COSC3106: Theory of Computing

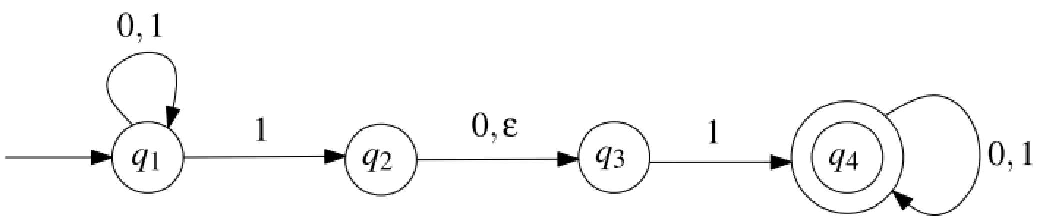
CH#2: Nondeterminism:

Lec#4: Non-deterministic Finite Automaton

School of Computer Science and Technology

- **By the end of this lecture, you will …**
  - **Understand the concept of the non-deterministic finite automaton**

- Let us start by giving examples of nondeterministic finite automata…
  - Ex#1: Consider the following state diagram:



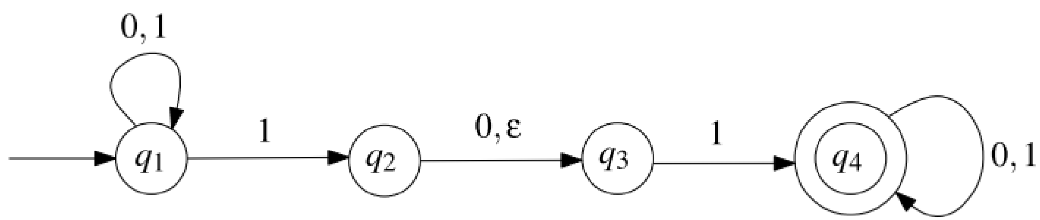
  - You will notice three differences with the FA that we have seen until now:

    1. If the automaton is in state q_1 and reads the symbol 1, then it has two options: Either it stays in state q_1, or it switches to state q_2 .

    2. If the automaton is in state q_2, then it can switch to state q_3 without reading an input symbol; this is indicated by the edge having the empty string ε as label.

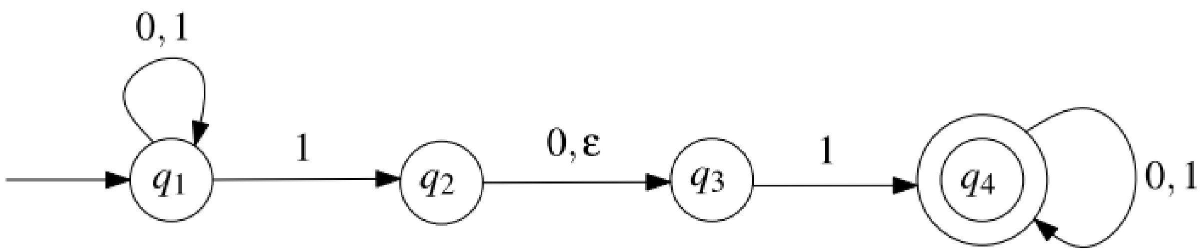
    3. If the automaton is in state q_3 and reads the symbol 0, then it cannot continue.

- **Ex#1 (cont.): What this automaton can do when it gets the string 010110 as input.**



- **An NFA accepts a string, if there exists at least one path in the state diagram that:**
  - **(i) starts in the start state,**
  - **(ii) does not stuck before the entire string has been read, and**
  - **(iii) ends in an accept state.**

- **Ex#2: If the input string is 010**



  - **In this case, there are three possible computations:**

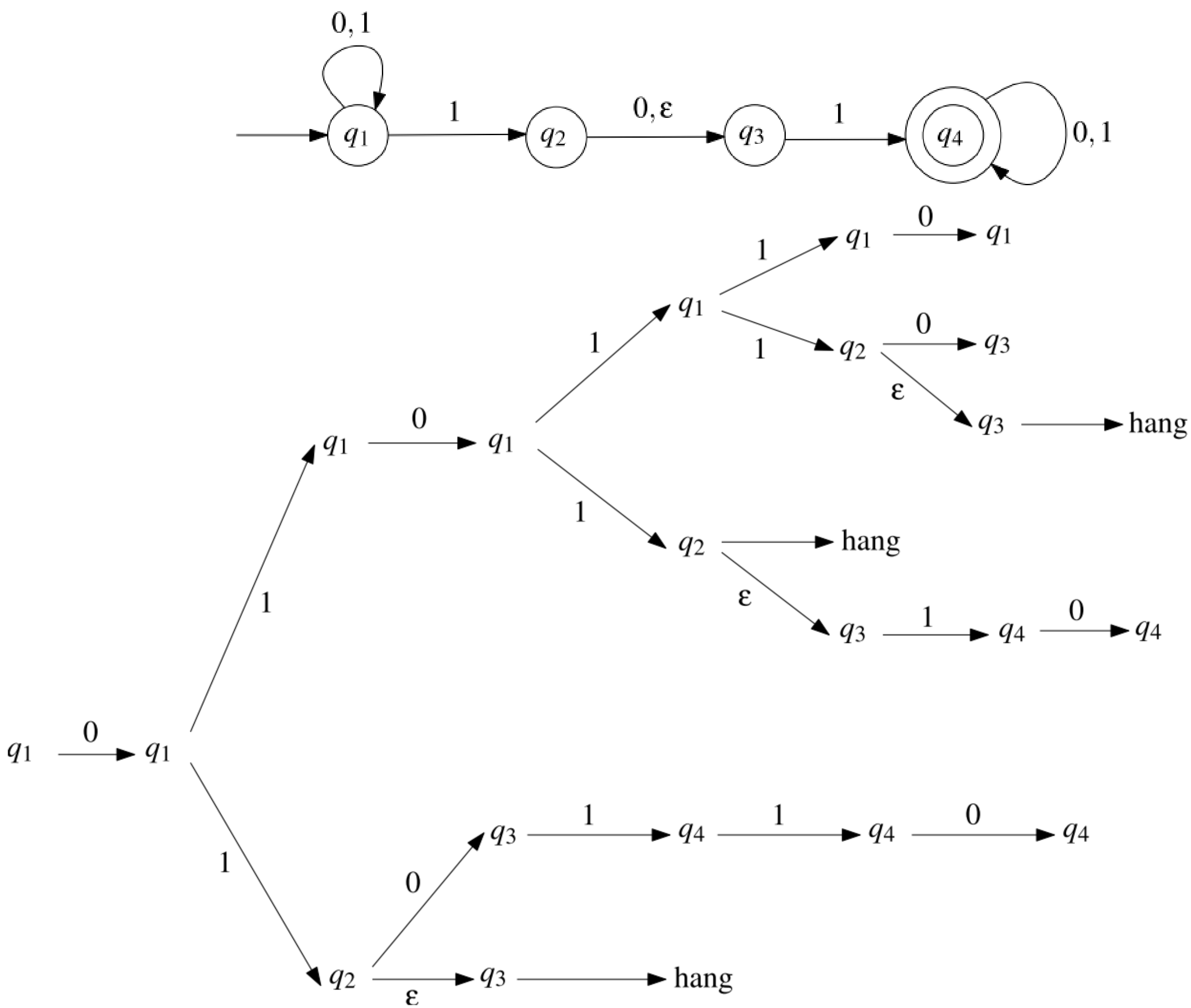
$$q_1 \xrightarrow{0} q_1 \xrightarrow{1} q_1 \xrightarrow{0} q_1$$

$$q_1 \xrightarrow{0} q_1 \xrightarrow{1} q_2 \xrightarrow{0} q_3$$

$$q_1 \xrightarrow{0} q_1 \xrightarrow{1} q_2 \xrightarrow{\epsilon} q_3 \rightarrow \text{hang}$$

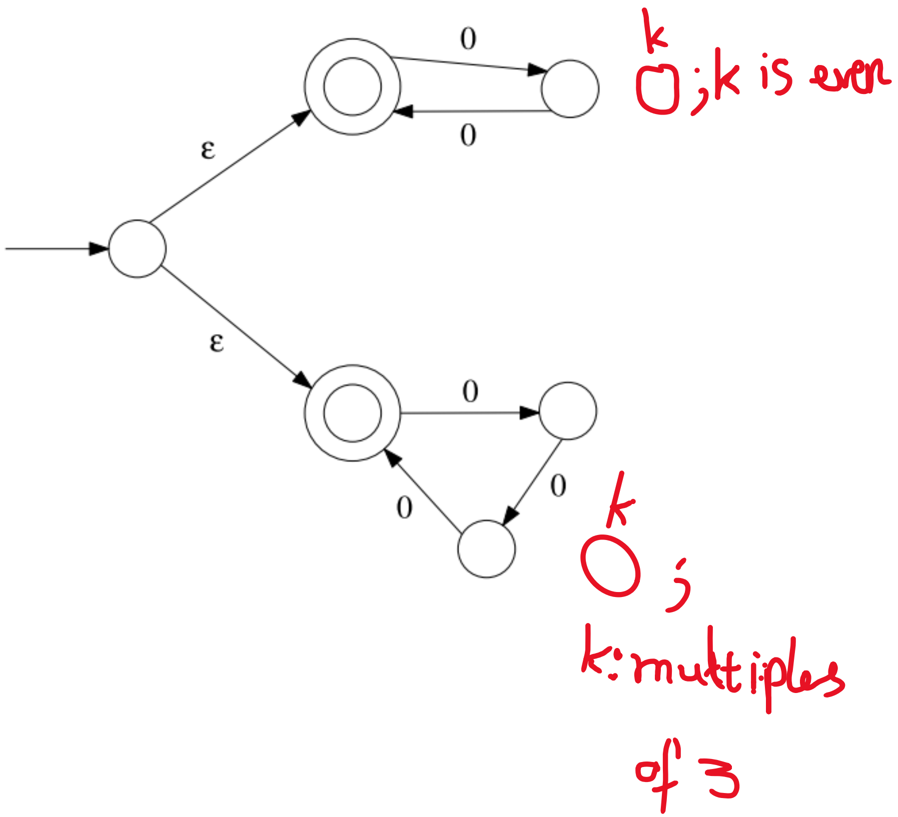
  - **A string w is accepted by this NFA if and only if the input string contains 101, or 11**

- **Ex#2: If the input string is 010110**



- **Ex#3: Let A be the language $A = \{ w \in \{0, 1\}: w$ has a 1 in the third position from the right $\}$.**
  - **We need to create an NFA that accepts A.**

- **Ex#4: Consider the following state diagram, which defines an NFA whose alphabet is $\{0\}$.**



  - **Define its accepted language?**

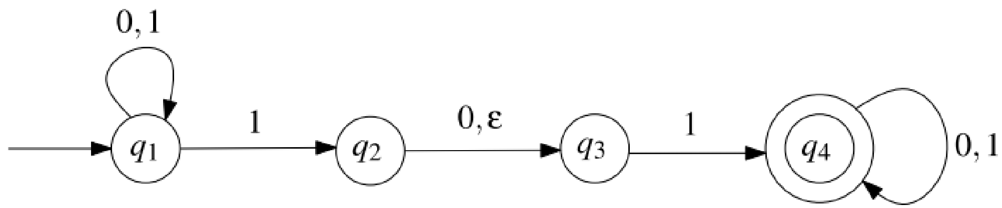
$\{w\!:\ 0^k;\ k$ **is even or multiples of** $3\}$

- **Definition of nondeterministic finite automaton:**
  - **The previous examples give you an idea what nondeterministic finite automata are and how they work.**
  - **In this section, we give a formal definition of these automata.**
    - **For any alphabet Σ, we define Σ_ε to be the set: $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$**

NFA: $N = \{Q, \Sigma, \delta, q, F\}$

- Q **is a finite set, whose elements are called states,**
- Σ **is a finite set, called the alphabet; the elements of Σ are called symbols,**
- $\delta : Q \times \Sigma_\varepsilon \rightarrow P(Q)$ **is the transition function,**
- q **is the start state,**
- F **is a subset of Q; the elements of F are called accept states.**

- **Ex: Consider the NFA in the figure, specify it as a formal NFA?**



- $Q = \{q_1, q_2, q_3, q_4\}$,
- $\Sigma = \{0, 1\}$,
- The start state is q_1,
- The set of accept states is $F = \{q_4\}$.
- The transition function δ is given by the following table:

	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	$\{q_4\}$	$\{q_4\}$	$\emptyset$

- N accepts string $w \in \Sigma^*$
  - if $w = y_1 y_2 \dots y_m$; $y_i \in \Sigma_\varepsilon$
  - $\exists$ a sequence of states:

$$r_0, r_1, \dots, r_m \in Q$$

  - $r_0 = q$
  - $r_{i+1} \in \delta(r_i, y_{i+1}), i = 0, 1, 2, \dots, m-1$
  - $r_m = F$

- In Words, an NFA accepts a string, if there exists at least one path in the state diagram that:
  - (i) starts in the start state,
  - (ii) does not stuck before the entire string has been read, and
  - (iii) ends in an accept state.

  - A string for which (i), (ii), and (iii) does not hold is rejected by the NFA.

COSC3106: Theory of Computing

Lec#5: Equivalence of DFAs and NFAs

School of Computer Science and Technology

- **By the end of this lecture, you will learn …**
  - **How to convert NFA to DFA**

- **Equivalence of DFAs and NFAs**
  - **NFA:** $N = \{Q, \Sigma, \delta, q, F\}$

$$\delta : Q \times \Sigma_\varepsilon \rightarrow P(Q)\ ;\ \Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$$

  - **DFA:** $M = \{Q', \Sigma, \delta', q', F'\}$

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

- **Converting NFA to Equivalent DFA**
  - **General Approach:**
    - **When an NFA reads part of an input string and has not yet processed the rest, we ask the following question:**
      - **What are the possible states that the NFA could be in at this point?**
      - **The key idea in converting an NFA to a DFA is that the DFA explicitly keeps track of all possible states the NFA could be in simultaneously.**

- **Easy Case: No ε-transitions in NFA "N"**
  - DFA construction: $M = \{Q', \Sigma, \delta', q', F'\}$
  - NFA Construction: $N = \{Q, \Sigma, \delta, q, F\}$
  - States: The set of states of DFA will be the power set of states of NFA

$$Q' = P(Q) = \{R: R \subseteq Q\}$$

    - For every subset R of the set of states Q, there is one state in DFA, this means if NFA has 5 states the power set has size of $2^5 = 32$ states
  - Start State: $q' = q$; (Same as the NFA start state.)
  - Accept States: $F' = \{R: R \subseteq Q, R \cap F \neq \phi\}$
    - Any DFA state that contains at least one NFA accept state is an accept state.
  - Transition Function:

$$\delta'(R, a) = \bigcup_{r \in R} \delta(r, a)$$

- **Easy Case: No ε-transitions in NFA "N"**
  - **Ex:**

- **General Case: ε-transitions in NFA "N"**
  - **NFA process a string:**
    - **Start in q, zero or more ε-transitions**
    - **Read one symbol and go to the next state**
    - **Zero or more ε-transitions**
    - **Read one symbol and go to the next state**
    - **Zero or more ε-transitions**
  - **ε-closure:**
    - **For $r \in Q$ of NFA, $C_\varepsilon(r)$ is a set of all states reachable from state "r" by making zero or more ε-transitions.**

- **General Case:**
  - **Ex:**

- General Case: Now we can define the equivalent DFA "M"
  - DFA construction: $M = \{Q', \Sigma, \delta', q', F'\}$
  - States:

$$Q' = P(Q)$$

    - Each DFA state corresponds to a subset of NFA states.
    - The DFA keeps track of "all possible states" the NFA could be in after reading some input.
  - Start State:

$$q' = q$$

  - Accept States:

$$F' = \{R \subseteq Q, R \cap F \neq \phi\}$$

    - Any DFA state that contains at least one NFA accept state is an accept state.
  - Transition Function:

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

$$\delta'(R, a) = \bigcup_{r \in R} \bigcup_{s \in \delta(r,a)} C_{\varepsilon}(r)$$

    - In words: from each NFA state in R, follow a-transitions; collect all reachable states including ε-transitions.

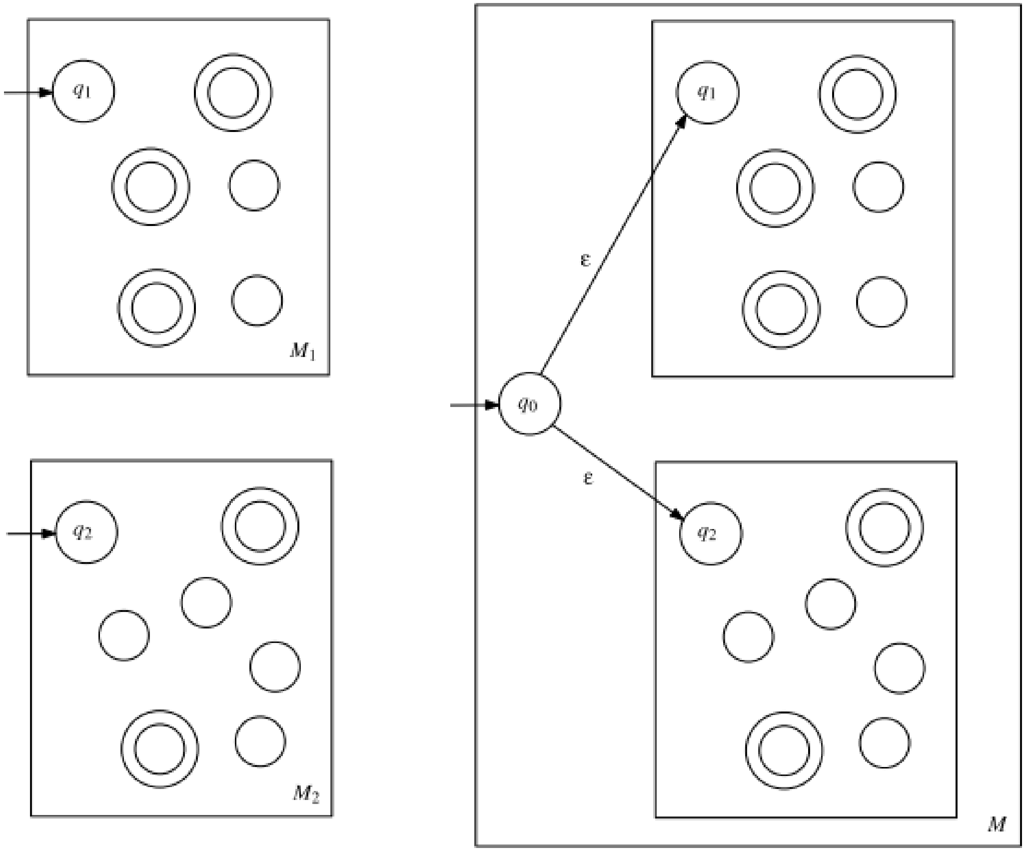
- **Theorem 2.1:**
  - **A language A is regular if and only if there exists a nondeterministic finite automaton NFA that accepts A.**

- **We have explained why DFA makes it not clear that the concatenation of two regular languages is regular, and that the star of a regular language is regular.**

- Now, we will see that the concept of NFA can be used to give a simple proof of the fact that the regular languages are indeed closed under the regular operations.

- Theorem 2.2:
  - The set of regular languages is closed under the union operation
  - i.e., if A_1 and A_2 are regular languages over the same alphabet Σ, then $A_1 \cup A_2$ is also a regular language.

- Proof:



- **Theorem 2.1:**
  - **A language A is regular if and only if there exists a nondeterministic finite automaton NFA that accepts A.**

- **We have explained why DFA makes it not clear that the concatenation of two regular languages is regular, and that the star of a regular language is regular.**

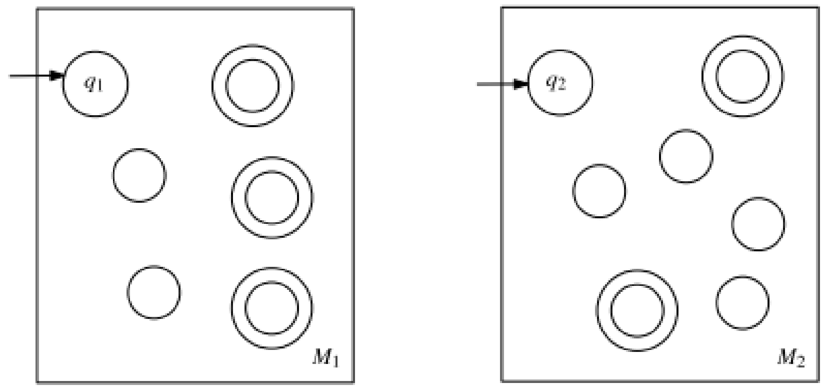
- **Now, we will see that the concept of NFA can be used to give a simple proof of the fact that the regular languages are indeed closed under the regular operations.**

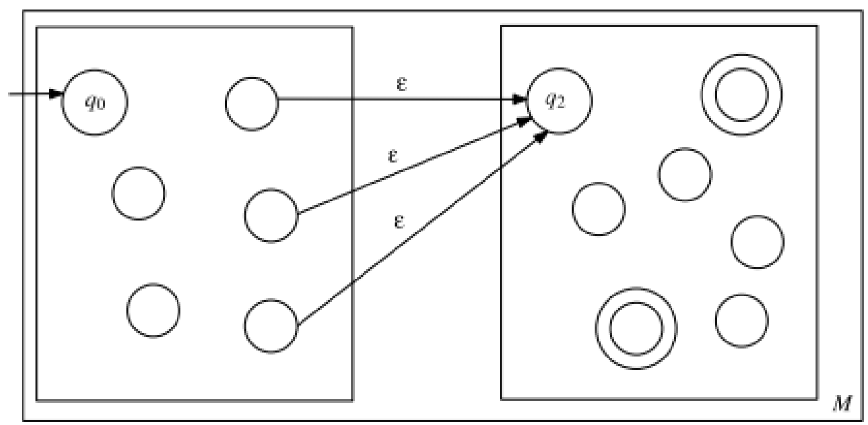
- **Theorem 2.2:**
  - **The set of regular languages is closed under the union operation**
  - **i.e., if A_1 and A_2 are regular languages over the same alphabet Σ, then $A_1 \cup A_2$ is also a regular language.**

- **Proof:**

- **Theorem 2.3:**
  - **The set of regular languages is closed under the concatenation operation,**
  - **i.e., if A_1 and A_2 are regular languages over the same alphabet Σ, then A_1A_2 is also a regular language.**

- **Proof:**





- **Theorem 2.4:**
  - **The set of regular languages is closed under the star operation,**
  - **i.e., if A is a regular language, then A^* is also a regular language.**

- **Proof:**