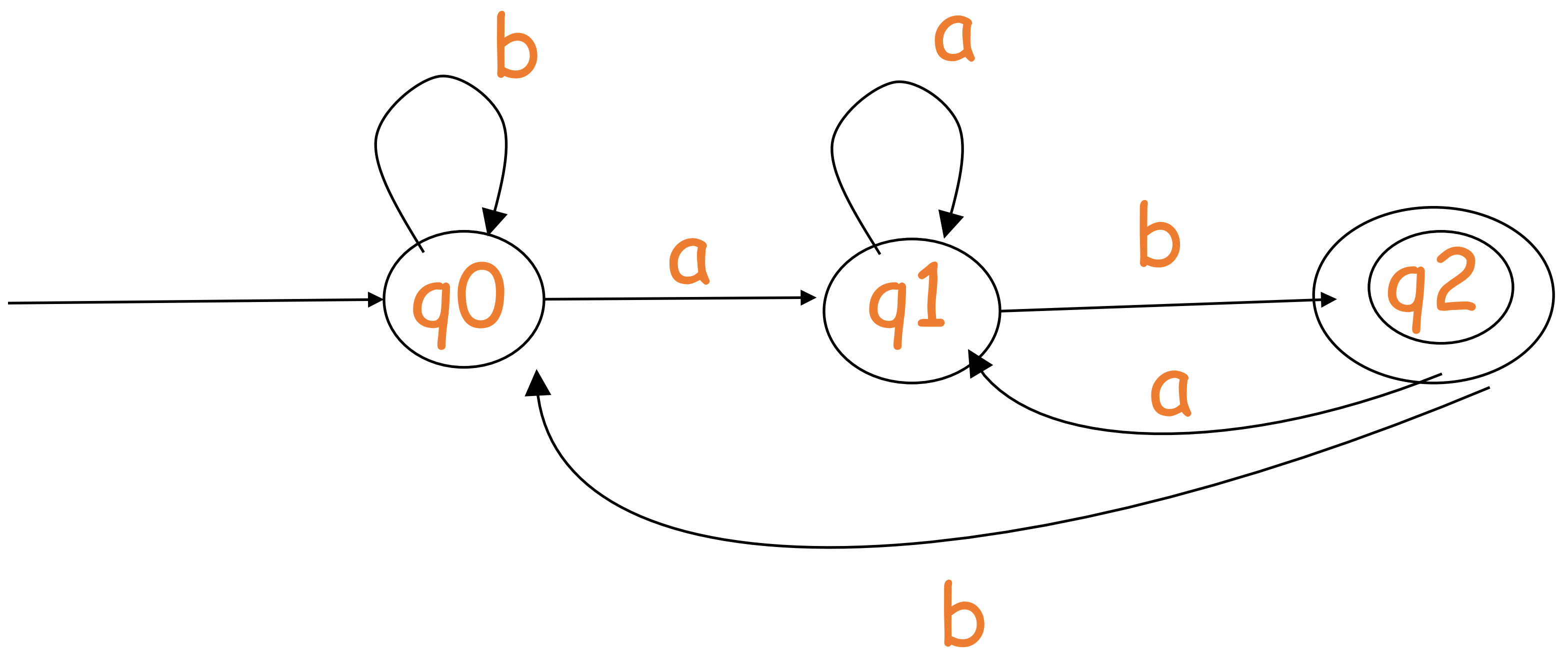
Ends with 1 , Σ={ 0,1)

- L={ 1,01,11,...}

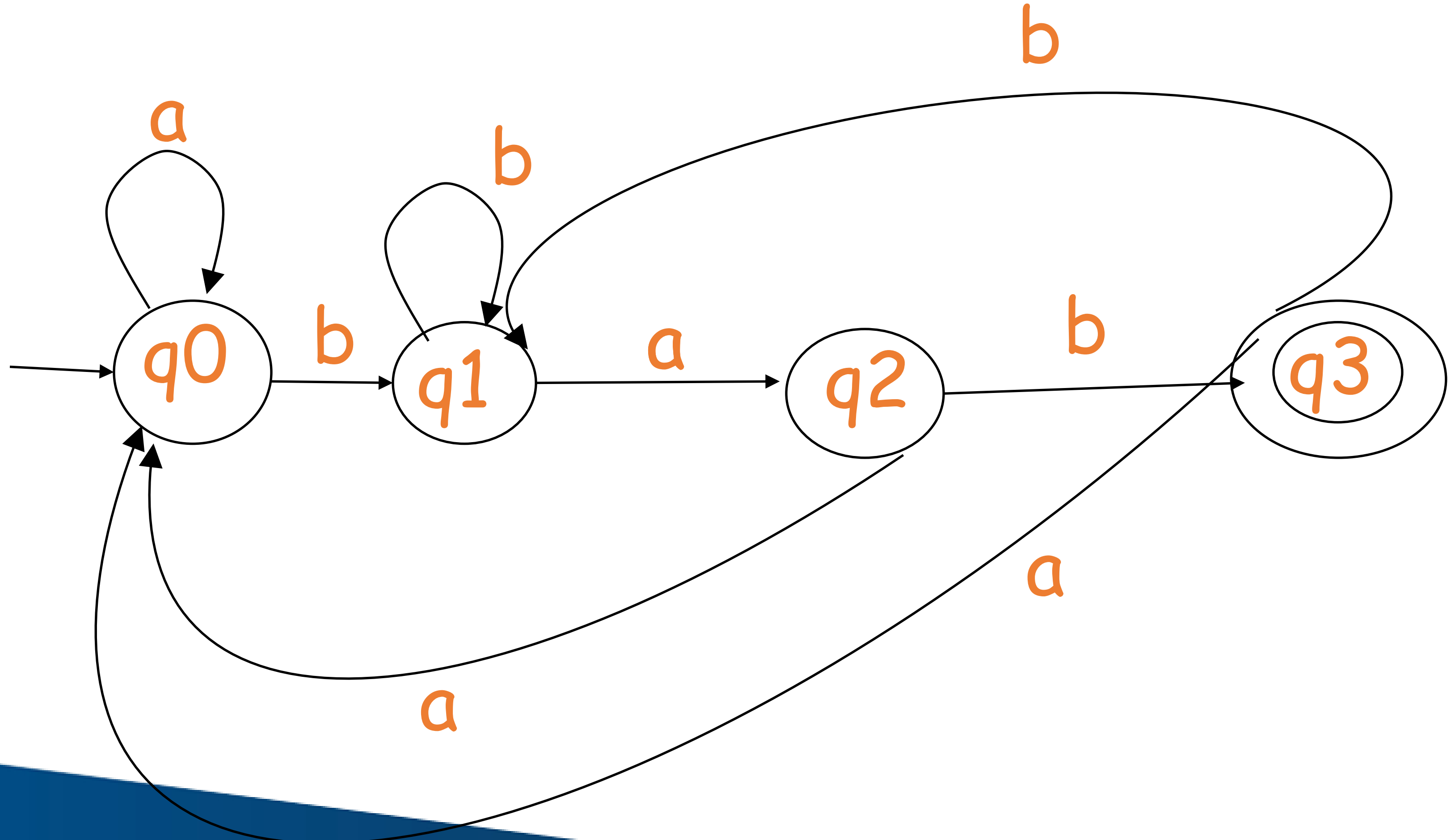


Ends with ab

- L={ab,aab,bab,aaab,bbbbab,abab,...}

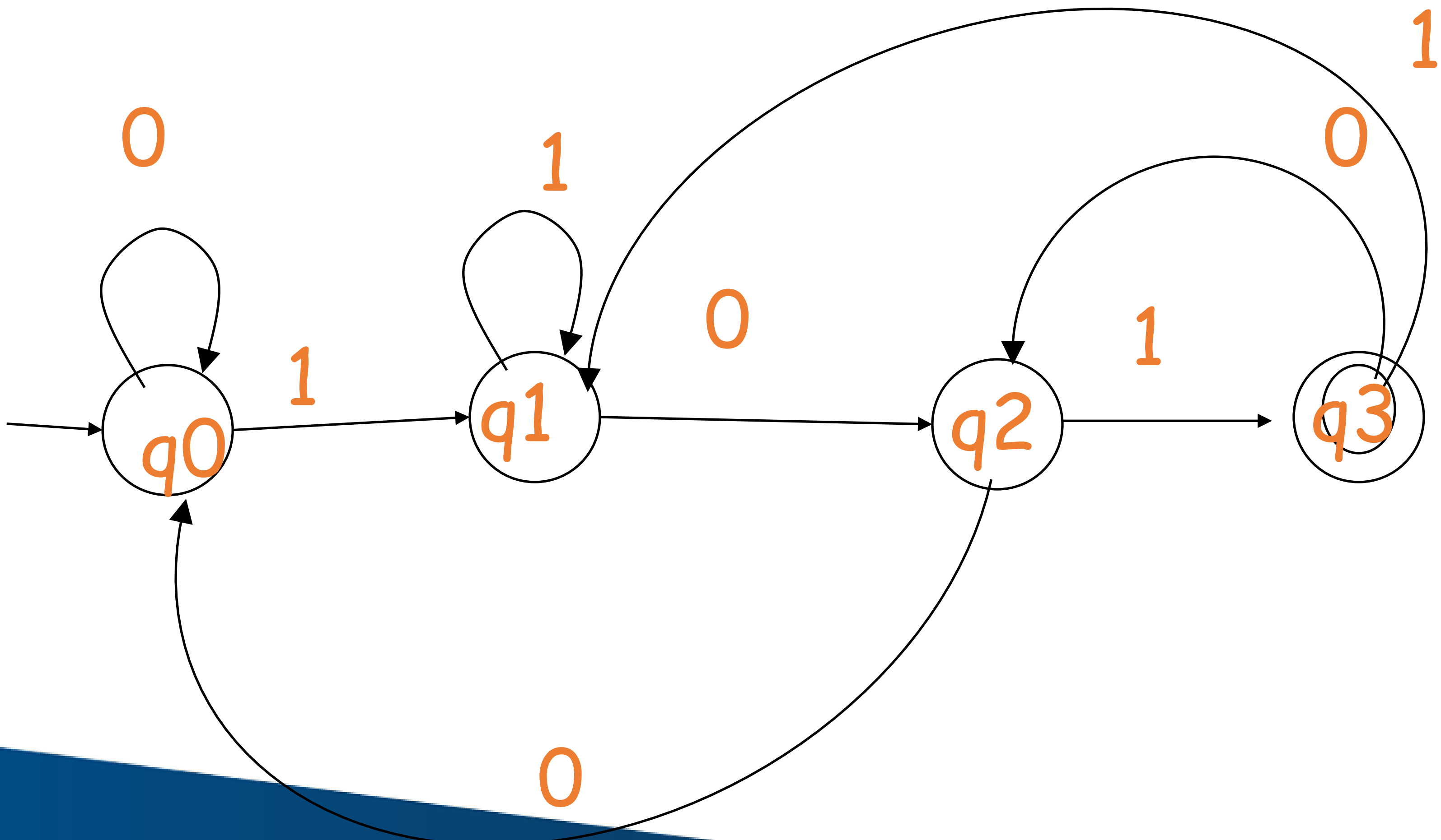


Ending with bab



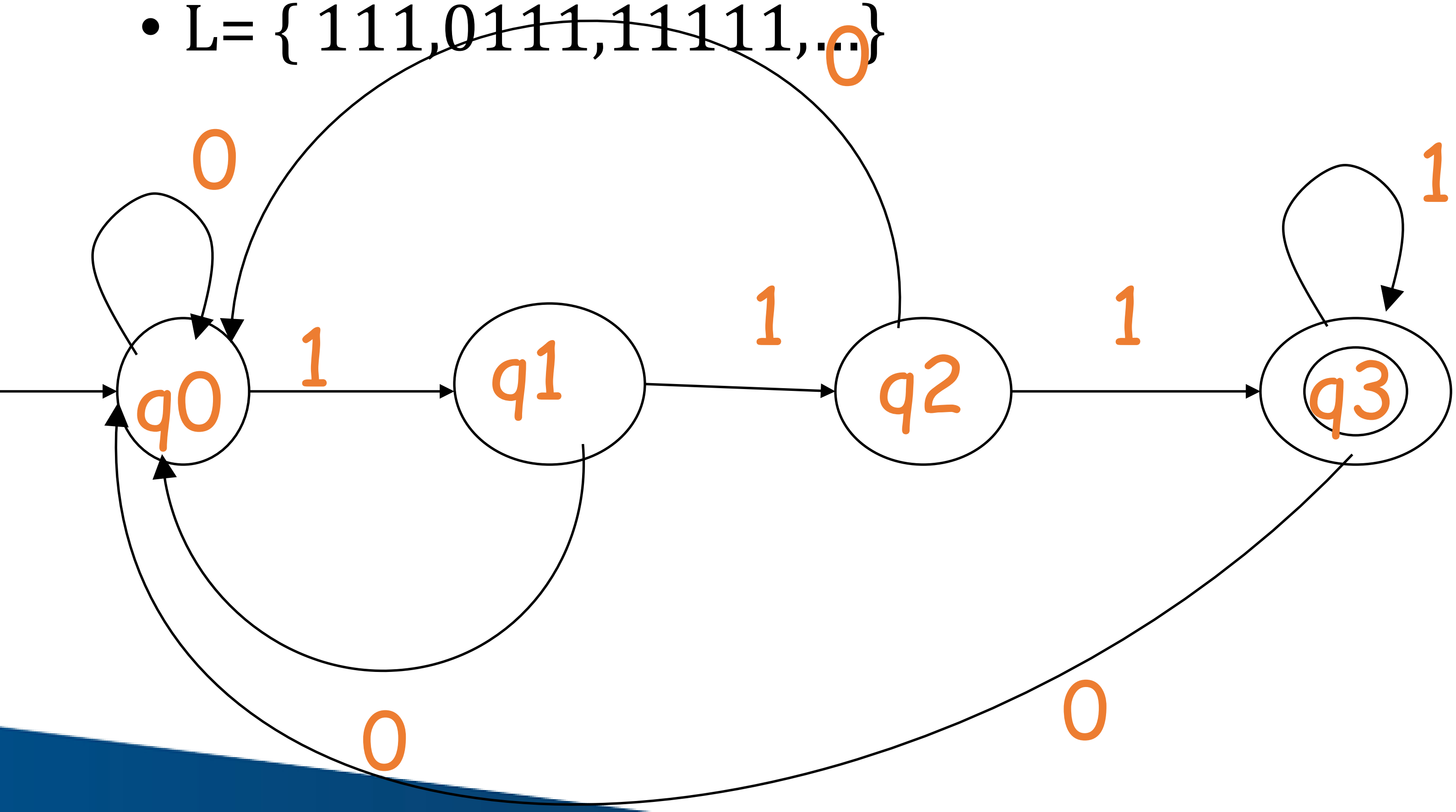
DFA – Ends with 101, ∑={ 0,1}

- L={ 101,0101,1101,000101,111101,…}



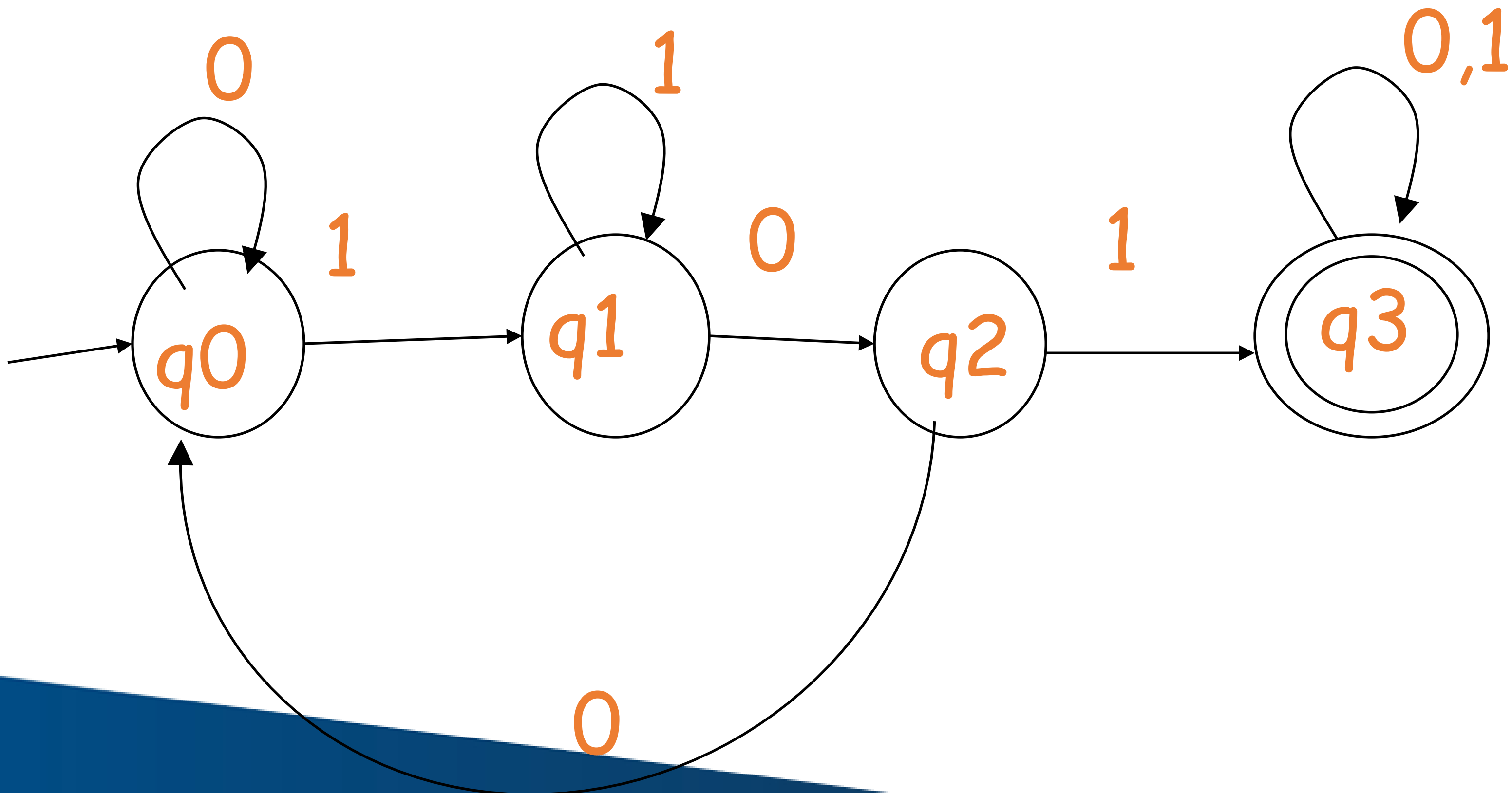
DFA – Ends with 111 , ∑={ 0,1}

- L= { 111,0111,11111,…}



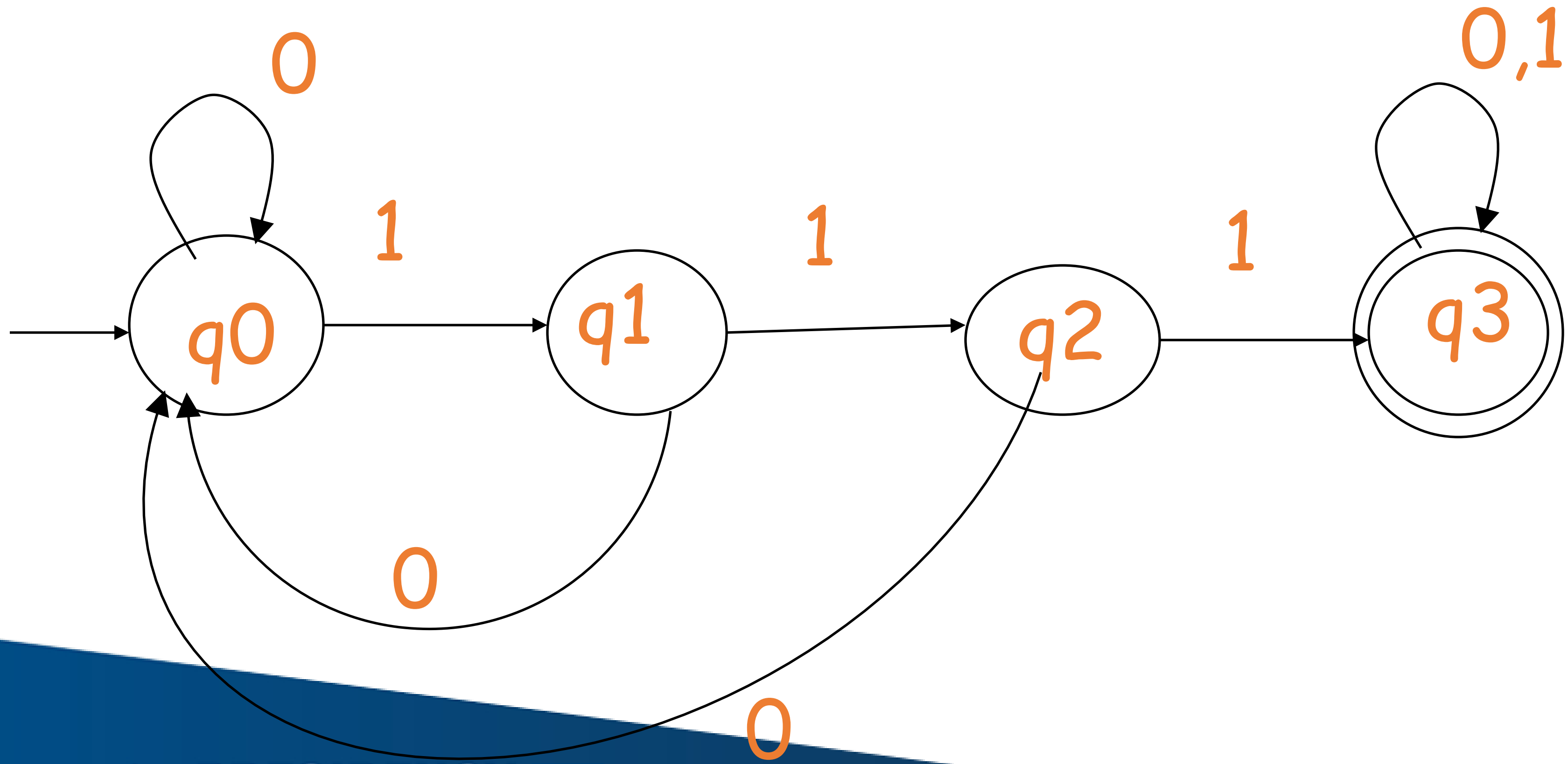
DfA – Substring 101

- L={ 101,0101,1101,1010,1011,00010111,...}



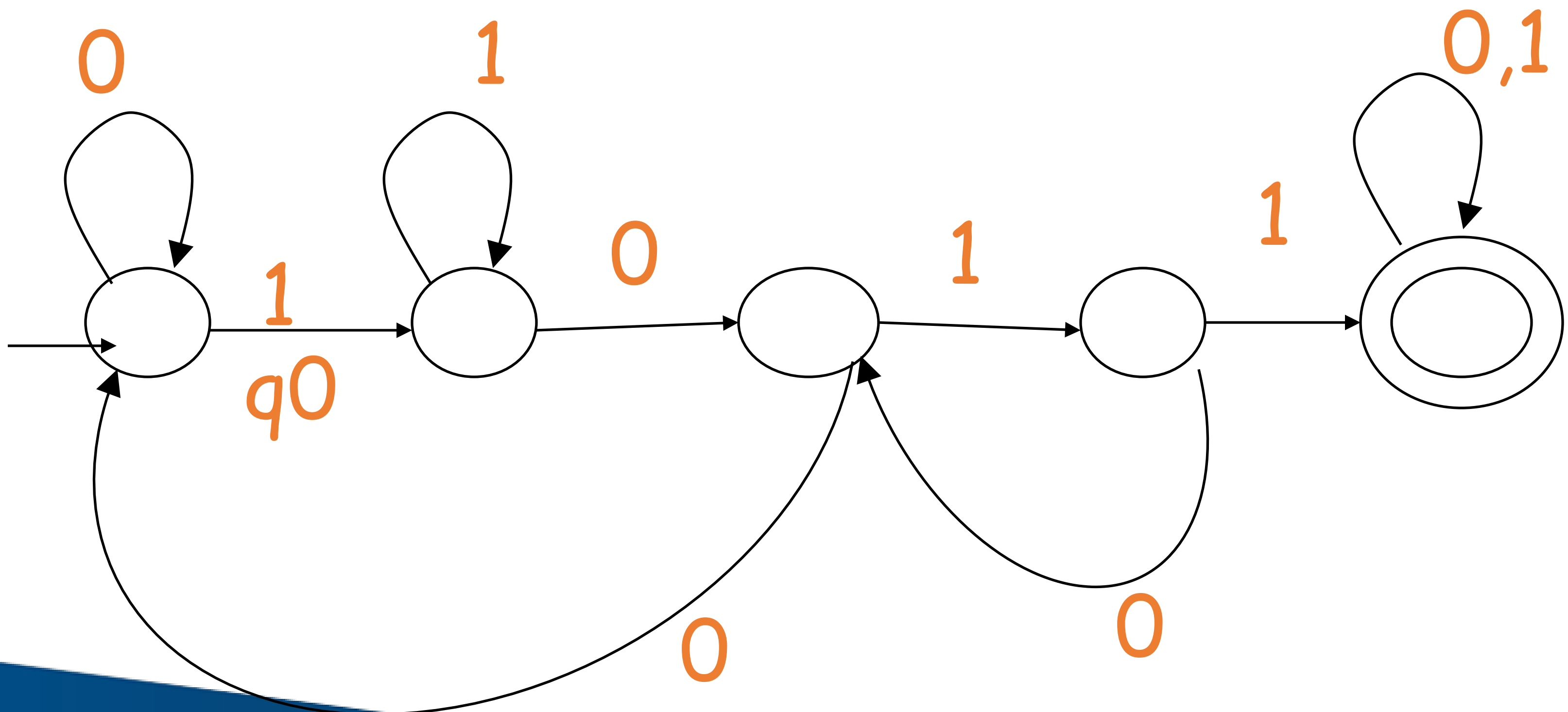
DFA- Contains 111

- L={ 111,0111,1111,000111000,11111000,..}



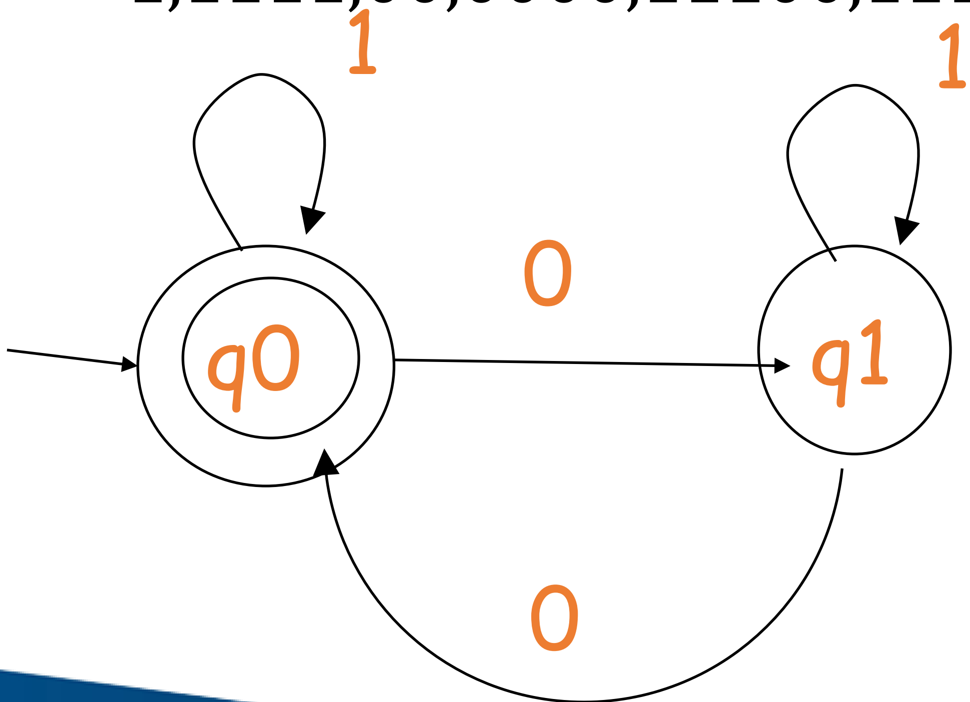
DFA – Contains 1011

- L= { 1011,01011,11011,00010111,..}



DFA- Even no of 0s, ∑={ 0,1} or n0(w)mod2=0

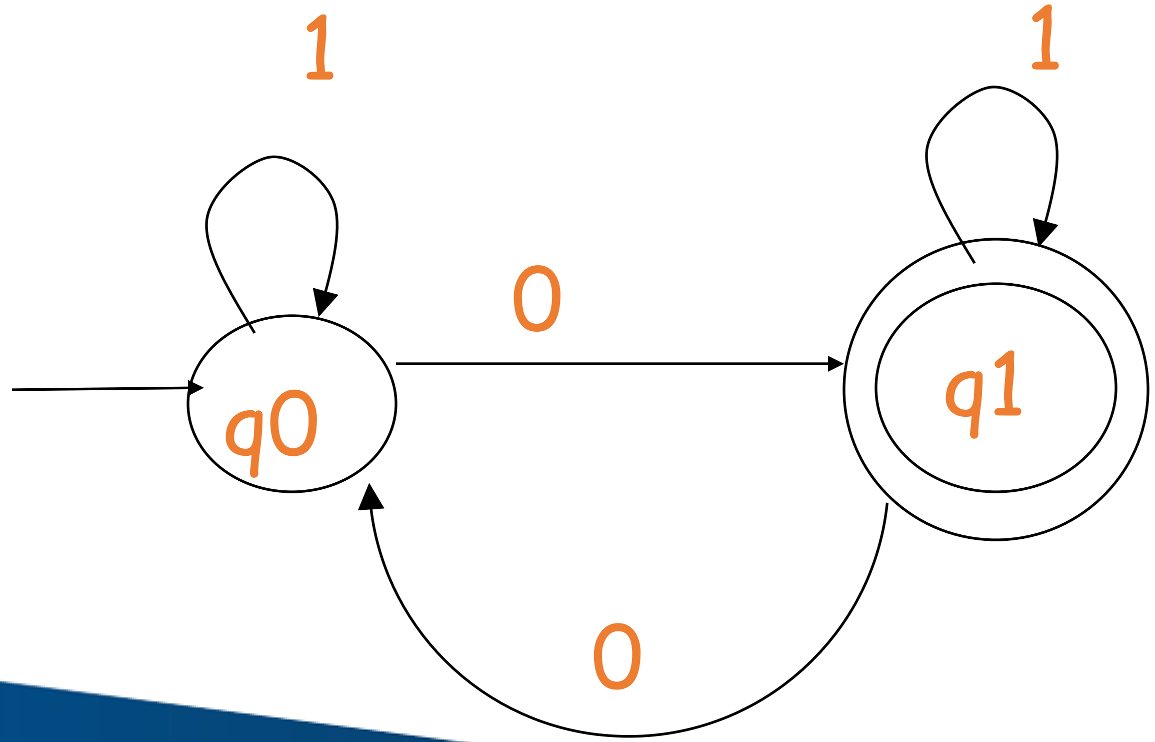
- L={ λ, 1,1111,00,0000,11100,1110000,0000111,..}



Transitio n	o	1
*->q0	q1	q0
q1	qo	q1

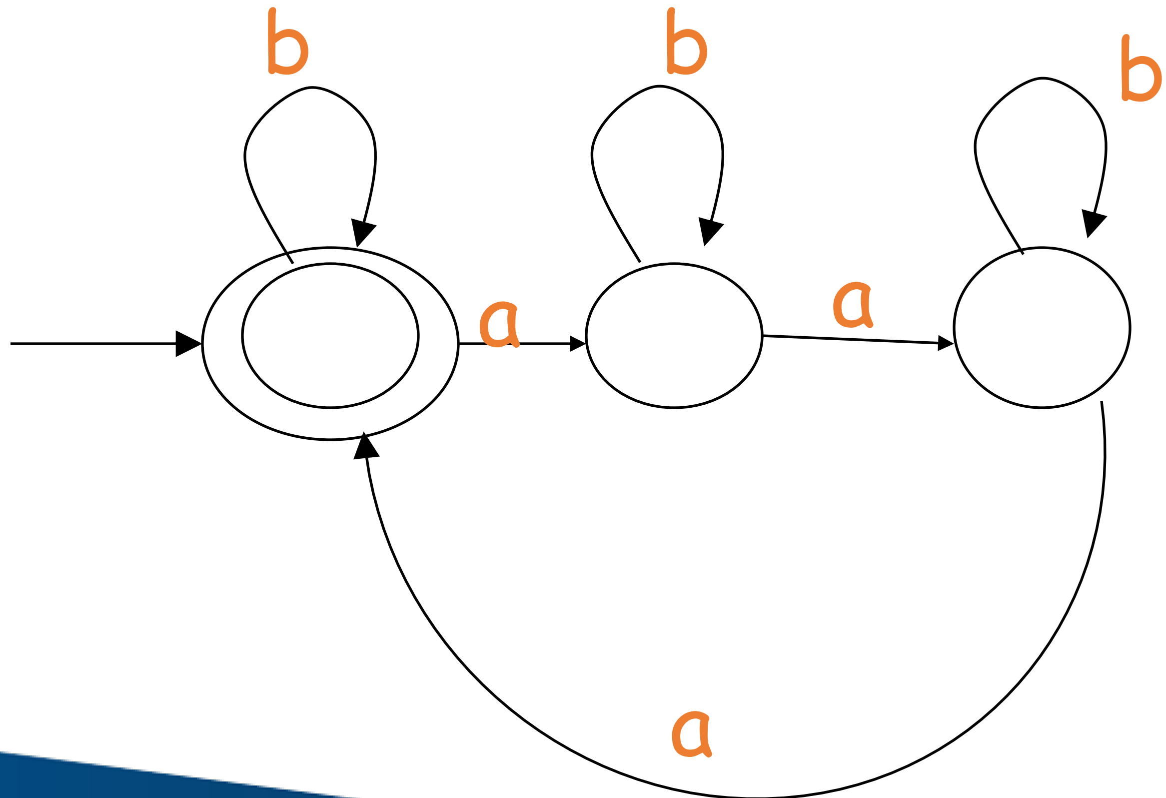
DFA – Odd no of 0's , ∑={ 0,1} or n0(w)mod2=1

- L={ 0,01,1110,000,000001111,...}

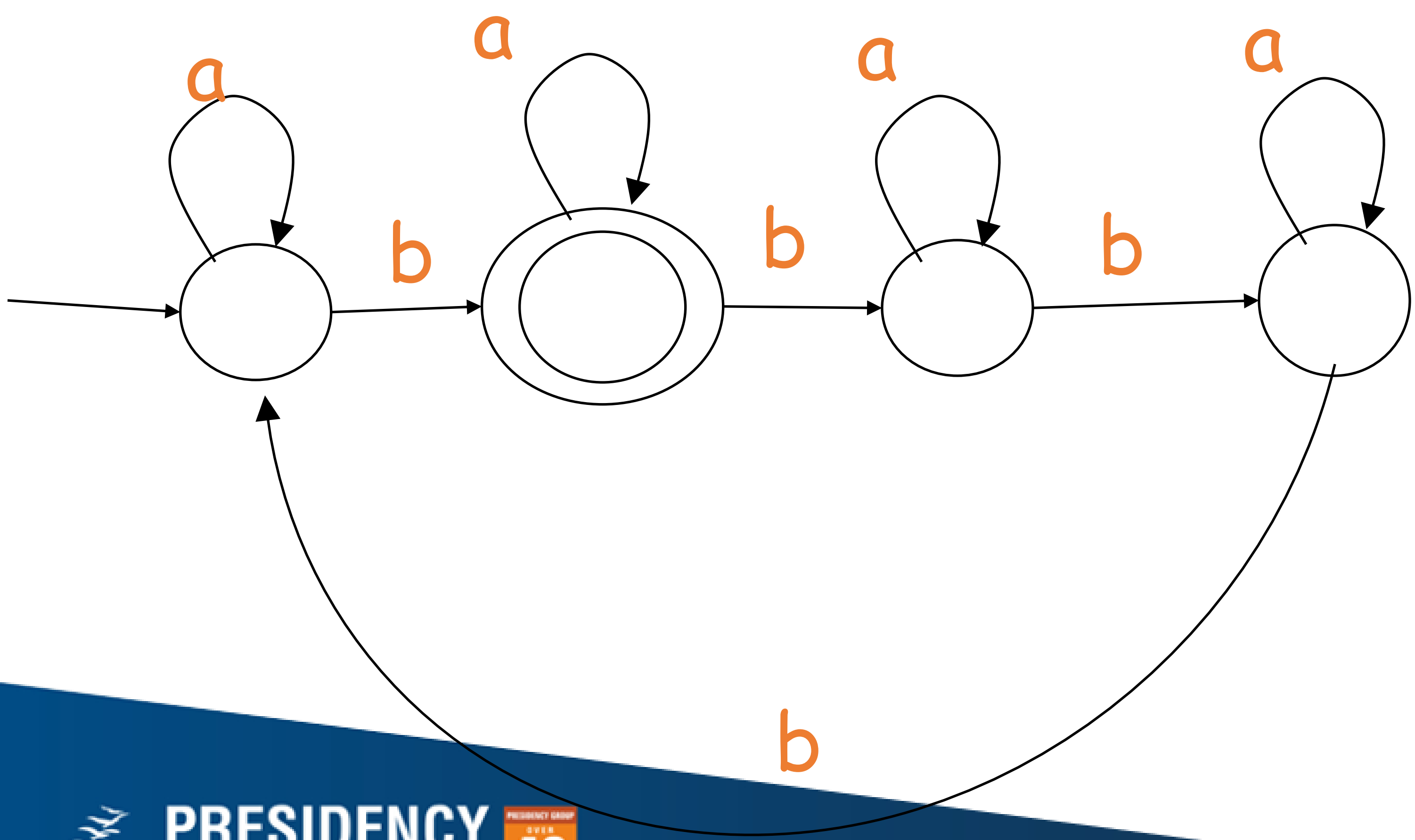


DFA - na(w)mod3=0, ∑={ a,b}

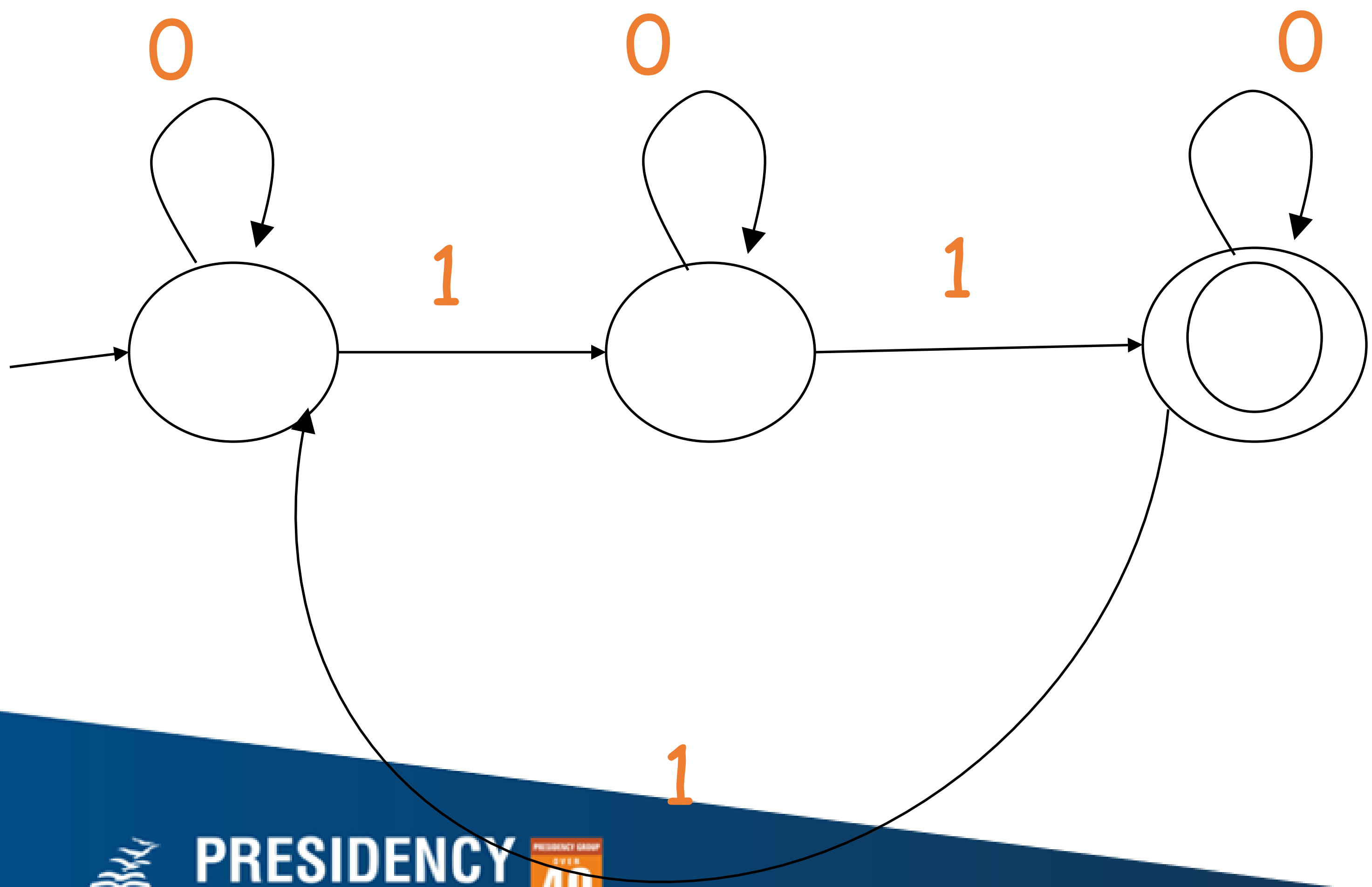
L={ λ,bbbb,aaabbb,aaaaaabbb,....}



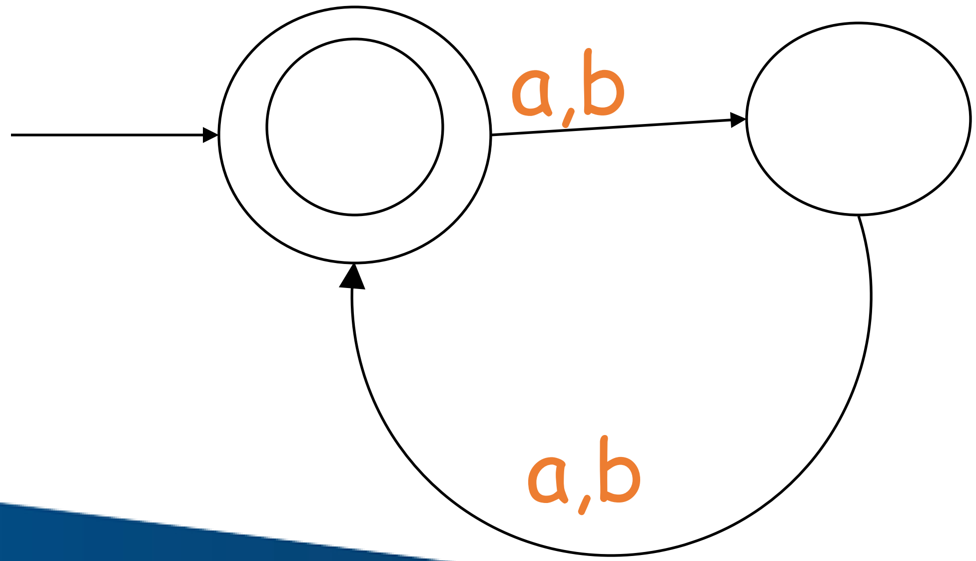
DFA- nb(w)mod4=1, ∑={ a,b}



DfA – n1(w)mod3=2, ∑={ 0,1}

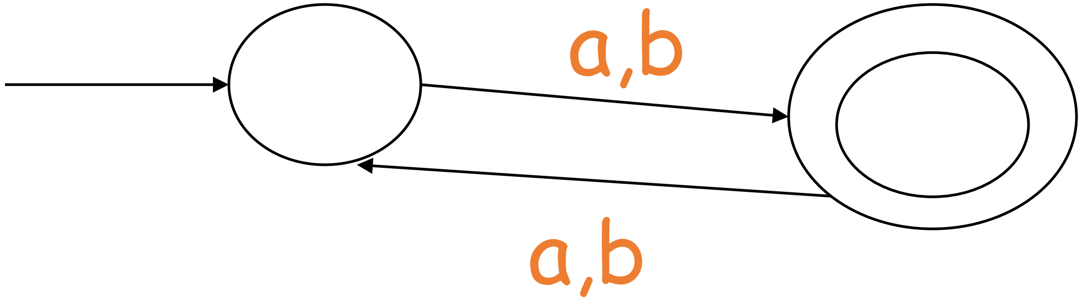


Dfa –lwlmod2=0,

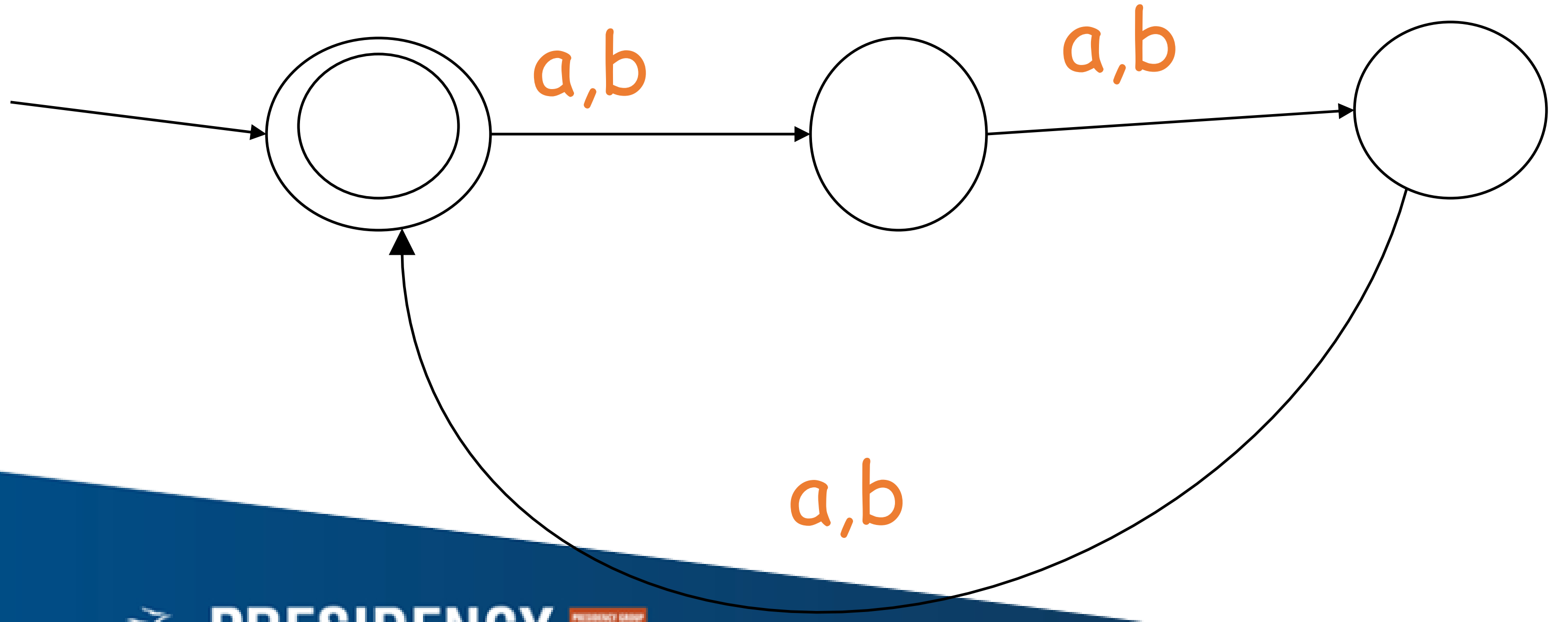


DfA – lwlmod2=1

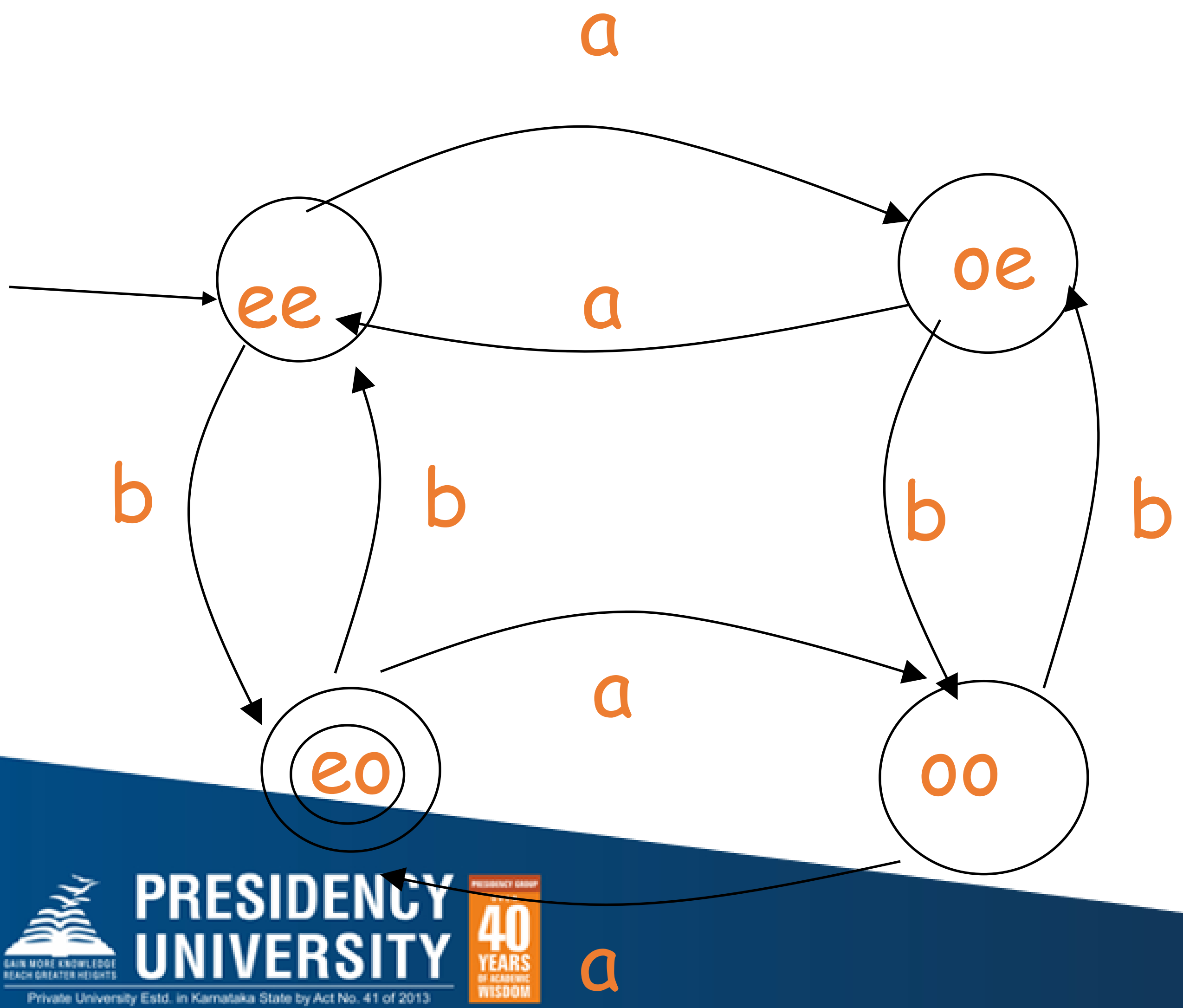
L={a,b,aaa,aba,aab,baa,bbb,bab
a,bba,....



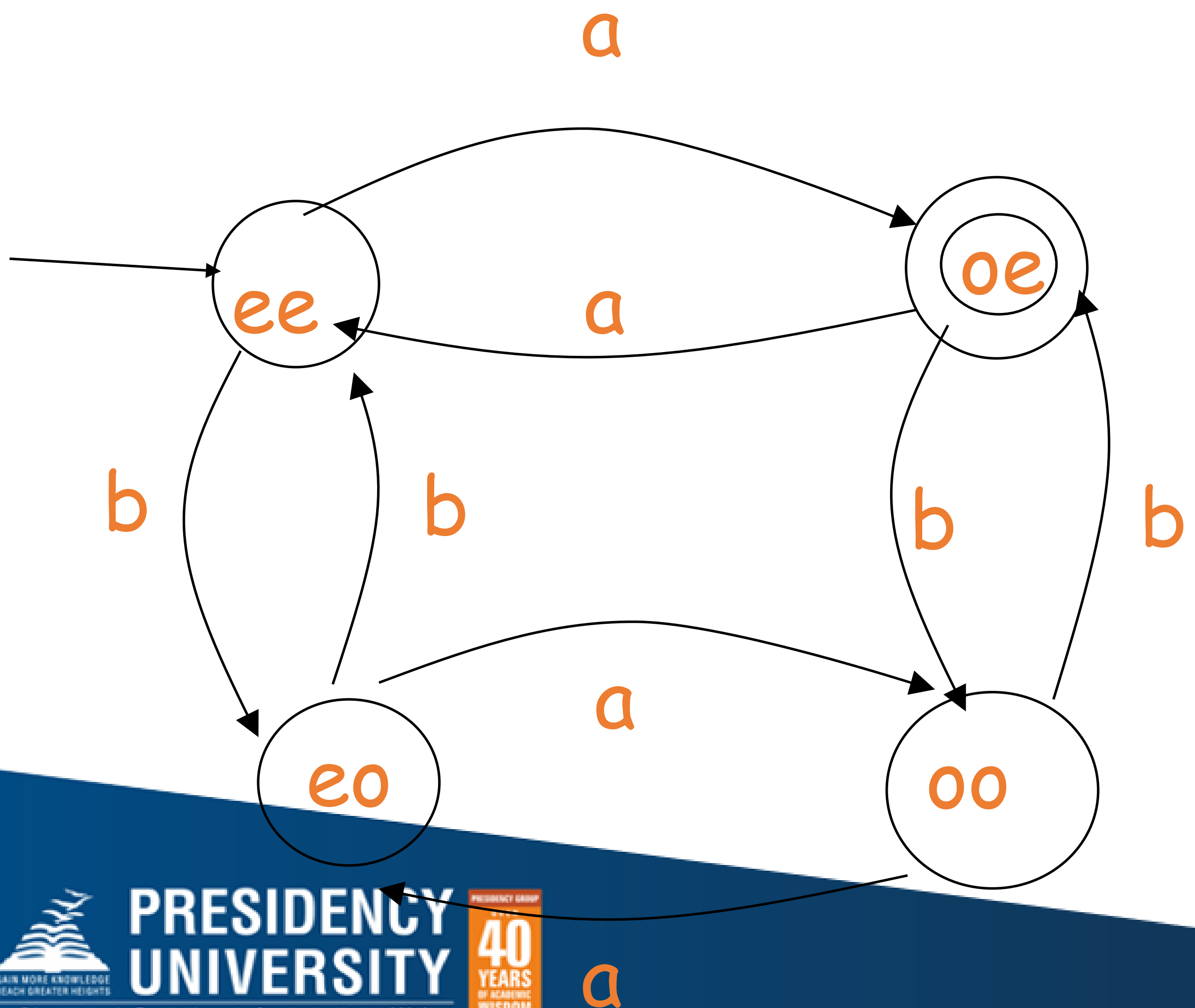
DFA –lwlmod3=0, ∑={ 0,1}



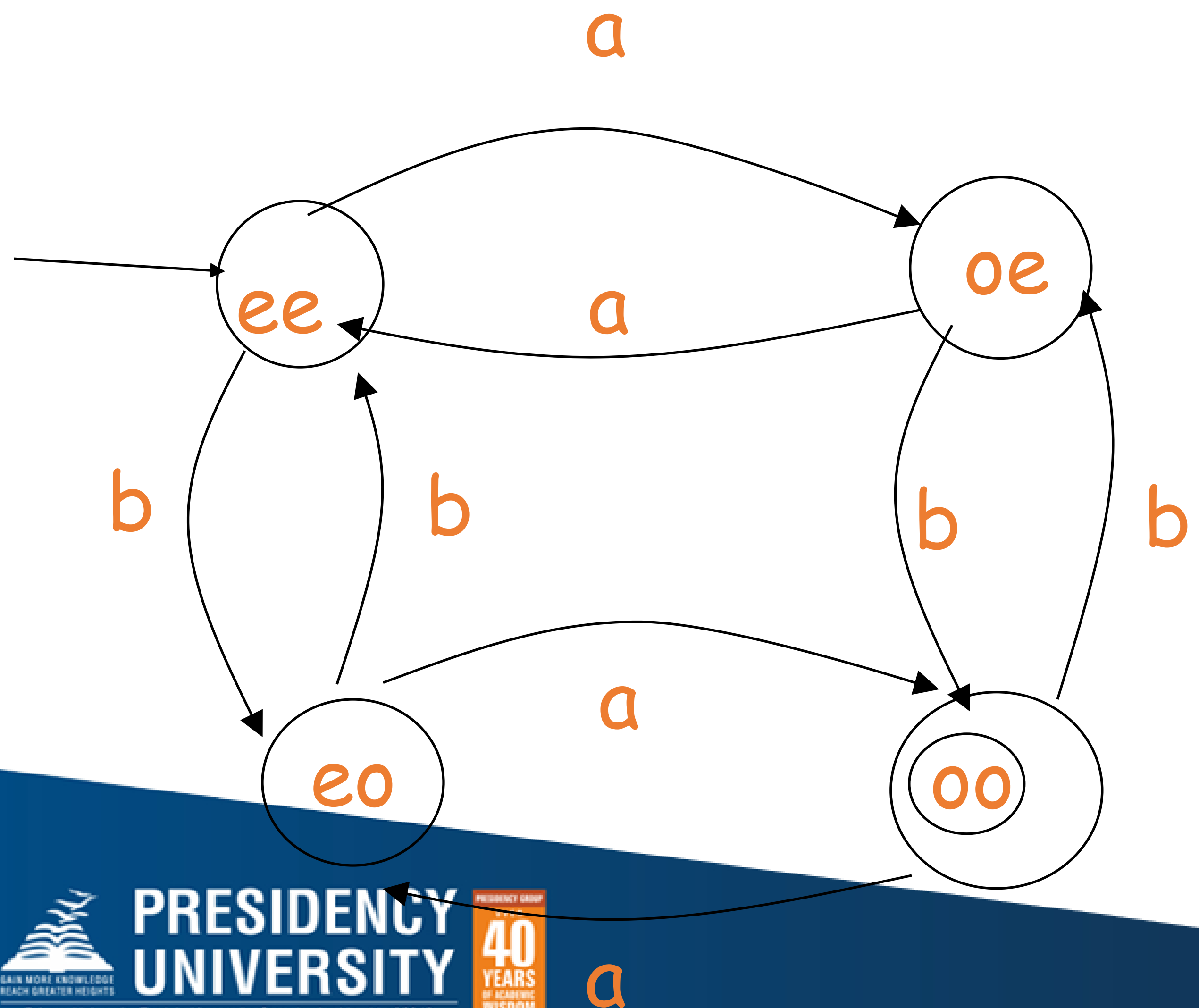
No of a's is even and no of b's odd-eo



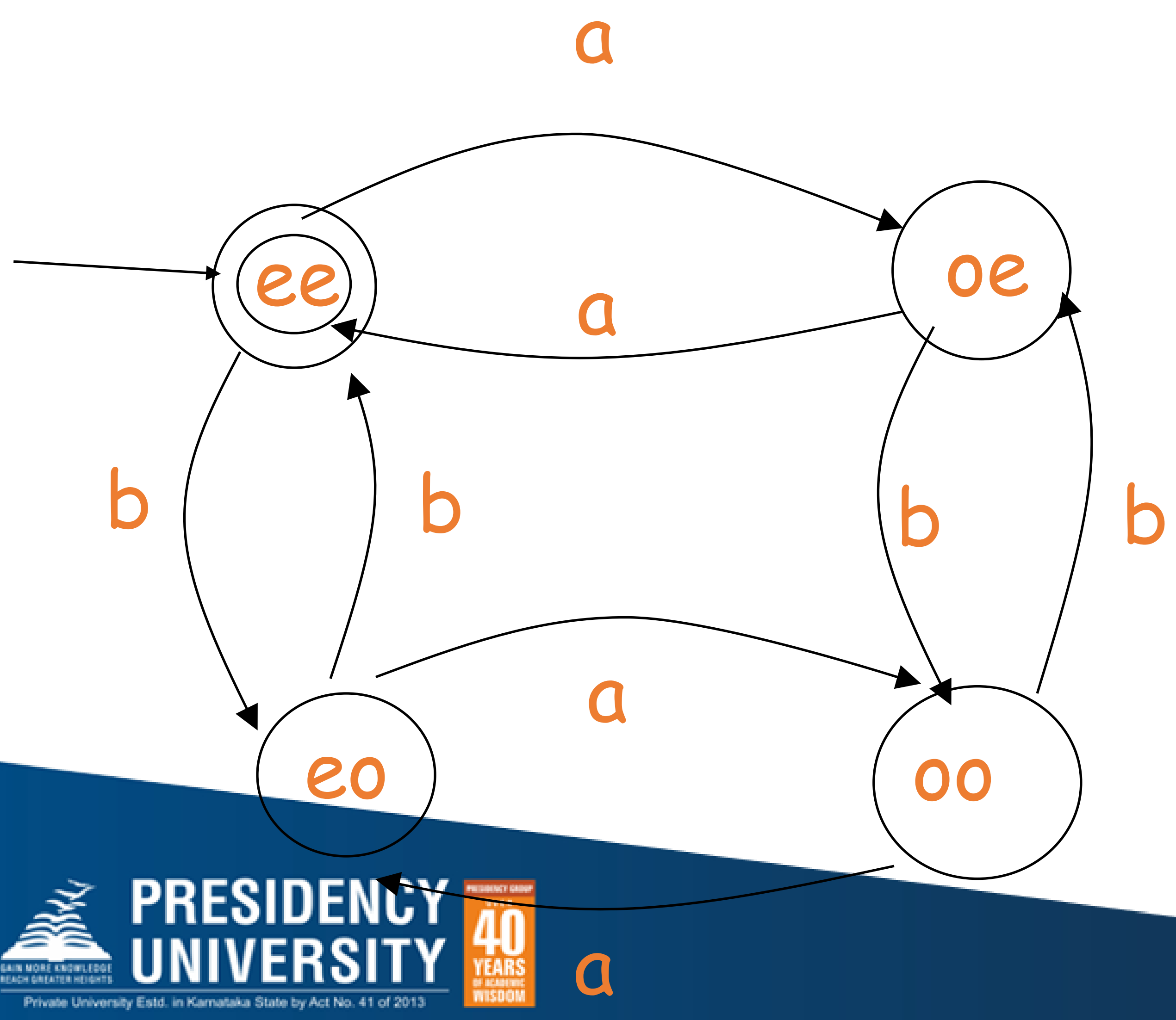
No of a's is odd and no of b's even=oe



No of a's is odd and no of b's odd-oo



No of a's is even and no of b's even- ee



NFA Definition

NFA is defined as $M=(Q, \Sigma, \partial, q0, F)$

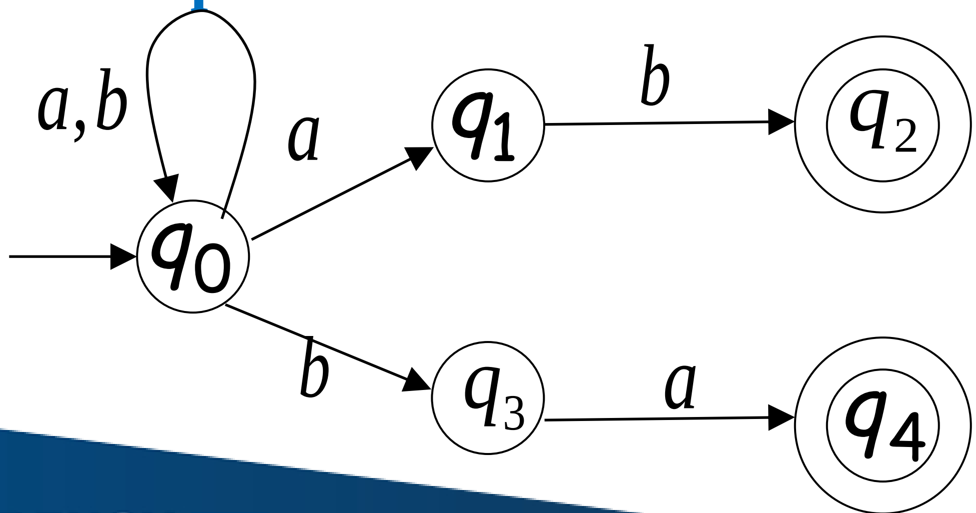
Where, Q= set of states

Σ = input alphabet

∂ = transition function $\partial: Q \times (\Sigma \cup \{\epsilon\}) \rightarrow 2^{Q}$

q0→ Start / Initial State

F → set of acceptance state

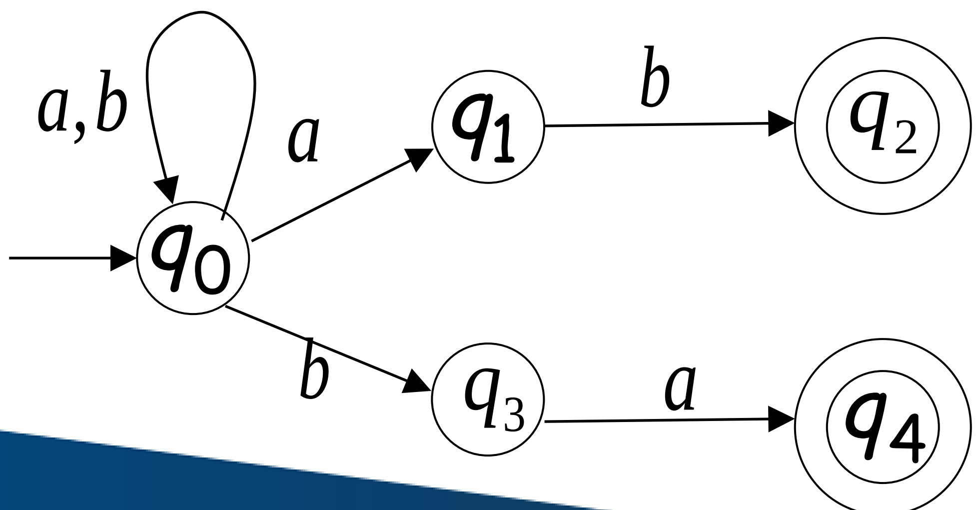


NFA Example

Design NFA that accepts language of strings ending with ab/ba over ∑ = {a, b}

Solution

- **Strings accepted= {ab, ba, abab, baba, bbab, aaaba, …}**
- **Strings rejected= {ϵ, a, b, aa, bb, aaa, abb, aaabbb, ...}**
- **Transition Diagram**

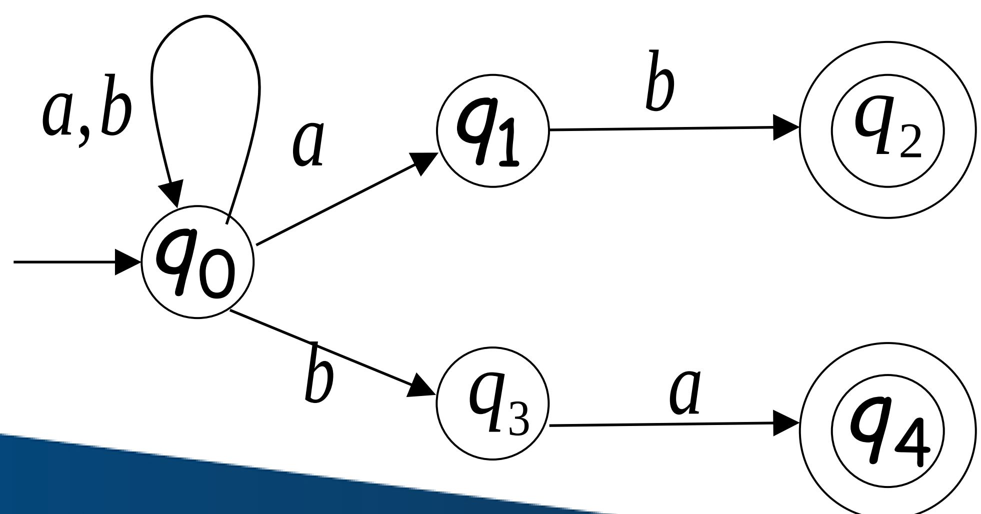


NFA to DFA Conversion Example

Convert NFA to DFA that accepts language of strings ending with ab/ba over ∑ = {a, b}

Solution

- **Draw NFA for given language**
- **Convert that NFA into DFA using subset construction method**



NFA to DFA Conversion Example

- Using Subset Construction Method
- Consider {q0} as Start State
- δ(q0, a)={q0 q1} newly formed state
- δ({q0 q1}, a)= δ(q0, a)u δ({q1, a})
  ={q0, q1} u {Ø}
  ={q0 q1}
- δ({q0 q1}, b)= δ(q0, b)u δ({q1, b})
  ={q0, q3} u {q2}
  ={q0 q2 q3}

NFA	a	b
q0	{q0, q1}	{q0, q3}
q1	-	{q2}
q2	-	-
q3	{q4}	-
q4	-	-

DFA	a	b
q0	{q0 q1}	{q0 q3}
{q0 q1}	{q0 q1}	{q0 q2 q3}

NFA to DFA Conversion Example

- Using Subset Construction Method
- Consider {q0} as Start State
- δ(q0, a)={q0 q1} newly formed state
- δ({q0 q1}, a)= δ(q0, a)∪ δ({q1, a})
  ={q0, q1} ∪ {Ø}
  ={q0 q1}
- δ({q0 q1}, b)= δ(q0, b)∪ δ({q1, b})
  ={q0, q3} ∪ {q2}
  ={q0 q2 q3}
- {q0 q2 q3} is newly formed state
- δ({q0 q2 q3}, a)= δ(q0, a) ∪ δ({q2, a}) ∪ δ({q3, a})
  ={q0, q1} ∪ {Ø} ∪ {q4}
  ={q0 q1 q4}
- δ({q0 q2 q3}, b)= δ(q0, b)∪ δ({q2, b}) ∪ δ({q3, b})
  ={q0, q3} ∪ {Ø} ∪ {Ø} ={q0 q3}

NFA	a	b
q0	{q0, q1}	{q0, q3}
q1	-	{q2}
q2	-	-
q3	{q4}	-
q4	-	-

DFA	a	b
q0	{q0 q1}	{q0 q3}
{q0 q1}	{q0 q1}	{q0 q2 q3}
{q0 q2 q3)	{q0 q1 q4}	{q0 q3}

NFA to DFA Conversion Example

- {q0 q3} is newly formed state
- δ({q0 q3}, a)= δ(q0, a)∪ δ({q3, a})
  ={q0, q1} ∪ {q4}
  ={q0 q1 q4}
- δ({q0 q3}, b)= δ(q0, b)∪ δ({q3, b})
  ={q0, q3} ∪ {Ø}
  ={q0 q3}

NFA	a	b
q0	{q0, q1}	{q0, q3}
q1	-	{q2}
q2	-	-
q3	{q4}	-
q4	-	-

DFA	a	b
q0	{q0 q1}	{q0 q3}
{q0 q1}	{q0 q1}	{q0 q2 q3}
{q0 q2 q3)	{q0 q1 q4}	{q0 q3}
{q0 q3}	{q0 q1 q4}	{q0 q3}

NFA to DFA Conversion Example

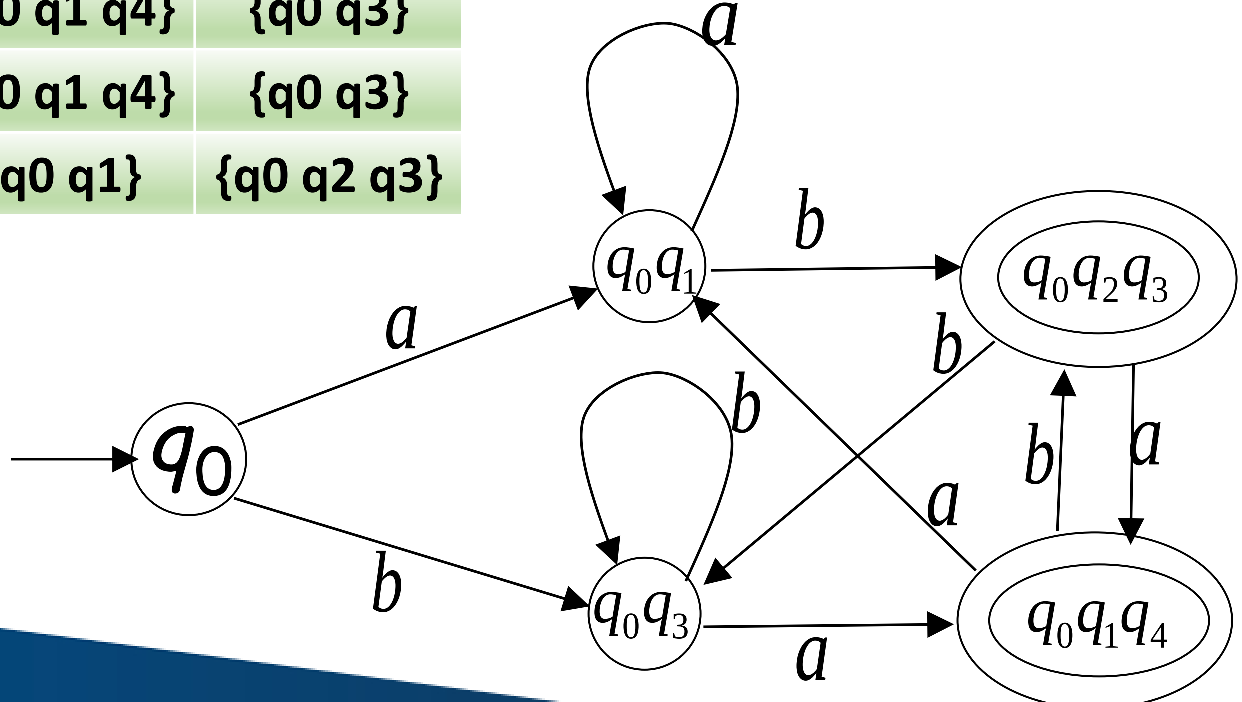
- {q0 q3} is newly formed state
- δ({q0 q3}, a)= δ(q0, a)∪ δ({q3, a})
  ={q0, q1} ∪ {q4}
  ={q0 q1 q4}
- δ({q0 q3}, b)= δ(q0, b)∪ δ({q3, b})
  ={q0, q3} ∪ {Ø}
  ={q0 q3}
- {q0 q1 q4} is newly formed state
- δ({q0 q1 q4}, a)= δ(q0, a) ∪ δ({q1, a}) ∪ δ({q4, a})
  ={q0, q1} ∪ {Ø} ∪ {Ø}
  ={q0 q1}
- δ({q0 q1 q4}, b)= δ(q0, b)∪ δ({q1, b}) ∪ δ({q4, b})
  ={q0, q3} ∪ {q2} ∪ {Ø}
  ={q0 q2 q3}

NFA	a	b
q0	{q0, q1}	{q0, q3}
q1	-	{q2}
q2	-	-
q3	{q4}	-
q4	-	-

DFA	a	b
q0	{q0 q1}	{q0 q3}
{q0 q1}	{q0 q1}	{q0 q2 q3}
{q0 q2 q3)	{q0 q1 q4}	{q0 q3}
{q0 q3}	{q0 q1 q4}	{q0 q3}
{q0 q1 q4}	{q0 q1}	{q0 q2 q3}

NFA to DFA Conversion Example

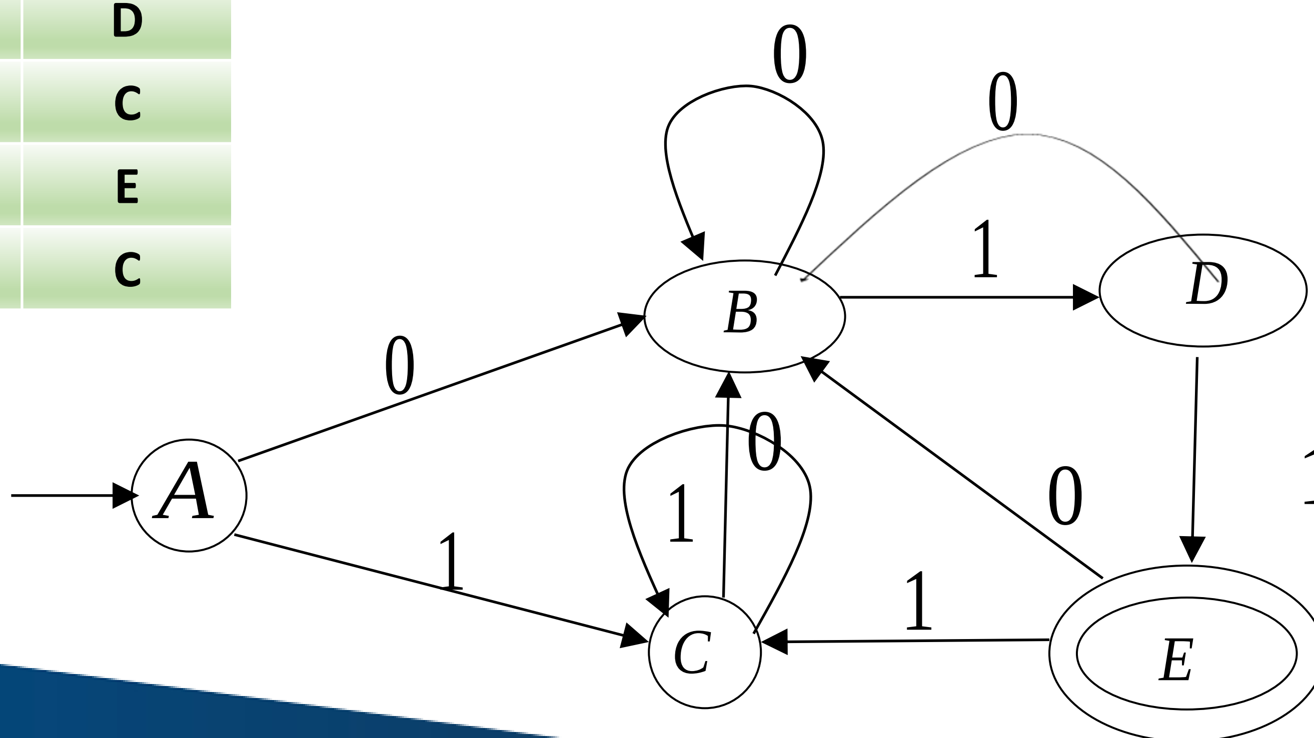
DFA	a	b
q0	{q0 q1}	{q0 q3}
{q0 q1}	{q0 q1}	{q0 q2 q3}
{q0 q2 q3)	{q0 q1 q4}	{q0 q3}
{q0 q3}	{q0 q1 q4}	{q0 q3}
{q0 q1 q4}	{q0 q1}	{q0 q2 q3}



Minimization of DFA Example

Minimize the following DFA using state equivalence method

DFA	0	1
→A	B	C
B	B	D
C	B	C
D	B	E
E*	B	C



Minimization of DFA Example

Minimize the following DFA

- **Use State Equivalence Method**
- **Write 0'Equivalence as**
  {A, B, C, D} {E}
- **Write 1'Equivalence**
  {A, B, C} {D} {E}
- **Write 2'Equivalence**
  {A, C} {B} {D} {E}
- **Write 3'Equivalence**
  {A, C} {B} {D} {E}

DFA	0	1
→A	B	C
B	B	D
C	B	C
D	B	E
E*	B	C

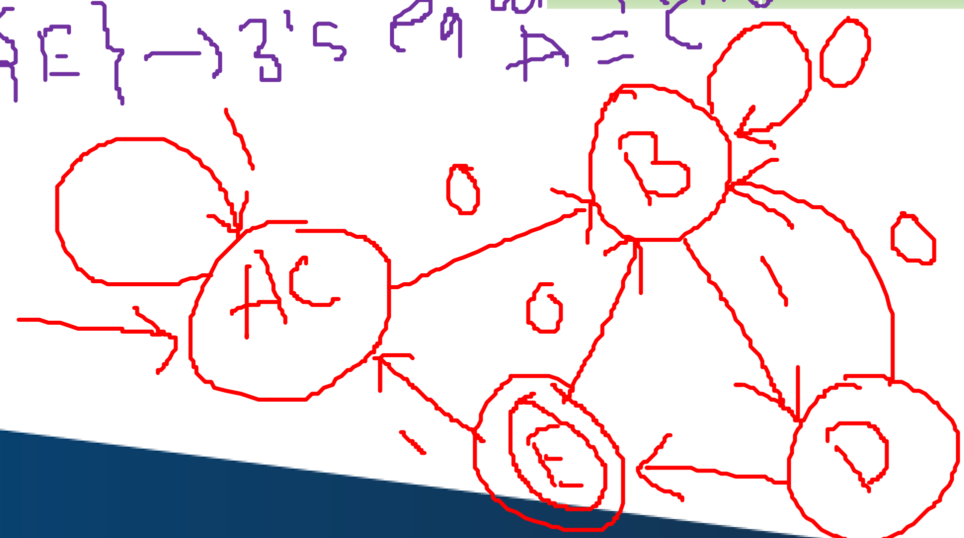
{A, B, C, D} {E} → 0's equivalence

{A, B, C} {D} {E} → 1's equivalence

{A, B, C} {D} {E} → 2's equivalence

{A, C} {B} {D} {E} → 3's equivalence

A = C



Minimization of DFA Example

Minimize the following DFA

- **Use State Equivalence Method**
- **Write 0'Equivalence as**

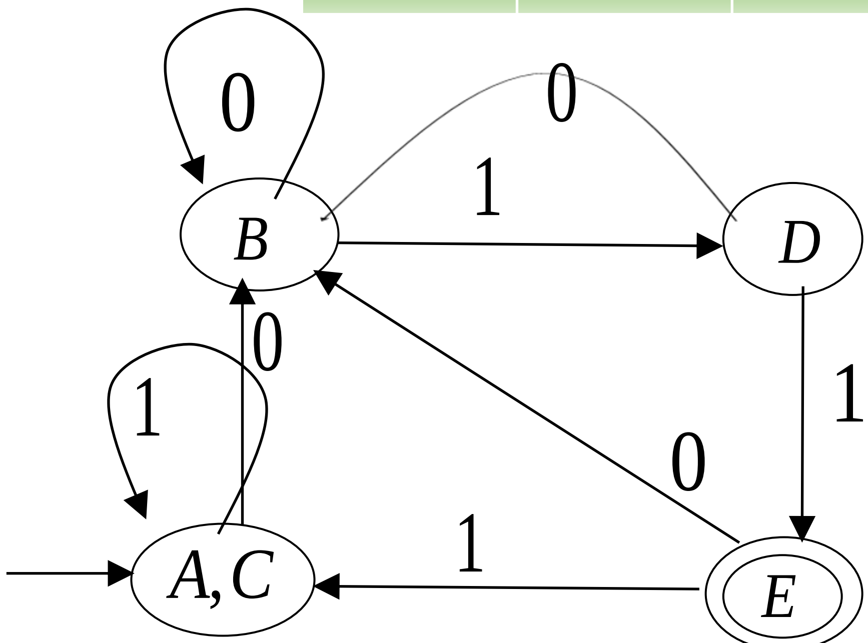
  {A, B, C, D} {E}
- **Write 1'Equivalence**

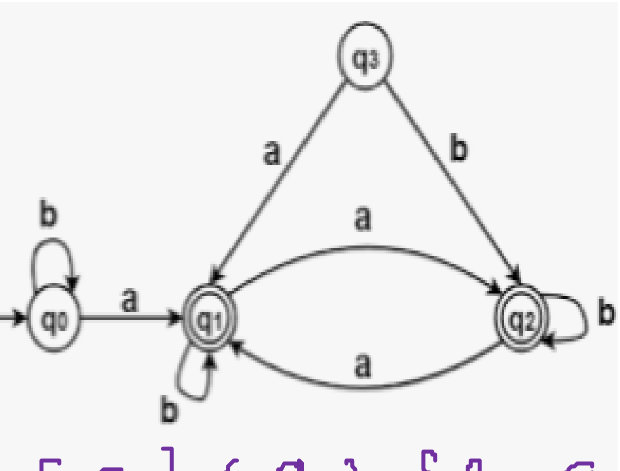
  {A, B, C} {D} {E}
- **Write 2'Equivalence**

  {A, C} {B} {D} {E}
- **Write 3'Equivalence**

  {A, C} {B} {D} {E}

DFA	0	1
→A,C	B	C
B	B	D
D	B	E
E*	B	C





$\{q_0, q_3\}\{q_1, q_2\} \rightarrow$ 0's equivalent

$\delta(q_0, a) = q_1 \quad \delta(q_0, b) = q_0$

$\delta(q_3, a) = q_1 \quad \delta(q_3, b) = q_2$

$\{q_0\}\{q_3\}\{q_1, q_2\} \rightarrow$ 1's $\quad q_0 \neq q_3$

$\{q_0\}\{q_3\}\{q_1, q_2\} \rightarrow$ 2's

$\delta(q_1, a) = q_2 \parallel \quad \delta(q_1, b) = q_1 \parallel$

$\delta(q_2, a) = q_1 \parallel \quad \delta(q_2, b) = q_2$

$q_1 = q_2$

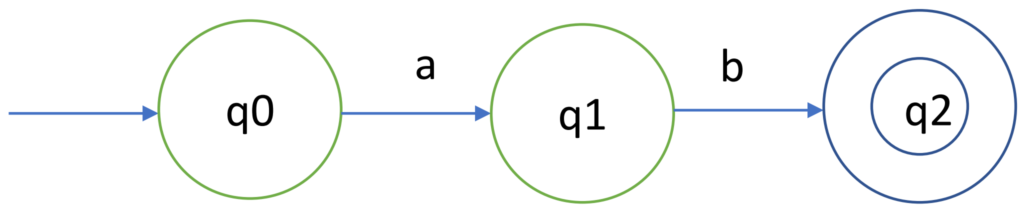
Ɛ-NFA (Finite Automaton with Epsilon Transitions)

- A transition with an empty string is called an Ɛ-transition
- The formal definition of ∈-NFA is represented through 5-tuple (Q, ∑, δ, q_0, F) where,
- **Q** is a finite set of all states ($q_{0,}$ $q_{1,}$ $q_{2,}$ … q_n) where n is finite number
- **∑** is a finite set of symbols called the alphabet. i.e. {0, 1},
- **δ : Q x (∑ U ∈) → 2^Q is a total function called as transition function**
- **q_0** is the initial state from where any input is processed (q_0 ∈ Q).
- **F** is a set of final state/states where F will be subset (⊆) of Q.

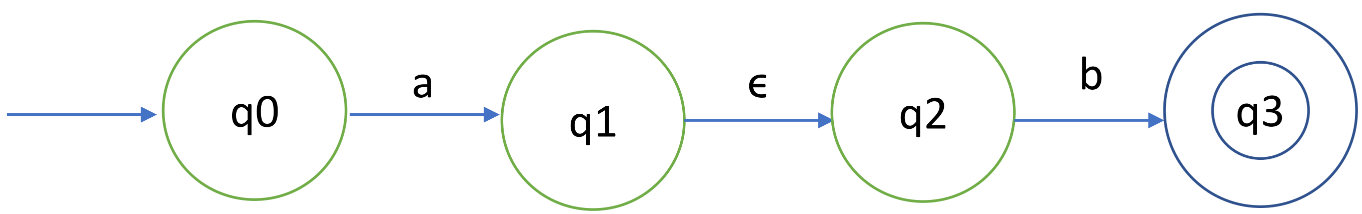
PRESIDENCY UNIVERSITY

Examples of Epsilon-NFA

- **Draw a Epsilon NFA which accept the string “ab”.**



NFA



ϵ-NFA

Example 2

Draw an Epsilon Finite Automata which can accept the string “a or b or c”

Conversion of ϵ-NFA to DFA

- The method for converting the NFA with ε to DFA is explained below −

Step 1 − Consider M={Q, Σ, δ,q0,F) is NFA with ε. We have to convert this NFA with ε to equivalent DFA denoted by

M0=(Q0,Σ, δ0,q0,F0)

Then obtain,

ε-closure(q0) ={p1,p2,p3,……pn}

then [p1,p2,p2,….pn] becomes a start state of DFA

now[p1,p2,p3,….pn] ∈ Q0

- **Step 2** – We will obtain δ transition on [p1,p2,p3,...pn] for each input.

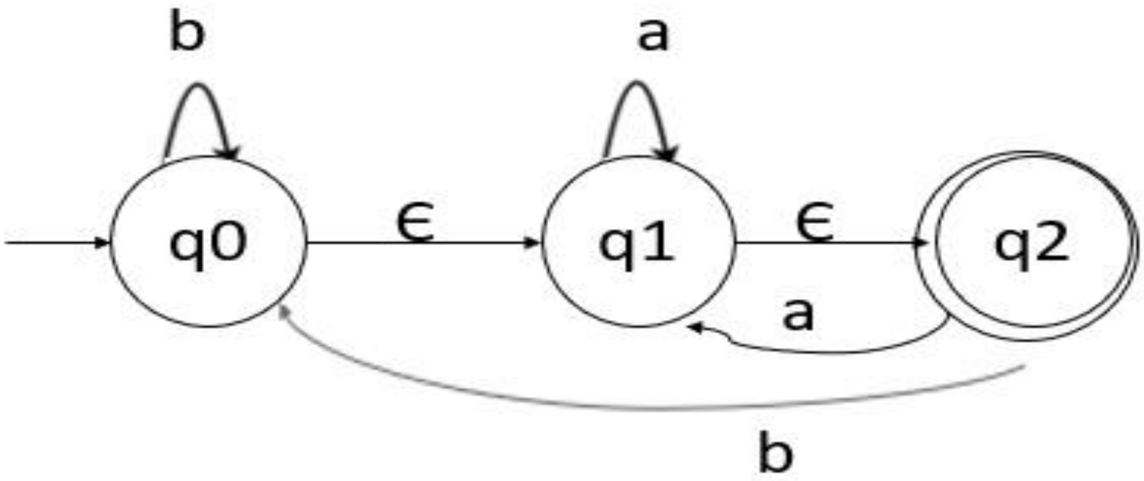
δ 0([p1,p2,p3,..pn],a) = ε-closure(δ(p1,a) U δ(p2,a)U……………… δ(pn,a))= U (i=1 to n) ε-closure d(pi,a)

Where a is input ∈Σ

- Step 3 – The state obtained [p1,p2,p3,...pn] ∈ Q0 .
- The states containing final state in pi is a final state in DFA.

Example of Conversion

- Convert the following NFA with epsilon to equivalent DFA



- To convert this NFA with epsilon, we will first find the ε-closures, as given below –

ε-closure(q0)={q0,q1,q2}

ε-closure(q1)={q1,q2}

ε-closure(q2)={q2}

- Let us start from **ε-closure of start state**, as mentioned below –
- When, ε-closure(q0)={q0,q1,q2}, we will call this state as A.
- Now, let us find transition on A with every input symbol, as shown below –

- $\delta'(A, a) = \varepsilon\text{-closure}(\delta(A,a))$

  $= \varepsilon\text{-closure}(\delta(q0,q1,q2), a))$

  $= \varepsilon\text{-closure}(\delta(q0, a) \cup \delta(q1,a) \cup \delta(q2,a))$

  $= \varepsilon\text{-closure}(\Phi \cup q1 \cup q1)$

  $= \varepsilon\text{-closure}(q1)$

  $= \{q1, q2\}$ let us call it as state B

- $\delta'(A, b) = \varepsilon\text{-closure}(\delta(A,b))$

$= \varepsilon\text{-closure}(\delta(q0,q1,q2), b))$

$= \varepsilon\text{-closure}(\delta(q0, b) \cup \delta(q1,b) \cup \delta(q2,b))$

$= \varepsilon\text{-closure}(q0 \cup \Phi \cup q0)$

$= \varepsilon\text{-closure}(q0)$

$= \{q0,q1, q2\}$ its nothing but state A

- $\delta'(B, a) = \varepsilon\text{-closure}(\delta(B,a))$

  $= \varepsilon\text{-closure}(\delta(q1,q2), a))$

  $= \varepsilon\text{-closure}(\delta(q1,a) \cup \delta(q2,a))$

  $= \varepsilon\text{-closure}(q1 \cup q1)$

  $= \varepsilon\text{-closure}(q1)$

  $= \{q1, q2\}$ its nothing but state B

- $\delta'(B, b) = \varepsilon\text{-closure}(\delta(B,b))$

$= \varepsilon\text{-closure}(\delta(q1,q2), b))$

$= \varepsilon\text{-closure}(\delta(q1,b) \cup \delta(q2,b))$

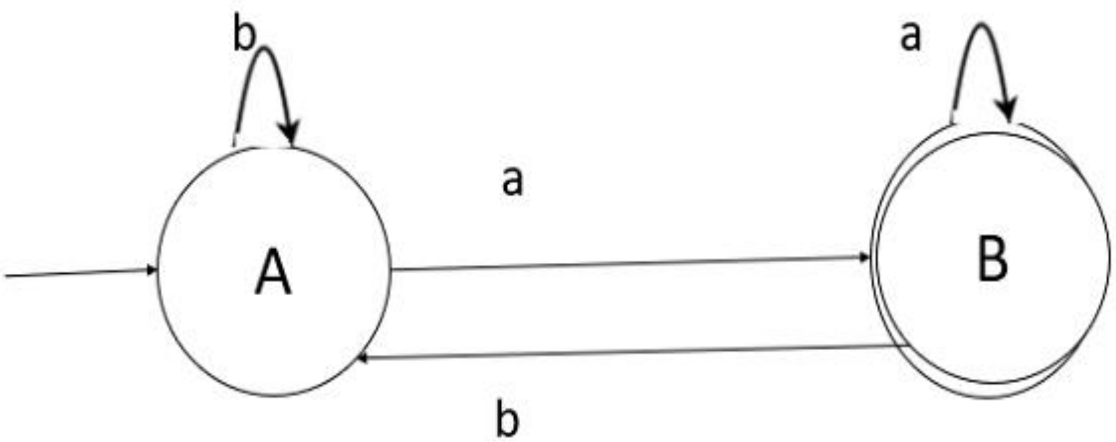
$= \varepsilon\text{-closure}(\Phi \cup q0)$

$= \varepsilon\text{-closure}(q0)$

= {q0,q1, q2} its nothing but state A

- Hence, the transition table for the generated DFA is as follows –

States\inputs	a	b
A	B	A
B	B	A



F' ={A,B}

- As, A={q0,q1,q2} in which the final state q2 lies. Hence, A is the final state.
- In B ={q1,q2} the state q2 lies. Hence, B is also the final state.

END of MODULE 2

Exercises