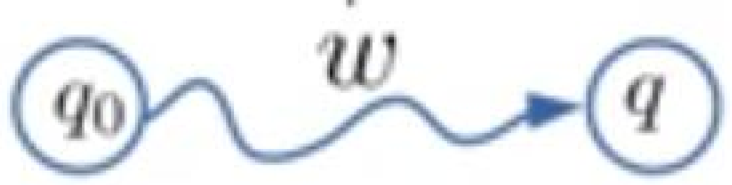
Few guidelines to construct a DFA for a given language

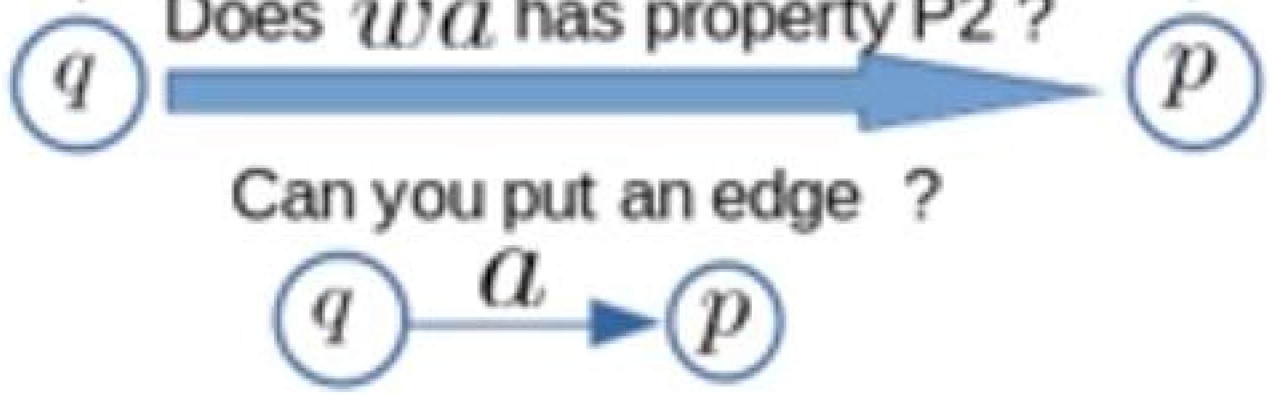
what property should w have s.t. $\delta(q_0, w) = q$?

What are the states / paths it must have ? Consider `short' input strings

For any state q (not only final state) what are the strings w s.t. there is a path



Once we know, w have property P1 s.t. $\delta(q_0, w) = q$ and x have property P2 s.t. $\delta(q_0, x) = p$



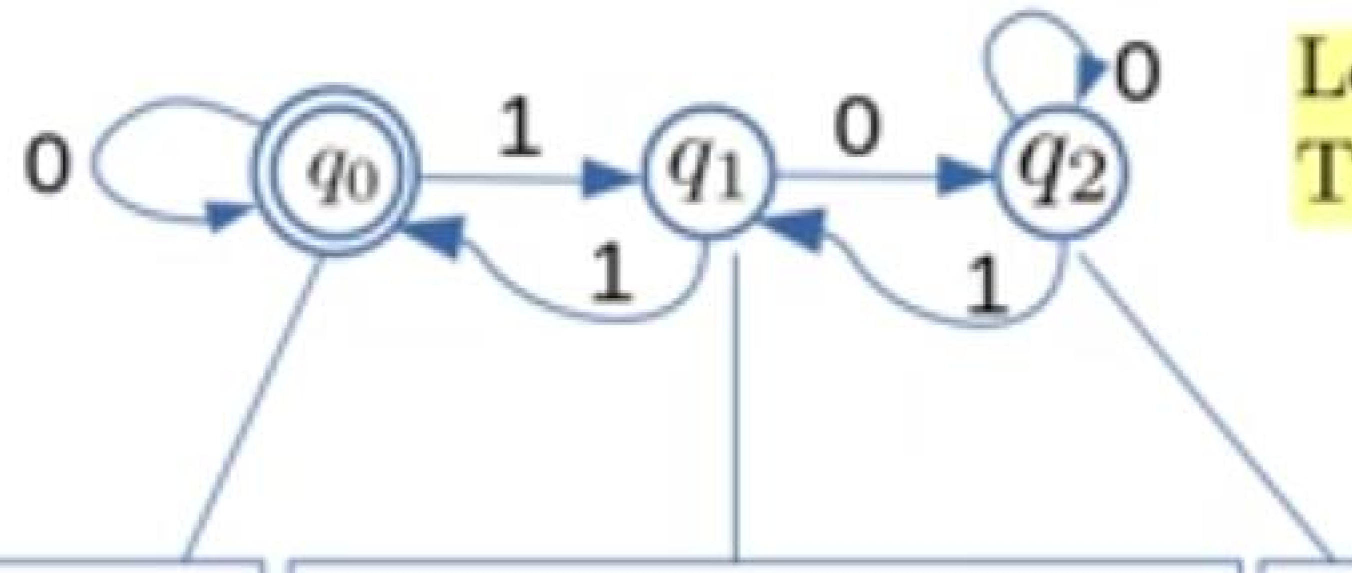
Does wa has property P2 ?

Can you put an edge ?

Example

$$\Sigma = \{0,1\} \quad L(M) = \{w \mid w \text{ is binary representation of an integer divisible by 3}\}$$

ϵ is considered integer 0



Let $f : \Sigma^* \to \mathbb{N}$ s.t. w is binary rep. of $f(w)$
Then, $f(w0) = 2f(w)$ and $f(w1) = 2f(w) + 1$

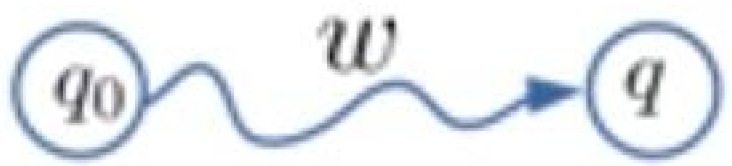
string represents integer i s.t. i = 0 mod 3

string represents integer j s.t. j = 1 mod 3

string represents integer k s.t. k = 2 mod 3

Few guidelines to construct a DFA for a given language

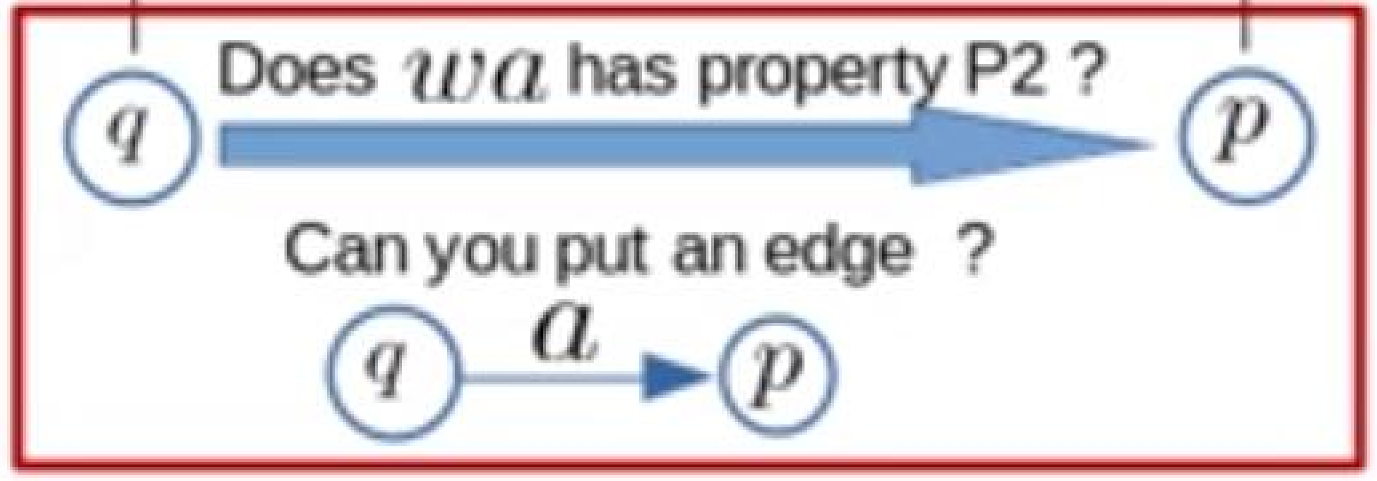
what property should w have s.t. $\delta(q_0, w) = q$?



What are the states / paths it must have ? Consider `short' input strings

For any state q (not only final state) what are the strings w s.t. there is a path

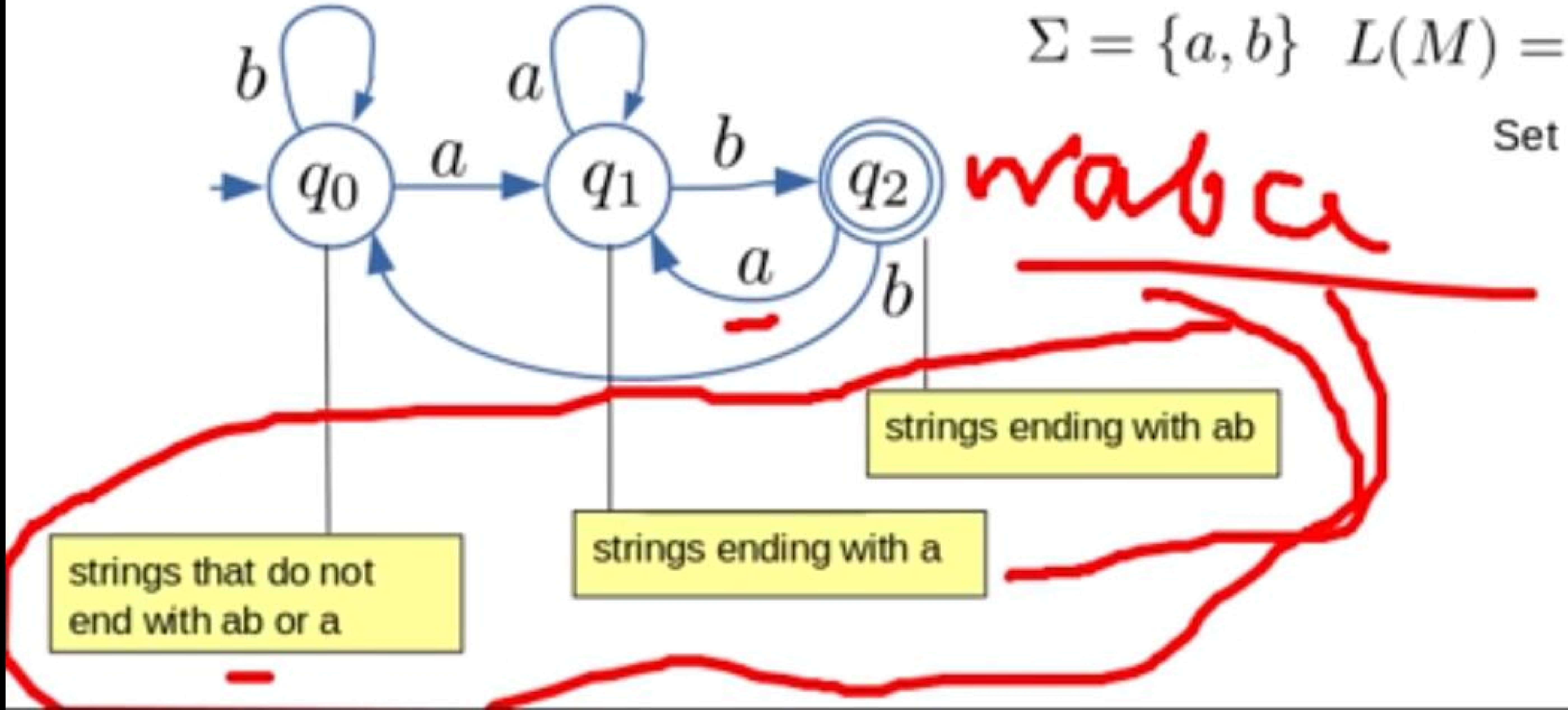
Once we know, w have property P1 s.t. $\delta(q_0, w) = q$ and x have property P2 s.t. $\delta(q_0, x) = p$



Does wa has property P2 ?

Can you put an edge ?

Example



strings that do not end with ab or a

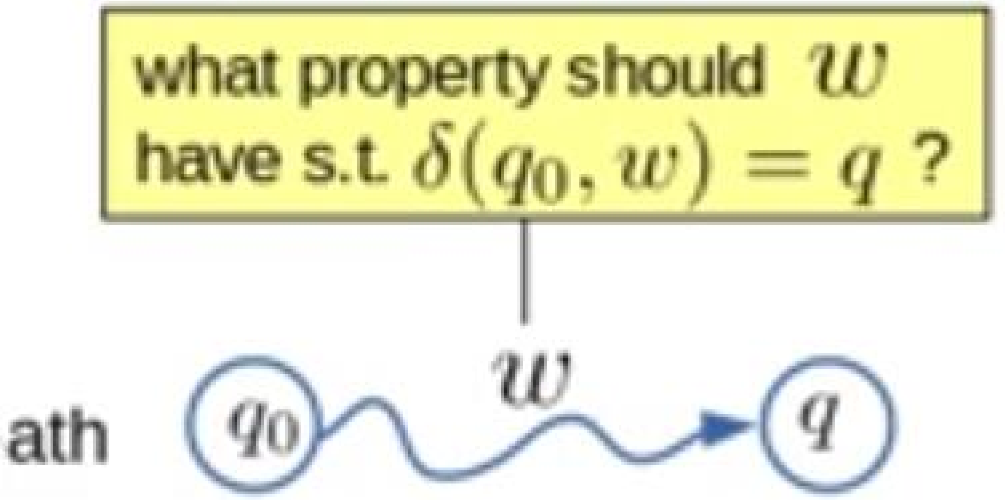
strings ending with a

strings ending with ab

$\Sigma = \{a, b\}$ $L(M) = \{wab \mid w \in \{a, b\}^*\}$

Set of all strings ending with ab

Few guidelines to construct a DFA for a given language

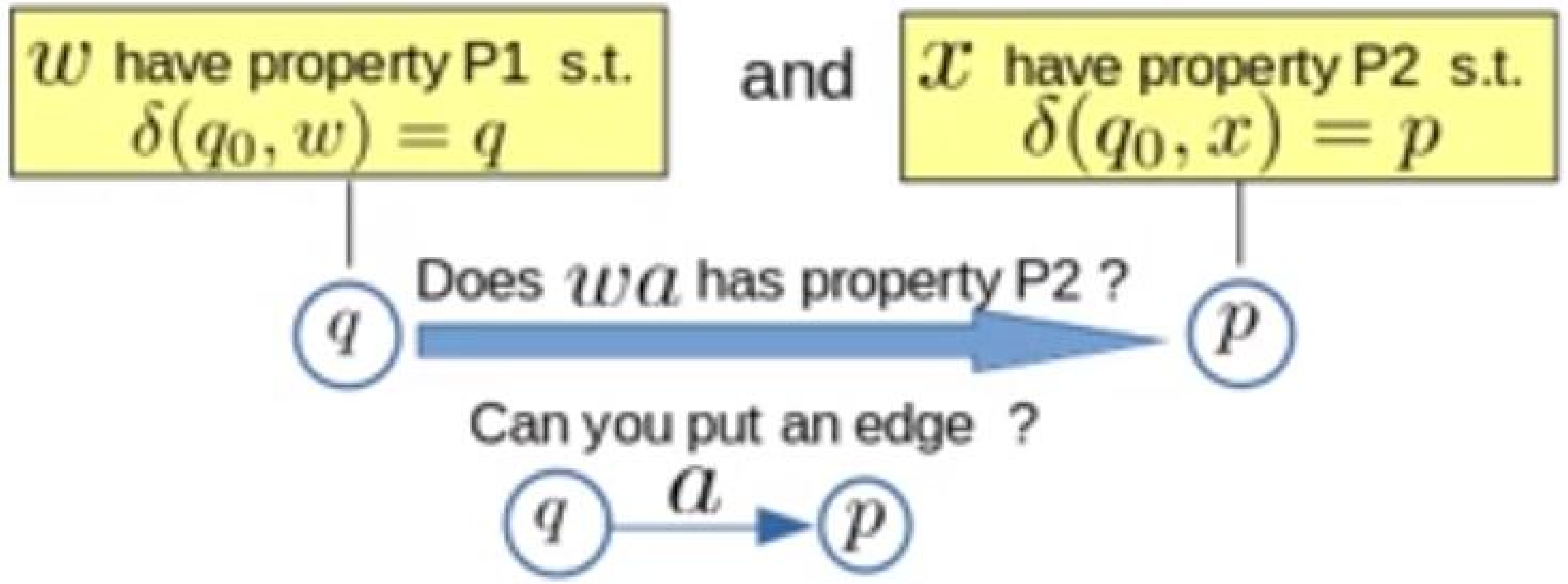


what property should w have s.t. $\delta(q_0, w) = q$?

What are the states / paths it must have ? Consider `short' input strings

For any state q (not only final state) what are the strings w s.t. there is a path

Once we know,



w have property P1 s.t. $\delta(q_0, w) = q$

and

x have property P2 s.t. $\delta(q_0, x) = p$

Does wa has property P2 ?

Can you put an edge ?

Example

$$\Sigma = \{a, b\} \quad L(M) = \{wab \mid w \in \{a, b\}^*\}$$

Set of all strings ending with ab

Converting Partial Transition Function To A Total Function

Theorem : Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$ with a partial transition function δ

We can construct another DFA $M' = (Q', \Sigma, \delta', q_0, F)$ with a total transition function δ'

s.t. $L(M) = L(M')$

We shall use partial transition function if it easier to do so.

But without loss of generality,
we can always assume the transition function to be total.

Converting Partial Transition Function To A Total Function

Theorem : Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$ with a partial transition function δ

We can construct another DFA $M' = (Q', \Sigma, \delta', q_0, F)$ with a total transition function δ'

s.t. $L(M) = L(M')$

Let $Q' = Q \cup \{q_{\text{dead}}\}$

and $\delta'(q, a) = \delta(q, a)$ if $\delta(q, a)$ is defined

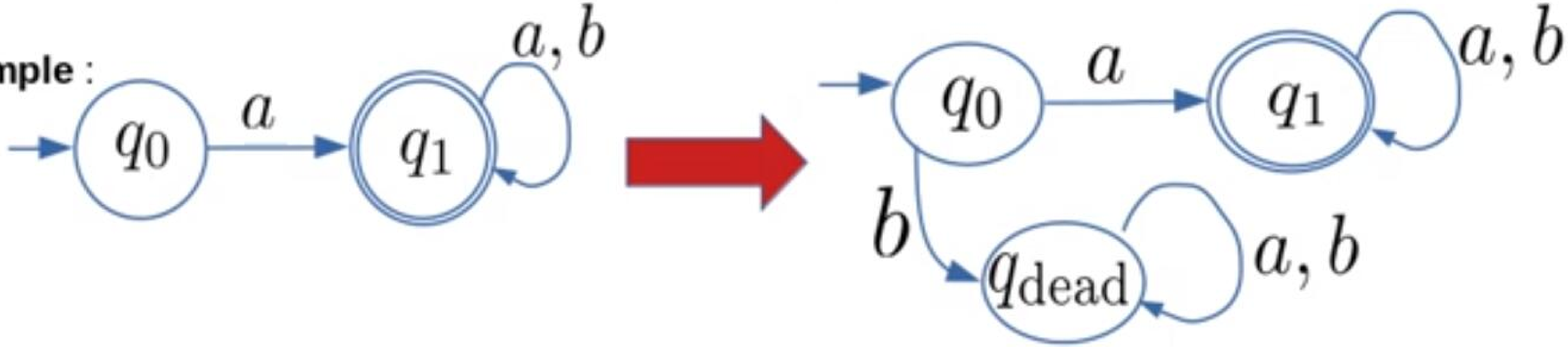
$\delta'(q, a) = q_{\text{dead}}$ if $\delta(q, a)$ is undefined

$\delta(q_{\text{dead}}, a) = q_{\text{dead}}$ for all input symbol a

Notice that $\delta'$is a total function

- Take input string w .
- Since δ and δ'agree on all valid moves, if $\hat{\delta}(q, w)$ is defined then, $\hat{\delta}(q, w) \in F \Leftrightarrow \hat{\delta}'(q, w) \in F$
- If $\hat{\delta}(q, w)$ is undefined, M must encounter an invalid move. So, $\hat{\delta}'(q, w) = q_{\text{dead}}$

Example :

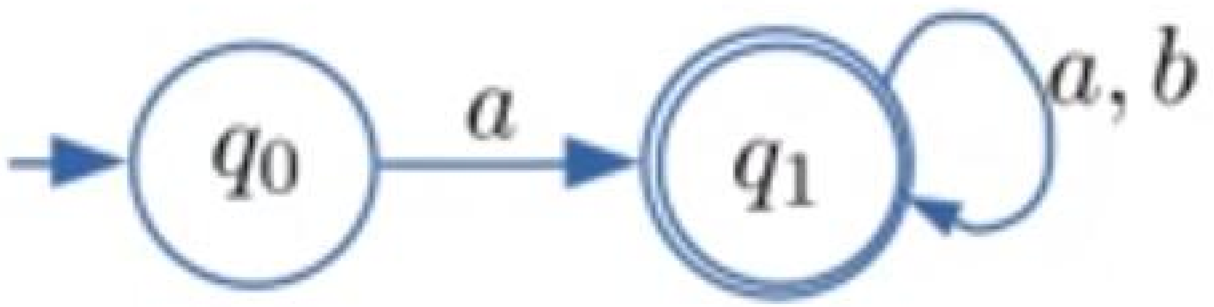


Partial Transition Function

Sometimes we consider the transition function $\delta : Q \times \Sigma \to Q$ as a partial function

That is, we allow $\delta(q, a)$ to be **undefined** for some $(q, a) \in Q \times \Sigma$

Consider the alphabet $\Sigma = \{a, b\}$ and the DFA M



Here $\delta(q_0, b)$ is undefined

We say there is **no valid move** for input symbol b from state q_0

What happens on input string baa **?**

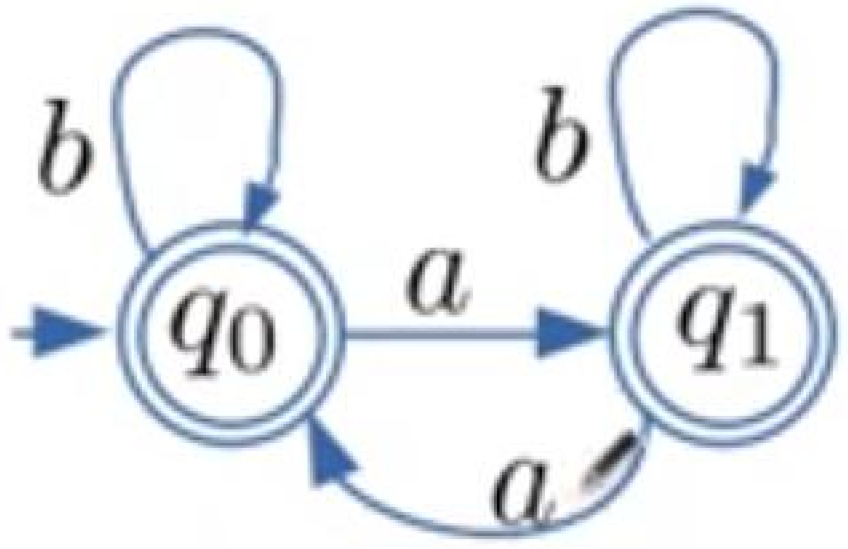
While processing the input string, if there is **no valid move** we **reject** **the input string**.

So, baa is **rejected** i.e. **not accepted**.

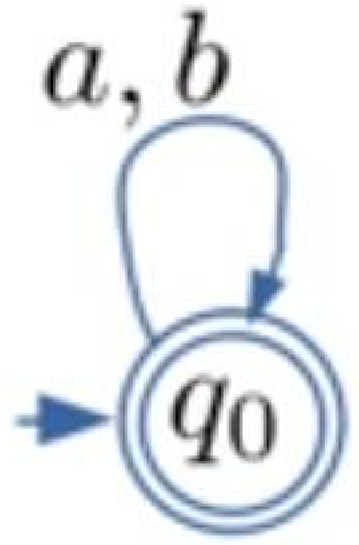
According to this convention for the given DFA M. $L(M) = \{aw \mid w \in \Sigma^*\}$

Examples of DFA (Contd.)

$M =$



$M' =$



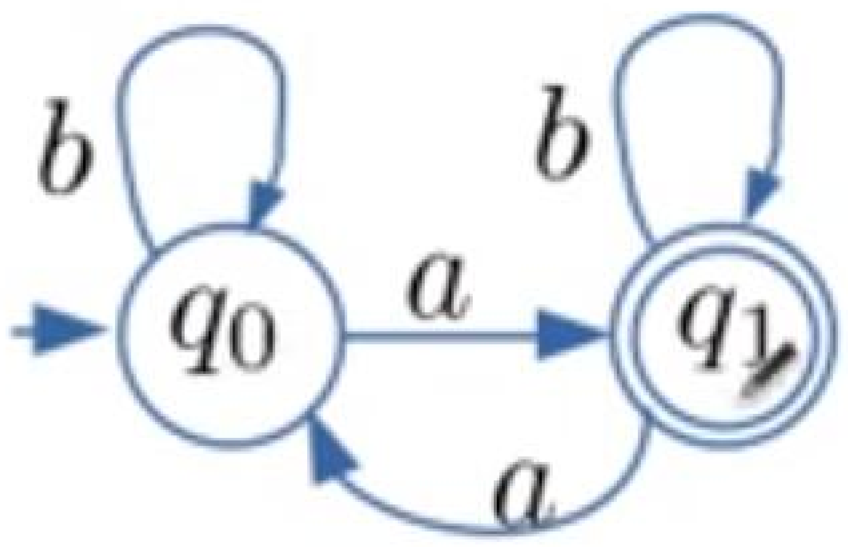
$$L(M) = \{a, b\}^*$$

$$L(M') = \{a, b\}^*$$

Two different automata accepting the same language

Examples of DFA (Contd.)

Example 4

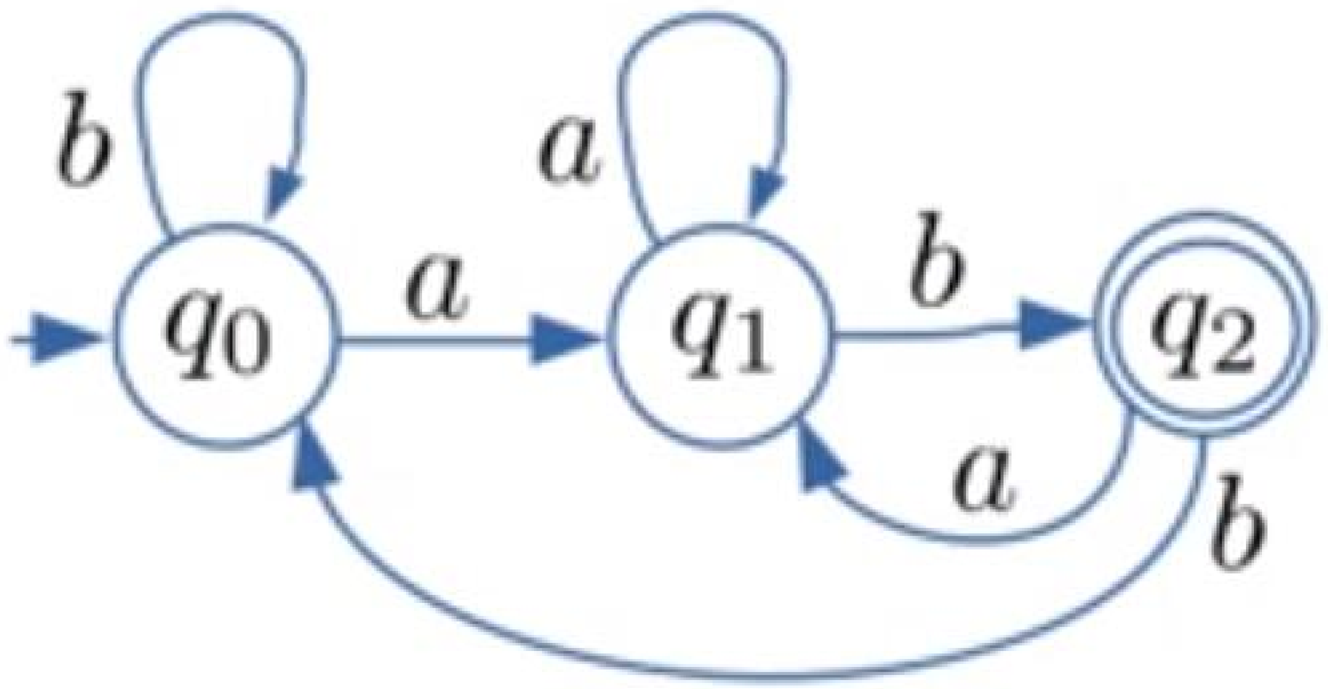


$$\Sigma = \{a, b\}$$

$$L(M) = \{w \in \{a,b\}^* \mid w \text{ contains odd no. of } a\text{s}\}$$

(Can you prove this ?)

Example 5



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

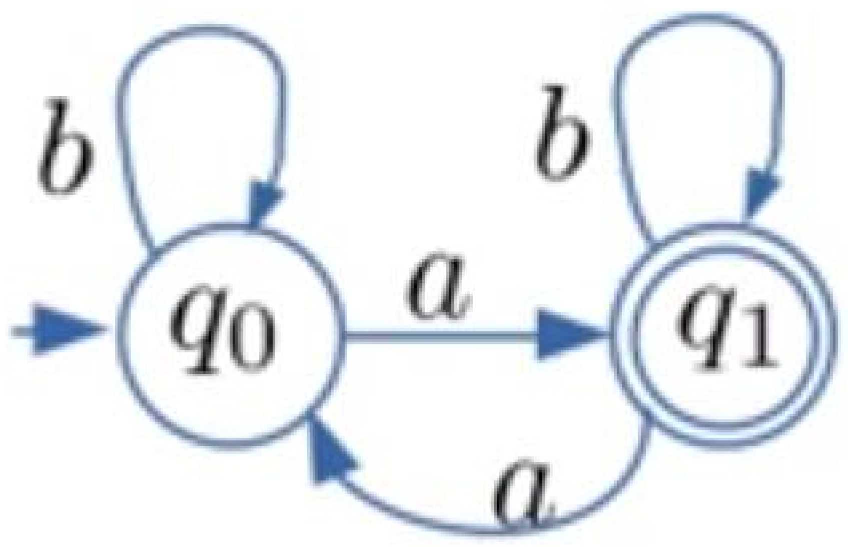
$$L(M) = \{wab \mid w \in \{a,b\}^*\}$$

i.e. Set of all strings ending with ab

(Can you prove this ?)

Examples of DFA (Contd.)

Example 4

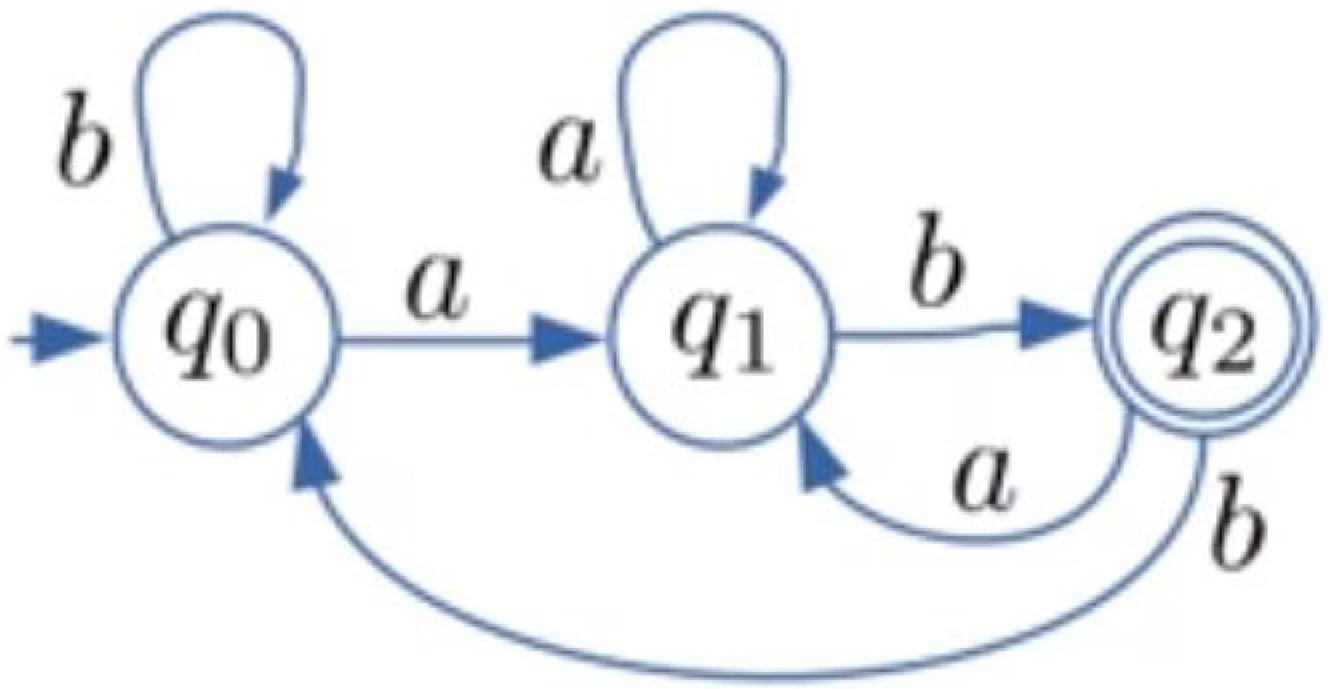


$$\Sigma = \{a, b\}$$

$$L(M) = \{w \in \{a,b\}^* \mid w \text{ contains odd no. of 1s}\}$$

(Can you prove this ?)

Example 5



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

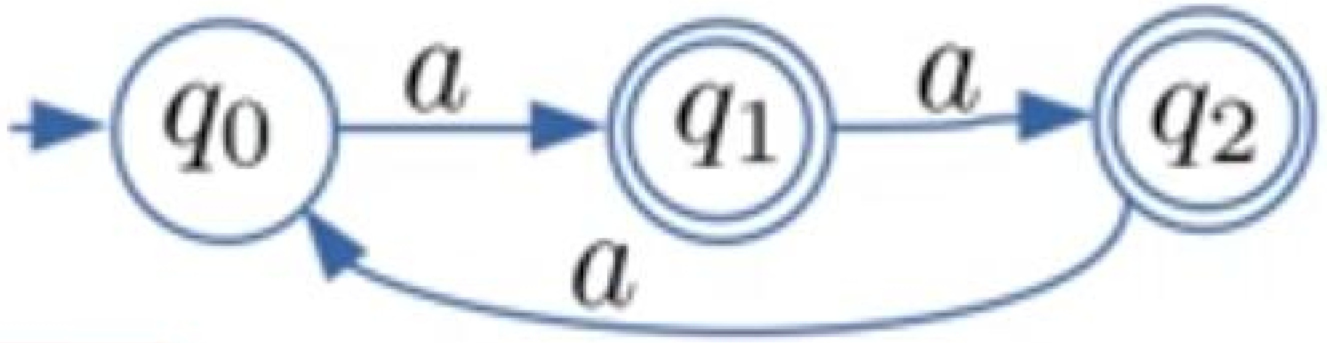
$$L(M) = \{wab \mid w \in \{a,b\}^*\}$$

i.e. Set of all strings ending with ab

(Can you prove this ?)

Examples of DFA

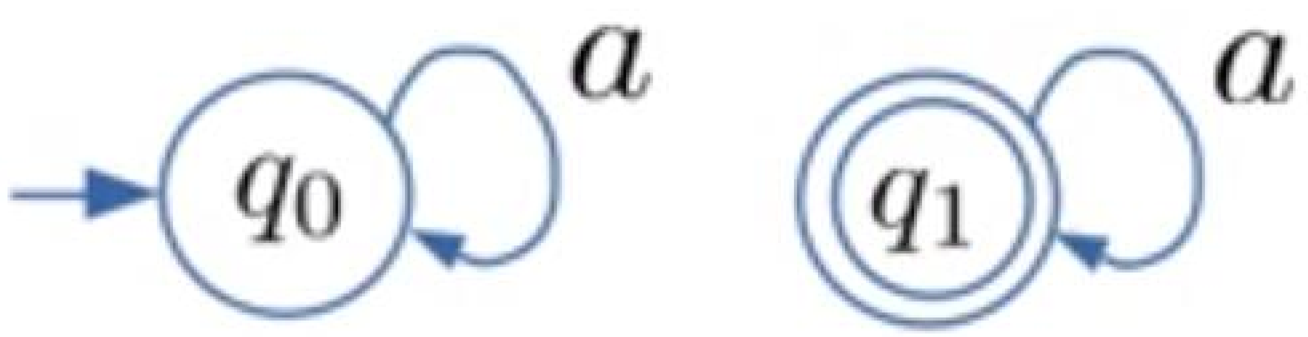
Example 1



$F = \{q_1, q_2\}$ **This DFA has more than one final states.**

$$L(M) = \{a^n \mid 3 \text{ does not divide } n\}$$

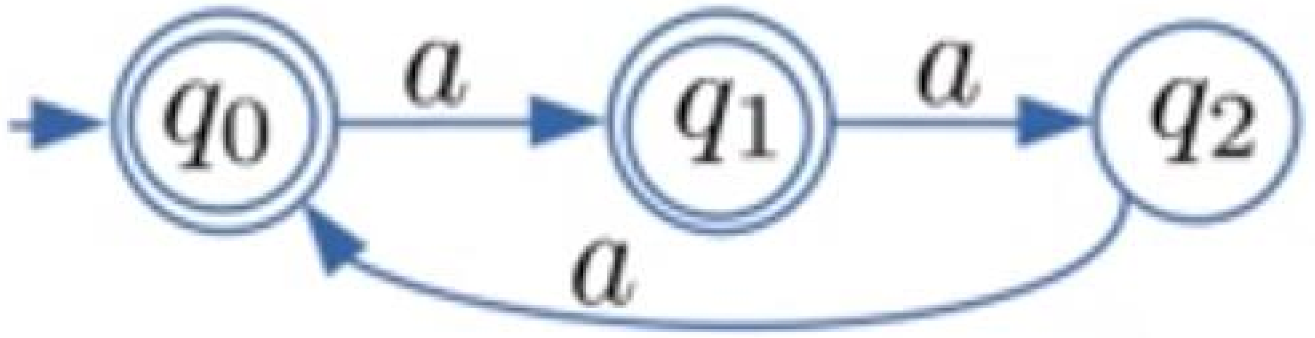
Example 2



No path from starting to final state.

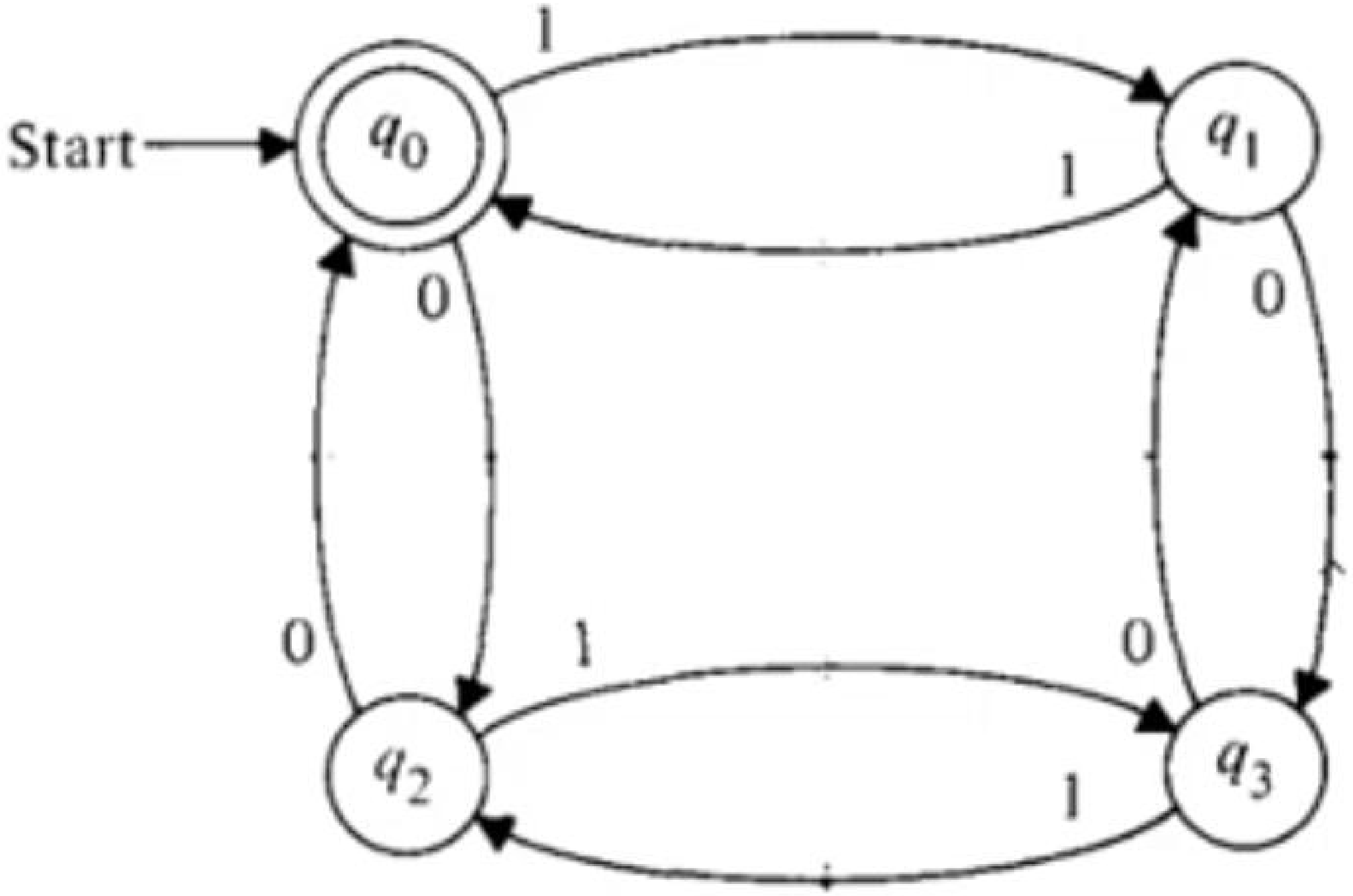
$L(M) = \phi$

Example 3



The starting state can also be a final state.

$$L(M) = \{a^n \mid n \not\equiv 2 \mod 3\}$$



$$\hat{\delta}(q_0, 1) = q_1$$

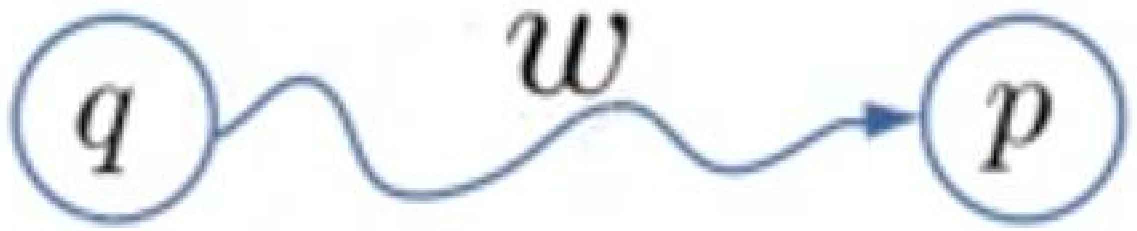
$$\hat{\delta}(q_0, 11) = \delta(\hat{\delta}(q_0, 1), 1) = \delta(q_1, 1) = q_0$$

$$\hat{\delta}(q_0, 110) = \delta(\hat{\delta}(q_0, 11), 0) = \delta(q_0, 0) = q_2$$

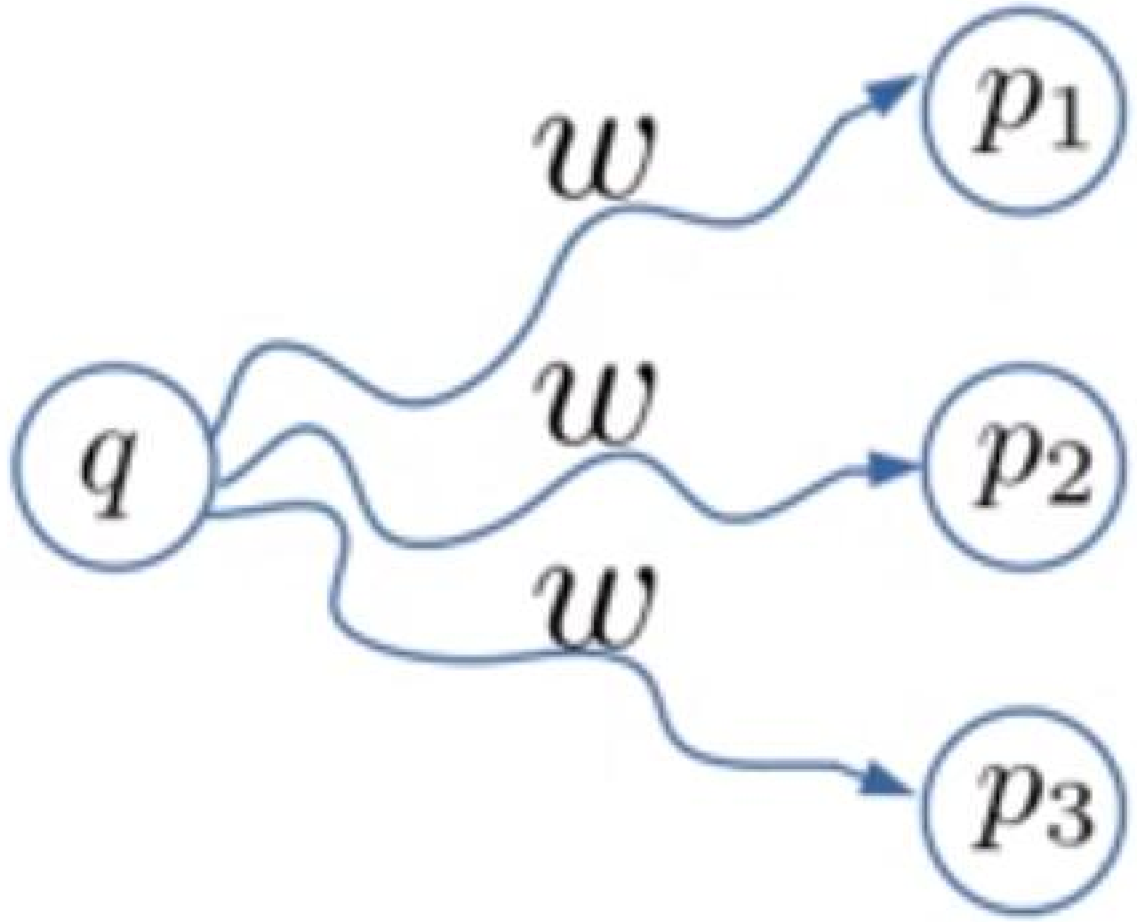
$$\hat{\delta}(q_0, 1101) = q_3, \qquad \hat{\delta}(q_0, 11010) = q_1 \qquad \hat{\delta}(q_0, \underset{\text{ACCEPTED}}{\underline{110101}}) = q_0 \in F$$

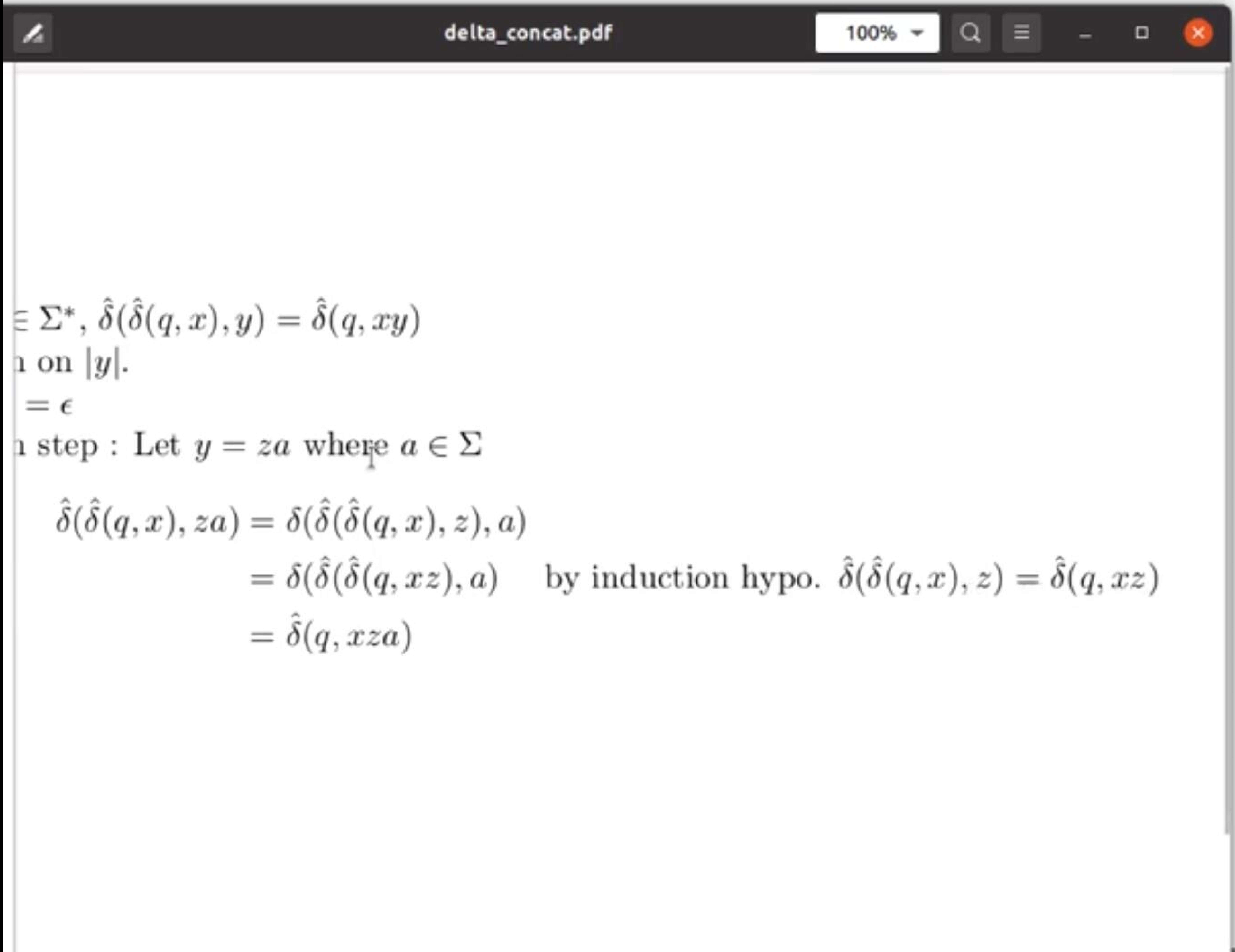
A Path In Transition Diagram Corresponds To A String

For DFAs, this path is unique (for a given string w and current state q)



For NFAs, there can be multiple paths (for a given string w and current state q)





$\in \Sigma^*$, $\hat{\delta}(\hat{\delta}(q,x),y) = \hat{\delta}(q,xy)$

n on $|y|$.

$= \epsilon$

n step : Let $y = za$ where $a \in \Sigma$

$$\begin{aligned} \hat{\delta}(\hat{\delta}(q,x),za) &= \delta(\hat{\delta}(\hat{\delta}(q,x),z),a) \\ &= \delta(\hat{\delta}(\hat{\delta}(q,xz),a) \quad \text{by induction hypo. } \hat{\delta}(\hat{\delta}(q,x),z) = \hat{\delta}(q,xz) \\ &= \hat{\delta}(q,xza) \end{aligned}$$

ToC06 DFA (Continued)
delta_concat.pdf
100%
18:07 / 34:29

For $x, y \in \Sigma^*$, $\hat{\delta}(\hat{\delta}(q, x), y) = \hat{\delta}(q, xy)$

Induction on $|y|$.

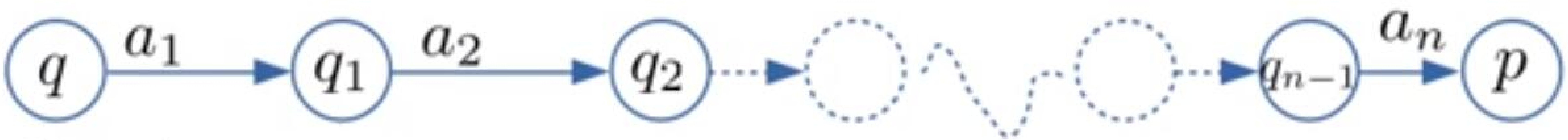
Basis : $y = \epsilon$

Induction step : Let $y = za$ where $a \in \Sigma$

$$\begin{aligned}\hat{\delta}(\hat{\delta}(q,x), za) &= \delta(\hat{\delta}(\hat{\delta}(q,x), z), a)\\ &= \delta(\hat{\delta}(\hat{\delta}(q,xz), a) \quad \text{by induction hypo. } \hat{\delta}(\\ &= \hat{\delta}(q, xza)\end{aligned}$$

A Path In Transition Diagram Corresponds To A String

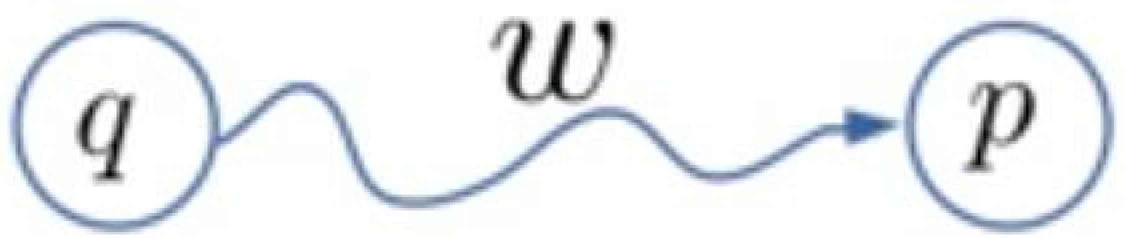
$$w = a_1 a_2 \cdots a_n$$



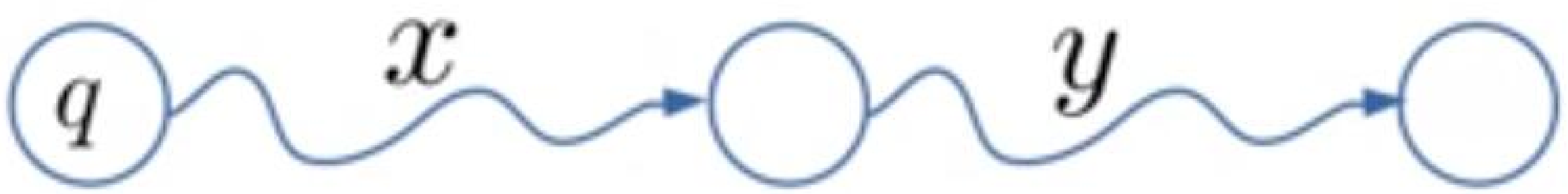
$\delta(q, a_1) = q_1$ $\delta(q_1, a_2) = q_2$ $\delta(q_{i-1}, a_i) = q_i$ $\delta(q_{n-1}, a_n) = p$

Notice that a_is are not necessarily distinct. p, q and q_is are not necessarily distinct

From now on, we shall simply draw such a path as a `wiggly line' labelled by the string



For **DFA** we have , $\forall x, y \in \Sigma^* \quad \hat{\delta}(\hat{\delta}(q, x), y) = \hat{\delta}(q, xy)$



Acceptance Criteria For DFA

Extended transition function

$\hat{\delta} : Q \times \Sigma^* \rightarrow Q$ is defined as,

For all state $q \in Q$
all string $w \in \Sigma^*$
and all symbol $a \in \Sigma$

$$\hat{\delta}(q, \epsilon) = q$$

$$\hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a)$$

Acceptance Criteria : An input string $w \in \Sigma^*$ is accepted if and only if $\hat{\delta}(q_0, w) \in F$

Language Accepted By DFA

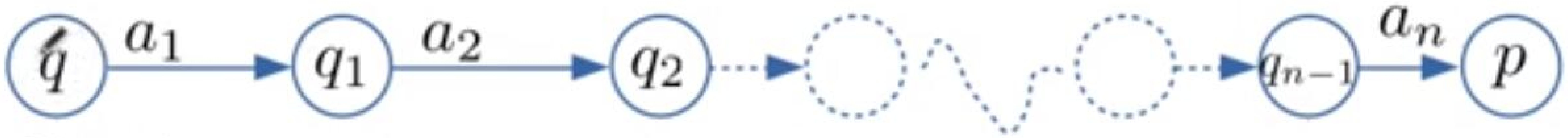
Recall that the language accepted by an automata is the set of all strings that are accepted.

For an automata M we generally denote the accepted language by $\underline{L(M)}$

So, for a DFA M, $L(M) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F\}$

Extended Transition Function

In a DFA, for all state $q \in Q$, a string $w = a_1 a_2 \cdots a_n$ (a_is are symbols) corresponds to a unique path in the transition diagram starting from q.

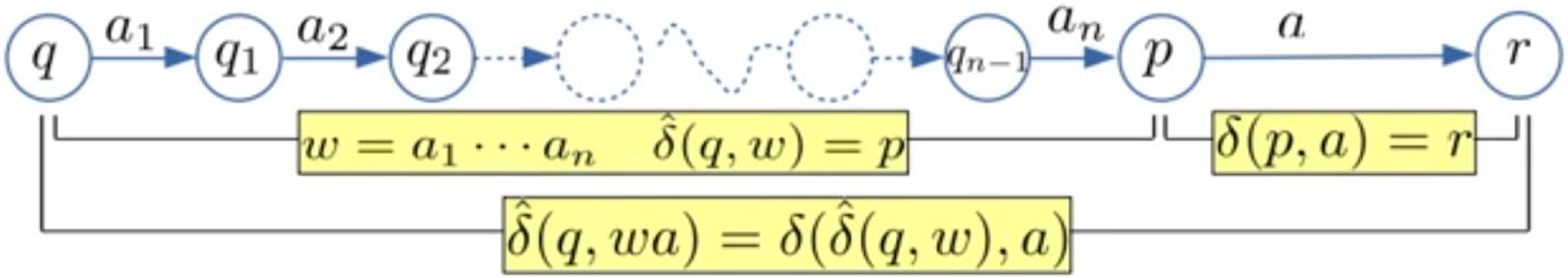


$\delta(q, a_1) = q_1$ $\delta(q_1, a_2) = q_2$ $\delta(q_{i-1}, a_i) = q_i$ $\delta(q_{n-1}, a_n) = p$

Notice that a_is are not necessarily distinct. p, q and q_is are not necessarily distinct

We capture this chain of transitions by the extended transition function

$$\hat{\delta} : Q \times \Sigma^* \to Q \quad \text{such that} \quad \hat{\delta}(q, a_1 \cdots a_n) = p$$



$w = a_1 \cdots a_n \quad \hat{\delta}(q, w) = p$

$\delta(p, a) = r$

$\hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a)$

Definition of Deterministic Finite Automata

In general this is how a DFA is defined.

A Deterministic Finite Automata is a 5-tuple $\underline{M = (Q, \Sigma, \delta, q_0, F)}$ where,

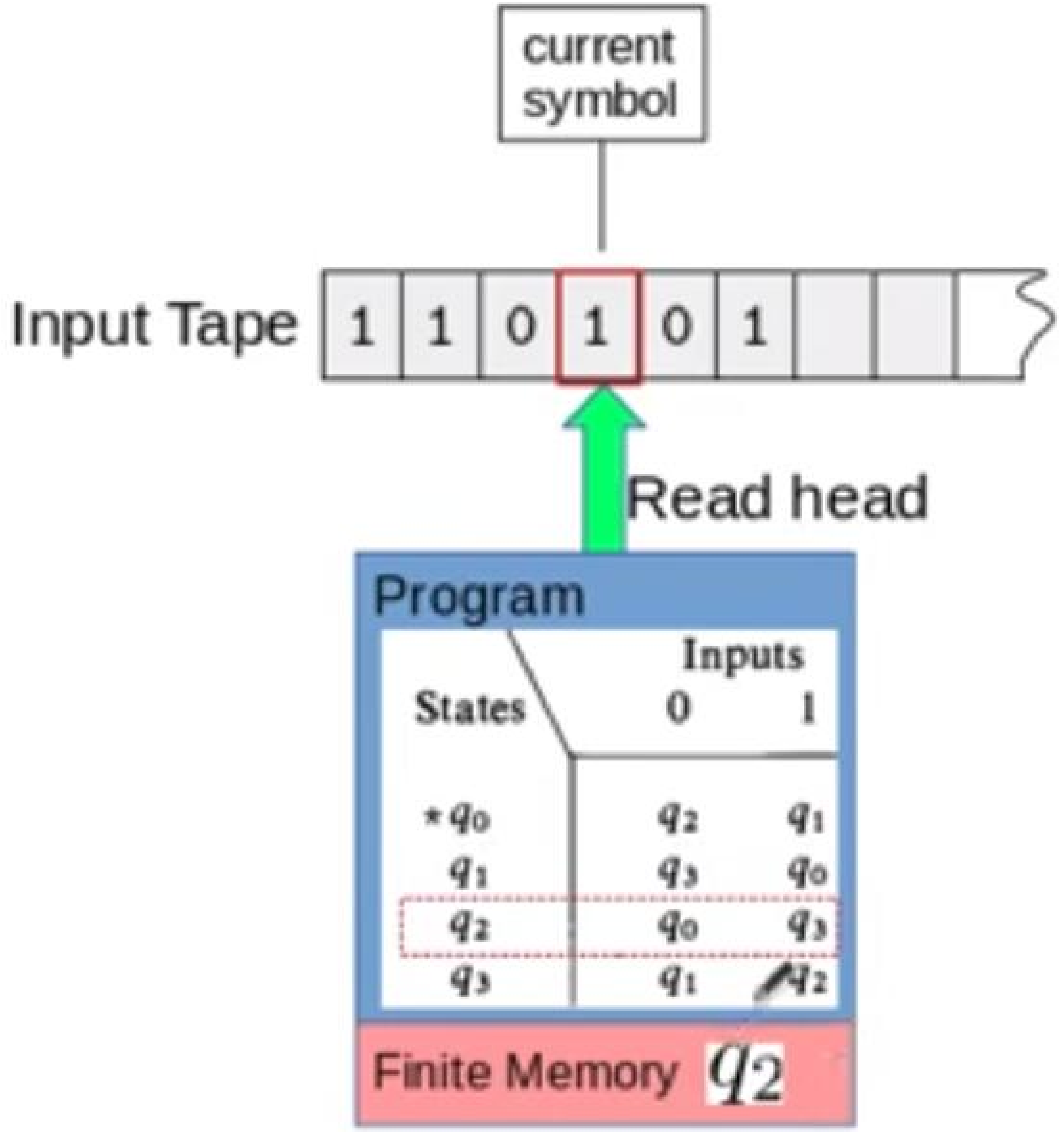
Q is a non-empty finite set known as the set of states.

Σ is a non-empty finite set known as the input alphabet.

$\delta : Q \times \Sigma \to Q$ is a function known as the transition function.

$q_0 \in Q$ is a state known as the starting state.

$F \subset Q$ is a subset of states known as the set of final states



States	Inputs 0	Inputs 1
$\star q_0$	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

Example 2.2 Consider the transition diagram of Fig. 2.2 again. In our formal notation this FA is denoted $M = (Q, \Sigma, \delta, q_0, F)$, where $Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{0, 1\}$, $F = \{q_0\}$, and δ is shown in Fig. 2.4.

	Inputs	
States	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

Fig. 2.4 $\delta(q, a)$ for the FA of Fig. 2.2.

→ denotes the starting state

* denotes final states

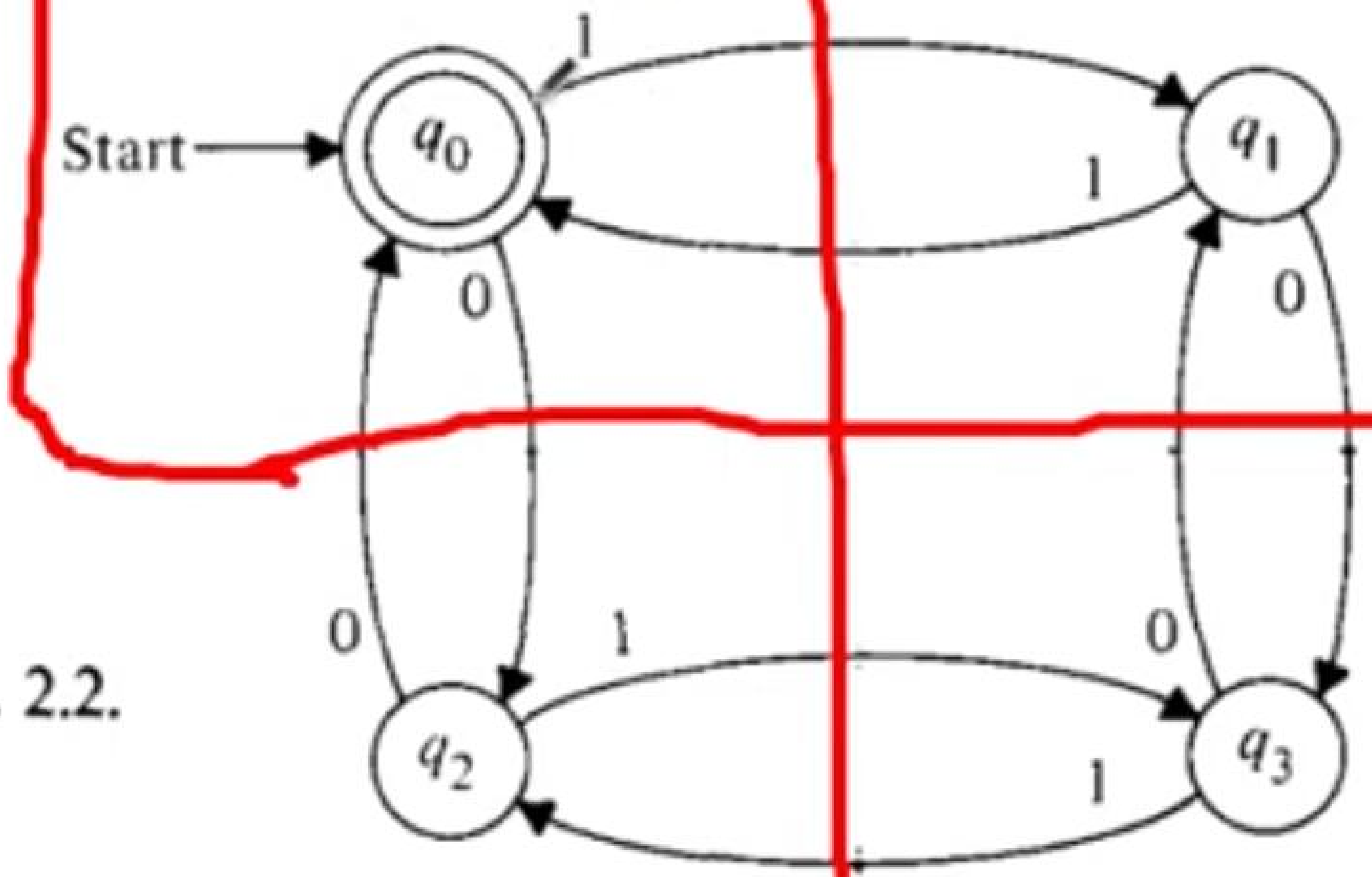
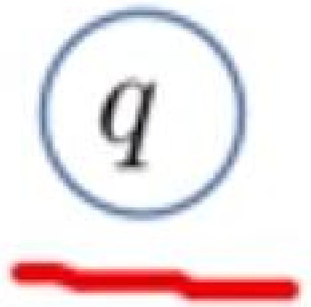


Fig. 2.2 The transition diagram of a finite automaton.

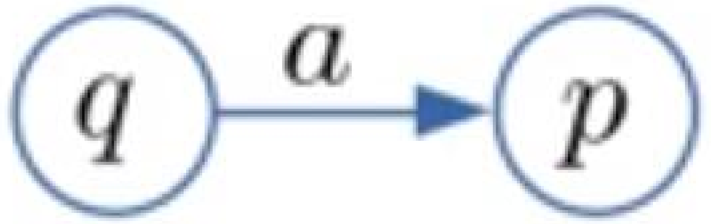
Convention for Drawing Transition Diagram

- A DFA is often depicted as a labelled directed graph, known as transition diagram.

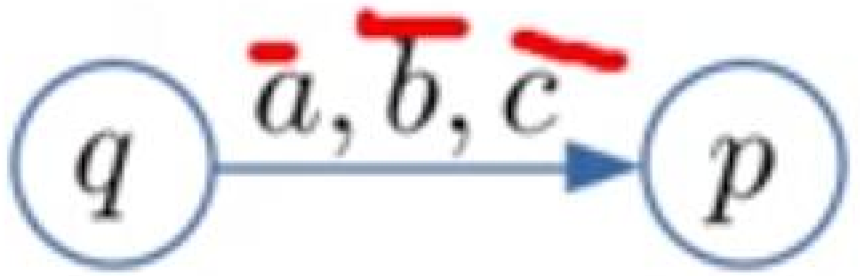
- All the states are shown as vertices.
  It is drawn as a circle labelled by the name of the state.



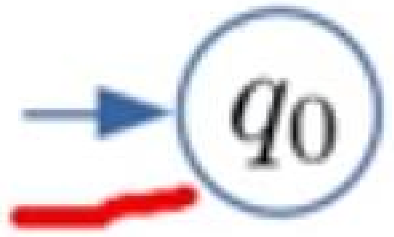
- A transition $\delta(q, a) = p$ is represented as an directed edge from vertex q to p and is labelled by the symbol a



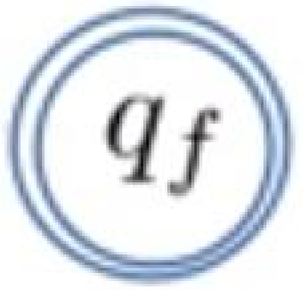
- An edge can be labelled with multiple symbols like.
  It means,
  $$\delta(q, a) = p$$
  $$\delta(q, b) = p$$
  $$\delta(q, c) = p$$



- The vertex for the starting state is marked with an extra incoming arrow.



- Vertices for all the final states are marked with double circles



Definition of Deterministic Finite Automata

In general this is how a DFA is defined.

A Deterministic Finite Automata is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$ where,

Q is a non-empty finite set known as the set of states.

Σ is a non-empty finite set known as the input alphabet.

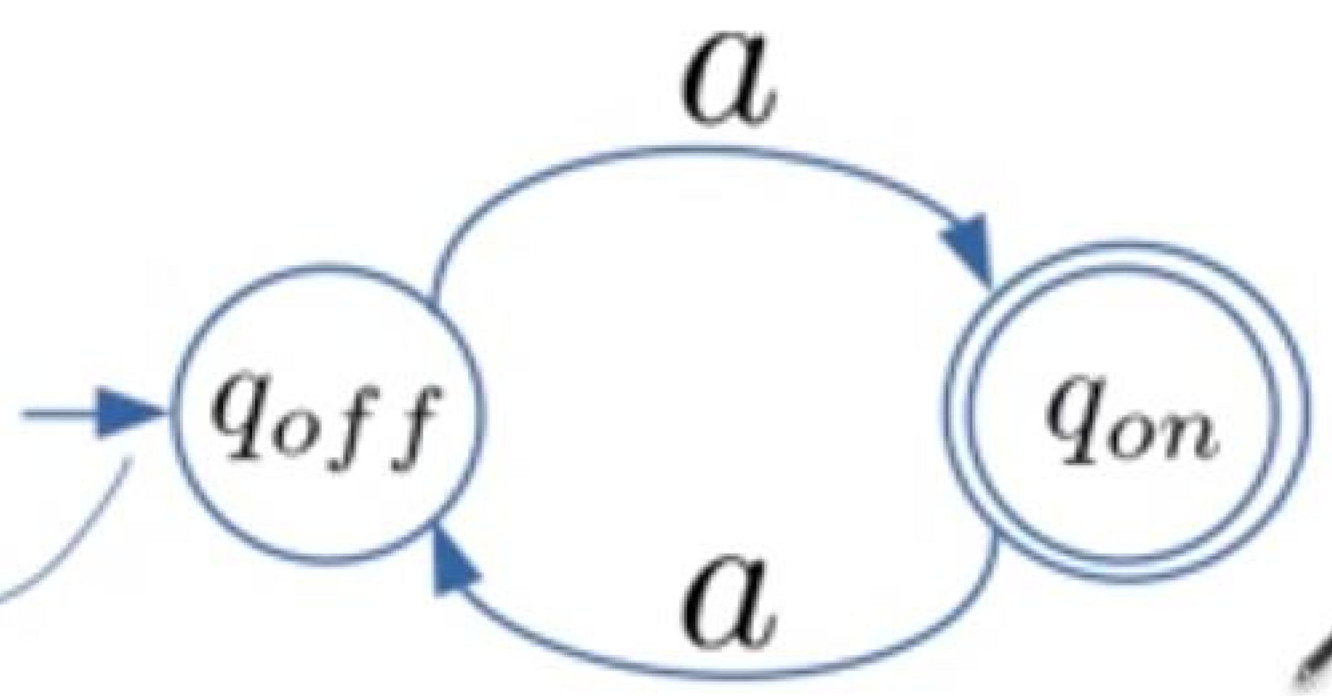
$\delta : Q \times \Sigma \rightarrow Q$ is a function known as the transition function.

$q_0 \in Q$ is a state known as the starting state.

$F \subset Q$ is a subset of states known as the set of final states

There can be **more than one** final states.

Transition Diagram of our automata



Most often a finite automata is expressed as a labelled directed graph.
This is known as the transition diagram of the automata.

Vertices are states.

Edges are labelled with input symbols which depict the transitions between states.

An extra arrow is used to mark the starting state.

Each final state is marked with double circle.

ToC05 Deterministic Finite Automata
28:47 / 40:06

The Mathematical Description Of Our Automata

Our automata is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$

where, $Q = \{q_{on}, q_{off}\}$ is the set of states which is a non-empty finite set.

$\Sigma = \{a\}$ input alphabet which is a non-empty finite set.

$\delta : Q \times \Sigma \to Q$ is the transition function.

defined as, $\delta(q_{off}, a) = q_{on}, \delta(q_{on}, a) = q_{off}$

$q_0 = q_{off} \in Q$ is the starting start

$F = \{q_{on}\} \subset Q$ is the set of final states

Acceptance Criteria : An input string $w \in \Sigma^*$ is accepted if and only if $\hat{\delta}(q_0, w) \in F$

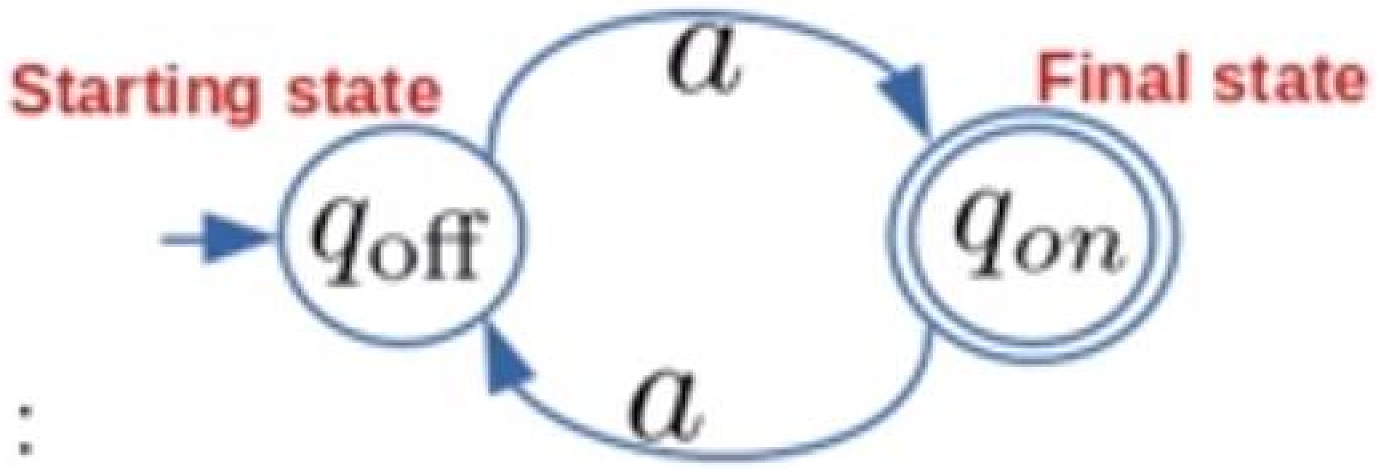
Let us solve a computational problem

Problem : Given any natural number n decide whether n is odd.

Encoding the input : Consider the **alphabet** $\Sigma = \{a\}$

Let us encode the natural number n as the string a^n

The **language** corresponding to our problem is $\{a^n \mid n \text{ is odd } \}$



Solution strategy :

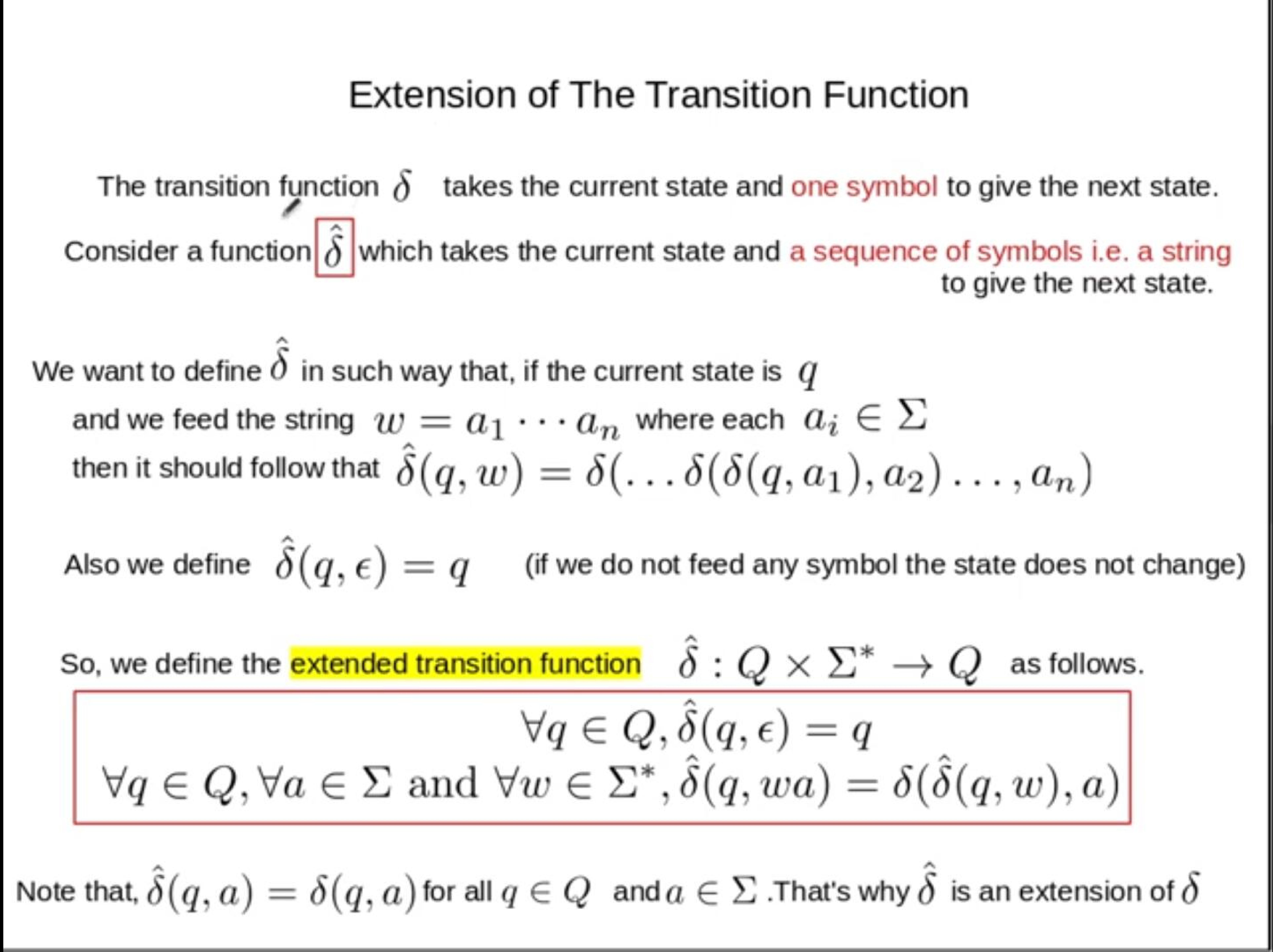
- Start the `machine' in q_{off} state. We call it the **starting state**.
- We apply the input string a^n. The machine processes the input one symbol at a time.
- On `reading' each input symbol, the machine changes states

  as dictated by the **transition function**. δ
- After reading all the input symbol if the machine is in q_{on} state then we say YES.

  i.e. we consider that the **input string is accepted**.

  we call this state a **final state**.

In our example we have only one final state. It is not necessarily true in general. That is why we must consider the **set of final states** which can be any subset of **the set of states** .



Extension of The Transition Function

The transition function δ takes the current state and one symbol to give the next state.

Consider a function $\boxed{\hat{\delta}}$ which takes the current state and a sequence of symbols i.e. a string to give the next state.

We want to define $\hat{\delta}$ in such way that, if the current state is q and we feed the string $w = a_1 \cdots a_n$ where each $a_i \in \Sigma$ then it should follow that $\hat{\delta}(q, w) = \delta(\ldots \delta(\delta(q, a_1), a_2) \ldots, a_n)$

Also we define $\hat{\delta}(q, \epsilon) = q$ (if we do not feed any symbol the state does not change)

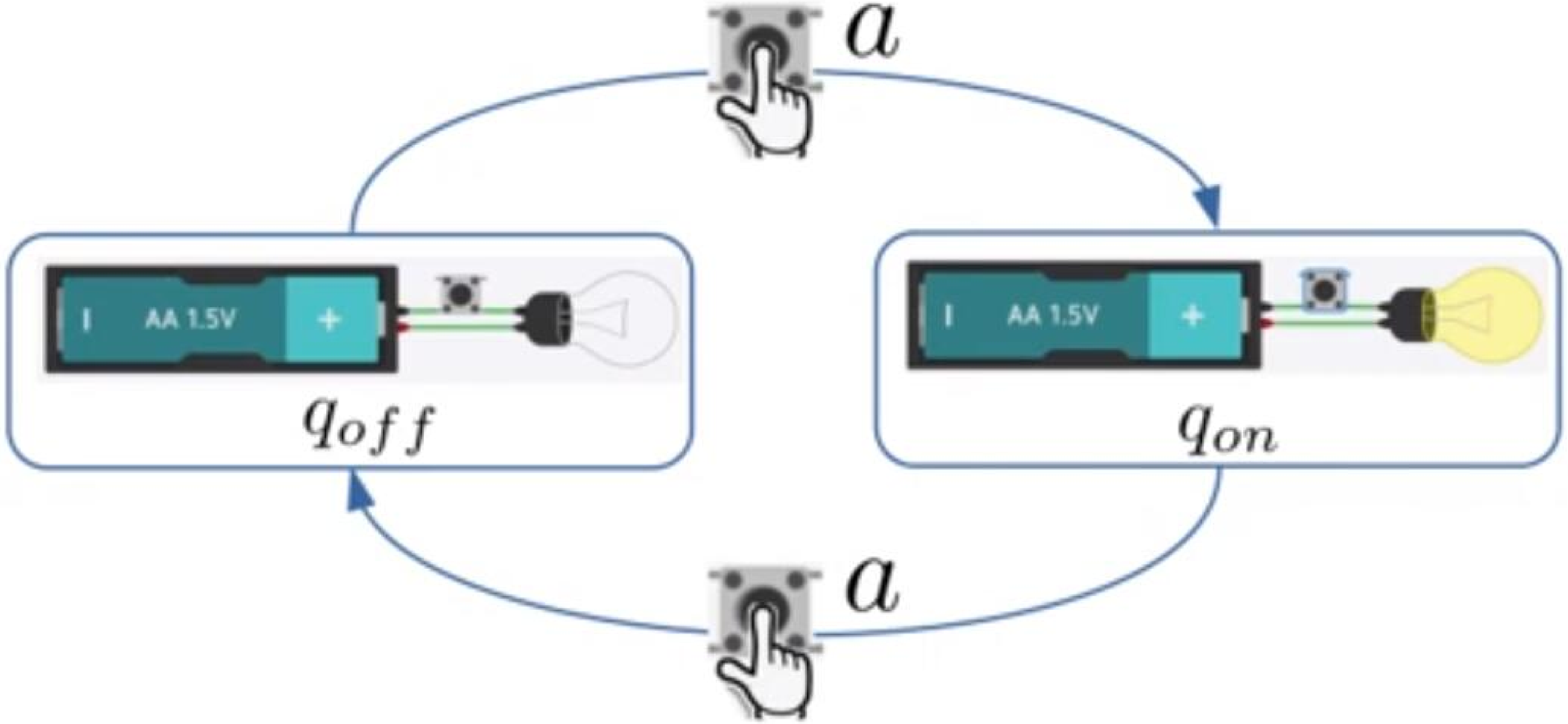
So, we define the extended transition function $\hat{\delta} : Q \times \Sigma^* \to Q$ as follows.

$$\boxed{\begin{array}{c} \forall q \in Q, \hat{\delta}(q, \epsilon) = q \\ \forall q \in Q, \forall a \in \Sigma \text{ and } \forall w \in \Sigma^*, \hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a) \end{array}}$$

Note that, $\hat{\delta}(q, a) = \delta(q, a)$ for all $q \in Q$ and $a \in \Sigma$ .That's why $\hat{\delta}$ is an extension of δ

Transition Between States

Pushing the switch toggles the state between on and off



We capture this behaviour by a **transition function** $\delta : Q \times \Sigma \to Q$

This function gives the `next state' depending on the `current state' and the `current input symbol'

$$\delta(q_{off}, a) = q_{on}$$
$$\delta(q_{on}, a) = q_{off}$$

Input to our machine

The only way to provide any input is to push the button.
We can push it multiple times.

We express the action of pushing the button once by a symbol a

To express the action of pushing the button twice we use the string aa

In general, to express the action of pushing the button i times we use the string a^i

The action of not pushing the button at all is expressed by the empty string $\epsilon = a^0$

The alphabet for the input strings is $\Sigma = \{a\}$

The set of all possible input is $\Sigma^* = \{\epsilon, a, aa, aaa, \ldots\}$

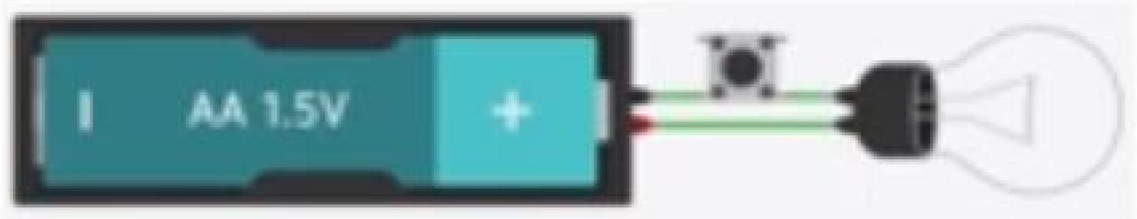
(Notice that the set of `states' $Q = \{q_{on}, q_{off}\}$ is finite but the set of inputs is infinite.)

Finite Automata

A **Finite Automata (FA)** is a mathematical model of a machine

The machine can be in any one of FINITELY many internal configurations or "states."

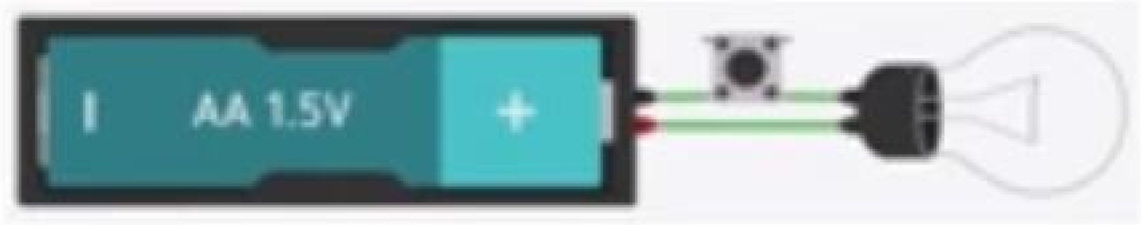
Example : Take the following circuit. How do we describe its configuration ?



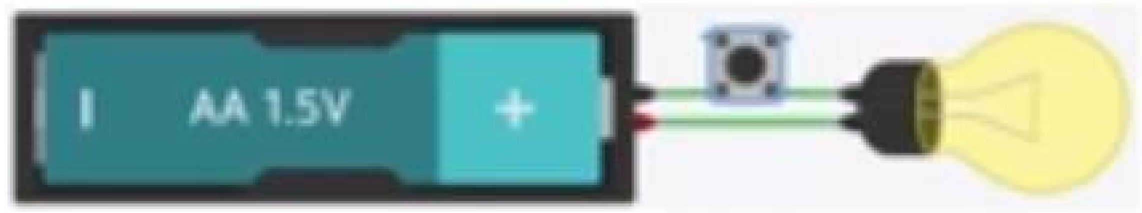
On one extreme we may consider all the information about all the electrons and quarks in the system !!!

However, suppose **we are only interested about whether the light is on or off**.

So, for our purpose the set of states is $Q = \{q_{on}, q_{off}\}$



q_{off}



q_{on}