CS 302.1 - Automata Theory

Lecture 02

Shantanav Chakraborty

Center for Quantum Science and Technology (CQST)
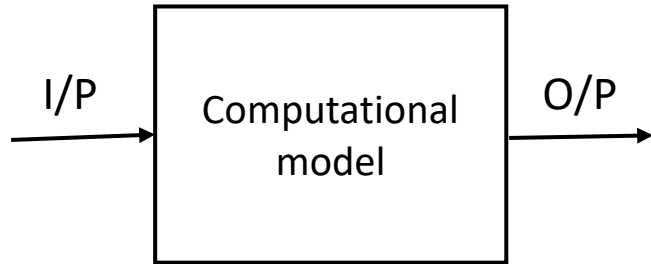
Center for Security, Theory and Algorithms (CSTAR)

IIIT Hyderabad

INTERNATIONAL INSTITUTE OF
INFORMATION TECHNOLOGY
H Y D E R A B A D

A quick recap

- Can a given problem be computed by a particular computational model?

A computational model solves a problem P if,

(i) For all inputs belonging to the YES instance of P, the device outputs **YES**

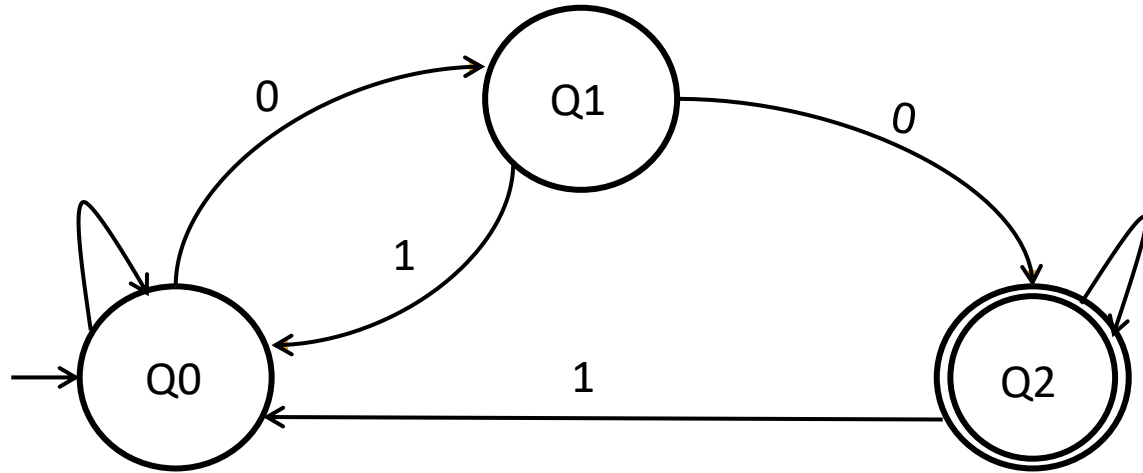(ii) For all inputs belonging to the NO instance of P, the device outputs **NO**.

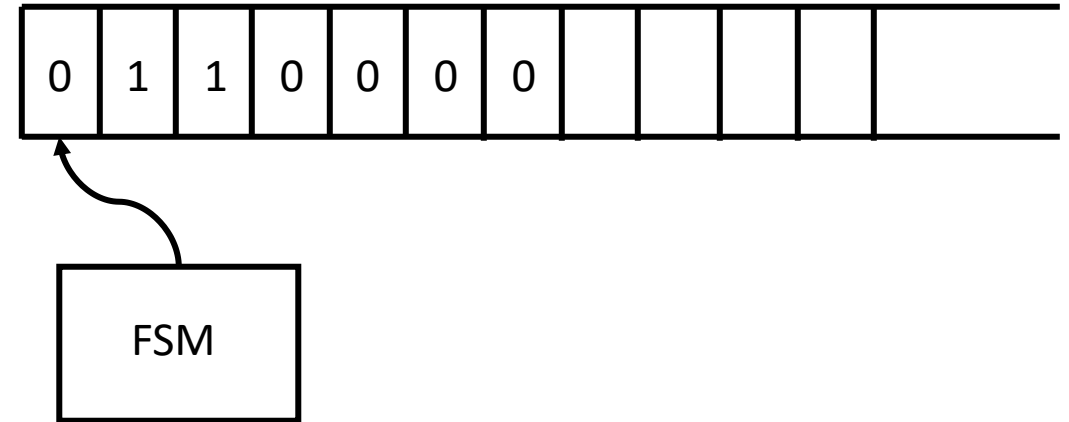If (i) and (ii) hold, we say that the problem **P is computable** by this computational model.
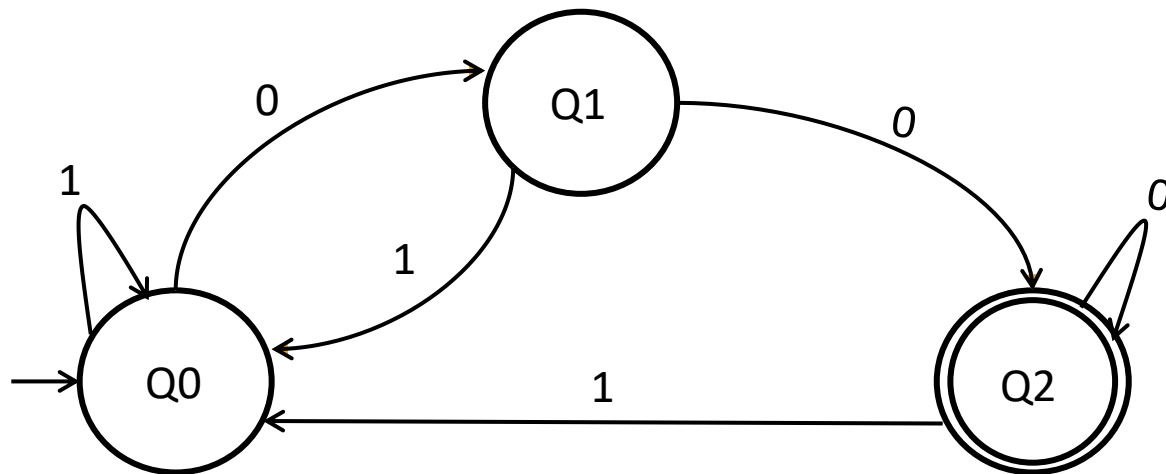
Deterministic Finite Automata (DFA)

Characteristics: (i) Single start State
(ii) Unique Transitions
(iii) Zero or more final states

A quick recap

Deterministic Finite Automata (DFA)

Run:

$$\boldsymbol{Q0} \xrightarrow{0} Q1 \xrightarrow{1} Q0 \xrightarrow{1} Q0 \xrightarrow{0} Q1 \xrightarrow{0} Q2 \xrightarrow{0} Q2 \xrightarrow{0} \boldsymbol{Q2}$$
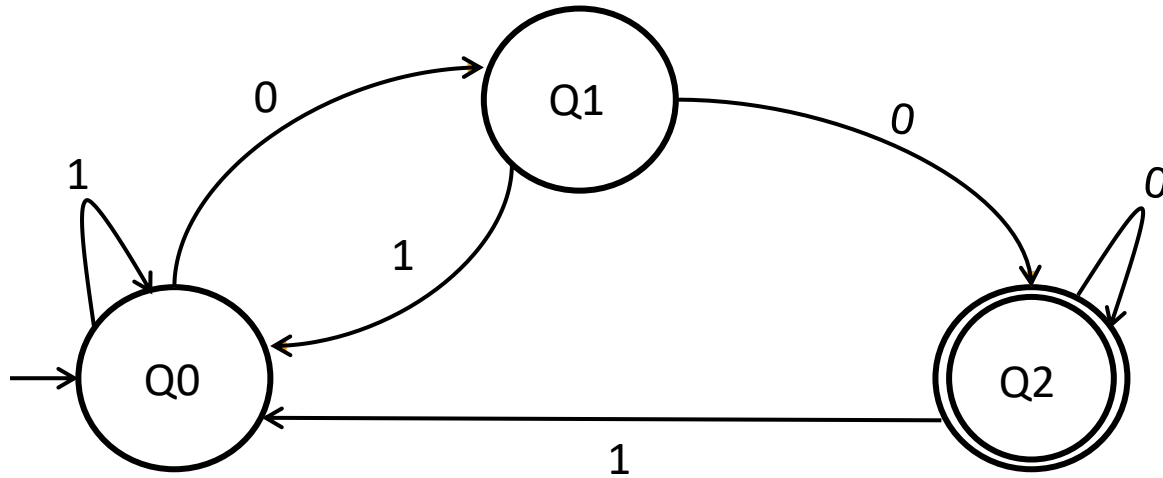
The DFA "accepts" an input string, if it corresponds to a *run* that ends up in the final state Q2. **(Accepting Run)**

The DFA "rejects" an input string, if it corresponds to a *run* that ends up in any non-final state. **(Rejecting Run)**
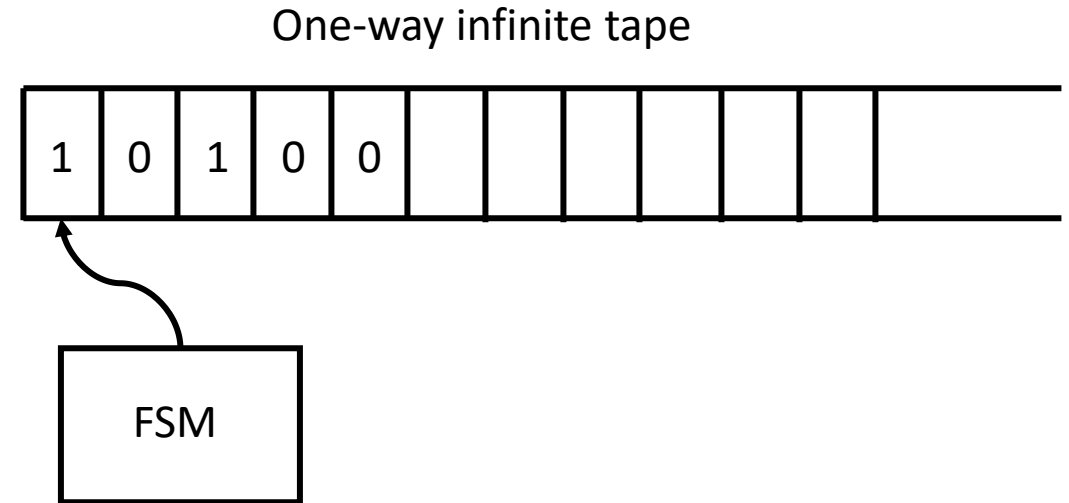
L(M) = {ω|ω results in an accepting run}

For the example above, **L(M) = {ω|ω ends in "00"}**

Deterministic Finite Automata (DFA)

One-way infinite tape

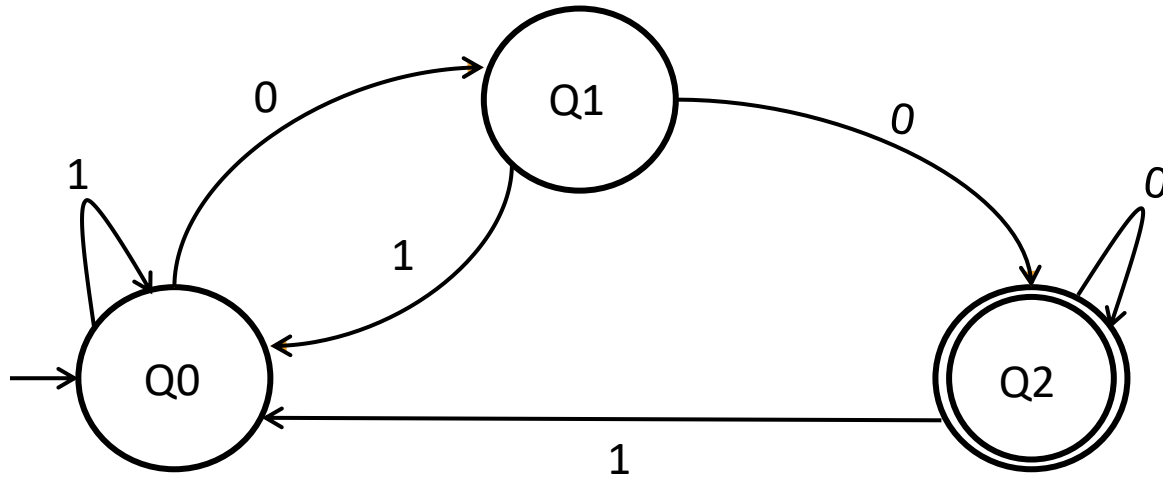State transition diagram of the Finite State Machine

For any language L, we say **M decides L or M solves L** if

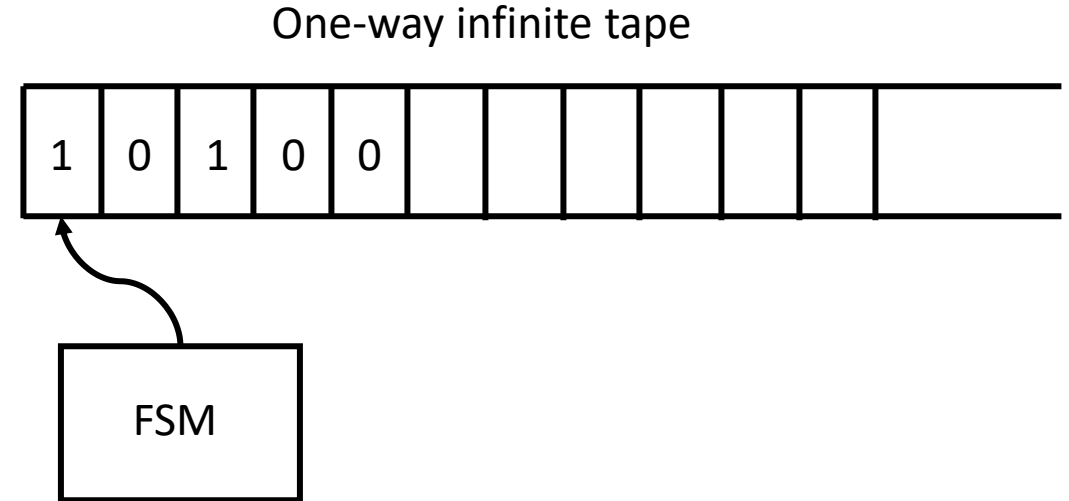$$\forall \omega \in L, M(\omega) \text{ accepts}$$
$$\forall \omega \notin L, M(\omega) \text{ rejects}$$

Deterministic Finite Automata (DFA)

0

Q1

0

1

1

1

0

Q0

Q2

1

State transition diagram of the Finite State Machine

One-way infinite tape

| 1 | 0 | 1 | 0 | 0 | | | | | | | | | |

FSM

Characteristics of DFA : (i) Single start state   (ii) Unique transitions   (iii) Zero or more final states

Formally, a finite automaton M is a 5-tuple ($Q, \Sigma, \delta, q_0, F$) where

- Q is a finite set called the **states.**
- Σ is a finite set called the **alphabet**.
- $\delta: Q \times \Sigma \mapsto Q$ is the **transition function** (unique).
- $q_0 \in Q$ is the **start state**.
- $F \subseteq Q$ are the **final/accepting states**.

Q = {Q0, Q1, Q2}

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

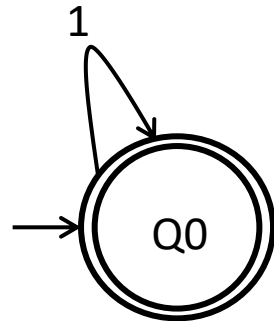$(Q0,0) \mapsto Q1; \ (Q0,1) \mapsto Q0,......,(Q2,1) \mapsto Q0$

$q_0 = Q0$

$F = Q2$

Constructing DFA for a language

Examples: $\Sigma = \{0, 1\}$, L(M)=$\{\omega|\omega$ has an even number of 0's$\}$

Constructing DFA for a language

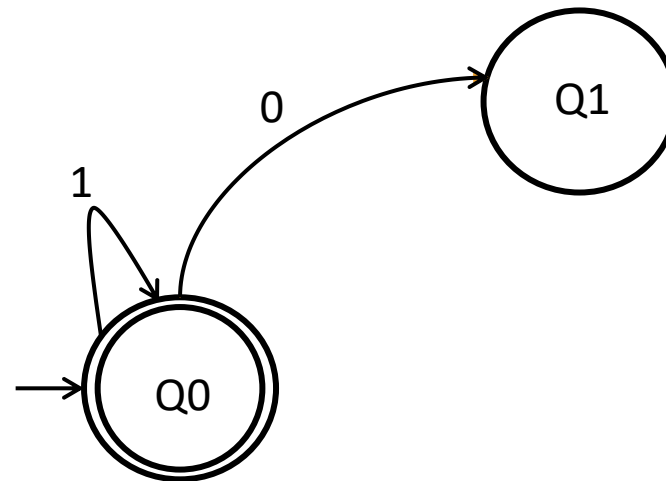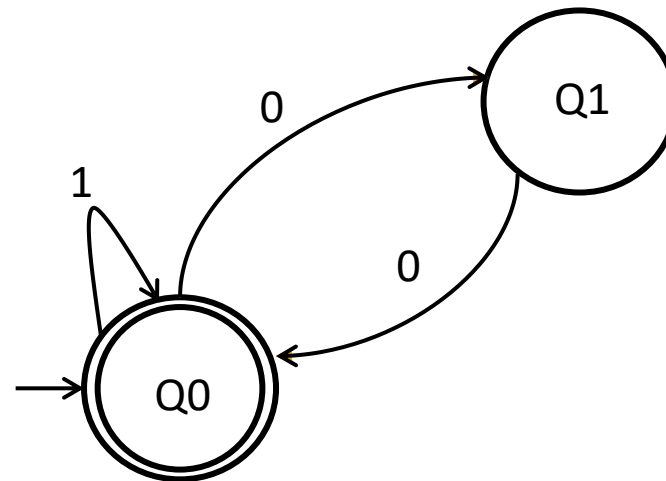Examples: $\Sigma = \{0, 1\}$, L(M)=$\{\omega | \omega$ has an even number of 0's$\}$

Constructing DFA for a language

Examples: $\Sigma = \{0, 1\}$, L(M)=$\{\omega | \omega$ has an even number of 0's$\}$

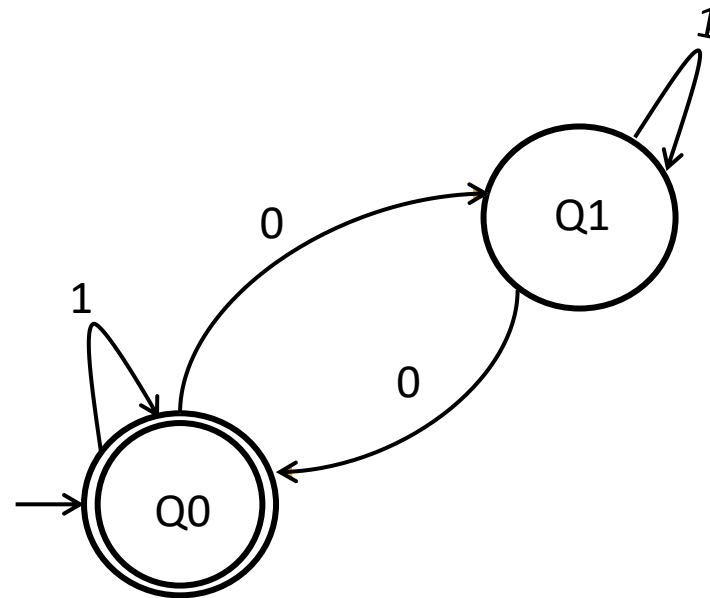# Constructing DFA for a language

Examples: $\Sigma = \{0, 1\}$, **L(M)={$\omega | \omega$ has an even number of 0's}**

Constructing DFA for a language

Examples: $\Sigma = \{0, 1\}$, L(M)=$\{\omega | \omega$ has an even number of 0's$\}$

	0	1
Q0	Q1	Q0
Q1	Q0	Q1

Constructing DFA for a language

Examples: $\Sigma = \{0, 1\}$, L(M)={$\omega | \omega$ is divisible by 3}

Any input string would leave three remainders: 0, 1 or 2.

Constructing DFA for a language

Examples: $\Sigma = \{0, 1\}$, **L(M)={ω|ω is divisible by 3}**

Any input string would leave three remainders: 0, 1 or 2.

Intuition: Let ω be any substring of the input string divisible by 3, i.e. $\omega = 0 (mod\ 3)$

$$\omega\ 0 = 2 \times value\ (\omega) = 0\ (mod\ 3)$$
$$\omega\ 1 = 2 \times value(\omega) + 1 = 1(mod\ 3)$$
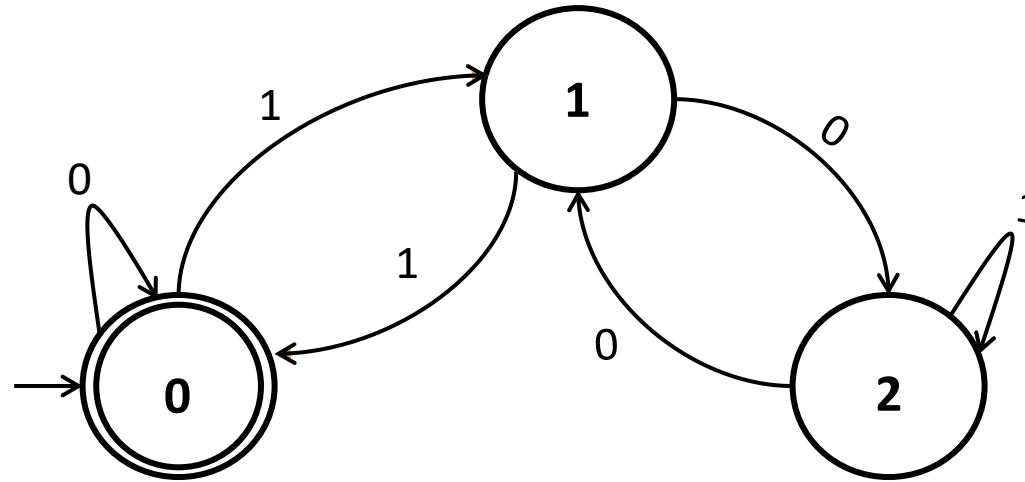$$\omega\ 10 = 2 \times value(\omega 1) = 2(mod\ 3)$$
$$\omega\ 11 = 2 \times value(\omega 1) + 1 = 0(mod\ 3)$$

…. And so on

- The DFA will have three states, each corresponding to the remainder of $value(\omega)/3$.

- The final state $= 0(mod\ 3)$ – the string ω is accepted if after reading it, the DFA ends in this state.

Constructing DFA for a language

Examples: $\Sigma = \{0, 1\}$, **L(M)=$\{\omega | \omega$ is divisible by 3$\}$**

Any input string would either leave remainders 0, 1 or 2.

Intuition: Let ω be any substring of the input string divisible by 3, i.e. $\omega = 0(mod\ 3)$

$$\omega\ 0 = 2 \times value\ (\omega) = 0\ (mod\ 3)$$
$$\omega\ 1 = 2 \times value(\omega) + 1 = 1(mod\ 3)$$
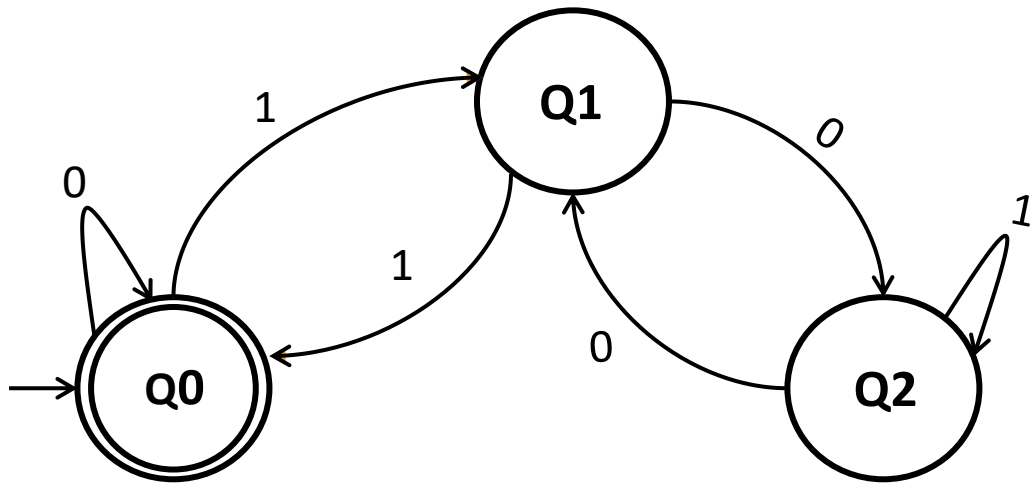$$\omega\ 10 = 2 \times value(\omega 1) = 2(mod\ 3)$$
$$\omega\ 11 = 2 \times value(\omega 1) + 1 = 0(mod\ 3)$$

…. And so on

Constructing DFA for a language

Examples: $\Sigma = \{0, 1\}$, L(M)=$\{\omega | \omega$ is divisible by 3$\}$

		0	1
Q0		Q0	Q1
Q1		Q2	Q0
Q2		Q1	Q2

Constructing DFA for a language

Examples: $\Sigma = \{0, 1\}$, **L(M)=$\{\omega | \omega$ is NOT divisible by 3\}**

Constructing DFA for a language

Examples: $\Sigma = \{0, 1\}$, **L(M)={ω|ω is NOT divisible by 3}**

Intuition - Construct a **Toggled DFA:** Toggle the final states and the non-final states!

Constructing DFA for a language

Examples: $\Sigma = \{0, 1\}$, **L(M)={ω|ω is NOT divisible by 3}**

Intuition - Construct a **Toggled DFA:** Toggle the final states and the non-final states!

In fact if any DFA accepts L, the toggled DFA accepts $\overline{L}$, the complement of L

Constructing DFA for a language

Examples: $\Sigma = \{0, 1\}$, **L(M)={$\omega | \omega$ is NOT divisible by 3}**

Intuition - Construct a **Toggled DFA:** Toggle the final states and the non-final states!

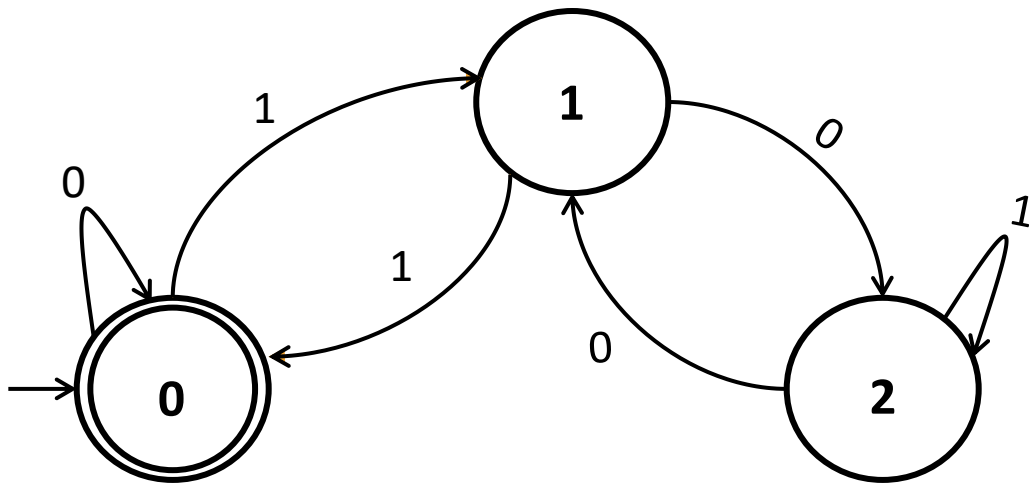**In fact if any DFA accepts L, the toggled DFA accepts $\overline{L}$, the complement of L**

Constructing DFA for a language

Examples: $\Sigma = \{0, 1\}$, **L(M)={ω|ω is NOT divisible by 3}**

Intuition - Construct a **Toggled DFA:** Toggle the final states and the non-final states!

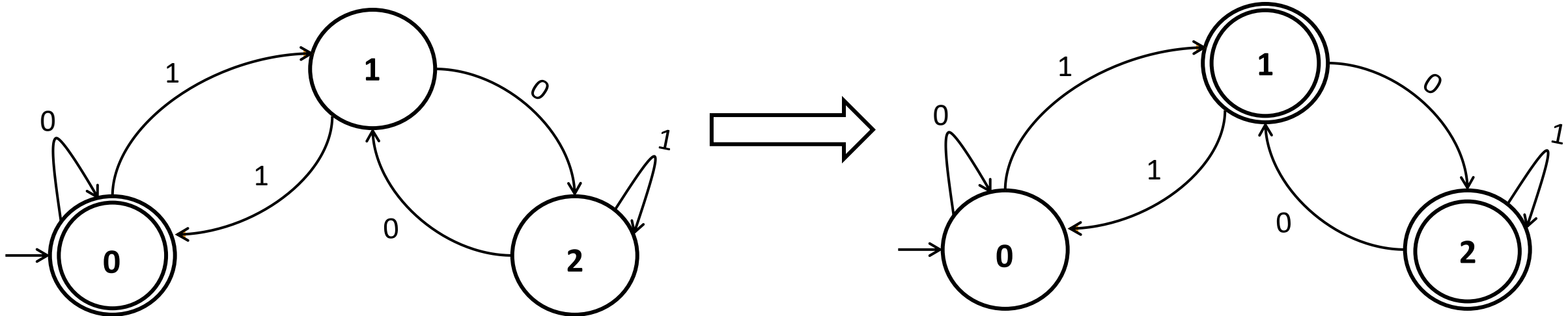**In fact if any DFA accepts L, the toggled DFA accepts $\overline{L}$, the complement of L**

Non-deterministic Finite Automata (NFA)

Characteristics of DFA : (i) Single start state   (ii) Unique transitions   (iii) Zero or more final states
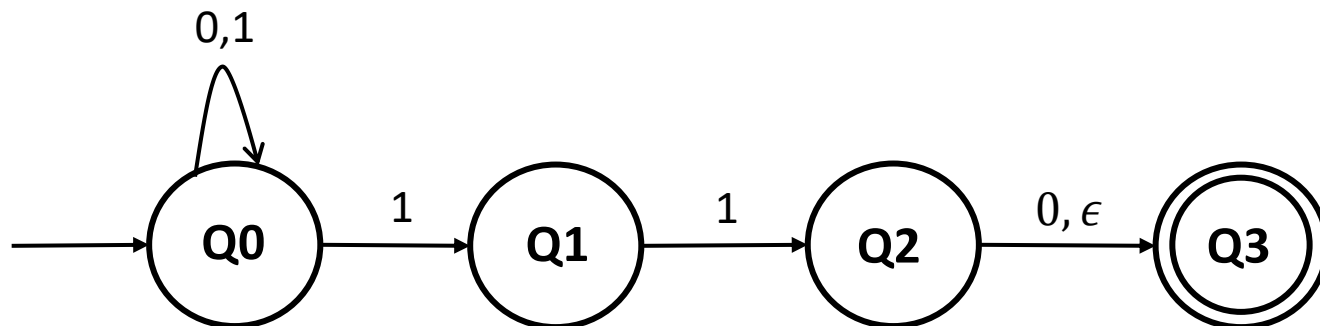
Non-deterministic Finite Automata (NFA)

Characteristics of DFA : (i) Single start state   (ii) Unique transitions   (iii) Zero or more final states

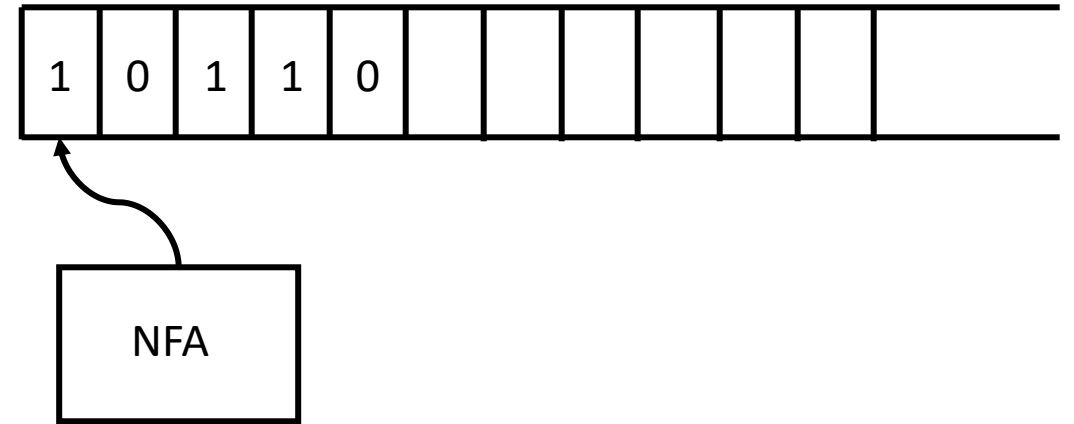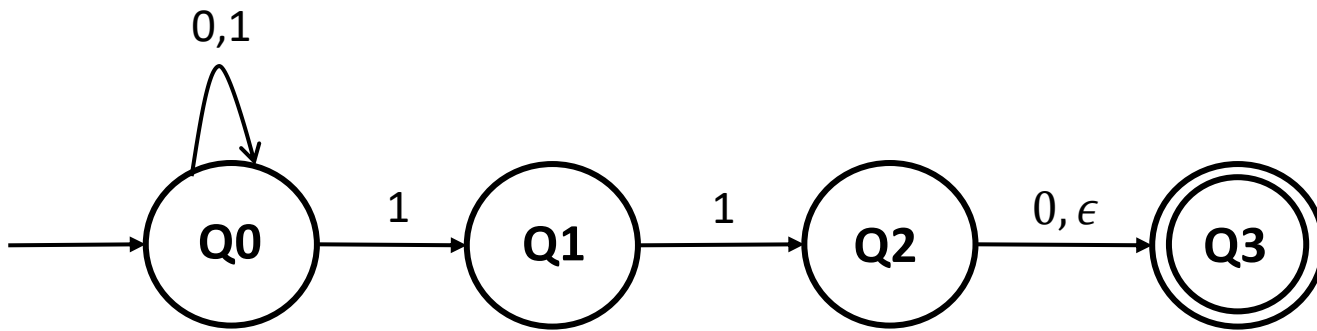**Characteristics of NFA** : (i) Single start state (ii) Zero or more final states

(iii) Multiple transitions are possible on the same input for a state

(iv) Some transitions might be missing

(v) ϵ - transitions

Non-deterministic Finite Automata (NFA)

Run 1: $Q0 \xrightarrow{1} Q0 \xrightarrow{0} Q0 \xrightarrow{1} Q0 \xrightarrow{1} Q0 \xrightarrow{0} Q0$ (**REJECT**)

Run 2: $Q0 \xrightarrow{1} Q0 \xrightarrow{0} Q0 \xrightarrow{1} Q1 \xrightarrow{1} Q2 \xrightarrow{0} Q3$ (**ACCEPT**)

Multiple **runs** per input is possible
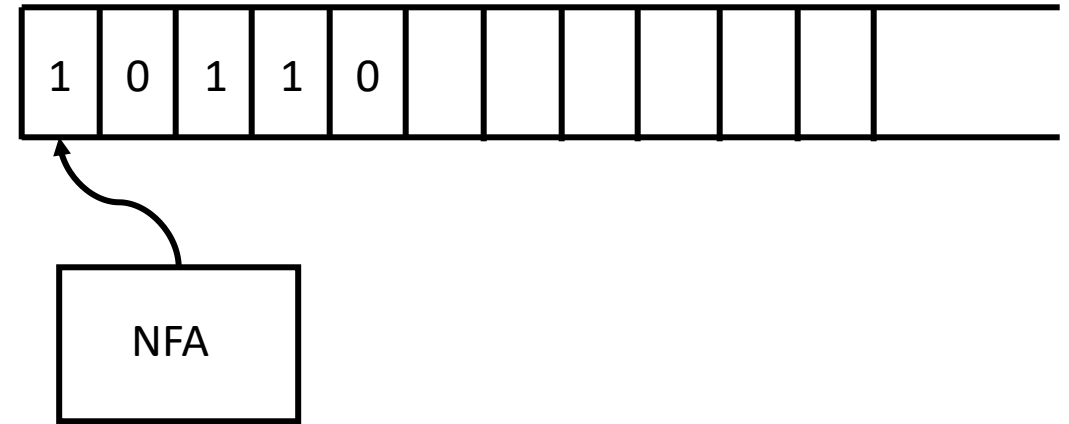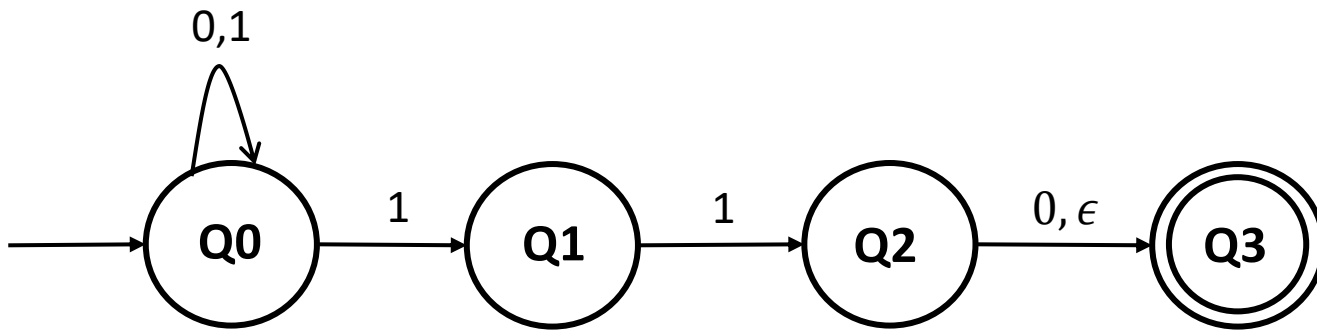
Non-deterministic Finite Automata (NFA)

Run 1: $Q0 \xrightarrow{1} Q0 \xrightarrow{0} Q0 \xrightarrow{1} Q0 \xrightarrow{1} Q0 \xrightarrow{0} Q0$ (**REJECT**)
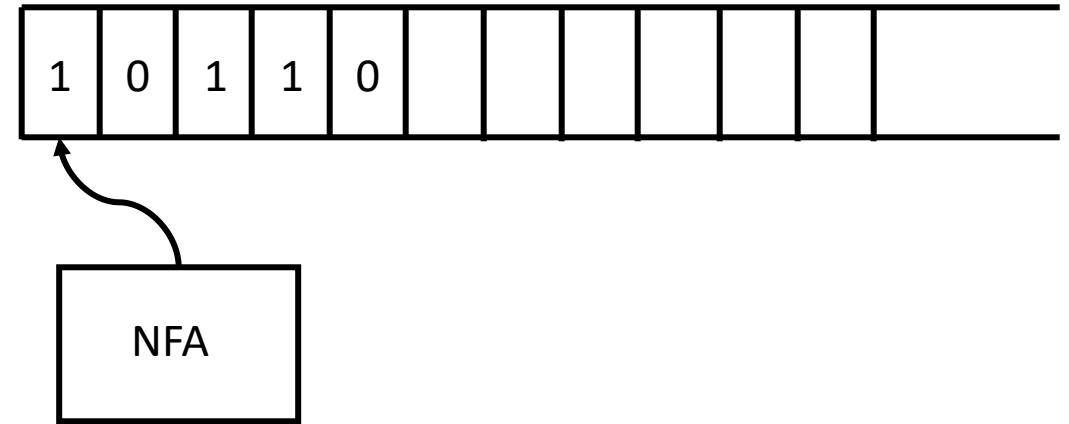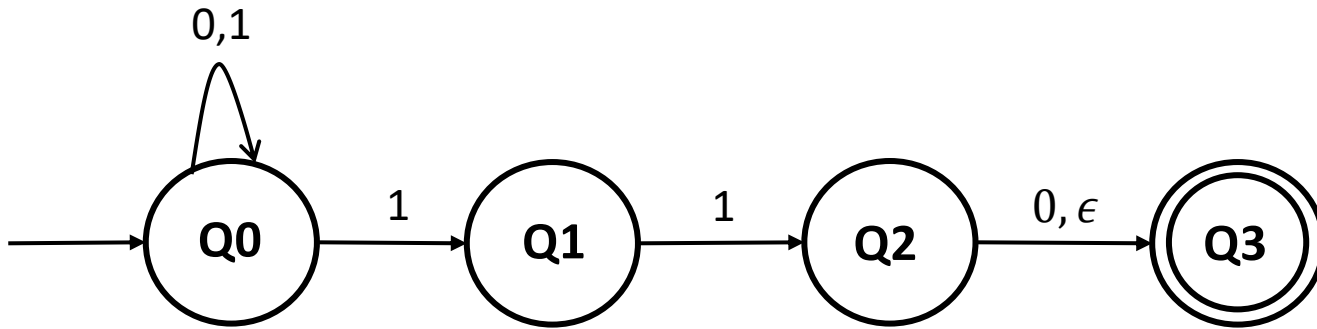
Run 2: $Q0 \xrightarrow{1} Q0 \xrightarrow{0} Q0 \xrightarrow{1} Q1 \xrightarrow{1} Q2 \xrightarrow{0} Q3$ (**ACCEPT**)

Run 3: $Q0 \xrightarrow{1} Q0 \xrightarrow{0} Q0 \xrightarrow{1} Q0 \xrightarrow{1} Q1 \xrightarrow{0}$ **CRASH**

Run 4: $Q0 \xrightarrow{1} Q0 \xrightarrow{0} Q0 \xrightarrow{1} Q1 \xrightarrow{1} Q2 \xrightarrow{\epsilon} Q3 \xrightarrow{0}$ **CRASH**

CRASH is a Rejecting Run

Non-deterministic Finite Automata (NFA)

The NFA "accepts" an input string, if it at **least one *run* ends up in the final state**. **(Accepting Run)**

The NFA "rejects" an input string, if there are **no runs that end up in a final state**. **(Rejecting Run)**

Run 1: $Q0 \xrightarrow{1} Q0 \xrightarrow{0} Q0 \xrightarrow{1} Q0 \xrightarrow{1} Q0 \xrightarrow{0} Q0$ **(REJECT)**

Run 2: $Q0 \xrightarrow{1} Q0 \xrightarrow{0} Q0 \xrightarrow{1} Q1 \xrightarrow{1} Q2 \xrightarrow{0} Q3$ **(ACCEPT)**
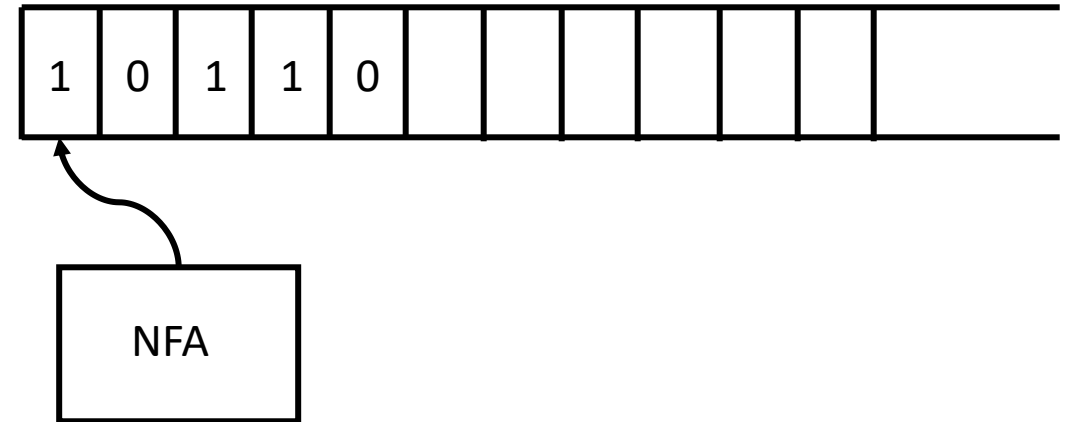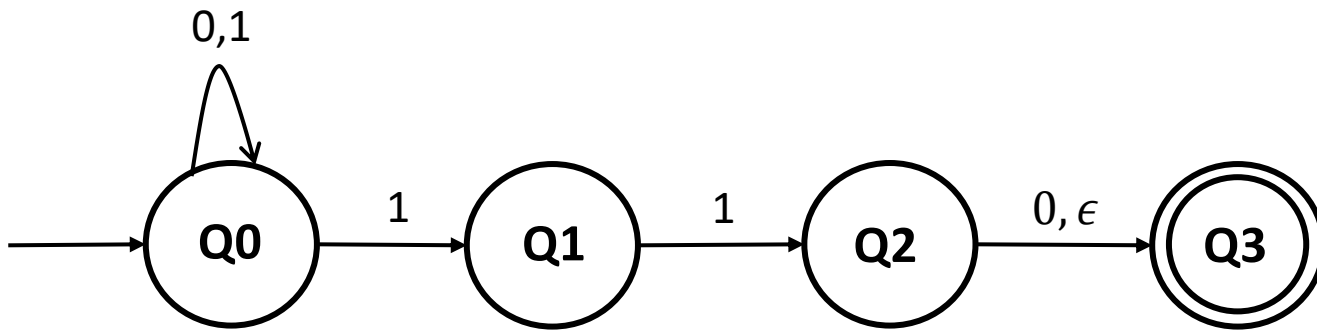
Run 3: $Q0 \xrightarrow{1} Q0 \xrightarrow{0} Q0 \xrightarrow{1} Q0 \xrightarrow{1} Q1 \xrightarrow{0}$ CRASH **(REJECT)**

Run 4: $Q0 \xrightarrow{1} Q0 \xrightarrow{0} Q0 \xrightarrow{1} Q1 \xrightarrow{1} Q2 \xrightarrow{\epsilon} Q3 \xrightarrow{0}$ CRASH **(REJECT)**

Non-deterministic Finite Automata (NFA)

	0	1	ϵ
Q0	Q0	Q0, Q1	
Q1		Q2	
Q2	Q3		Q3
Q3			

Non-deterministic Finite Automata (NFA)
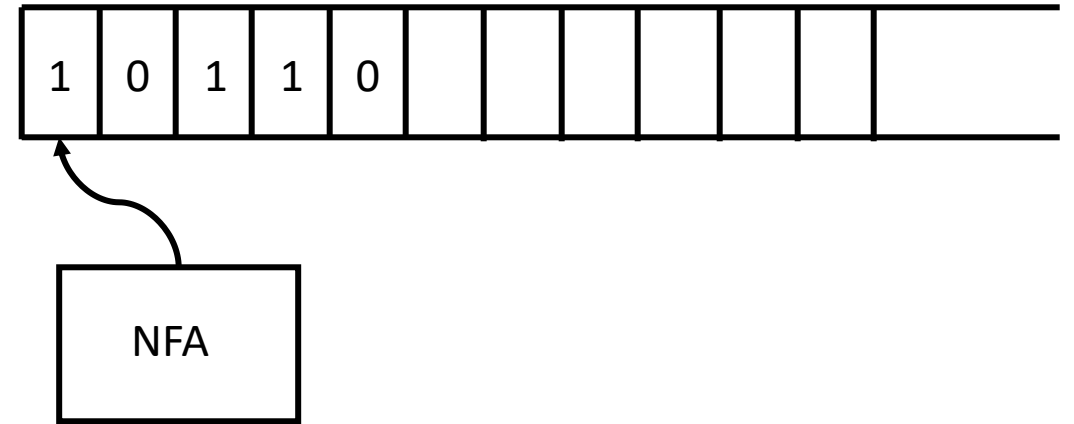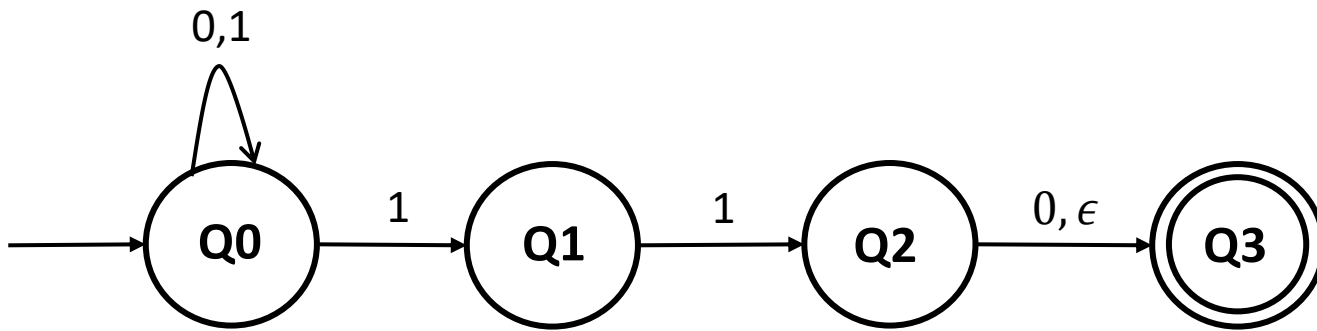
Formally, a NFA M is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

- Q is a finite set called the **states.**
- Σ is a finite set called the **alphabet**.
- $\delta : Q \times \Sigma \mapsto P(Q)$ is the **transition function**. $P(Q)$ is the power set of Q
- $q_0 \in Q$ is the **start state**.
- $F \subseteq Q$ is the set of **final/accepting states**.

	0	1	ϵ
Q0	Q0	Q0, Q1	
Q1		Q2	
Q2	Q3		Q3
Q3			

NFA vs DFA

- Are NFAs more powerful than DFAs?

- Intuitively, non-determinism seems to be adding more "power".

NFA vs DFA

- Are NFAs more powerful than DFAs?

- Intuitively, non-determinism seems to be adding more "power".

- Let L_1 be the language accepted by NFAs and L_2 be the language accepted by DFAs

- Is $L_2 \subseteq L_1$?

NFA vs DFA

- Are NFAs more powerful than DFAs?

- Intuitively, non-determinism seems to be adding more "power".

- Let L_1 be the language accepted by NFAs and L_2 be the language accepted by DFAs

- Is $L_2 \subseteq L_1$? Clearly true, because a DFA is just a special case of an NFA.

NFA vs DFA

- Are NFAs more powerful than DFAs?

- Intuitively, non-determinism seems to be adding more "power".

- Let L_1 be the language accepted by NFAs and L_2 be the language accepted by DFAs

- Is $L_2 \subseteq L_1$? Clearly true, because a DFA is just a special case of an NFA.

- Surprisingly, what we will show next is that $\boldsymbol{L_1 \subseteq L_2}$!

- That is, **given an NFA, we can convert it to a DFA that accepts the same language**.

- Such a DFA is called a "**Remembering DFA**".

**Thus, DFAs and NFAs are completely equivalent and $L_1 = L_2$!**

Converting an NFA to a DFA

Intuitive idea for the construction of a Remembering DFA from an NFA:

- Let R be the Remembering DFA corresponding to an NFA N.

- R on an input enters a state that is labelled by all possible states that N can enter on that input.

- Note that this "trims away" the non-determinism of the NFA N without "losing" the language it accepts.

- Also note that if N has k states, then R has at most 2^k states. Why?

Converting an NFA to a DFA

Intuitive idea for the construction of a Remembering DFA from an NFA:

- Let R be the Remembering DFA corresponding to an NFA N.

- R on an input enters a state that is labelled by all possible states that N can enter on that input.

- Note that this "trims away" the non-determinism of the NFA N without "losing" the language it accepts.

- Also note that if N has k states, then R has at most 2^k states. Why?

- Any label in the Remembering DFA is a subset of $\{Q_0, \ Q_1, \ Q_2, \ldots \ldots, Q_{k-1}\}$, where Q_i = State of the NFA.

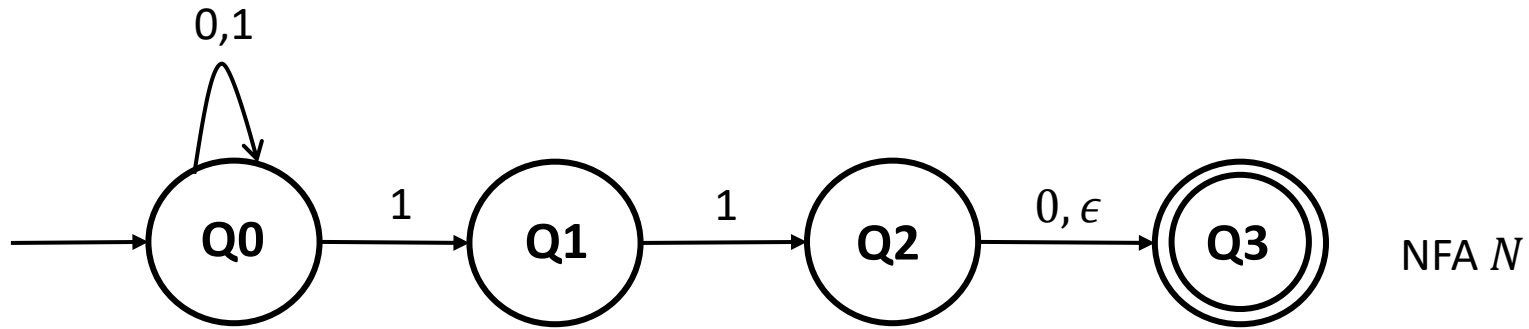- There are at most 2^k labels for the DFA.

Converting an NFA to a DFA

- R on an input enters a state that is labelled by all possible states that N can enter on that input.
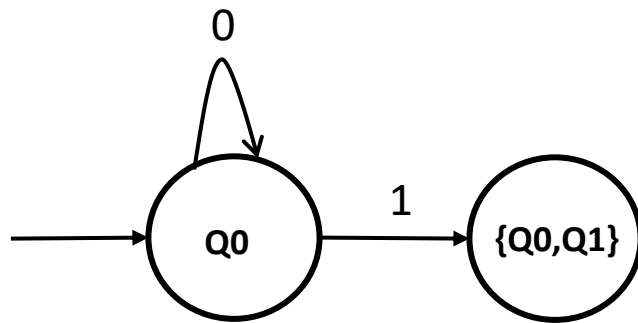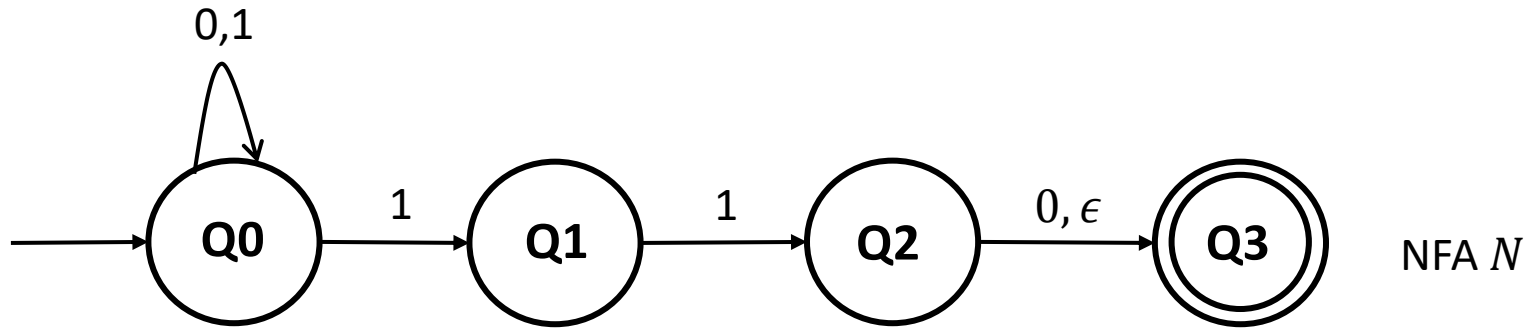

NFA N
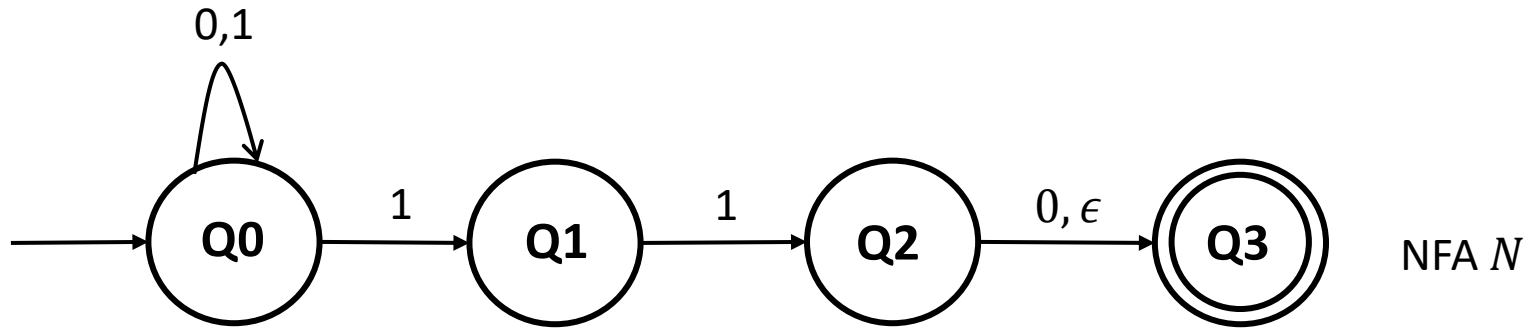
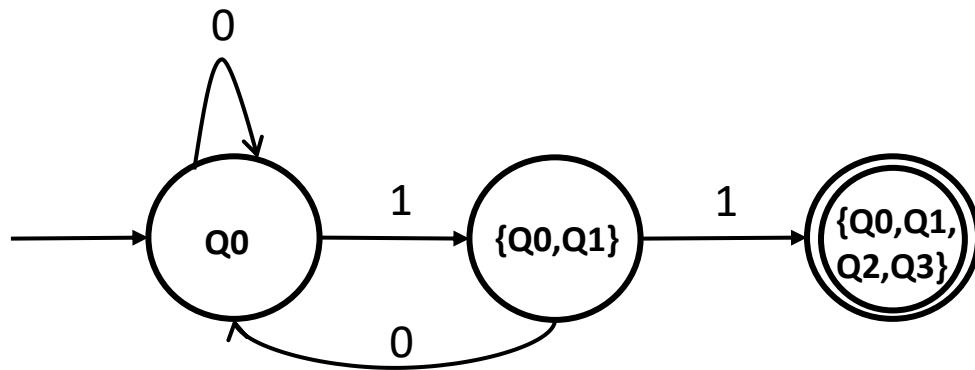|	0	1	ϵ
Q0	Q0	Q0, Q1	
Q1		Q2	
Q2	Q3		Q3
Q3			

Converting an NFA to a DFA

- R on an input enters a state that is labelled by all possible states that N can enter on that input.

NFA N

	0	1	ϵ
Q0	Q0	Q0, Q1	
Q1		Q2	
Q2	Q3		Q3
Q3			

Remembering DFA R

Converting an NFA to a DFA

- R on an input enters a state that is labelled by all possible states that N can enter on that input.

NFA N

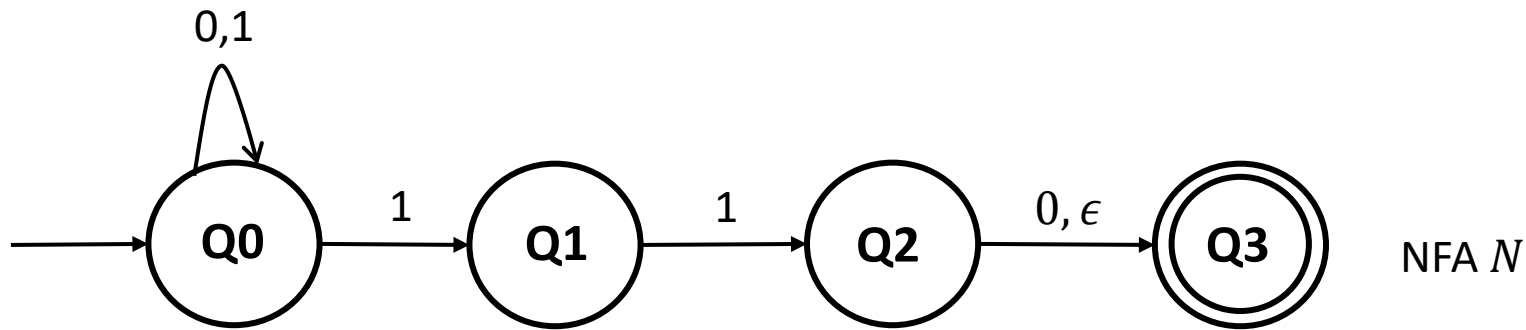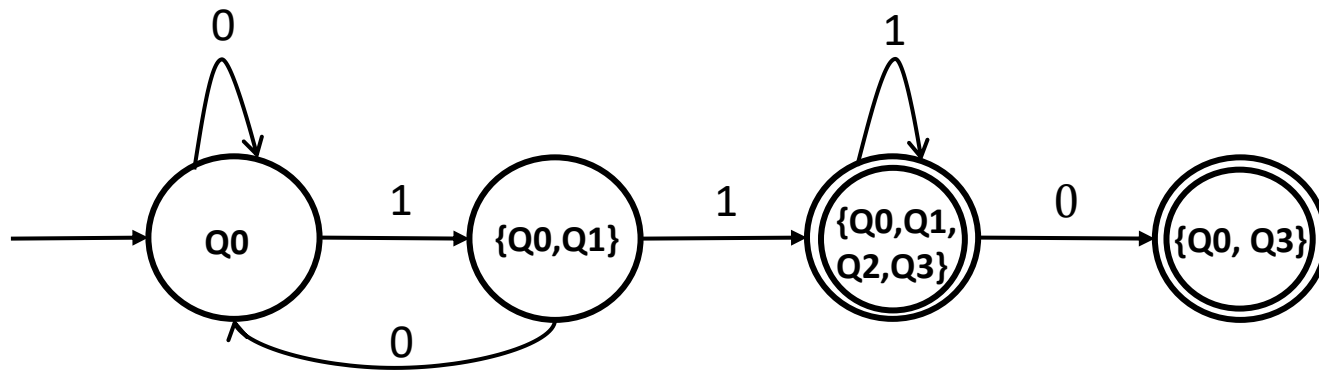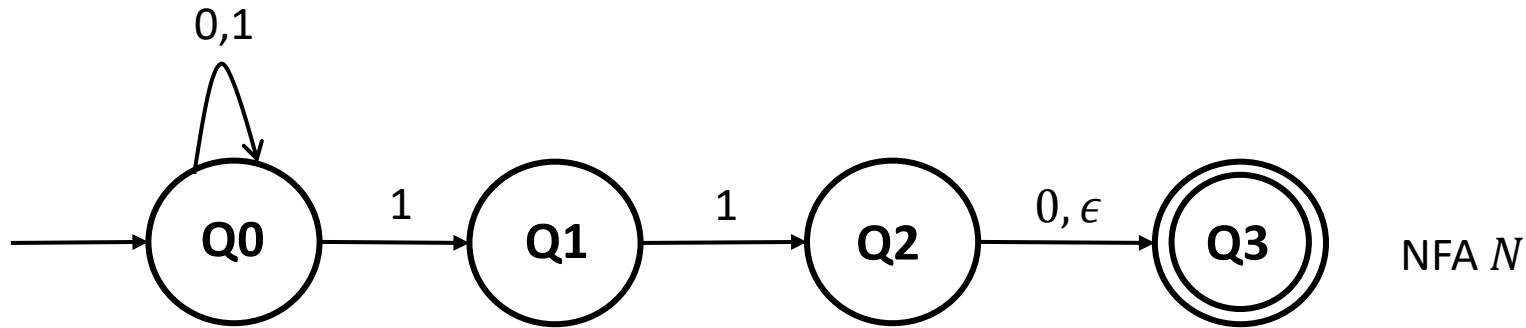|	0	1	ϵ
Q0	Q0	Q0, Q1	
Q1		Q2	
Q2	Q3		Q3
Q3			

Remembering DFA R

Any state of R that contains in its label, an accepting state of N is an accepting state of R.

Converting an NFA to a DFA

- R on an input enters a state that is labelled by all possible states that N can enter on that input.

NFA N

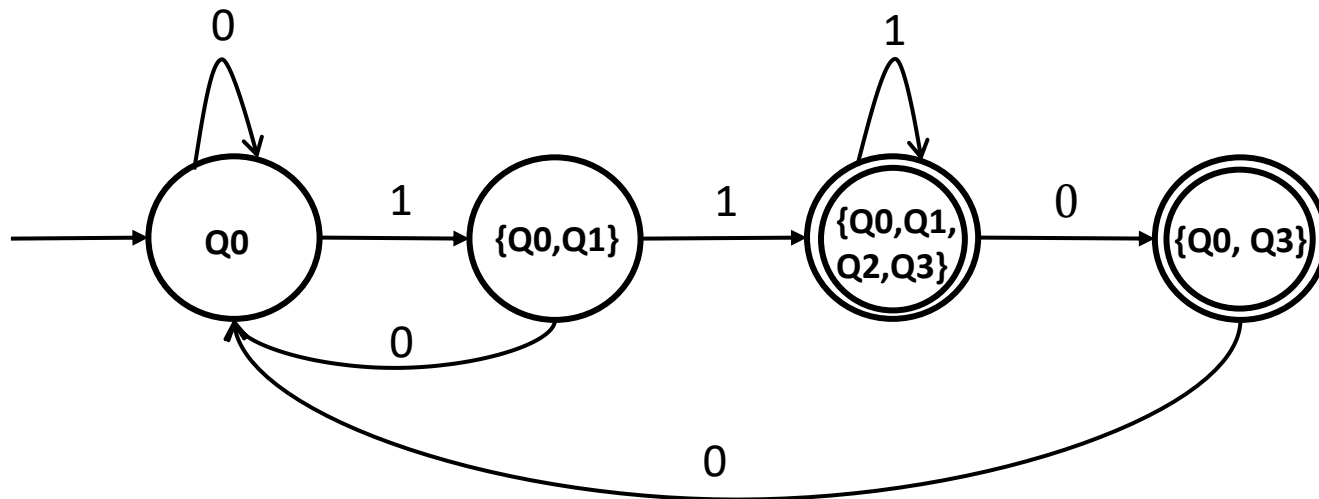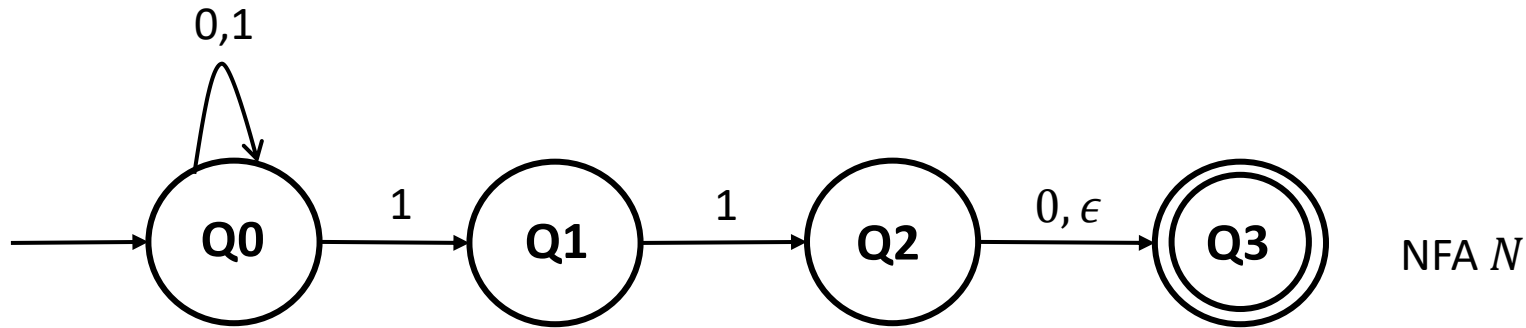|	0	1	ϵ
Q0	Q0	Q0, Q1	
Q1		Q2	
Q2	Q3		Q3
Q3			

Remembering DFA R

Any state of R that contains in its label, an accepting state of N is an accepting state of R.

Converting an NFA to a DFA

- M_2 on an input enters a state that is labelled by all possible states that M_1 can enter on that input.

NFA N

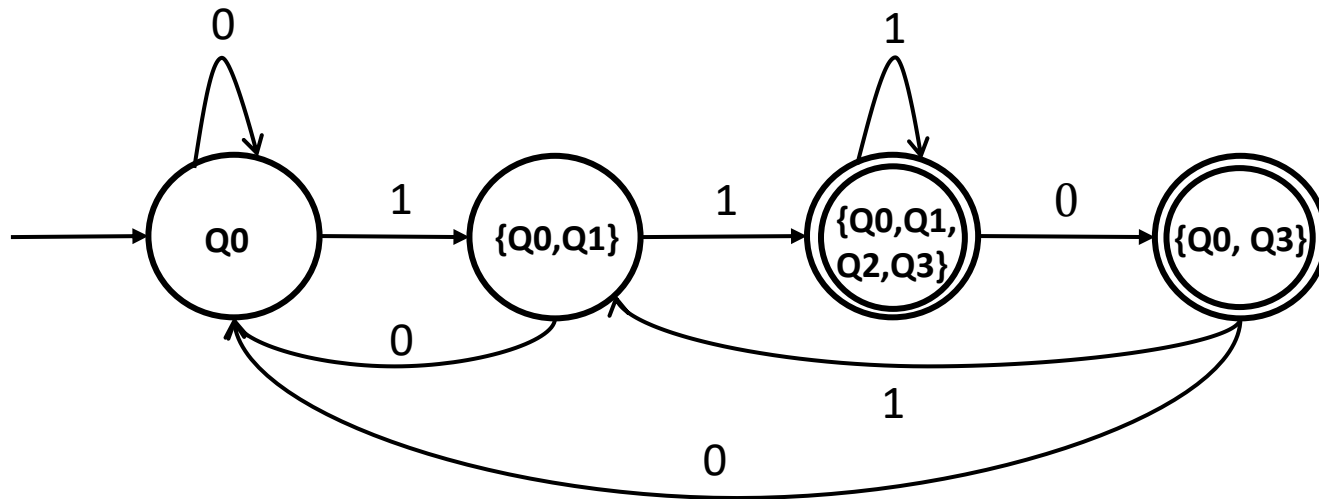|	0	1	ϵ
Q0	Q0	Q0, Q1	
Q1		Q2	
Q2	Q3		Q3
Q3			

Remembering DFA R

Any state of R that contains in its label, an accepting state of N is an accepting state of R.

Converting an NFA to a DFA

- M_2 on an input enters a state that is labelled by all possible states that M_1 can enter on that input.

NFA N

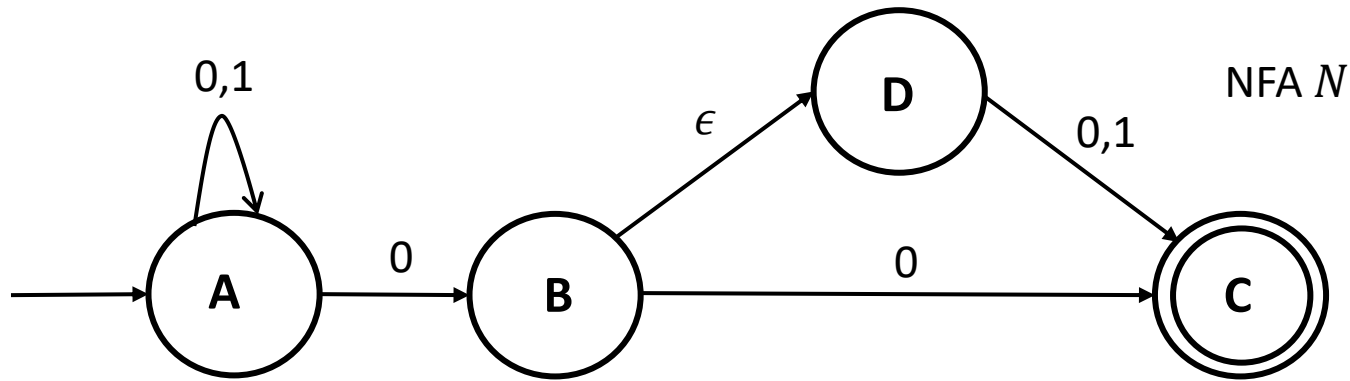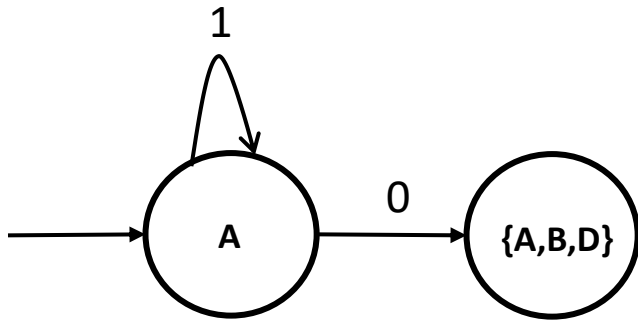|	0	1	ϵ
Q0	Q0	Q0, Q1	
Q1		Q2	
Q2	Q3		Q3
Q3			

Remembering DFA R

Any state of R that contains in its label, an accepting state of N is an accepting state of R.

Converting an NFA to a DFA

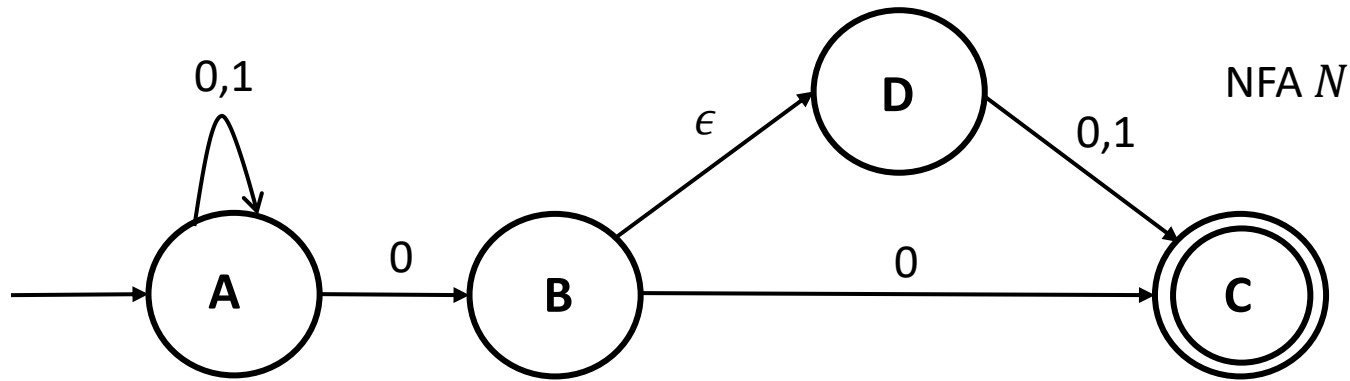- M_2 on an input enters a state that is labelled by all possible states that M_1 can enter on that input.

NFA N

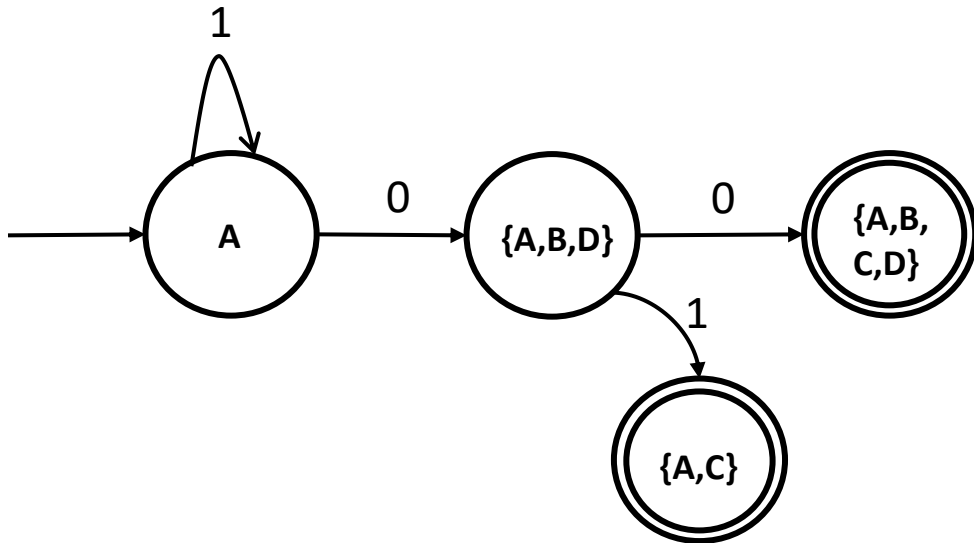|	0	1	ϵ
A	A, B	A	
B	C		D
C			
D	C	C	

Remembering DFA R

Converting an NFA to a DFA

- M_2 on an input enters a state that is labelled by all possible states that M_1 can enter on that input.

NFA N

	0	1	ϵ
A	A, B	A	
B	C		D
C			
D	C	C	

Remembering DFA R

Converting an NFA to a DFA

- M_2 on an input enters a state that is labelled by all possible states that M_1 can enter on that input.
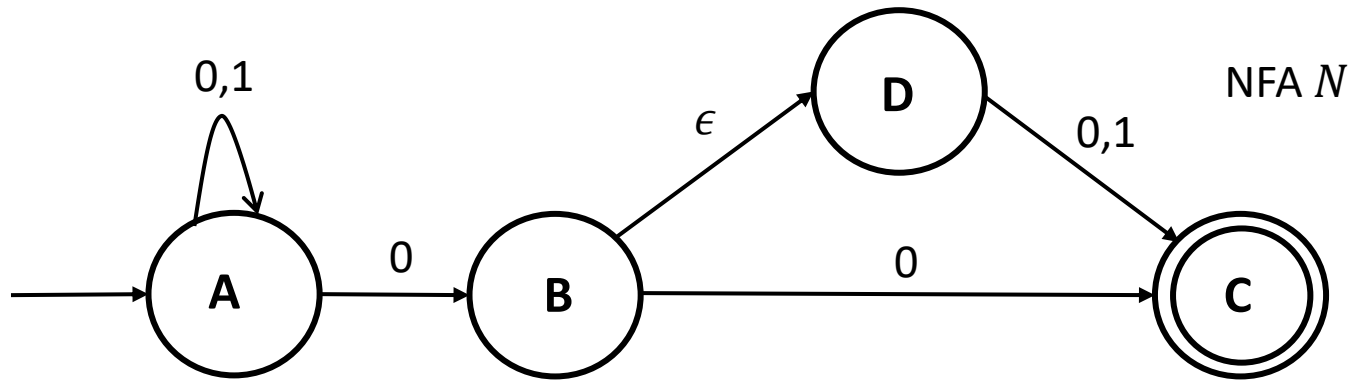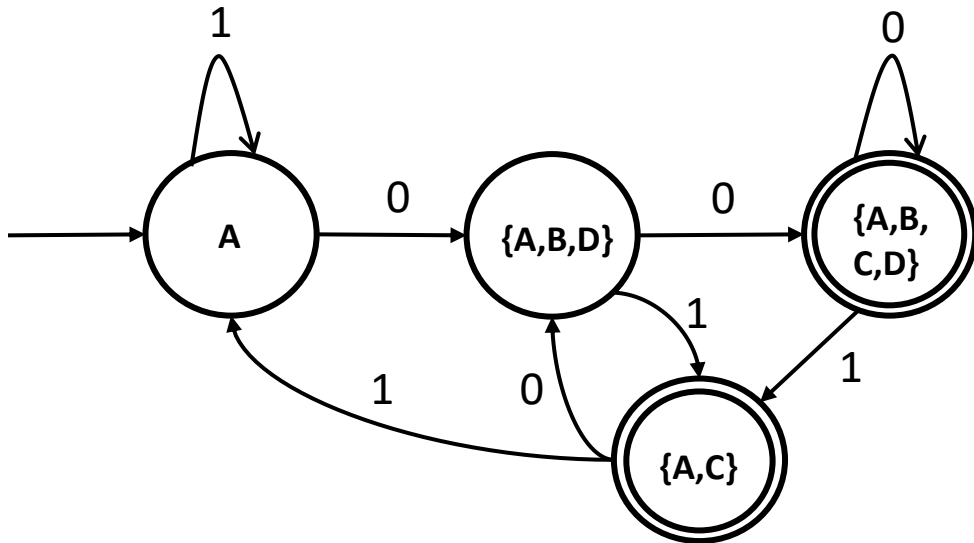

NFA N

	0	1	ϵ
A	A, B	A	
B	C		D
C			
D	C	C	

Remembering DFA R

Thank You!