CS F351: Theory of Computation

03 – Non-Deterministic Finite Automata

BITS Pilani
Dubai Campus

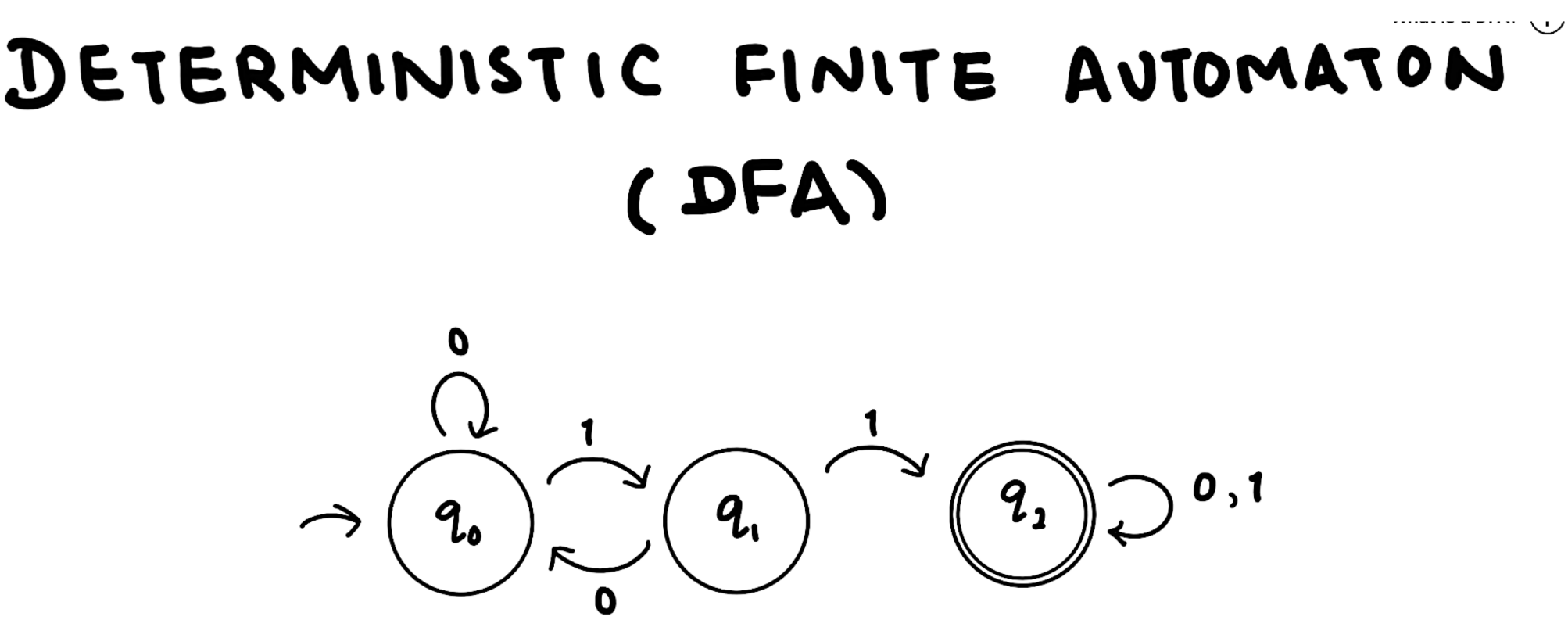
Dr Elakkiya R, AP/CS

Nondeterministic Finite Automaton

- Motivation for NFA
- NFA definition and computation
- Examples
- Reg Lang, NFA, DFA Equivalence
- NFA => DFA (Subset Construction)
- Closure Properties of Regular Language (NFA)

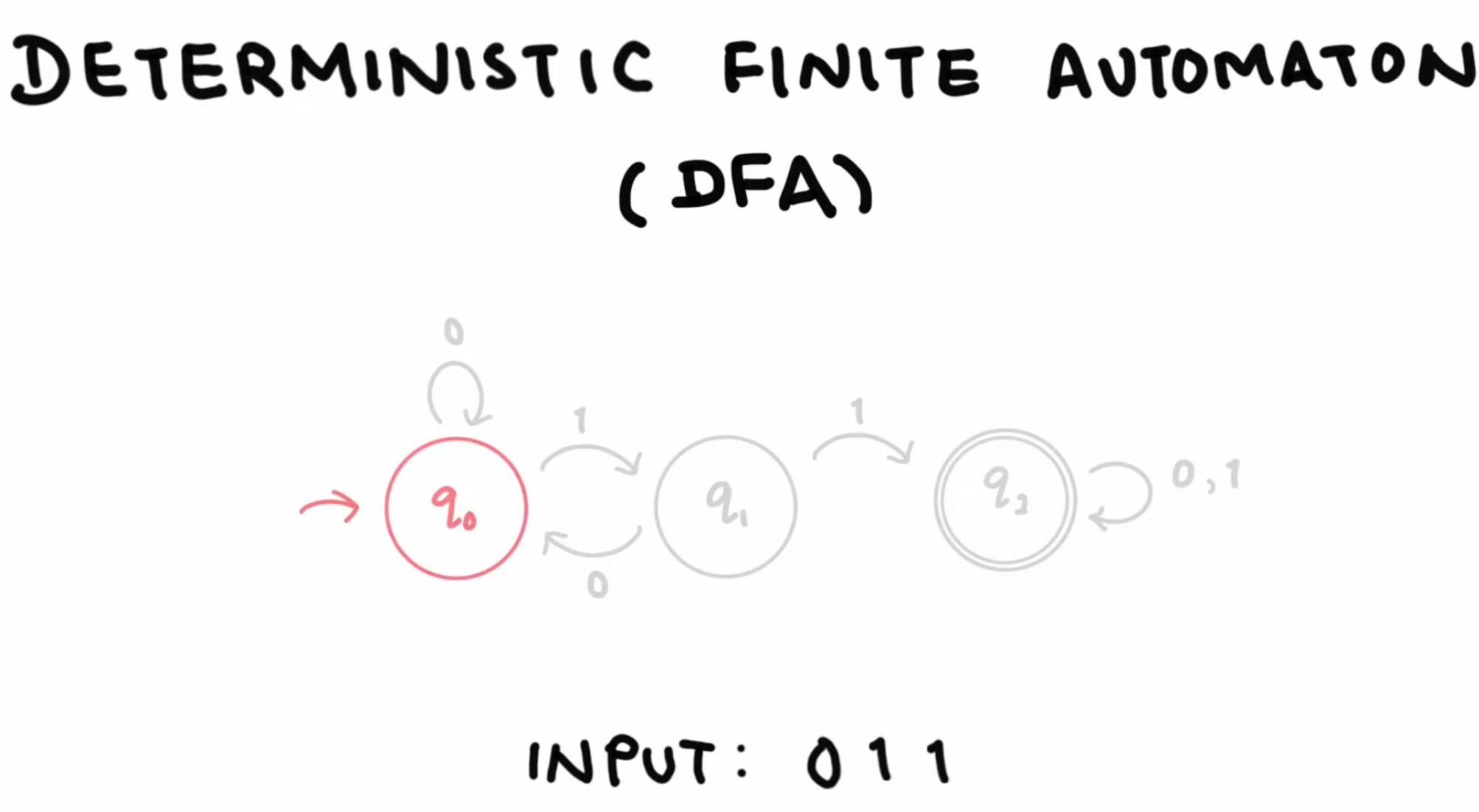
Formal Language, chapter 1, slide 5

Nondeterministic Finite Automaton



Formal Language, chapter 1, slide 5

Nondeterministic Finite Automaton



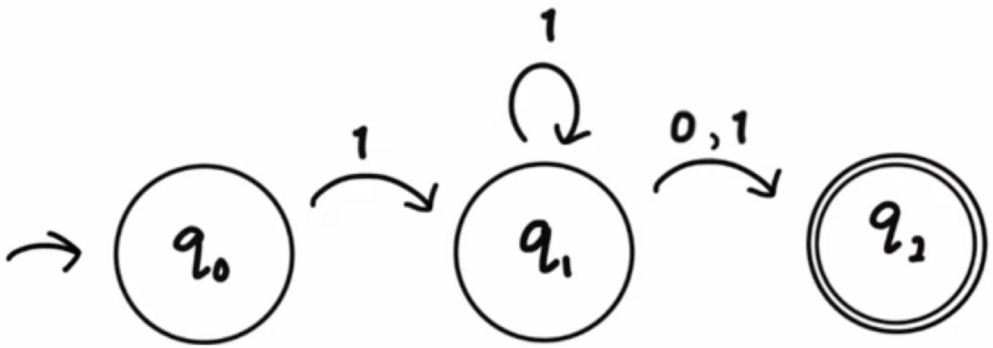
Formal Language, chapter 1, slide 5

Nondeterministic Finite Automaton

innovate achieve lead

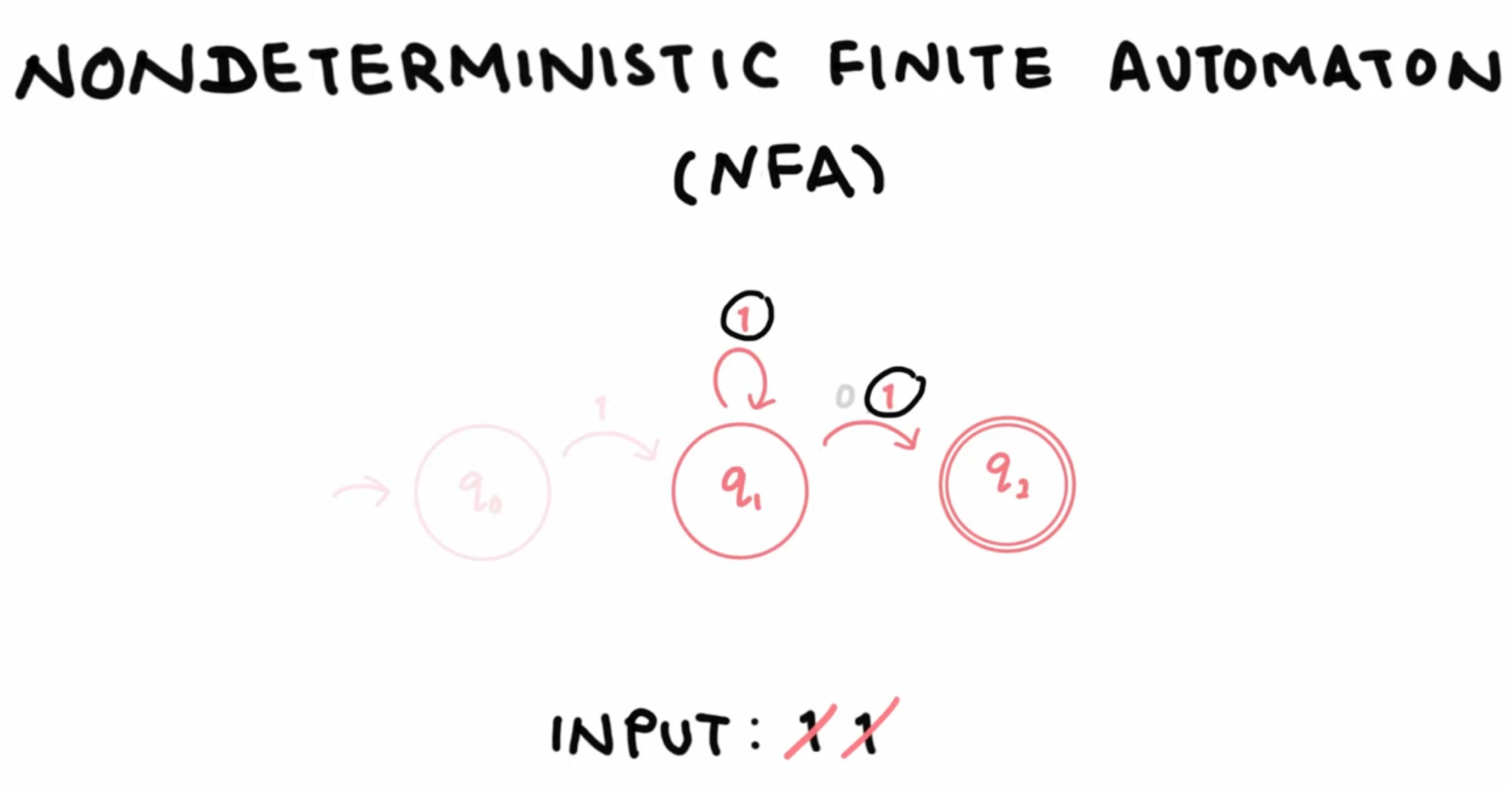
NONDETERMINISTIC FINITE AUTOMATON

(NFA)



Formal Language, chapter 1, slide 5

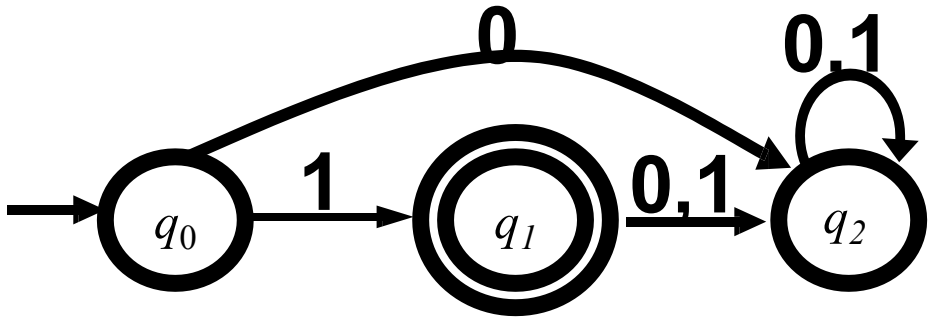
Nondeterministic Finite Automaton



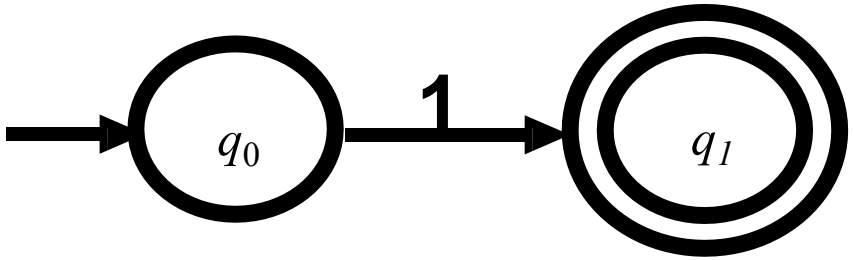
Formal Language, chapter 1, slide 5

Motivating eg for NFA

- DFA M for recognizing the $L = \{ 1 \}$



- NFA N for recognizing the $L = \{ 1 \}$



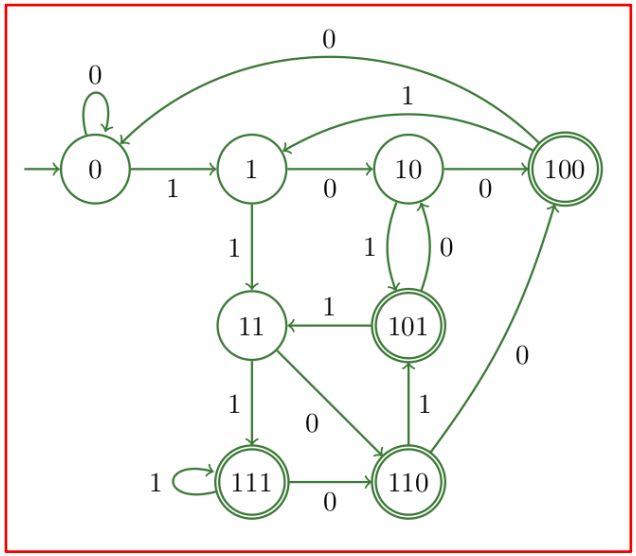
Informally :-
DFA : for every state, for each symbol, transition to next state has to be defined, possibly leading to more no. of states.

NFA: Only account for the transitions required for the pattern to be recognized.. Still the NFA will work correctly (why??, important to understand this concept)

Other reasons to follow later.
NFAs are easier to combine than DFAs for operations like union, concatenation etc. on regular languages

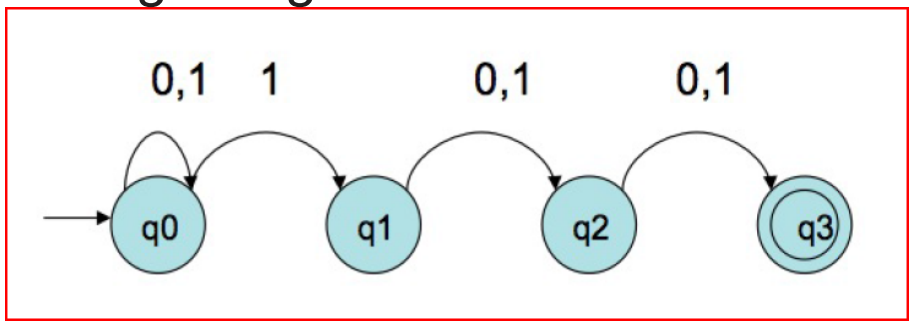
Motivating eg for NFA

- DFA M for recognizing L containing strings where the 3rd symbol from last is a 1 $(\Sigma = \{0, 1\})$

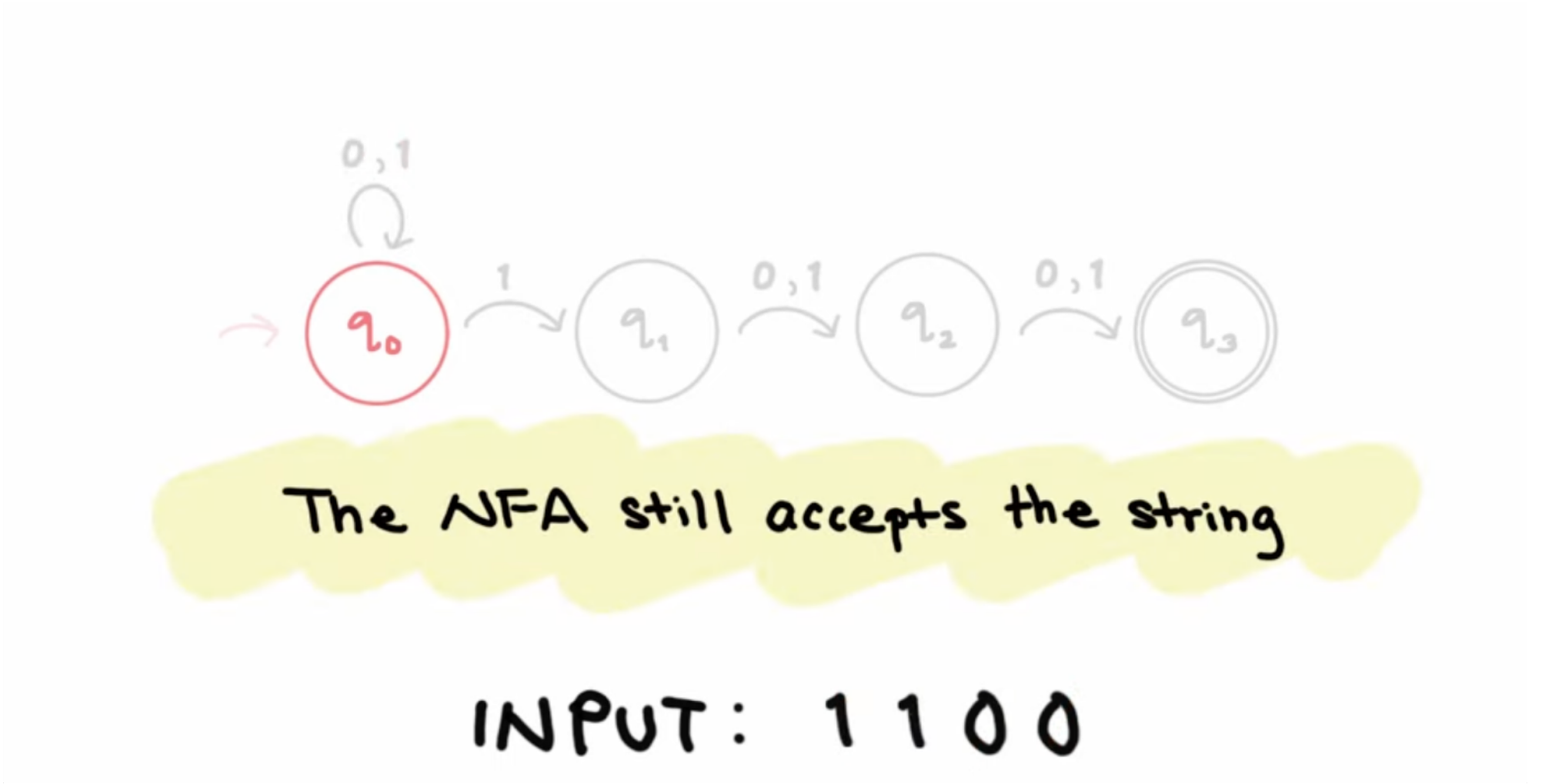


- Eg of such strings are 01010100, 0101, 010110, 0111
- -ve examles are 11, 0000, 0010, 0011, 011011

- NFA M for recognizing the same language *L*
- ie. the 3rd symbol from end is a 1 follow by either 0 or 1, followed by 0 or 1
- the string can have any no of 0s or 1s in the beginning.



Nondeterministic Finite Automaton



Formal Language, chapter 1, slide 5

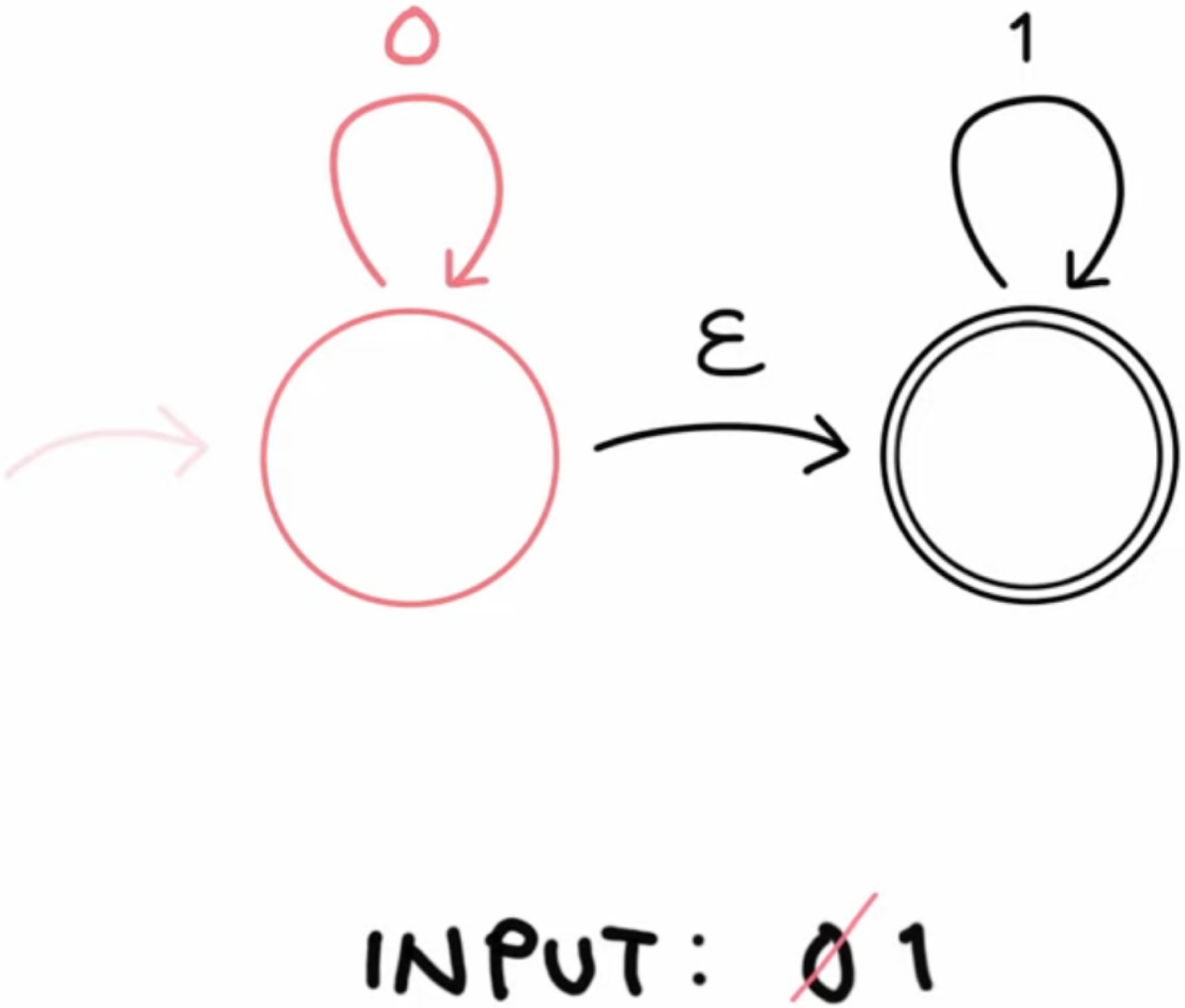
Nondeterministic Finite Automaton

The NFA still accepts the string because there is a path to an accept state.

INPUT: ~~1~~ ~~1~~ ~~0~~ ~~0~~

Formal Language, chapter 1, slide 5

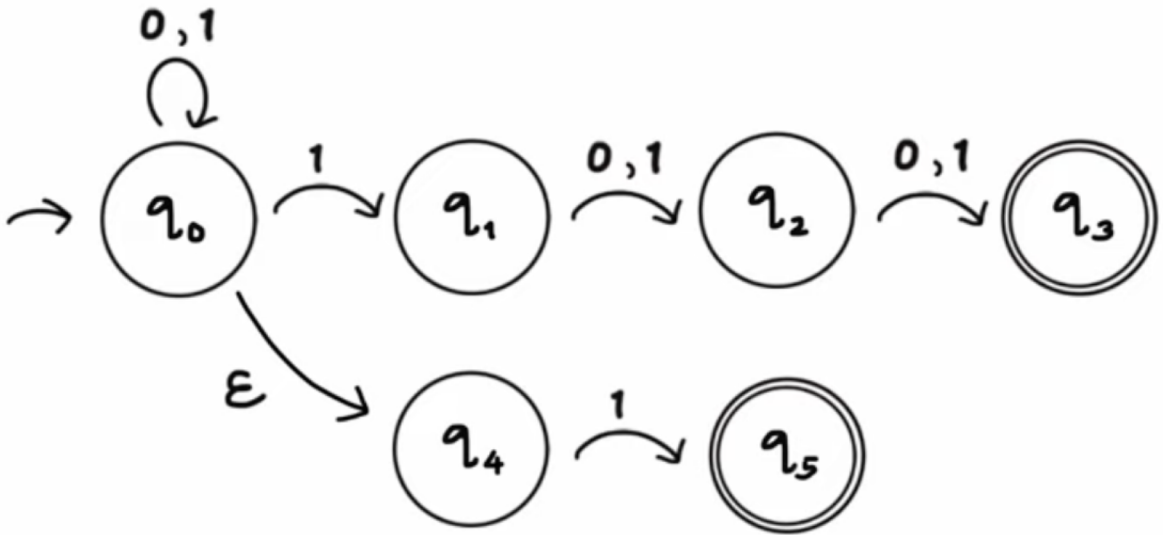
Nondeterministic Finite Automaton



Formal Language, chapter 1, slide 3

Nondeterministic Finite Automaton

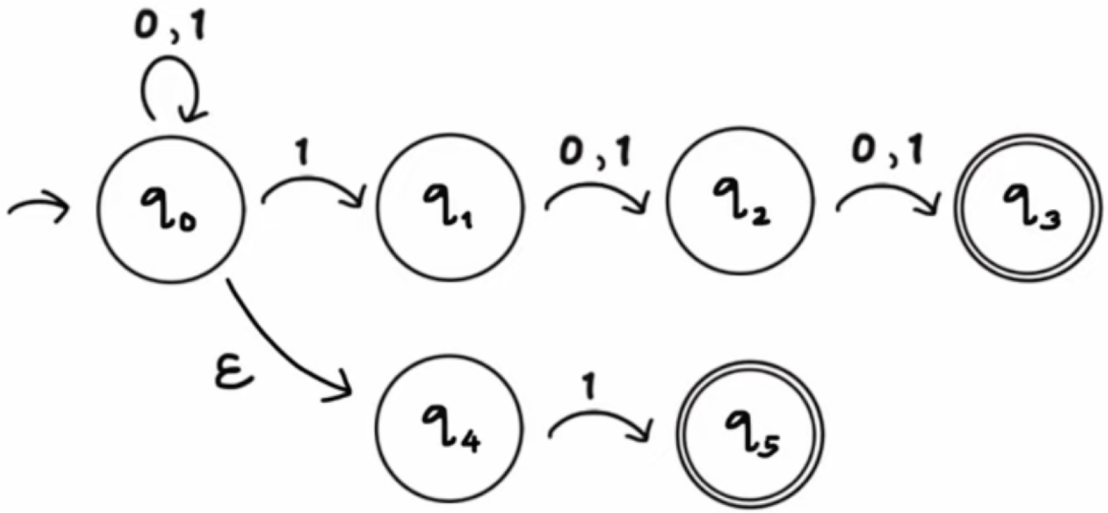
$L_M = \{ x : x \text{ contains a } 1 \text{ in the third final position} \} \cup \{1\}$



Formal Language, chapter 1, slide 5

Nondeterministic Finite Automaton

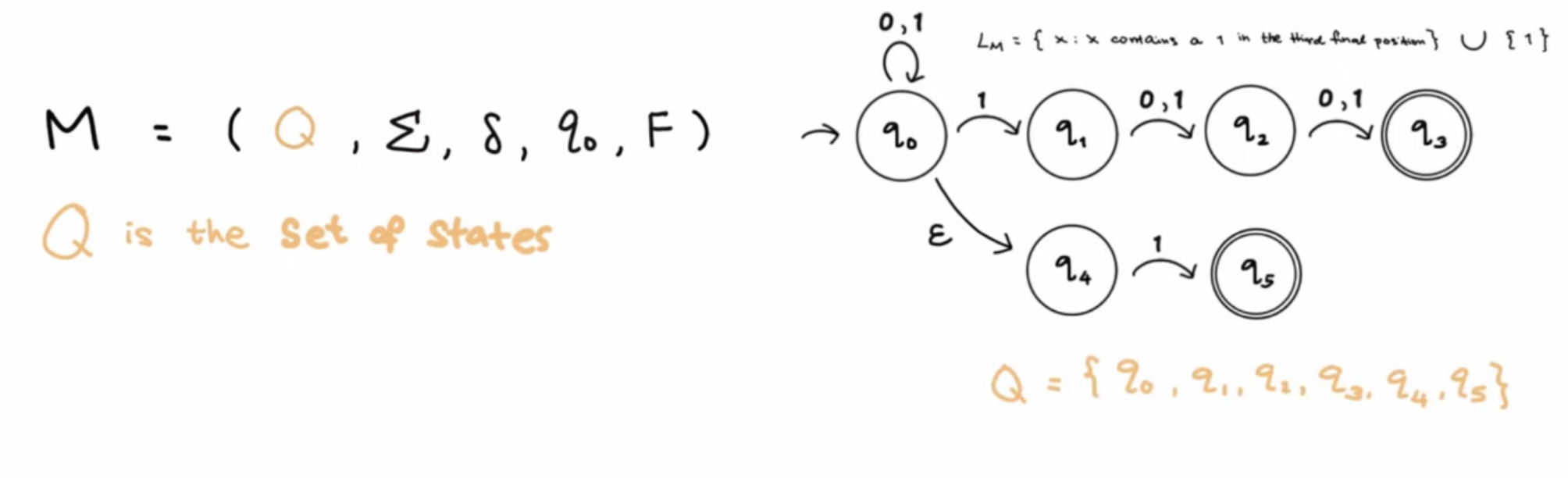
$L_M = \{ x : x \text{ contains a } 1 \text{ in the third final position} \} \cup \{ 1 \}$



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

Formal Language, chapter 1, slide 5

Nondeterministic Finite Automaton



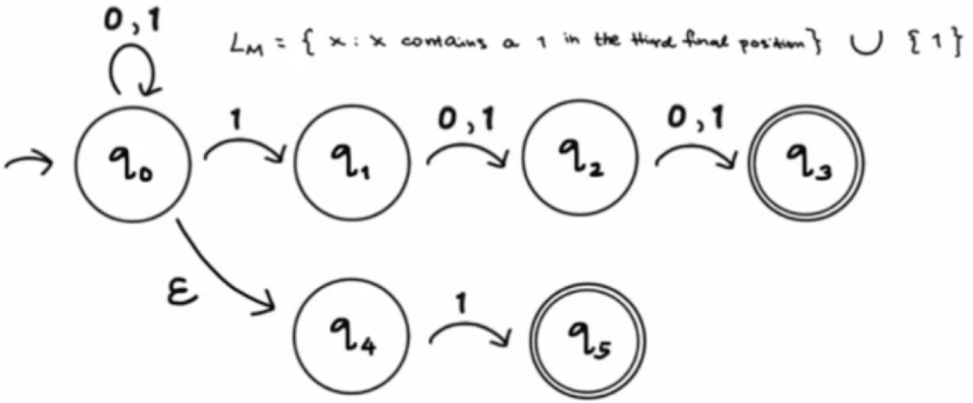
Formal Language, chapter 1, slide 5

Nondeterministic Finite Automaton

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

Q is the set of states

Σ is the alphabet



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

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

Formal Language, chapter 1, slide 5

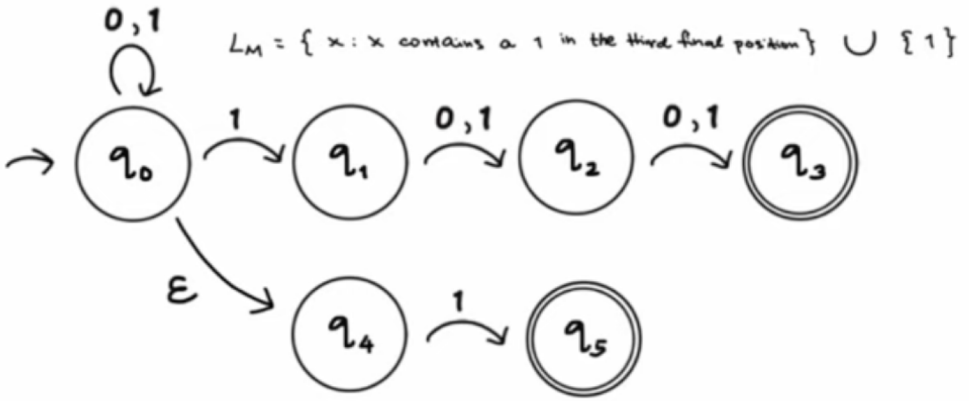
Nondeterministic Finite Automaton

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

Q is the set of states

Σ is the alphabet

$\delta : Q \times \Sigma_\varepsilon \rightarrow P(Q)$ is the transition function



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

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

δ :

	0	1	ε
q_0	$\{q_0\}$	$\{q_0, q_1\}$	$\{q_4\}$
q_1	$\{q_2\}$	$\{q_2\}$	ϕ
q_2	$\{q_3\}$	$\{q_3\}$	ϕ
q_3	ϕ	ϕ	ϕ
q_4	ϕ	$\{q_5\}$	ϕ
q_5	ϕ	ϕ	ϕ

Formal Language, chapter 1, slide 5

Nondeterministic Finite Automaton

innovate achieve lead

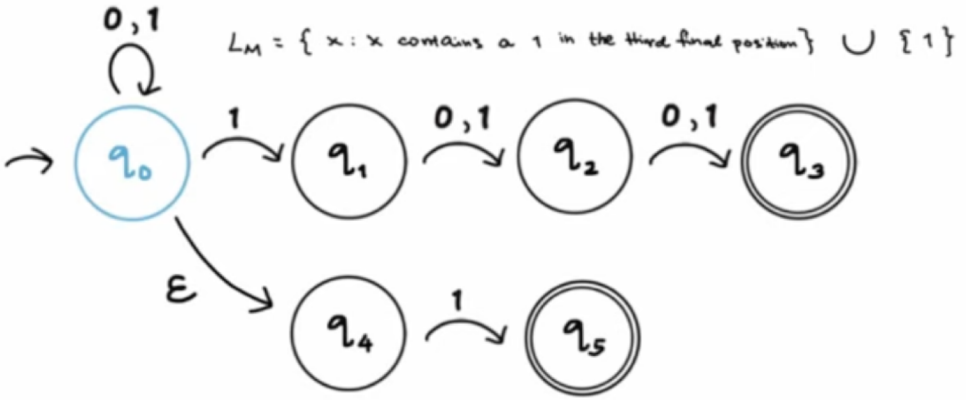
$M = (Q, \Sigma, \delta, q_0, F)$

Q is the set of states

Σ is the alphabet

$\delta : Q \times \Sigma_\varepsilon \to P(Q)$ is the transition function

q_0 is the start state



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

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

δ :

	0	1	ε
q_0	$\{q_0\}$	$\{q_0, q_1\}$	$\{q_4\}$
q_1	$\{q_2\}$	$\{q_2\}$	ϕ
q_2	$\{q_3\}$	$\{q_3\}$	ϕ
q_3	ϕ	ϕ	ϕ
q_4	ϕ	$\{q_5\}$	ϕ
q_5	ϕ	ϕ	ϕ

Formal Language, chapter 1, slide 5

Nondeterministic Finite Automaton

innovate achieve lead

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

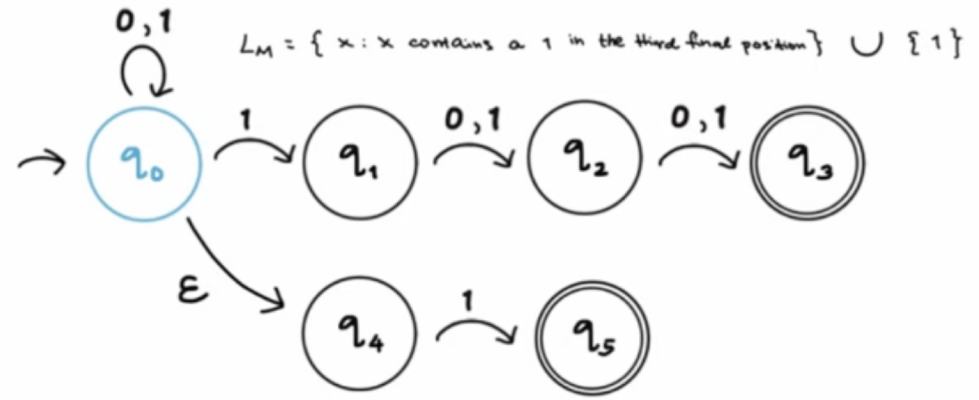
Q is the set of states

Σ is the alphabet

$\delta : Q \times \Sigma_\varepsilon \to P(Q)$ is the transition function

q_0 is the start state

F is the set of accept/final states



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

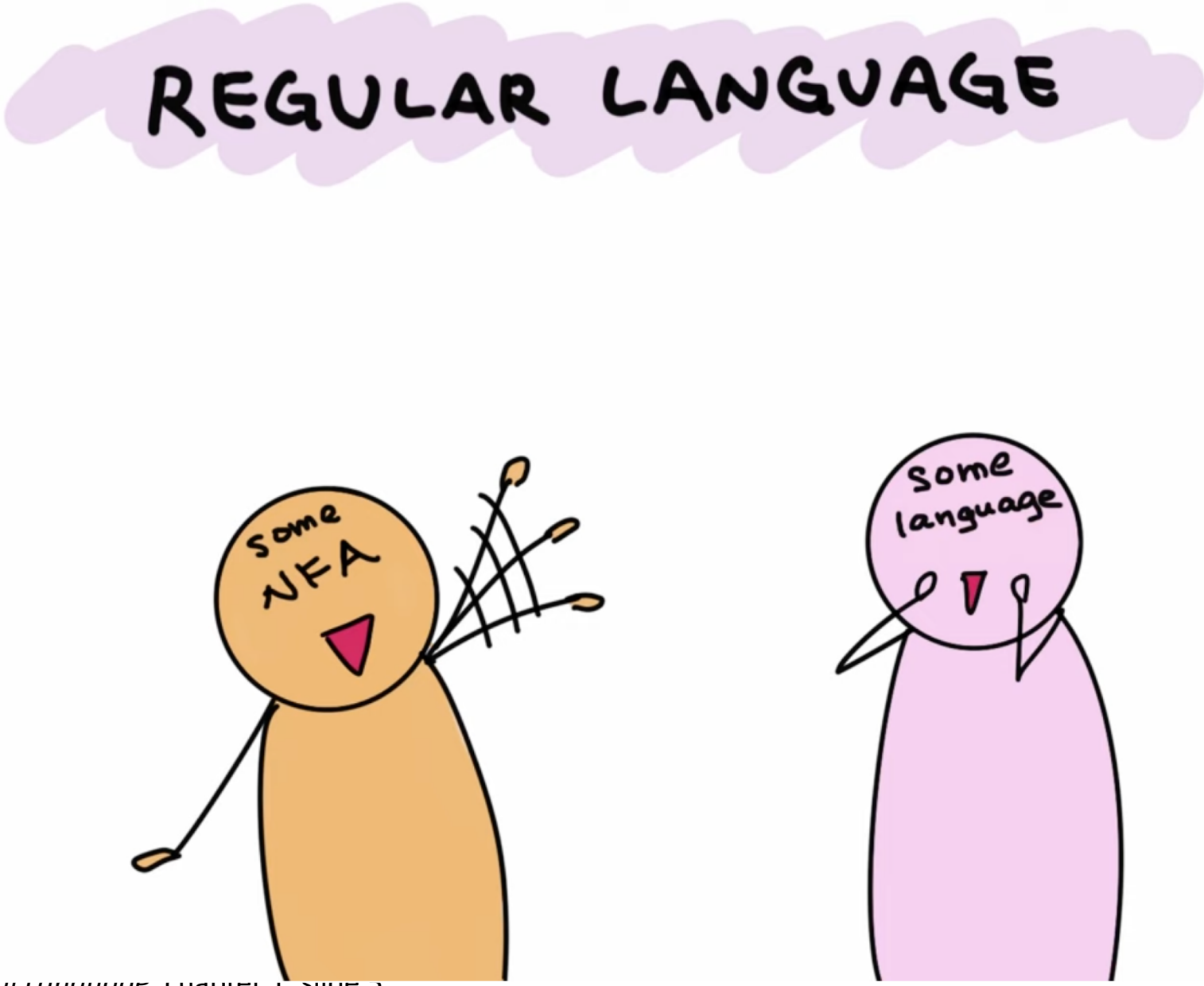
$\Sigma = \{1, 0\}$ $F = \{q_3, q_5\}$

δ :

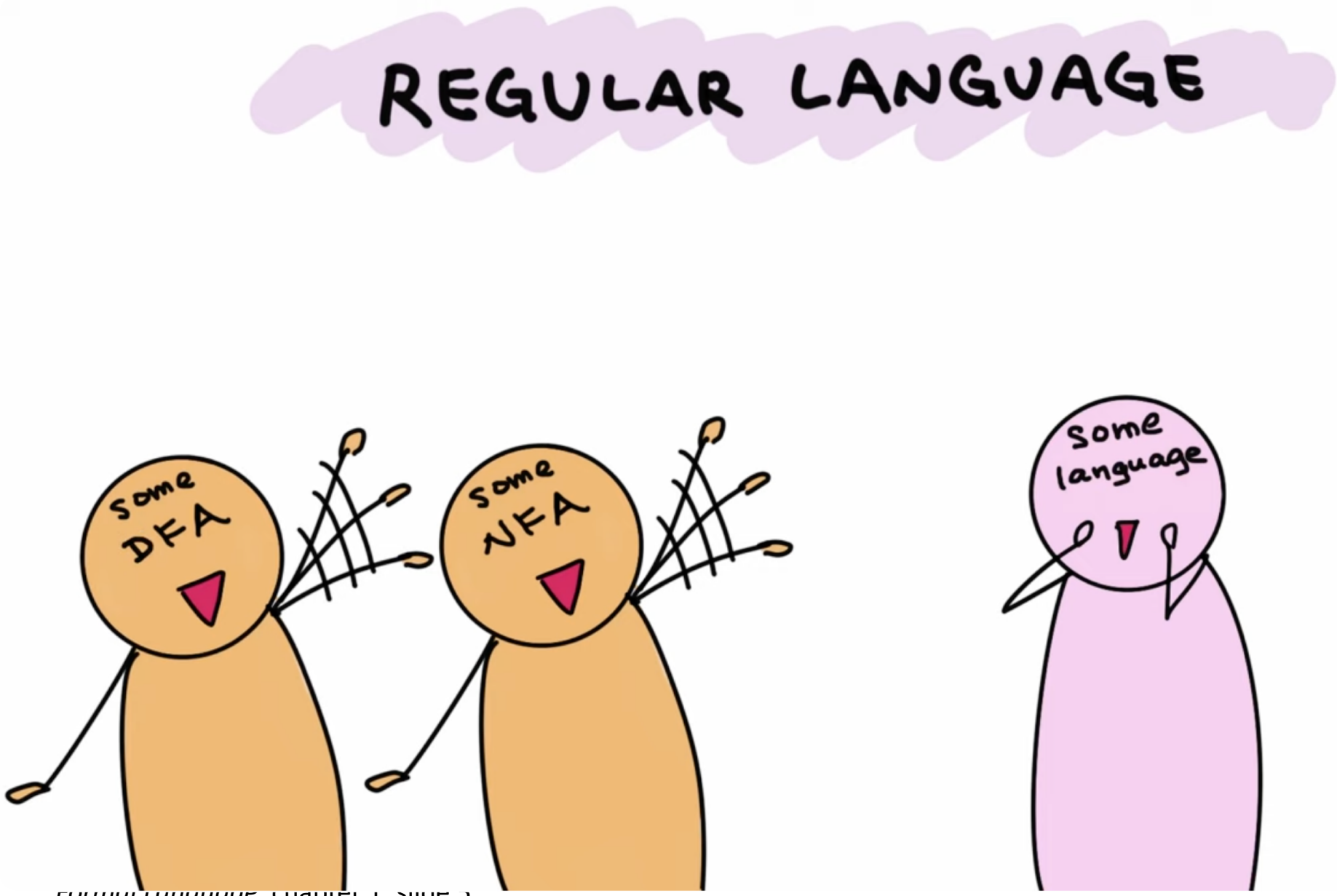
	0	1	ε
q_0	$\{q_0\}$	$\{q_0, q_1\}$	$\{q_4\}$
q_1	$\{q_2\}$	$\{q_2\}$	ϕ
q_2	$\{q_3\}$	$\{q_3\}$	ϕ
q_3	ϕ	ϕ	ϕ
q_4	ϕ	$\{q_5\}$	ϕ
q_5	ϕ	ϕ	ϕ

Formal Language, chapter 1, slide 5

Nondeterministic Finite Automaton



Nondeterministic Finite Automaton



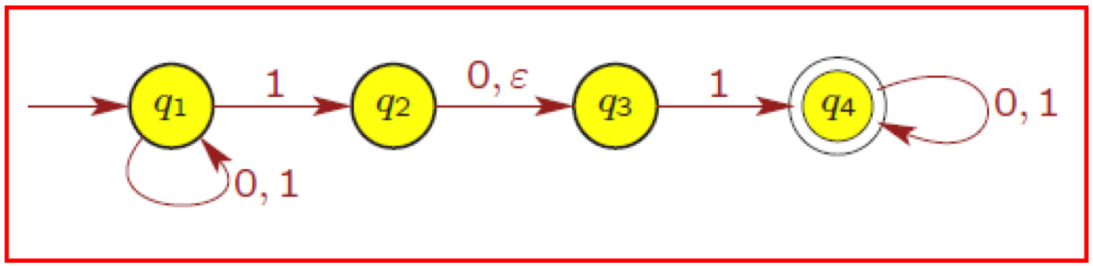
NFA (Informal Description)

innovate achieve lead

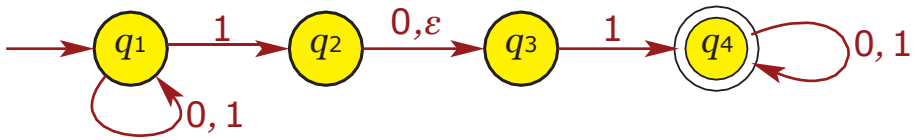
- Nondeterministic finite automata (NFAs) allow for **several** or **no choices** to exist for the next state on a given symbol.
- For a state q and symbol $a \in \Sigma$, NFA can have
  - multiple edges leaving q labelled with the same symbol a
  - no edge leaving q labelled with symbol a
  - edges leaving q labelled with ε **Note**: ε is empty symbol (null input)
  - can take ε-edge without reading any symbol from input string.

Eg: NFA N_1 for Strings containing 101 or 11 over $\Sigma = \{0, 1\}$.

This can be also described as **any no. of 0 or 1** followed by a **1** followed by **0 or ε** followed by **1** followed by **any no. of 0's or 1's**.
And can easily be translated in to the NFA shown below.



NFA Computation

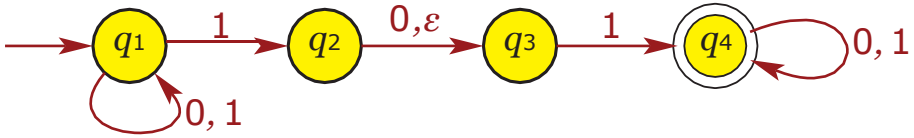


- When NFA is in a state with multiple ways to proceed, e.g., in state q_1 and the next symbol in input string is 1.
- The machine splits into multiple copies of itself (threads).
  - Each copy proceeds with computation independently of others.
  - NFA may be in a set of states, instead of a single state.
  - NFA follows all possible computation paths in parallel.
  - If a copy is in a state and next input symbol doesn't appear on any outgoing edge from the state, then the copy dies or crashes.

- If **any** copy ends in an **accept** state after reading entire input string, the NFA **accepts** the string.

- If **no** copy ends in an **accept** state after reading entire input string, then NFA does **not accept (rejects)** the string.

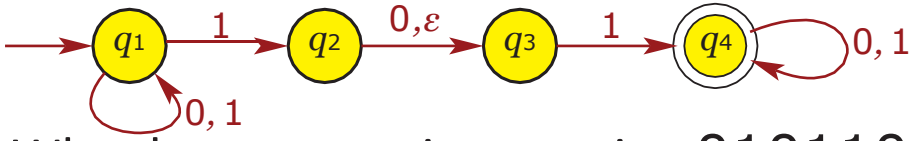
NFA Computation



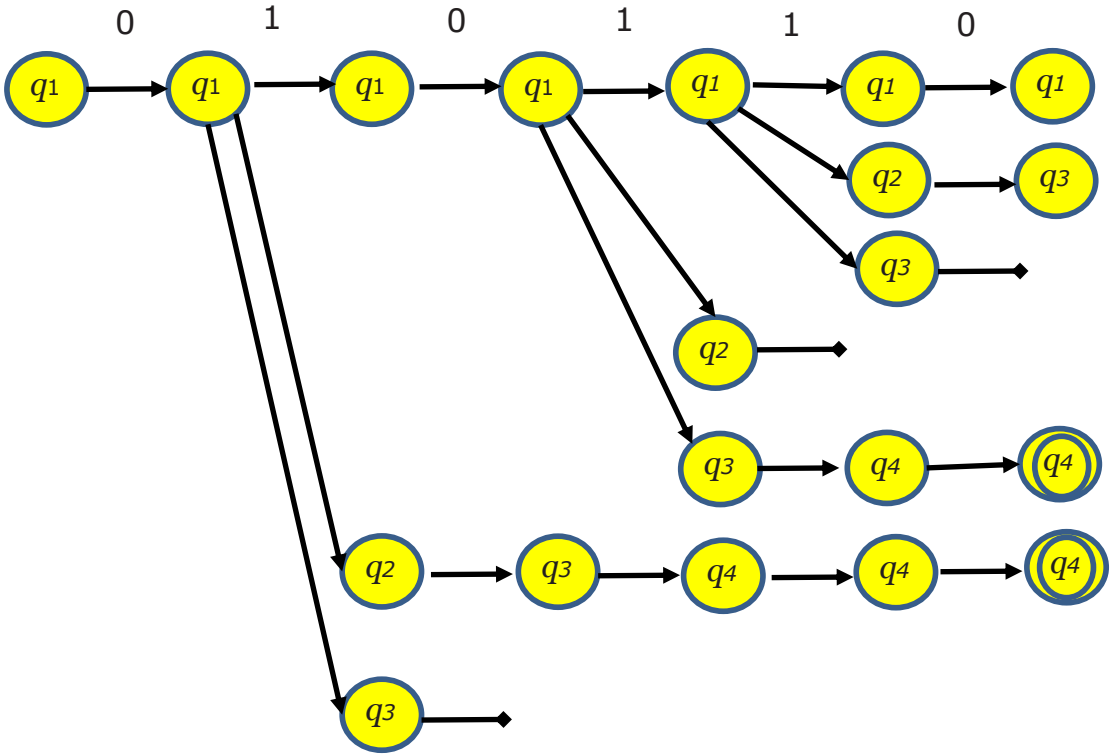
- Similarly, if a state with an ε-transition is encountered,
  - Without reading an input symbol, NFA splits into multiple copies,
  - Each one following an exiting ε-transition (or staying put).
  - NFA follows all possible paths in parallel (independently of others)
  - NFA proceeds **nondeterministically** as before.

- Tracing a NFA.
- What happens on input string 010110 ?

NFA Computation



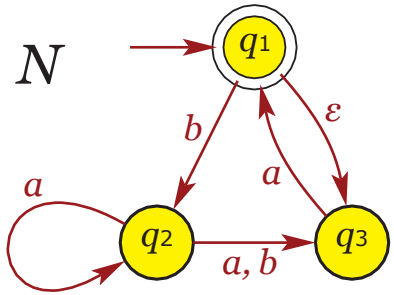
- Tracing a NFA..What happens on input string 010110 ?



- NFA splits into multiple copies of itself..
  - Each copy proceeds with computation independently of others.
  - NFA may be in a set of states, instead of a single state.
  - NFA follows all possible computation paths in parallel.
  - If a copy is in a state and next input symbol doesn't appear on any outgoing edge from the state, then the copy dies or crashes.

NFA Eg.

Example: NFA N



- N accepts strings $\varepsilon, a, aa, baa, baba, \ldots.$

e.g., $aa = \varepsilon a \varepsilon a$

- N does not accept (i.e., rejects) strings $b, ba, bb, bbb, \ldots.$

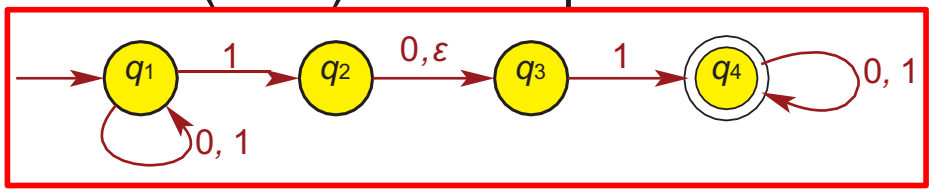
- "Outside Observer": is there a path labeled by x from the start state, to the final state (if we know the input in advance can we tell the NFA which decisions to make)
- "Perfect Guesser": The NFA has input x, and whenever there is a choice of what to do, it magically guesses a transition that will eventually lead to acceptance (if one exists)
- "Parallel exploration": The NFA computation runs all possible computations on x in parallel (updating each possible one at every step)

NFA Formal Definition

Definition: For an alphabet Σ, define $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$.

Σ_ε is set of possible labels on NFA edges.

Definition: A nondeterministic finite automaton (NFA) is a 5-tuple $N = (Q, \Sigma, \delta, q_0, F)$, where



1. Q is a finite set of states — $Q = \{q_1, q_2, q_3, q_4\}$
2. Σ is an alphabet — $\Sigma = \{0, 1\}$
3. $\delta : Q \times \Sigma_\varepsilon \rightarrow P(Q)$ is the transition function, where $P(Q)$ is the power set of Q

δ	0	1	ε
q_1	$\{q_1\}$	$\{q_1, q_2\}$	$\emptyset$
q_2	$\{q_3\}$	$\emptyset$	$\{q_3\}$
q_3	$\emptyset$	$\{q_4\}$	$\emptyset$
q_4	$\{q_4\}$	$\{q_4\}$	$\emptyset$

4. $q_0 \in Q$ is the start state
5. $F \subseteq Q$ is the set of accept states.

- q_1 is the start state
- $F = \{q_4\}$ is the set of accept states

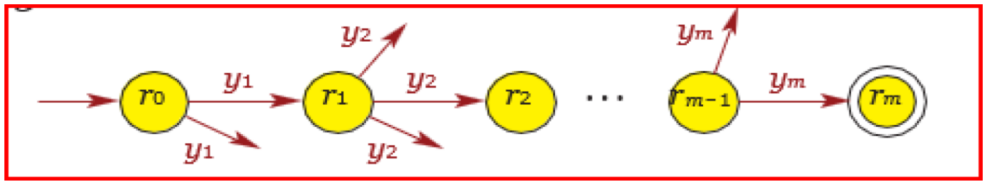
Power Set of a set S is all subsets of S – Denoted by $P(S)$ or 2^S

$s = \{q_1, q_2\}$; $P(S) = \{\emptyset, \{q_1\}, \{q_2\}, \{q_1, q_2\}\}$

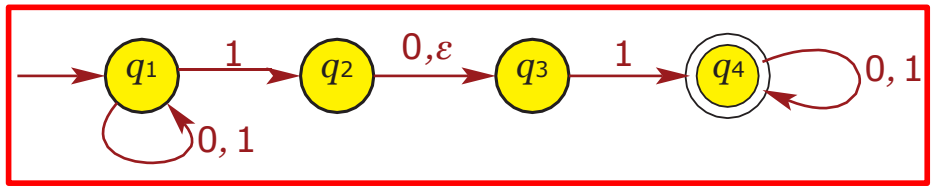
$|P(S)| = 2^{|S|}$

Formal Definition of NFA Computation

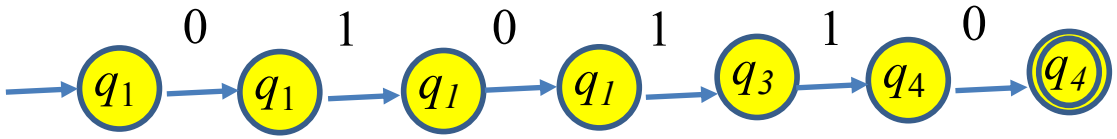
- Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA and $w \in \Sigma^*$.
- Then N **accepts** w if
  - If there is **some** $w = y_1 y_2 \cdots y_m$ for some $m \geq 0$, where each $y_i \in \Sigma_\varepsilon$, and there is a sequence of states $r_0, r_1, r_2, \ldots, r_m$ in Q such that
    1. $r_0 = q_0$
    2. $r_{i+1} \in \delta(r_i, y_{i+1})$ for each $i = 0, 1, 2, \ldots, m-1$
    3. $r_m \in F$



Eg: Continuing…



	0	1	ε
q_1	$\{q_1\}$	$\{q_1, q_2\}$	$\emptyset$
q_2	$\{q_3\}$	$\emptyset$	$\{q_3\}$
q_3	$\emptyset$	$\{q_4\}$	$\emptyset$
q_4	$\{q_4\}$	$\{q_4\}$	$\emptyset$



DFA vs NFA (Comparison)

innovate achieve lead

Difference	
DFA	NFA
Σ consist of only symbols excluding ε	$\Sigma_\varepsilon = \Sigma \cup \epsilon$
For each state and for each symbol of the input alphabet, the transition to the next is unique and must be specified.	Allows for several or no choices to exist for the transition from the next state on a given input symbol (including ϵ).
transition function $\delta : Q \times \Sigma \rightarrow Q$	$\delta : Q \times \Sigma_\varepsilon \rightarrow P(Q)$ ($P(Q)$:Power Set)
Input is accepted if the only path starting from the start state leads to a final/accept state.	Input is accepted if any of the possible paths from the start state leads to an accept state.
Usually harder to construct and may have more states that the equivalent NFA	Usually its easier to construct NFA and has lesser states than the corresponding DFA.
Efficient execution ($T_{DFA} = O(k)$) k = $\|w\|$ length of input string	Inefficient execution ($T_{NFA} = O(k*f(m,n))$) k = $\|w\|$ length of input string m,n = states, transitions
Every DFA is a NFA	Every NFA is not a DFA

DFA vs NFA (Comparison)

innovate achieve lead

Similarity

- Both DFA and NFA accept the **same class of Languages** called **Regular Languages**.
- Parallel (or non deterministic execution) **does not** add any computation **power** to NFAs.
- NFAs have the same power as DFAs.
- Power of a machine is defined based on the class of languages recognized by the machine.

NFAs more powerful than DFA??

- We know that DFAs accept regular languages.
- Are NFAs strictly more powerful than DFAs?

- Are there languages that some NFA will accept but no DFA can accept?

- It turns out that NFAs and DFAs accept the same set of languages.

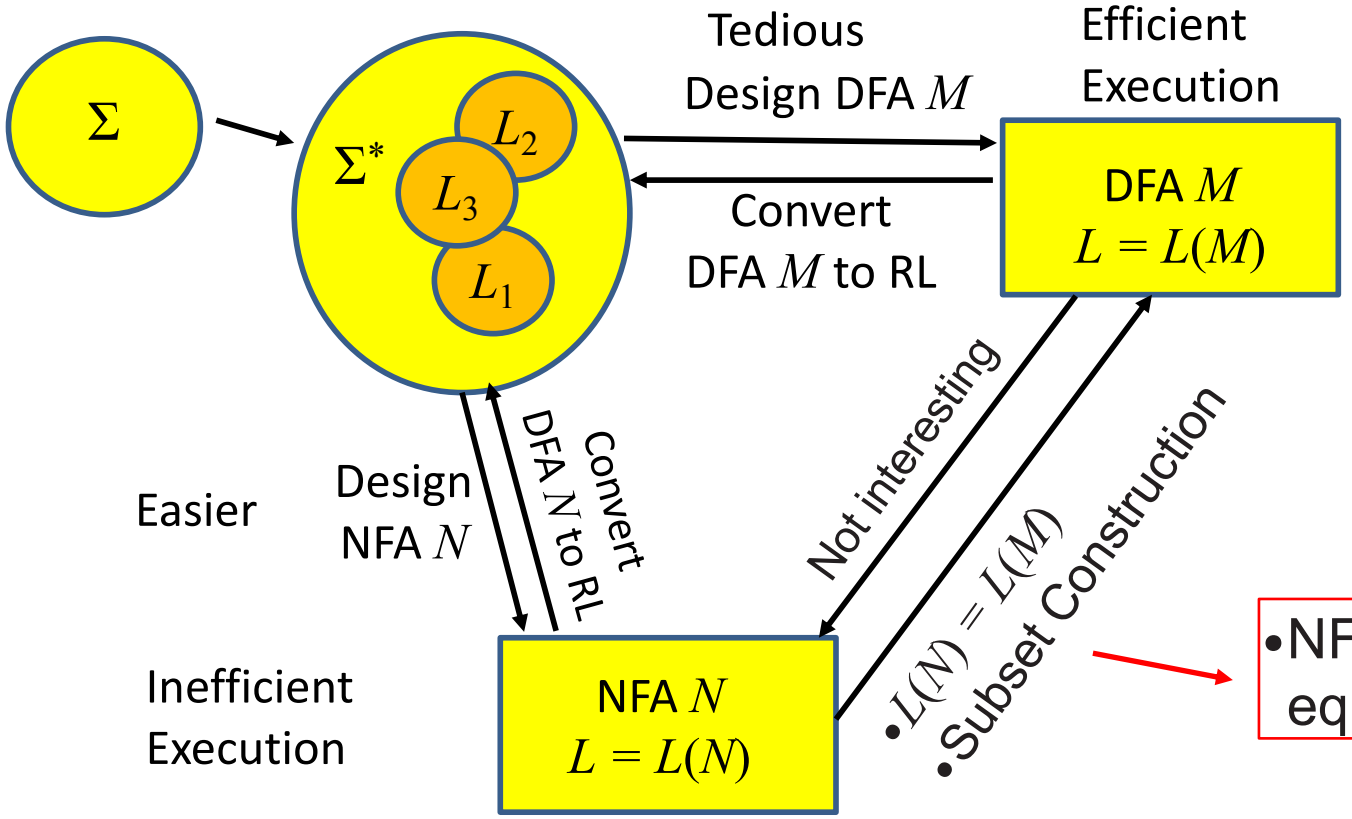
- Q is finite $\Rightarrow |2^Q| = |\{P|P \subseteq Q`\ | = 2^{|Q|}$ is also finite.

- Surprising that adding something as powerful as guessing/parallelism to the DFA model could turn out to not increase the power of the model!

- NFA execution is inefficient (due to backtracking)

- DFA execution is efficient (proportional to length of input string)

RL ⇔ NFA ⇔ DFA



- $L(M) = L$.

- Closure Property
- $L = L_1 \cup L_2$ *is regular*
- $L(M_1) = L_1$, $L(M_2) = L_2$
- $L(M) = L$
- $M = M_1 \times M_2$
- Tedious

- NFAs and DFAs are equivalent

- Closure Property
- $L = L_1 \cup L_2$ *is regular*
- $L(N_1) = L_1$, $L(N_2) = L_2$
- $L(N) = L$
- $N = N_1 \cup N_2$
- Use ε transitions
- Easier

- Allows DFAs to be constructed from RLs indirectly through a 2 step process
  - Design NFA
  - NFA => DFA

RL ⇔ NFA ⇔ DFA

- A language *L* is regular if and only if it is recognized by some DFA(M)
- Class of Regular Language are closed under Regular Operations
  - Regular Operations: Union, Concatenation, Kleene Star
  - ie. There exits DFA *M* such that *L* can be recognized.

- $L(M) = L$.
- $L = L_1 \cup L_2$ *is regular*
- $L(M_1) = L_1$, $L(M_2) = L$
- $L(M) = L$

- Every NFA (N) has a equivalent DFA (M)
  - Subset construction (Proof)

- $L(M) = L(N)$

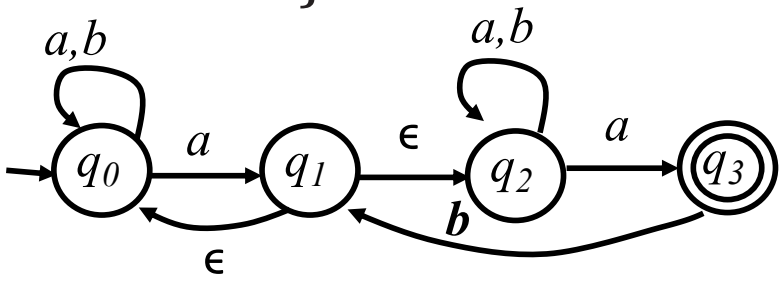
- Language L is regular if and only if some NFA (N) recognizes L

- $L(N) = L$

  - Proof. ($L \Rightarrow L(N)$)
  - (⇒) If L is regular then there is an NFA N that accepts L. $L \Rightarrow L(N)$
    - If *L* is regular, then there is a DFA for it. ($L(M) \Rightarrow L$.)
    - Every DFA M has an equivalent NFA N, so there is an NFA for L.
    - i.e. $L(M) \Rightarrow L$, $L(M) \Rightarrow L(N)$. So $L \Rightarrow L(N)$.
  - (⇐) If NFA N exists for L then L is regular. $L(N) \Rightarrow L$.
  - Every NFA has an equivalent DFA. $L(N) \Rightarrow L(M)$; $L(M) \Rightarrow L$; $L(N) \Rightarrow L$

ε-Closure (of a set of states)

- Definition: The ε-closure of a set of states $R \subseteq Q$ is $\varepsilon C(R) = \{ q \mid q$ can be reached transitively from R by travelling over 0 or more ε transitions $\}$.



$\varepsilon C(q_0) = \{q_0\}$

$\varepsilon C(q_1) = \{q_1, q_0, q_2\}$

$\varepsilon C(q_2) = \{q_2\}$

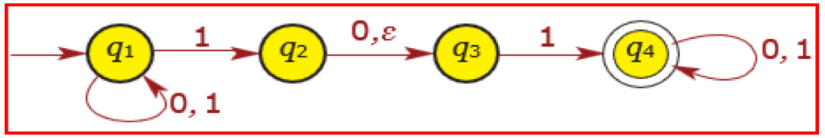
$\varepsilon C(q_3) = \{q_3\}$

NFA conversion to DFA (Subset)

Definition: Two machines (of any types) are equivalent if they recognize the same language. (Recollect: L = { w | w has some property}

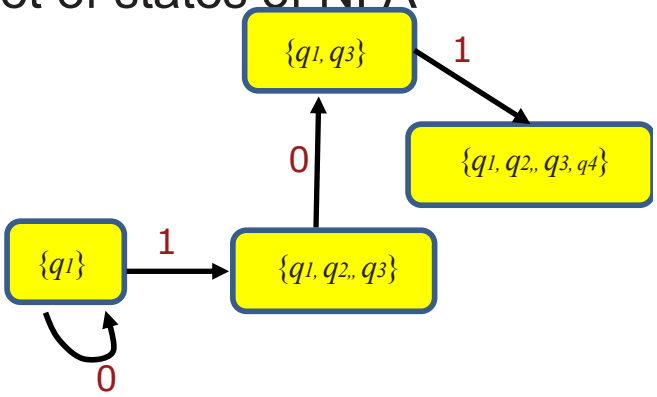
Theorem: Every NFA *N* has an equivalent DFA *M* .

- i.e., if *N* is some NFA, then ∃ DFA *M* such that $L(M) = L(N)$.



Proof Idea:

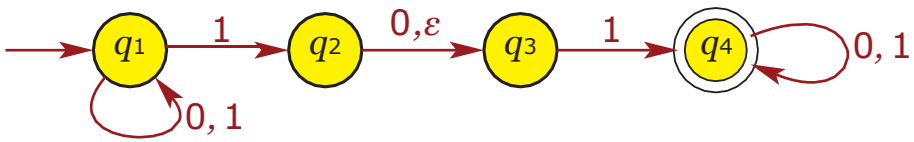
- NFA *N* splits into multiple copies of itself on nondeterministic moves.
- NFA can be in a set of states at any one time.
- Build DFA *M* whose set of states is the power set of the set of states of NFA



NFA to DFA conversion

innovate achieve lead

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



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

- Definition: The ε-closure of a set of states $R \subseteq Q$ is $E(R)$ = { q | q can be reached transitively from R by travelling over 0 or more ε transitions }.

q_0' = εC({q_1}) = {q_1} = A

$\delta'(q_1, 0)$ = εC(δ({q_1},0)) = εC({q_1}) = {q_1}

$\delta'(q_1, 1)$ = εC(δ({q_1},1))
= εC({q_1, q_2})
= εC({q_1}) ∪ εC({q_2})
= {q_1} ∪ {q_2, q_3} = {q_1, q_2, q_3}

$\delta'(\{q_1, q_2, q_3\}, 0)$ = εC(δ({q_1, q_2, q_3},0))
= εC(δ({q_1},0) ∪ δ({q_2},0) ∪ δ({q_3},0))
= εC({q_1}) ∪ εC({q_3}) ∪ ∅
= {q_1} ∪ {q_3} = {q_1, q_3}

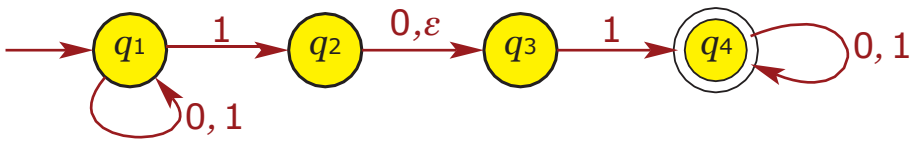
$\delta'(\{q_1, q_2, q_3\}, 1)$ = εC(δ({q_1, q_2, q_3},1))
= εC(δ({q_1},1) ∪ δ({q_2},1) ∪ δ({q_3},1))
= εC({q_1, q_2}) ∪ ∅ ∪ εC({q_4})
= {q_1, q_2, q_3} ∪ {q_4} = {q_1, q_2, q_3, q_4}

{q_1, q_2} {q_1, q_2, q_3, q_4}

	0	1
{q_1}	{q_1}	{q_1, q_2, q_3}
{q_1, q_2, q_3}	{q_1, q_3}	{q_1, q_2, q_3, q_4}

NFA to DFA conversion

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



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

	0	1
$\{q_1\}$	$\{q_1\}$	$\{q_1, q_2, q_3\}$
$\{q_1, q_2, q_3\}$	$\{q_1, q_3\}$	$\{q_1, q_2, q_3, q_4\}$
$\{q_1, q_3\}$	$\{q_1\}$	$\{q_1, q_2, q_3, q_4\}$
$\{q_1, q_2, q_3, q_4\}$	$\{q_1, q_3, q_4\}$	$\{q_1, q_2, q_3, q_4\}$

$\delta'(\{q_1, q_3\}, 0) = \varepsilon C(\delta(\{q_1, q_3\},0))$
$= \varepsilon C(\delta(\{q_1\},0) \cup \delta(\{q_3\},0))$
$= \varepsilon C(\{q_1\}) \cup \emptyset$
$= \{q_1\}$

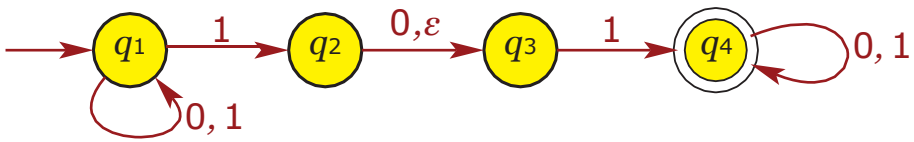
$\delta'(\{q_1, q_3\}, 1) = \varepsilon C(\delta(\{q_1, q_3\},1))$
$= \varepsilon C(\delta(\{q_1\},1) \cup \delta(\{q_3\},1))$
$= \varepsilon C(\{q_1, q_2\}) \cup \varepsilon C(\{q_4\})$
$= \{q_1, q_2, q_3\} \cup \{q_4\} = \{q_1, q_2, q_3, q_4\}$

$\delta'(\{q_1, q_2, q_3, q_4\}, 0) = \varepsilon C(\delta(\{q_1, q_2, q_3, q_4\},0))$
$= \varepsilon C(\delta(\{q_1\},0) \cup \delta(\{q_2\},0) \cup \delta(\{q_3\},0)) \cup \delta(\{q_4\},0)$
$= \varepsilon C(\{q_1\}) \cup \varepsilon C(\{q_3\}) \cup \emptyset \cup \varepsilon C(\{q_4\})$
$= \{q_1\} \cup \{q_3\} \cup \{q_4\} = \{q_1, q_3, q_4\}$

$\delta'(\{q_1, q_2, q_3, q_4\}, 1) = \varepsilon C(\delta(\{q_1, q_2, q_3, q_4\},1))$
$= \varepsilon C(\delta(\{q_1\},1) \cup \delta(\{q_2\},1) \cup \delta(\{q_3\},1)) \cup \delta(\{q_4\},1)$
$= \varepsilon C(\{q_1, q_2\}) \cup \emptyset \cup \varepsilon C(\{q_4\}) \cup \varepsilon C(\{q_4\})$
$= \{q_1, q_2, q_3\} \cup \{q_4\} \cup \{q_4\} = \{q_1, q_2, q_3, q_4\}$

NFA to DFA conversion

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



$\delta'(\{q_1, q_3, q_4\}, 0) = \varepsilon C(\delta(\{q_1, q_3, q_4\},0))$
$= \varepsilon C(\delta(\{q_1\},0) \cup \delta(\{q_3\},0) \cup \delta(\{q_4\},0))$
$= \varepsilon C(\{q_1\}) \cup \emptyset \cup \varepsilon C(\{q_4\})$
$= \{q_1\} \cup \{q_4\} = \{q_1, q_4\}$

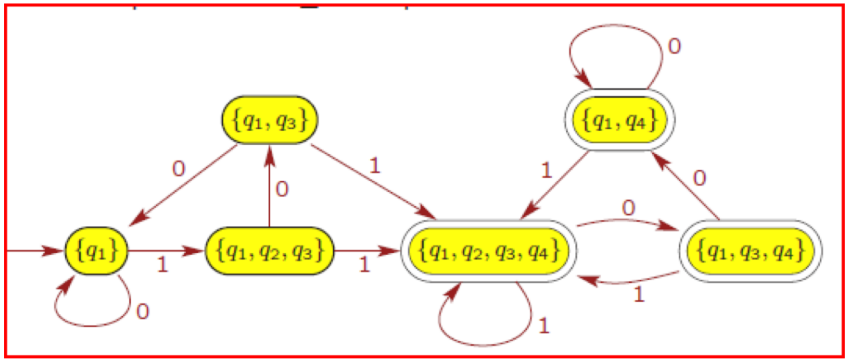
$\delta'(\{q_1, q_3, q_4\}, 1) = \varepsilon C(\delta(\{q_1, q_3, q_4\},1))$
$= \varepsilon C(\delta(\{q_1\},1) \cup \delta(\{q_3\},1) \cup \delta(\{q_4\},1))$
$= \varepsilon C(\{q_1, q_2\}) \cup \varepsilon C(\{q_4\}) \cup \varepsilon C(\{q_4\})$
$= \{q_1, q_2, q_3\} \cup \{q_4\} = \{q_1, q_2, q_3, q_4\}$

$\delta'(\{q_1, q_4\}, 0) = \varepsilon C(\delta(\{q_1, q_4\},0))$
$= \varepsilon C(\delta(\{q_1\},0) \cup \delta(\{q_4\},0))$
$= \varepsilon C(\{q_1\}) \cup \varepsilon C(\{q_4\})$
$= \{q_1\} \cup \{q_4\} = \{q_1, q_4\}$

$\delta'(\{q_1, q_4\}\ 1) = \varepsilon C(\delta(\{q_1, q_4\},1))$
$= \varepsilon C(\delta(\{q_1\},1) \cup \delta(\{q_4\},1)$
$= \varepsilon C(\{q_1, q_2\}) \cup \varepsilon C(\{q_4\})$
$= \{q_1, q_2, q_3\} \cup \{q_4\} = \{q_1, q_2, q_3, q_4\} = D$

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

	0	**1**
$\{q_1\}$	$\{q_1\}$	$\{q_1, q_2, q_3\}$
$\{q_1, q_2, q_3\}$	$\{q_1, q_3\}$	$\{q_1, q_2, q_3, q_4\}$
$\{q_1, q_3\}$	$\{q_1\}$	$\{q_1, q_2, q_3, q_4\}$
$\{q_1, q_2, q_3, q_4\}$	$\{q_1, q_3, q_4\}$	$\{q_1, q_2, q_3, q_4\}$
$\{q_1, q_3, q_4\}$	$\{q_1, q_4\}$	$\{q_1, q_2, q_3, q_4\}$
$\{q_1, q_4\}$	$\{q_1, q_4\}$	$\{q_1, q_2, q_3, q_4\}$



NFA => DFA Algo

- Every NFA $N = (Q, \Sigma, \delta, q_0, F)$ has an equivalent DFA $M = (Q', \Sigma, \delta', q_0', F')$.

a. $q_0' = \varepsilon C(\{q_0\})$, $Q' = q_0'$
b. while there is an unmarked state X in Q'
c. mark X
d. for a ∈ **Σ**
e. Y = {}
f. for q ∈ X
g. Y = Y U Q(q, a)
h. Z = εC(Y)
i. if Z ∉ Q'
j. $Q' = Q' \cup Z$
k. $\delta'[X, a] = Z$
l. $F' = \{X \in Q' \mid X \cap F \neq \emptyset\}$

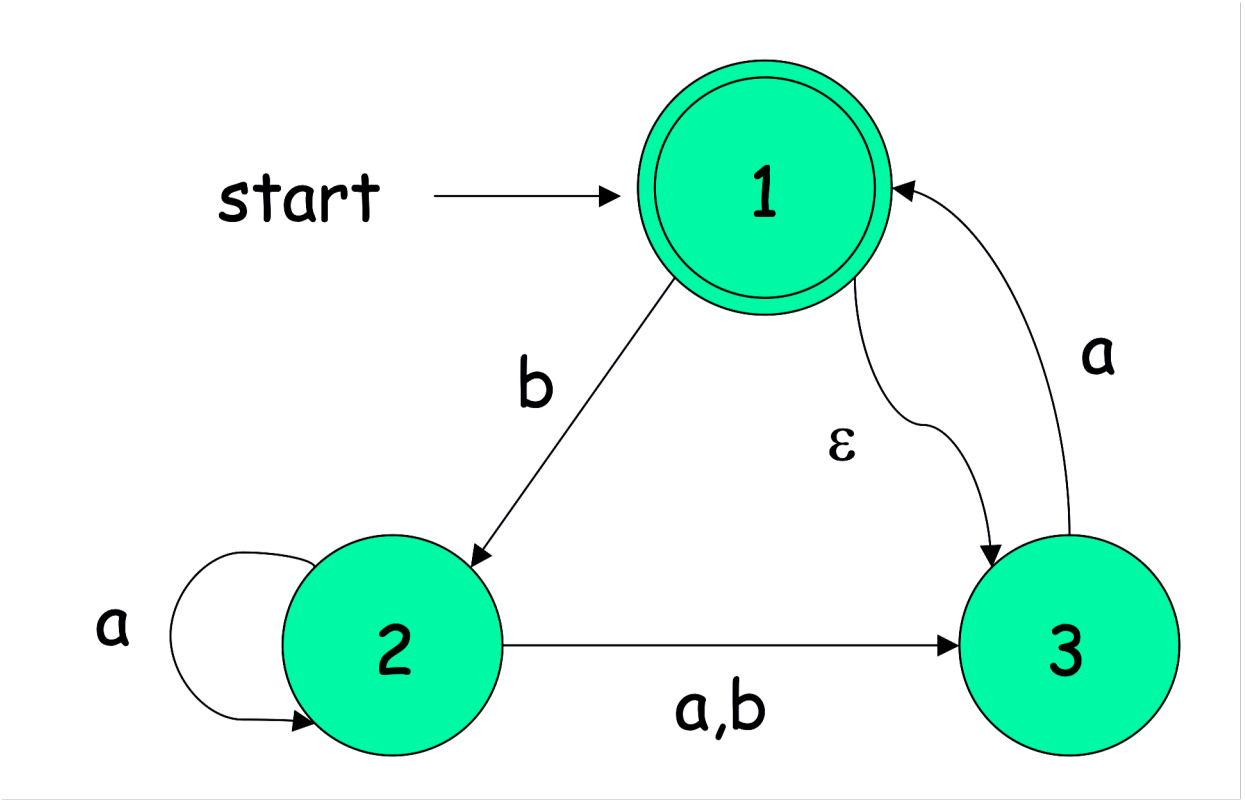
Example: NFA to DFA Conversion

Design a NFA N where

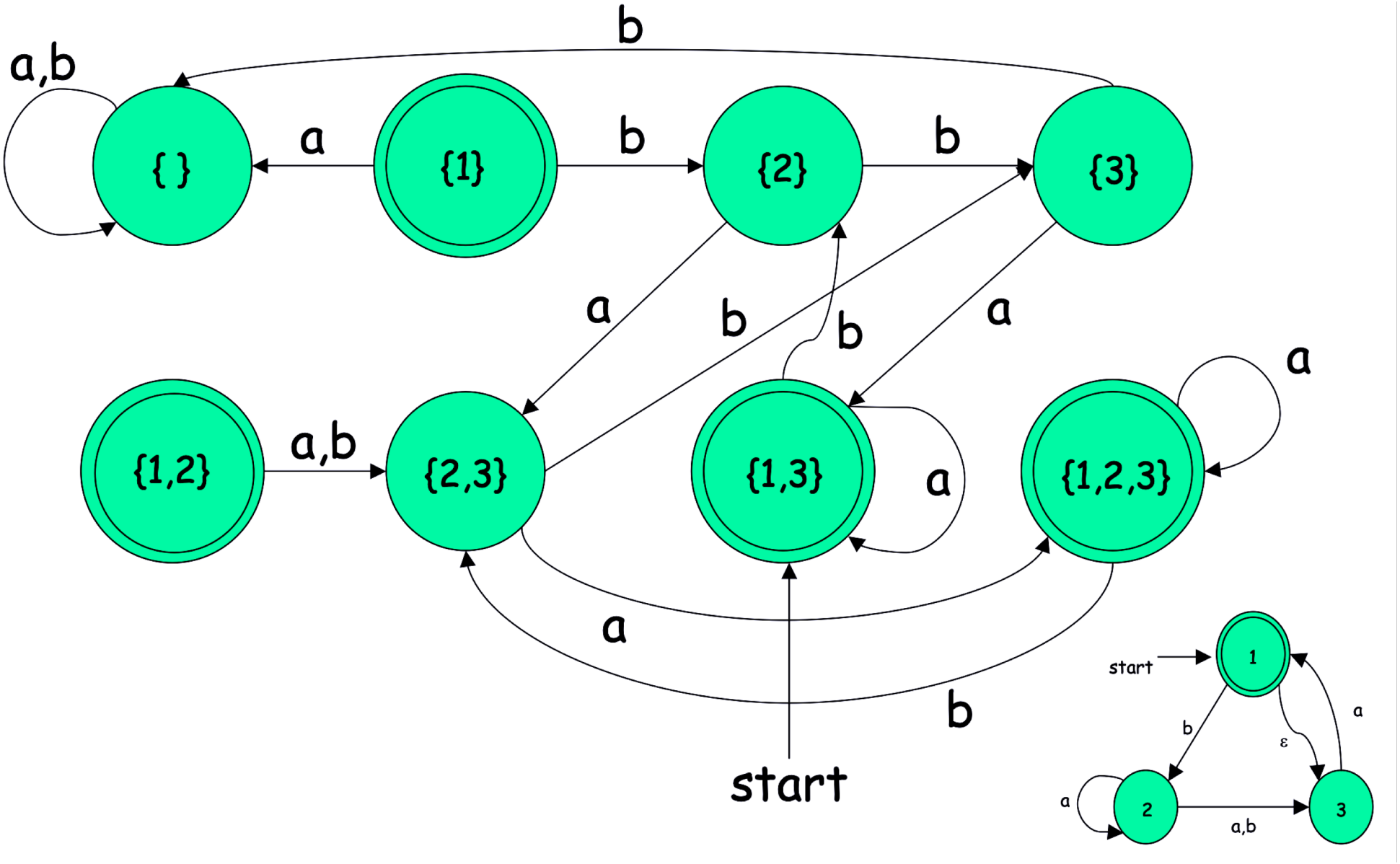
- N accepts strings $\varepsilon, a, aa, baa, baba, \ldots.$
- N does not accept (i.e., rejects) strings $b, ba, bb, bbb, \ldots.$

And derive the equivalent DFA M for the given N

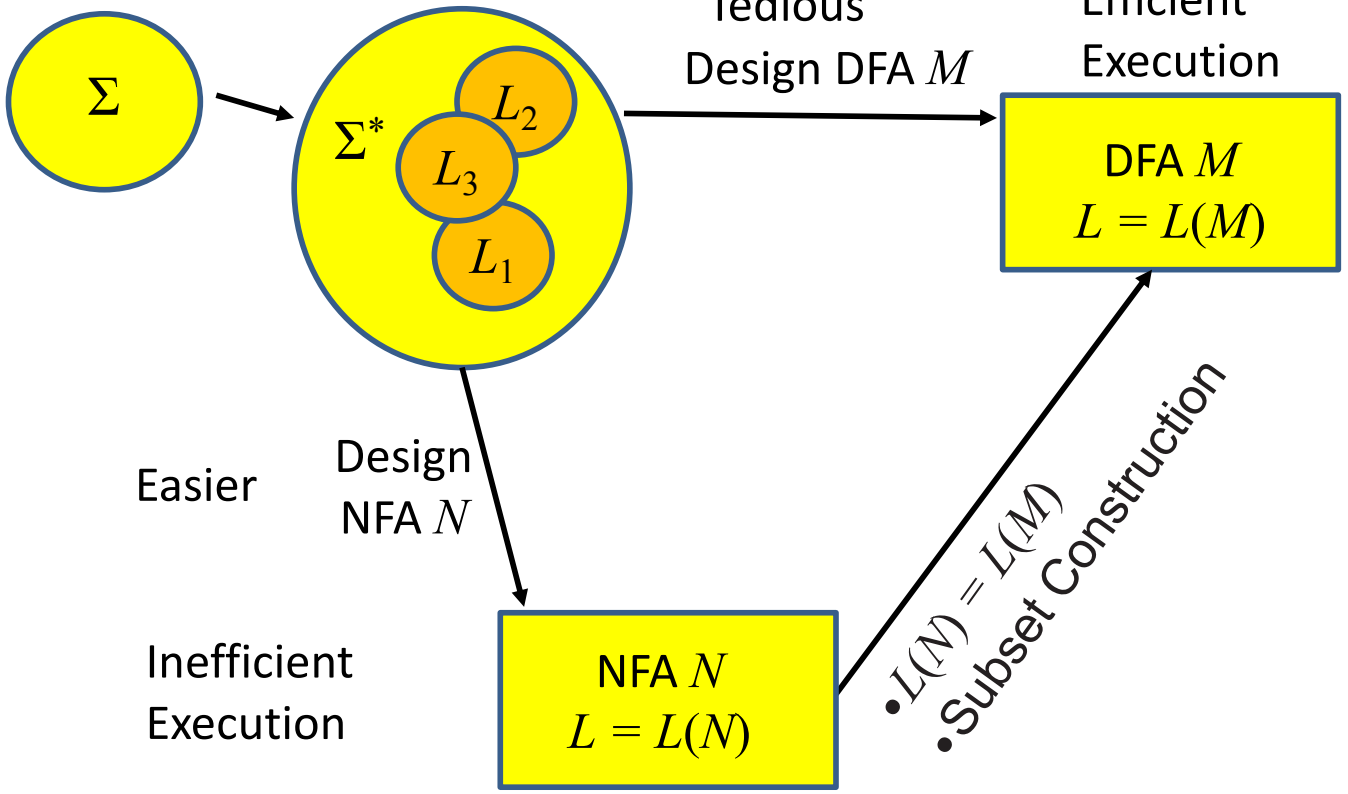
Example NFA to DFA conversion



Example NFA to DFA conversion



RL ⇔ NFA ⇔ DFA



- $L(M) = L$

- Closure Property
- $L = L_1 \cup L_2$ *is regular*
- $L(M_1) = L_1$, $L(M_2) = L_2$
- $L(M) = L$
- $M = M_1 \times M_2$
- Tedious

- Closure Property
- $L = L_1 \cup L_2$ *is regular*
- $L(N_1) = L_1$, $L(N_2) = L_2$
- $L(N) = L$
- $N = N_1 \cup N_2$
- Use ε transitions
- Easier

- Every NFA (N) has a equivalent DFA (M)

- Can we combine NFA's and produce a single equivalent NFA
  - Combine: Union, Concatenation, Kleen Star etc..

Closure properties of Regular Lang

- Class of Regular Language is close under Regular Operations
  - Regular Operations: Union, Concatenation, Kleene Star
  - ie. There exits some DFA M such that L can be recognized.

- Union : The question is
  L = L1 U L2
  Can we have a DFA(M) such that DFA(M) will accept L. i.e L(M) = L1 U L2
  How to construct a DFA(M) which is DFA(M1) U DFA(M2)
  - Cross production Construction process is “cumbersome” (informally)

- Alternate Method (Why?)
- NFAs are usually easier to construction
- And also to combine (i.e. union, intersection etc.)

- Can we have a NFA(N) such that NFA(N) will accept L. i.e L(N) = L1 U L2
  How to construct a NFA(N) which is NFA(N1) U NFA(N2)

Reg Lang is closed under Union

Union:

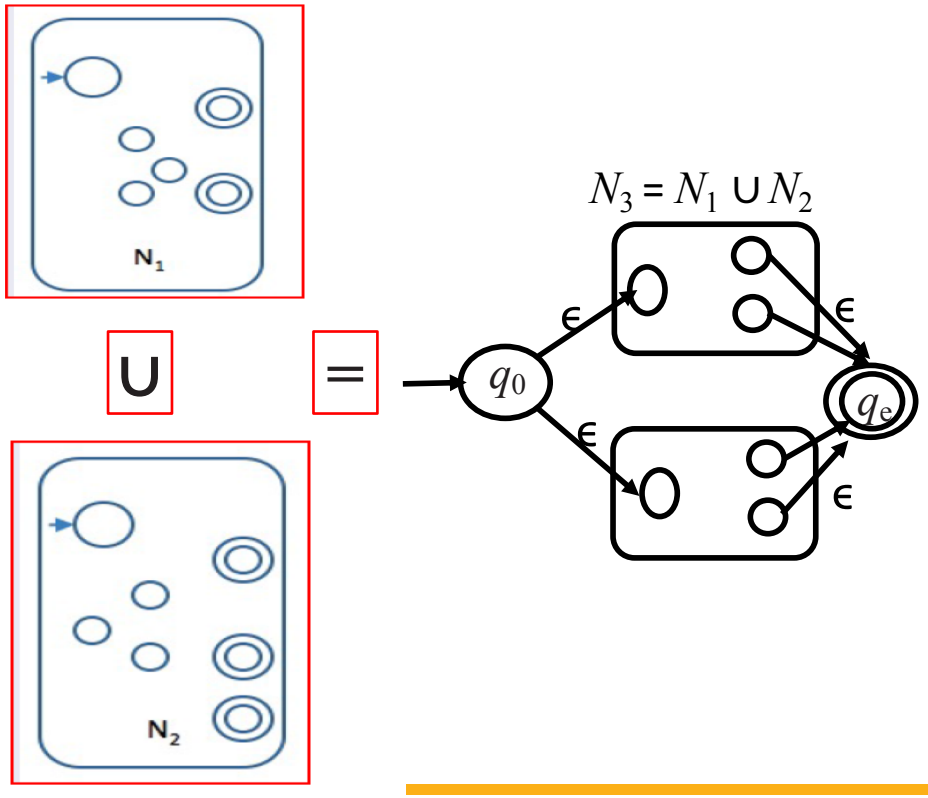
$$L_1 \cup L_2 = \{ w \mid w \in L_1 \text{ or } w \in L_2 \}.$$

Theorem:
The class of regular languages is closed under union.

Proof Idea: Given NFAs N_1 and N_2 for L_1 and L_2, resp., construct NFA N for $L_1 \cup L_2$ as follows:

- Construct NFA for $L_1 \cup L_2$ from NFAs for L_1 and L_2

- L(N1) = L1; NFA $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$
- L(N2) = L2; NFA $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$
- Construct NFA $N = (Q, \Sigma, \delta, q_0, F)$ for $L_1 \cup L_2$:



- Introduce q_0, q_e as a new start,end state of N.
- $Q = \{q_0, q_e\} \cup Q_1 \cup Q_2$ is set of states of N.
- Set of accept states $F = \{q_e\}$
- For $q \in Q$ and $a \in \Sigma_\varepsilon$, δ satisfies

$$\delta(q,a) = \begin{cases} \delta_1(q,a), & \text{if } q \in Q_1 \\ \delta_2(q,a) & \text{if } q \in Q_2 \\ \{q_1, q_2\} & q = q_0 \text{ and } a = \epsilon \\ \{q, q_e\} & q \in N_1 \cup N_2 \text{and } a = \epsilon \\ \phi & q = q_0 \text{ and } a \neq \epsilon \end{cases}$$

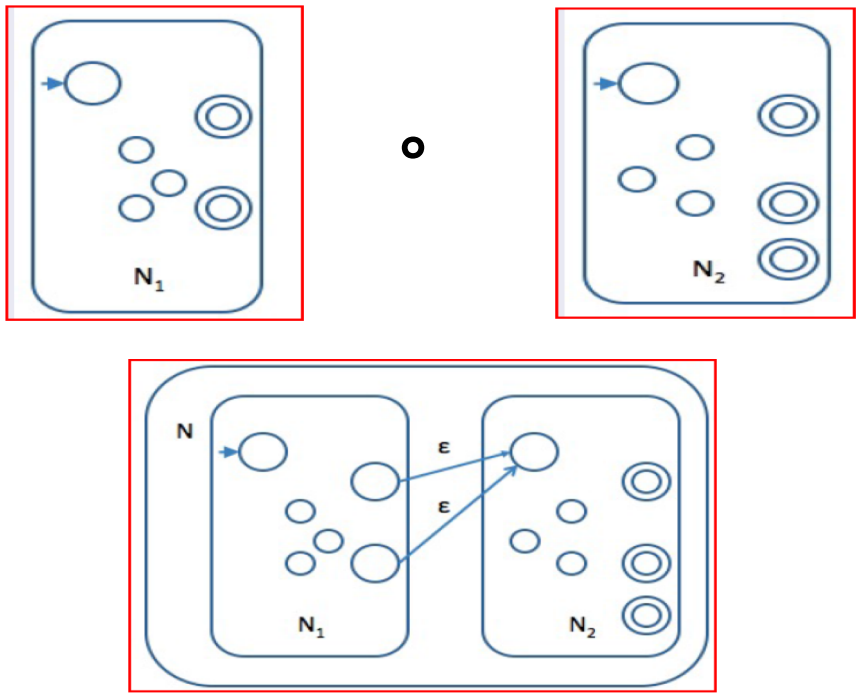
Reg Lang is closed under Concatenation

Concatenation:

$$L_1 \circ L_2 = \{ xy \mid x \in L_1 \text{ and } y \in L_2 \}.$$

Theorem:
The class of regular languages is closed under concatenation.

Proof Idea: Given NFAs N_1 and N_2 for L_1 and L_2, resp., construct NFA N for $L_1 \circ L_2$ as follows:



- Construct NFA for $L_1 \circ L_2$ from NFAs for L_1 and L_2

- $L(N_1) = L_1$; NFA $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$.
- $L(N_2) = L_2$; NFA $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$.
- Construct NFA $N = (Q, \Sigma, \delta, q_0, F)$ for $L_1 \circ L_2$:

- q_1 is start state of N.
- $Q = Q_1 \cup Q_2$ is set of states of N.
- Set of accept states $F = F_2$.
- For $q \in Q$ and $a \in \Sigma_\varepsilon$, δ satisfies

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & \text{if } q \in Q_1 - F_1 \\ \delta_1(q,a) & \text{if } q \in F_2 \text{ and } a \neq \epsilon \\ \delta_1(q,a) \cup \{q_2\} & \text{if } q = F_1 \text{ and } a = \epsilon \\ \delta_2(q,a) & \text{if } q \in Q_2 \end{cases}$$

Reg Lang is closed under Kleene Star

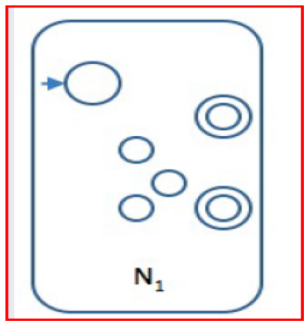
Kleene Star:

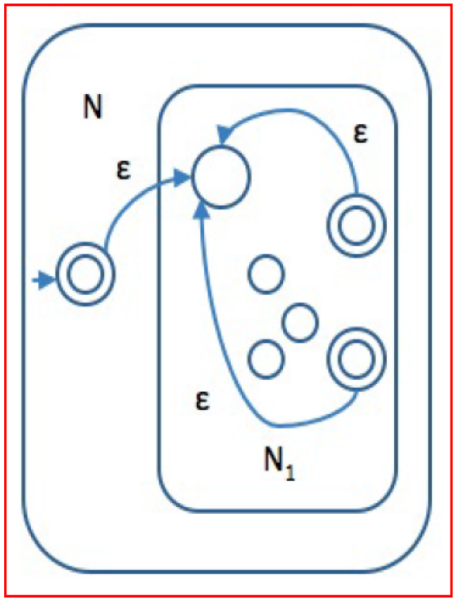
$L^* = \{ x_1 x_2 \cdots x_k \mid k \geq 0 \text{ and each } x_i \in A \}.$

Theorem:

The class of regular languages is closed under Kleene-start operation

Proof Idea: Given NFA N_1 for L, construct NFA N for L^*:





- Construct NFA for L^* from NFA for L

- $L(N_1) = L$; NFA $N_1 = (Q, \Sigma, \delta_1, q_1, F_1)$.
- Construct NFA $N = (Q, \Sigma, \delta, q_0, F)$ for L^* :

- $Q = \{q_0\} \cup Q_2$ is set of states of N.
- q_0 is start state of N.
- Set of accept states $F = \{q_0\} \cup F_1$.
- For $q \in Q$ and $a \in \Sigma_\varepsilon$, δ satisfies

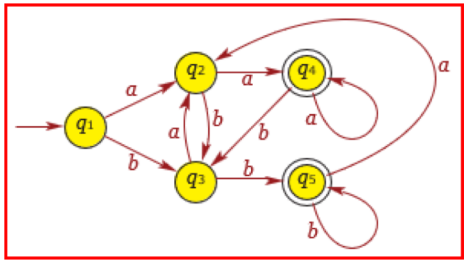
$$\delta(q,a) = \begin{cases} \delta_1(q,a) & \text{if } q \in Q_1 - F_1 \\ \delta_1(q,a) & \text{if } q \in F_1 \text{ and } a \neq \epsilon \\ \delta_1(q,a) \cup \{q_1\} & \text{if } q = F_1 \text{ and } a = \epsilon \\ \{q_1\} & \text{if } q = q_0 \text{ and } a = \epsilon \\ \phi & \text{if } q = q_0 \text{ and } a = \epsilon \end{cases}$$

Eg. Design automaton (L1 U L2)

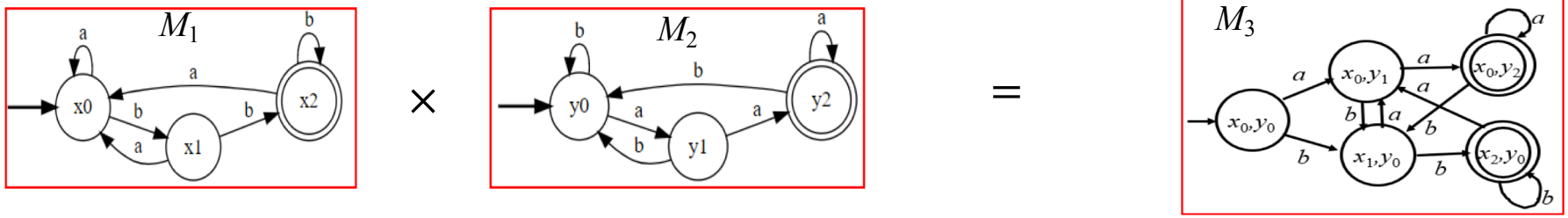
- $L_1 \cup L_2 = \{x \mid x \in L_1 \text{ or } x \in L_2\}$
- Example:
  - $L_1 = \{wbb \mid w \in \{a,b\}^*\}$ = strings that ends with bb
  - $L_2 = \{waa \mid w \in \{a,b\}^*\}$ = strings that ends with aa
  - $L_3 = L_1 \cup L_2 = \{x \in \{a,b\}^* \mid x$ ends with bb **or** $aa\}$

- How to construct DFA M_3 such that M_3 recognizes L_3 (i.e. $L(M_3) = M_3$)

  1. Construct the DFA M_3 directly from scratch

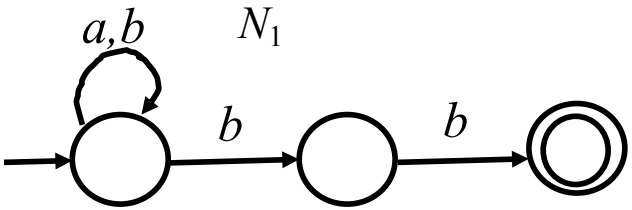


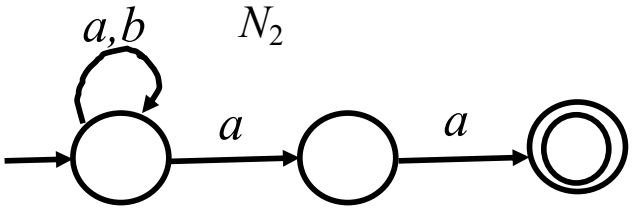
  2. Construct the DFA M_3 from M_1 and M_2 (cross product of DFAs)

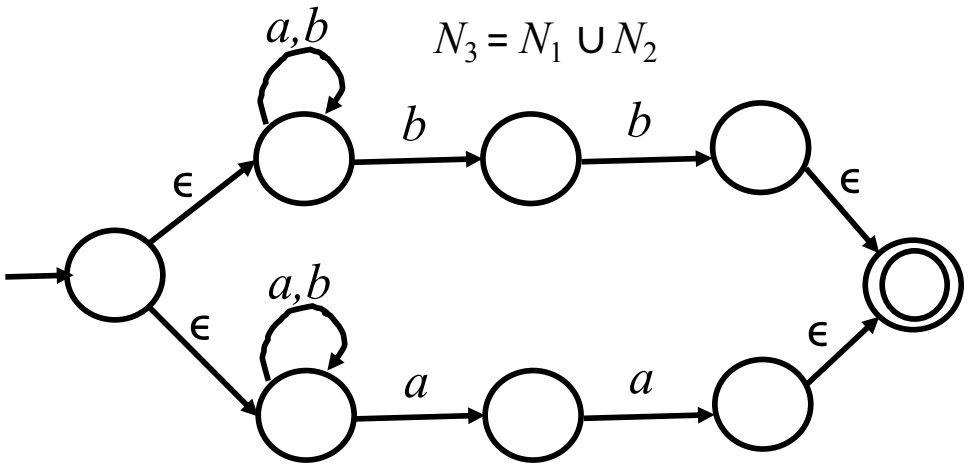


Eg. Design automaton (L1 U L2)

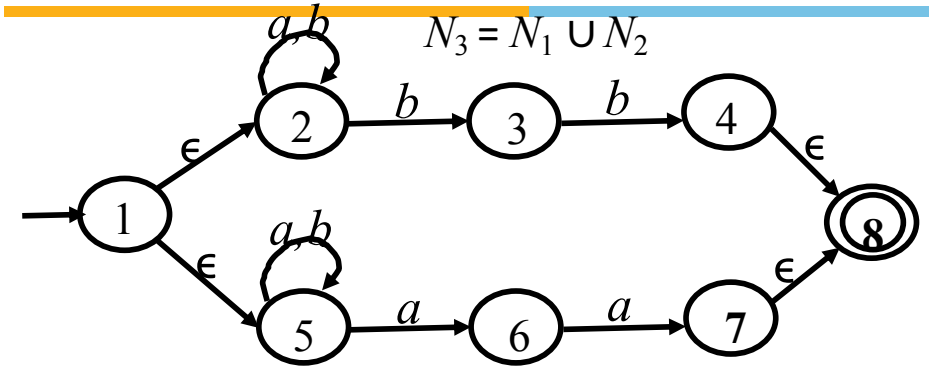
- 3. Construct the DFA M_3 from NFAs of L_1 and L_2
  - a) Construct NFA N_1 ,for L_1 (i.e. $L(N_1) = L_1$), NFA N_2 ,for L_2 (i.e. $L(N_2) = L_2$)
  - b) Construct NFA N_3 = $N_1 \cup N_2$ (i.e. $L(N_3) = L_3$)
  - c) Convert NFA N_3 to DFA M_3 (i.e. $L(M_3) = L_3$) (Subset construction)



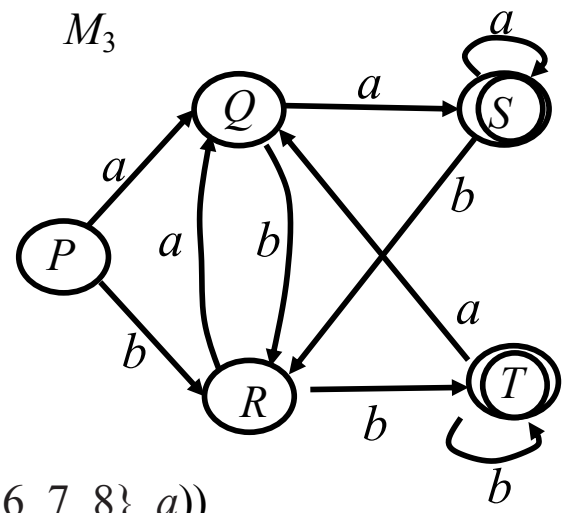




Eg. Design automaton (L1 U L2)



δ	a	b



$q_0 = \varepsilon C(\{1\}) = \{1, 2, 5\} = P$

$\delta'(P, a) = \varepsilon C(\delta(\{1, 2, 5\}, a))$
$= \varepsilon C(\{2, 5, 6\}) = \{2, 5, 6\} = Q$

$\delta'(P, b) = \varepsilon C(\delta(\{1, 2, 5\}, b))$
$= \varepsilon C(\{2, 3, 5\}) = \{2, 3, 5\} = R$

$\delta'(Q, a) = \varepsilon C(\delta(\{2, 5, 6\}, a))$
$= \varepsilon C(\{2, 5, 6, 7\}) = \{2, 5, 6, 7, 8\} = S$

$\delta'(Q, b) = \varepsilon C(\delta(\{2, 5, 6\}, b))$
$= \varepsilon C(\{2, 3, 5\}) = \{2, 3, 5\} = R$

$\delta'(R, a) = \varepsilon C(\delta(\{2, 3, 5\}, a))$
$= \varepsilon C(\{2, 5, 6\}) = \{2, 5, 6\} = Q$

$\delta'(R, b) = \varepsilon C(\delta(\{2, 3, 5\}, b))$
$= \varepsilon C(\{2, 3, 4, 5\}) = \{2, 3, 4, 5, 8\} = T$

$\delta'(S, a) = \varepsilon C(\delta(\{2, 5, 6, 7, 8\}, a))$
$= \varepsilon C(\{2, 5, 6, 7, 8\}) = \{2, 5, 6, 7, 8\} = S$

$\delta'(S, b) = \varepsilon C(\delta(\{2, 5, 6, 7, 8\}, b))$
$= \varepsilon C(\{2, 3, 5\}) = \{2, 3, 5\} = R$

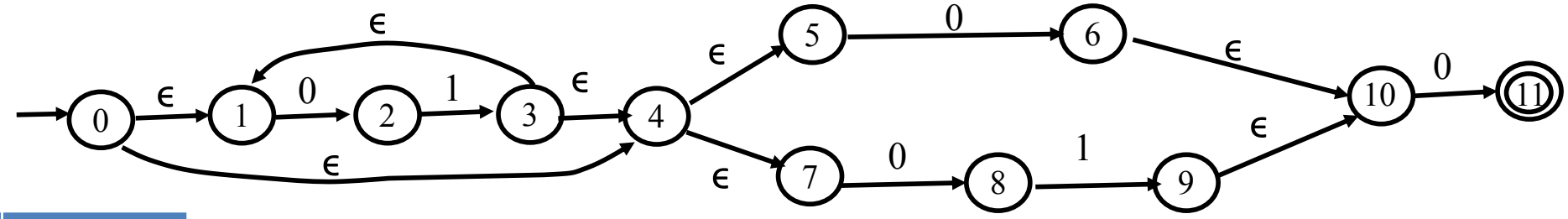
$\delta'(T, a) = \varepsilon C(\delta(\{2, 3, 4, 5, 8\}, a))$
$= \varepsilon C(\{2, 5, 6\}) = \{2, 5, 6\} = Q$

$\delta'(T\, b) = \varepsilon C(\delta(\{2, 3, 4, 5, 8\}, b))$
$= \varepsilon C(\{2, 3, 4, 5\}) = \{2, 3, 4, 5, 8\} = T$

$F_{N3} = \{8\}$ $\quad 8 \in S, T$

$F_{M3} = \{S, T\}$

Eg. NFA –> DFA Conversion



s-NFA	Ec(s)
0	{0,1,4,5,7}
1	{1}
2	{2}
3	{1,3,4,5,7}
4	{4,5,7}
5	{5}
6	{6,10}
7	{7}
8	{8}
9	{9,10}
10	{10}
11	{11}

$q_0 = \varepsilon C(\{0\}) = \{0, 1, 4, 5, 7\} = A$

$\delta'(A, 0) = \varepsilon C(\delta(\{0, 1, 4, 5, 7\}, 0))$
$= \varepsilon C(\{2, 6, 8\}) = \{2, 6, 8, 10\} = B$

$\delta'(A, 1) = \varepsilon C(\delta(\{0, 1, 4, 5, 7\}, 1)) = \varepsilon C(\emptyset) = \emptyset$

$\delta'(B, 0) = \varepsilon C(\delta(\{2, 6, 8, 10\}, 0))$
$= \varepsilon C(\{11\}) = \{11\} = C$

$\delta'(B, 1) = \varepsilon C(\delta(\{2, 6, 8, 10\}, 1))$
$= \varepsilon C(\{3, 9\}) = \{1,3,4,5,7, 9\ 10\} = D$

$\delta'(C, 0) = \varepsilon C(\delta(\{11\}, 0)) = \varepsilon C(\emptyset) = \emptyset$

$\delta'(C, 1) = \varepsilon C(\delta(\{11\}, 0)) = \varepsilon C(\emptyset) = \emptyset$

$\delta'(D, 0) = \varepsilon C(\delta(\{1, 3, 4, 5, 7, 9, 10\}, 0))$
$= \varepsilon C(\{2, 6, 8, 11\})$
$= \{2, 6, 8, 10, 11\} = E$

$\delta'(D, 1) = \varepsilon C(\delta(\{1, 3, 4, 5, 7, 9, 10\}, 1))$
$= \varepsilon C(\emptyset) = \emptyset$

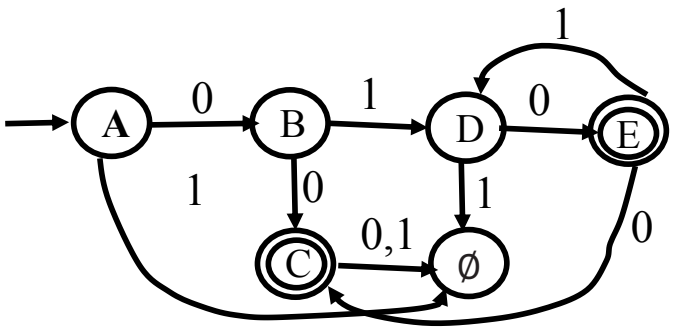
$\delta'(E, 0) = \varepsilon C(\delta(\{2, 6, 8, 10, 11\}, 0))$
$= \varepsilon C(\{11\}) = \{11\} = C$

$\delta'(E, 1) = \varepsilon C(\delta(\{2, 6, 8, 10, 11\}, 1))$
$= \varepsilon C(\{3, 9\}) = \{1,3,4,5,7, 9, 10\}$
$= D$

δ-DFA	0	1
A	B	∅
B	C	D
C	∅	∅
D	E	∅
E	C	D

$F_{N3} = \{11\}$ $\quad 8 \in C, D$

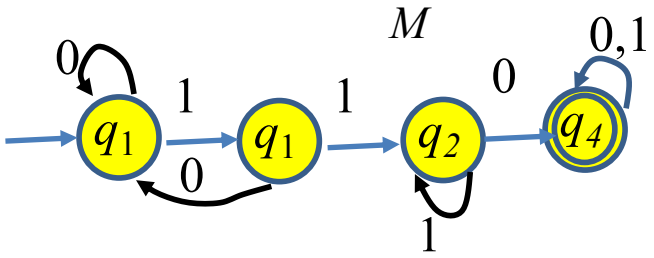
$F_{M3} = \{C, D\}$



Show that w^R is regular

- if $w = w_1 w_2 \cdots w_n$ is any string, then reverse of w denoted as w^R is the string $w_n w_{n-1} \cdots w_1$
- For any regular language L, $L^R = \{w^R \mid w \in L\}$ is regular.
- Eg. $L = \{\epsilon, 01, 110, 0101\}$; $L^R = \{\epsilon, 10, 011, 1010\}$

- $L = \{w \mid w \text{ contains } 110\}$ $\quad L(M_1) = L$

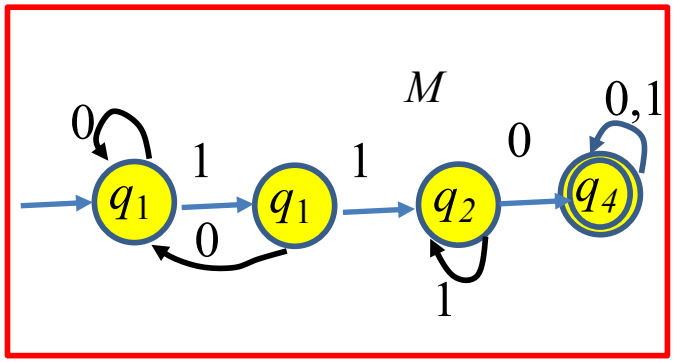


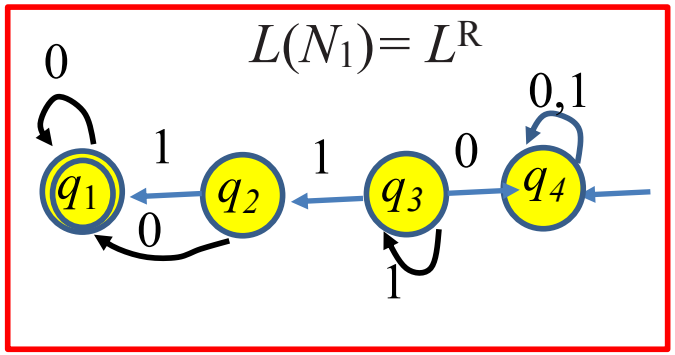
- To obtain the DFA M_2 for L^R
  - Design M_2 from scratch. i,.e $L^R = \{w \mid w \text{ contains } 011\}$
  - Construct M_2 from M_1

Show that w^R is regular

- $L = \{w \mid w \text{ contains } 110\}$
- To obtain the DFA M_2 for L^R
- Construct M_2 from M_1
  1. Reverse the direction of the transitions of M_1
  2. Make the accepting state the Start State
  3. Make the start state of M the accepting state

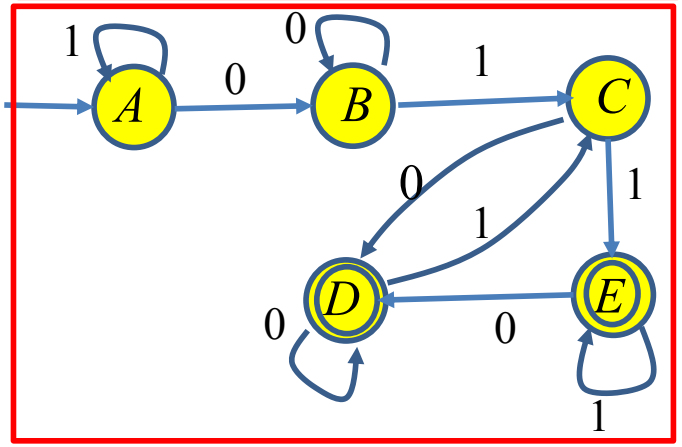
  Note that the resulting automaton from [1-3] is an NFA N_1 for L^R .i.e $L(N_1) = L^R$
  4. So convert NFA N_1 to DFA M_2





	0	1
$\{q_4\}$ **A**	$\{q_3,q_4\}$ B	$\{q_4\}$ A
$\{q_3,q_4\}$ B	$\{q_3,q_4\}$ B	$\{q_2,q_3,q_4\}$ C
$\{q_2,q_3,q_4\}$ C	$\{q_1,q_3,q_4\}$ D	$\{q_1,q_2,q_3,q_4\}$ E
$\{q_1,q_3,q_4\}$ **D**	$\{q_1,q_3,q_4\}$ D	$\{q_2,q_3,q_4\}$ C
$\{q_1,q_2,q_3,q_4\}$ **E**	$\{q_1,q_3,q_4\}$D	$\{q_1,q_2,q_3,q_4\}$E

$q_0 = A$
$F = \{D, E\}$



Thank You!

Tutorial - IV

Draw NFA's

Give NFAs with the specified number of states recognizing each of the following languages. In all cases, the alphabet is $\Sigma = \{0, 1\}$.

a. The language $\{w \in \Sigma^* \mid w \text{ ends with } 00\}$ with three states.

b. The language $\{w \in \Sigma^* \mid w \text{ contains the substring } 0101, \text{ i.e., } w = x0101y \text{ for some } x, y \in \Sigma^*\}$ with five states.

c. The language $\{w \in \Sigma^* \mid w \text{ contains at least two 0s, or exactly two 1s}\}$ with six states.

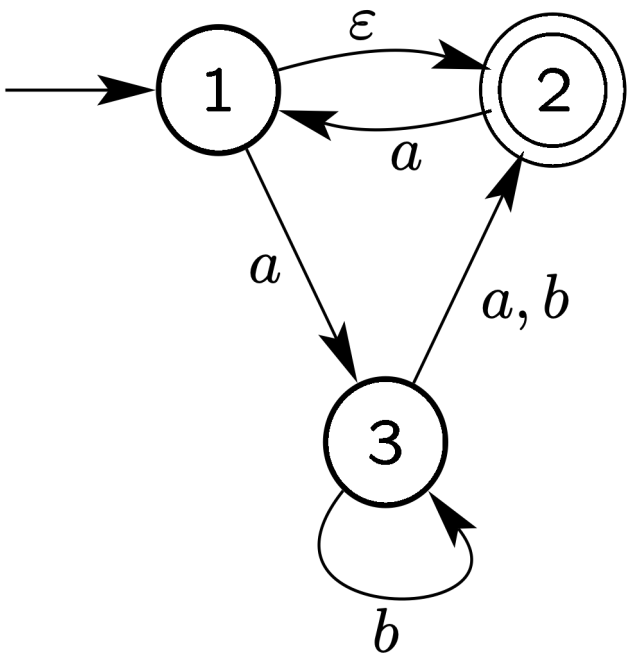
d. The language $\{\varepsilon\}$ with one state.

e. The language $0^*1^*0^*0$ with three states.

Construct DFA's from NFA

Use the construction given in Theorem to convert the following NFA N into an equivalent DFA.

a)



b)