Theory of Computation
Chapter 1

Finite Automata

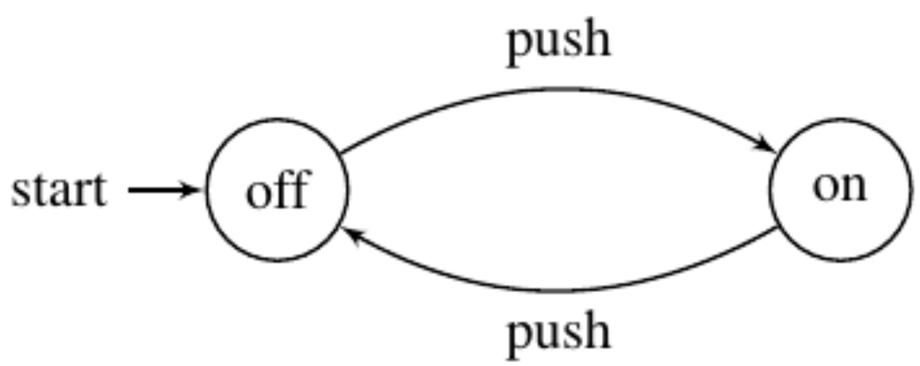
School of Engineering | Computer Science

Grace Murray Hopper 1906-1992

- PhD in mathematics, Yale, 1934
- Volunteered for Navy during WWII, became a programmer on Harvard Mark I
- Helped develop UNIVAC I
- Developed first compiler
- Led development of COBOL programming language
- Popularized the term “debugging”
- First female Admiral in US Navy
- USS Hopper, launched in 1996, named for her

Finite Automata

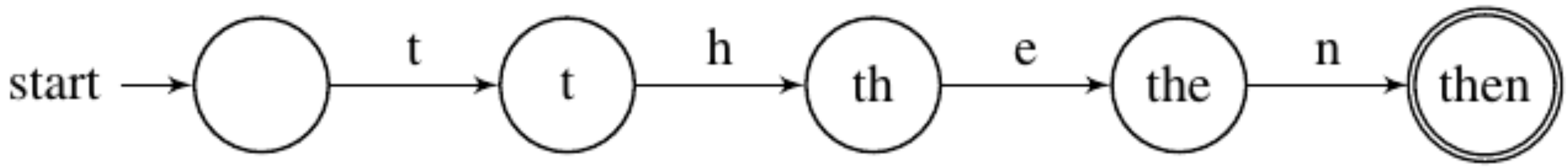
- Good models for computers with extremely limited amount of memory
- Ex: An on/off switch



VCU
School of Engineering | Computer Science

Finite Automata

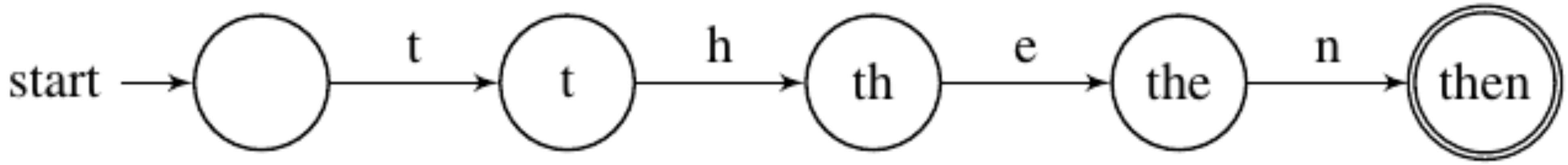
- Ex: Part of a lexical analyzer for recognizing the word then



- Move from state to state on input given, starting with start state
- States with double circles are accept states. All others are reject states.
- Final state is where end up after processing the given input.

Finite Automata

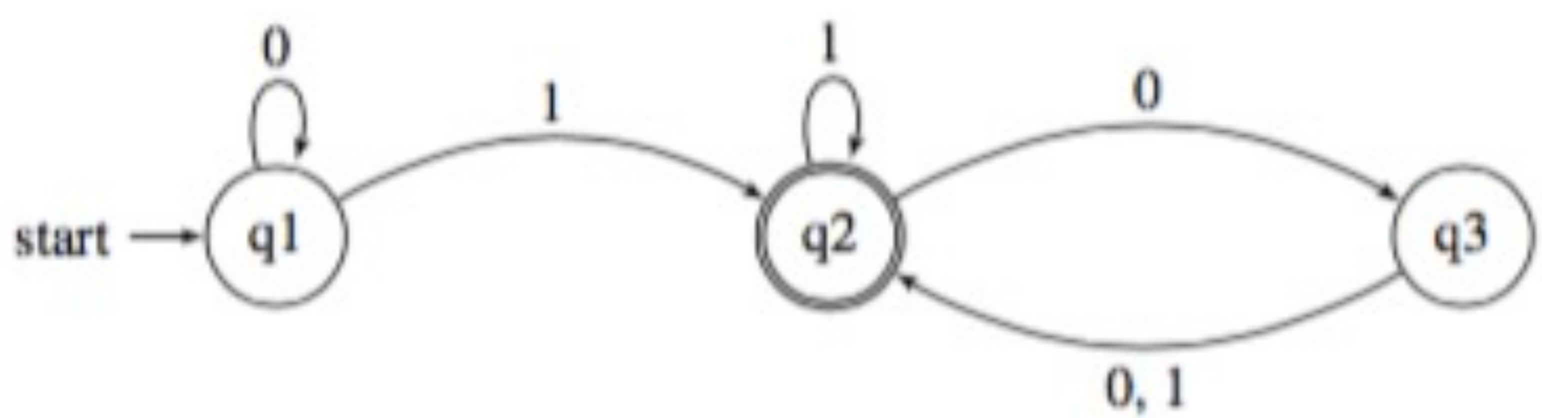
- Ex: Part of a lexical analyzer for recognizing the word then



- Input 1:
  - String "the"
  - States: t, th, the – end on a non-accepting state
- Input 2:
  - String "then"
  - States: t, th, the, then – end on an accepting state

Finite Automata

- Formal Example
  - Alphabet: $\Sigma = \{0, 1\}$
  - State Diagram:

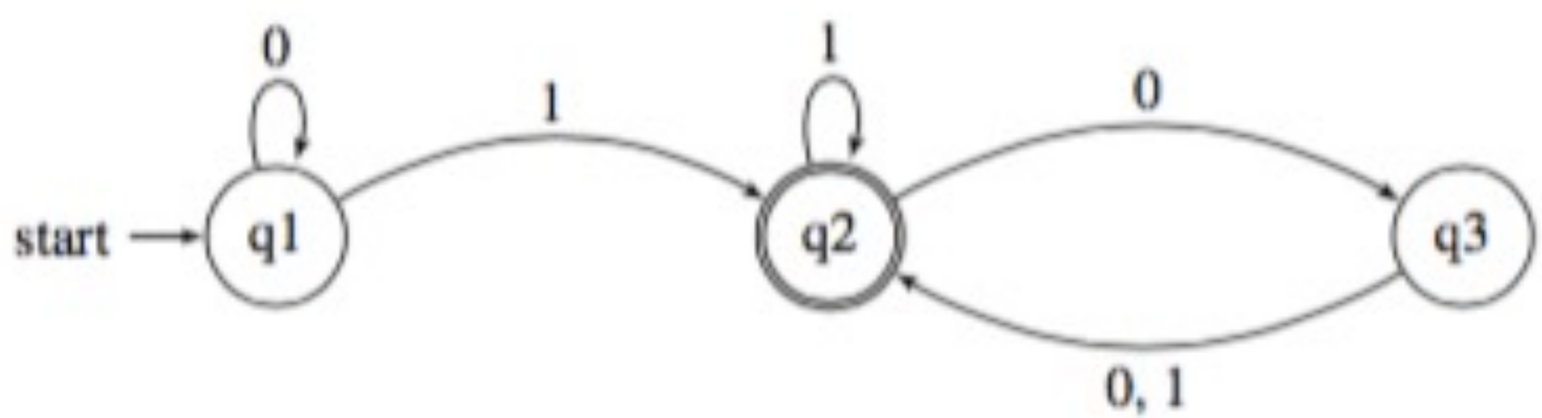


- Start State is q1, Accept state is q2
- Arrow are the transitions between states

School of Engineering | Computer Science

Finite Automata

- Formal Example
  - Alphabet: $\Sigma = \{0, 1\}$
  - State Diagram:

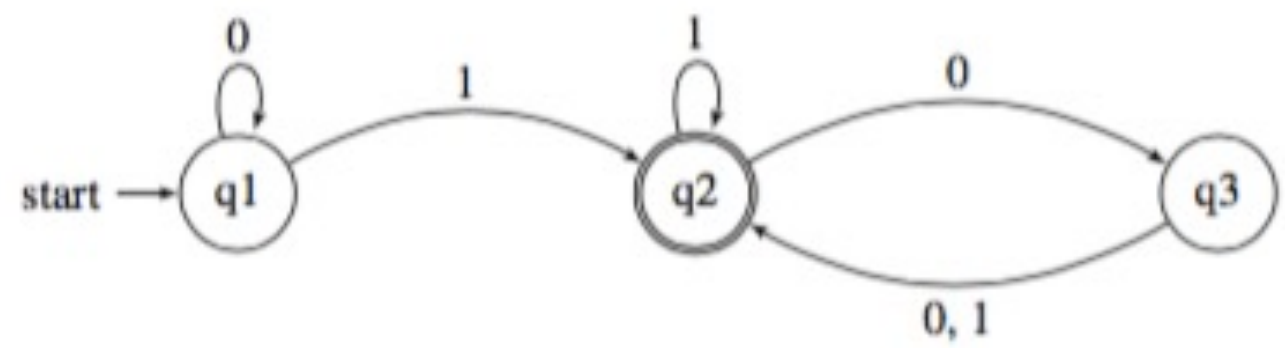


- Will this automaton accept the string 1101?
  - $q1 \rightarrow q2 \rightarrow q2 \rightarrow q3 \rightarrow q2$ (accept state – Yes!)

    1 1 0 1
  - Move from state to state on string characters. When characters end, determine if in an accept state or not

Finite Automata

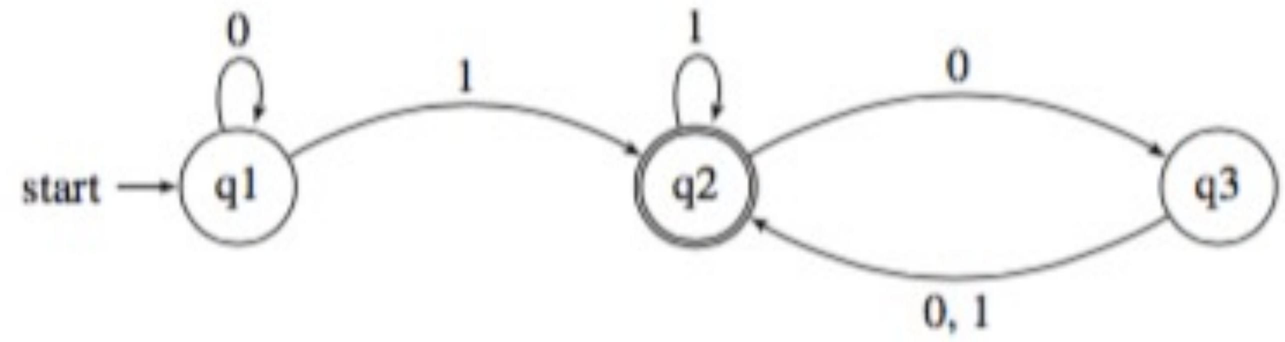
- Formal Example
  - Alphabet: $\Sigma = \{0, 1\}$
  - State Diagram:



- Will this automaton accept the string 0001?
  - q1 → q1 → q1 → q1 → q2 (accept state – Yes!)
    0 0 0 1
- Will this automaton accept the empty string ε?
  - q1 (not an accept state – No)

Finite Automata

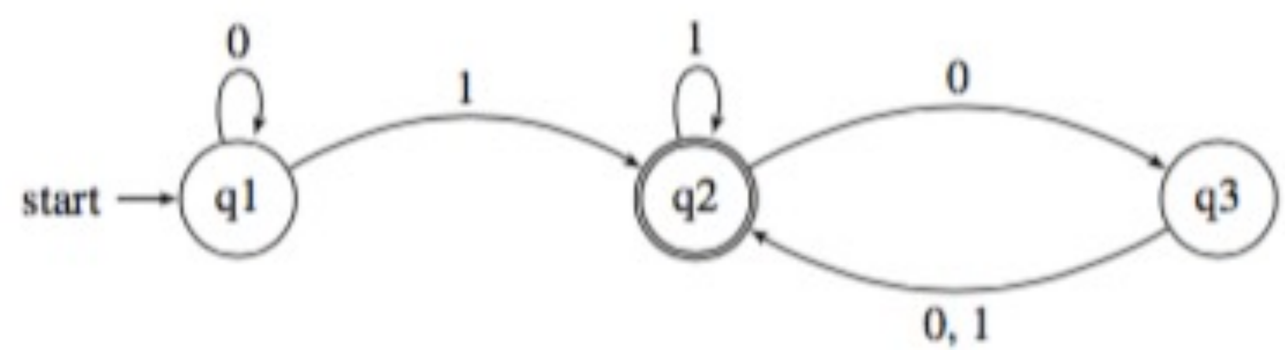
- Formal Example
  - Alphabet: $\Sigma = \{0, 1\}$
  - State Diagram:



- Will this automaton accept:
  - 0?
  - 0110101?
  - 011000?

Finite Automata

- Formal Example
  - Alphabet: $\Sigma = \{0, 1\}$
  - State Diagram:



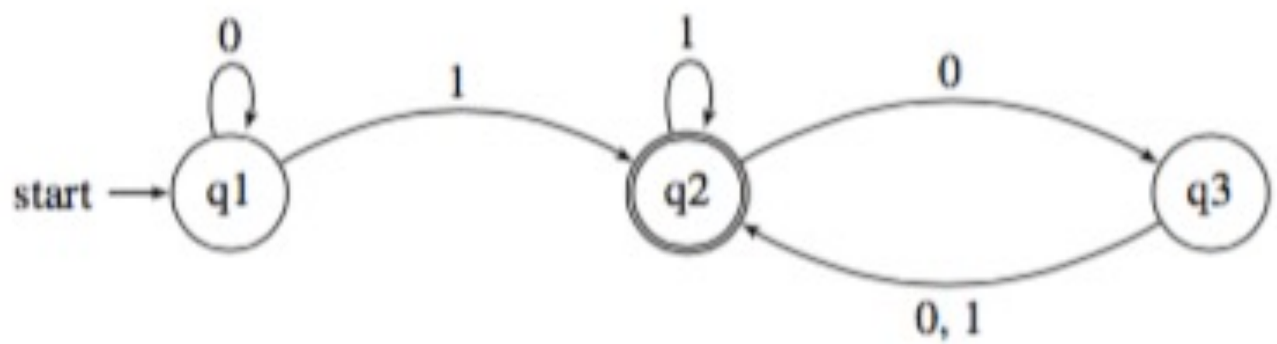
- Will this automaton accept:
  - 0? Ans: $q1 \xrightarrow{1} q1$ (no)
  - 0110101?
    - Ans: $q1 \xrightarrow{0} q1 \xrightarrow{1} q2 \xrightarrow{1} q2 \xrightarrow{0} q3 \xrightarrow{1} q2 \xrightarrow{0} q3 \xrightarrow{1} q2$ (yes)
  - 011000?
    - Ans: $q1 \xrightarrow{0} q1 \xrightarrow{1} q2 \xrightarrow{1} q2 \xrightarrow{0} q3 \xrightarrow{0} q2 \xrightarrow{0} q3$ (no)

Deterministic Finite Automata

- <u>Formal Definition (Deterministic Finite Automata (DFA))</u>
  - A finite automata is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where:
    1. Q is a finite set of states
    2. Σ is a finite set called the alphabet
    3. $\delta: Q \times \Sigma \rightarrow Q$ is the transition function
    4. $q_0 \in Q$ is the start state
    5. $F \subseteq Q$ is a set of accept states

Deterministic Finite Automata

- Ex: M_1



1. $Q =$
2. $\Sigma =$
3. $q_0 =$
4. $F =$

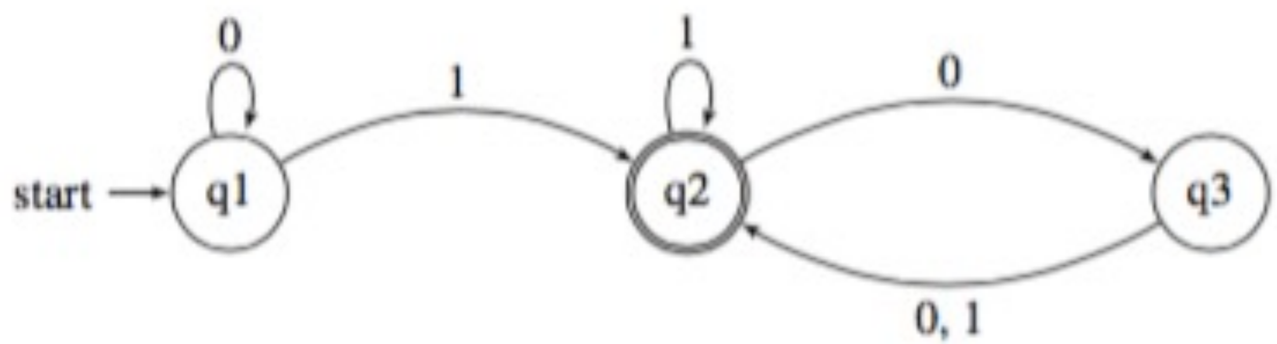
δ	**0**	**1**
$\mathbf{q_1}$		
$\mathbf{q_2}$		
$\mathbf{q_3}$		

When at state ? With an input of ? Move to state ?

- Deterministic means that each state has exactly one transition for each symbol in the alphabet

Deterministic Finite Automata

- Ex: M_1



1. $Q = \{q_1, q_2, q_3\}$
2. $\Sigma = \{0, 1\}$
3. $q_0 = q_1$
4. $F = \{q_2\}$

δ	**0**	**1**
$\mathbf{q_1}$	q_1	q_2
$\mathbf{q_2}$	q_3	q_2
$\mathbf{q_3}$	q_2	q_2

When at state ? With an input of ? Move to state ?

Ex: State q_1 (row), input of 0 (column), move to q_1 (output in grid)

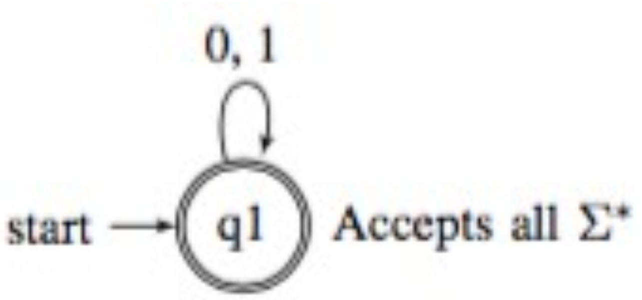
- Deterministic means that each state has <u>exactly one</u> transition <u>for each</u> symbol in the alphabet

Regular Languages

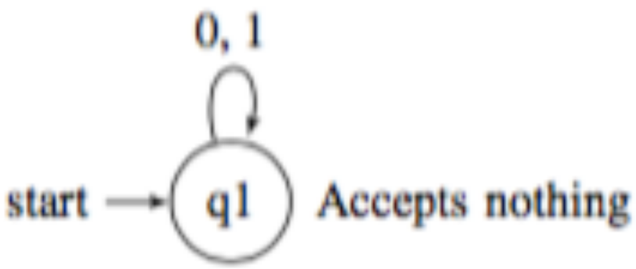
- Languages are just a set of strings
- <u>Language of Machine M:</u>
  - If A is a set of all strings that machine M accepts, say A is the language of M (L(M) = A)
  - M recognizes A
    - Ex: A = {w | w contains at least one 1 and an even number of 0's following the last 1}
    - L(M) = A
  - Generic: L(M) = {x | x is accepted by M}
    - Build a machine M to accept L(M) or A

Regular Languages, cont.

- Language of Machine M:
  - If A is a set of all strings that machine M accepts, say A is the language of M (L(M) = A)
  - $A \subseteq \Sigma^*$ (Σ^* means all possible strings formed from symbols in the alphabet Σ including the empty string)
  - Say Σ = {0, 1}
    - Can build a machine to accept all Σ^*



  - F = ∅, then the language of the machine L(M) = ∅
    - No accept state



Design an Automata

- Ex: Design M where the language is:
  - $L(M) = \{x \mid x \in \{0, 1, 2\}^* \text{ such that the sum of digits of } x \text{ is divisible by } 3\}$
    - How do we build this?

Design an Automata

- Ex: Design M where the language is:
  - L(M) = {x | x ∈ {0, 1, 2}* such that the sum of digits of x is divisible by 3}
    - How do we build this?
      - Σ = {0, 1, 2}
      - First x can be the empty set, ε.
        - The start state should be an accept state since can add the empty set to itself and it should be included.
      - Try numbers: $\varepsilon + 0 = 0$ (divisible by 3, so loop back to start state on 0)

Design an Automata, cont.

- Ex: Design M where the language is:
  - $L(M) = \{x \mid x \in \{0, 1, 2\}^*$ such that the sum of digits of x is divisible by 3$\}$
    - How do we build this?
      - $\Sigma = \{0, 1, 2\}$
      - Try numbers: $\varepsilon + 1 = 1$ (not divisible by 3, so move to a new state: q_1)
      - Try numbers: $\varepsilon + 2 = 2$ (not divisible by 3, so move to a new state: q_2)
      - Try numbers: $1 + 1 = 2$ (not divisible by 3, but same as q_2, so move to state: q_2)
      - Try numbers: $2 + 1 = 3$ (divisible by 3, so move back to start state: q_0)

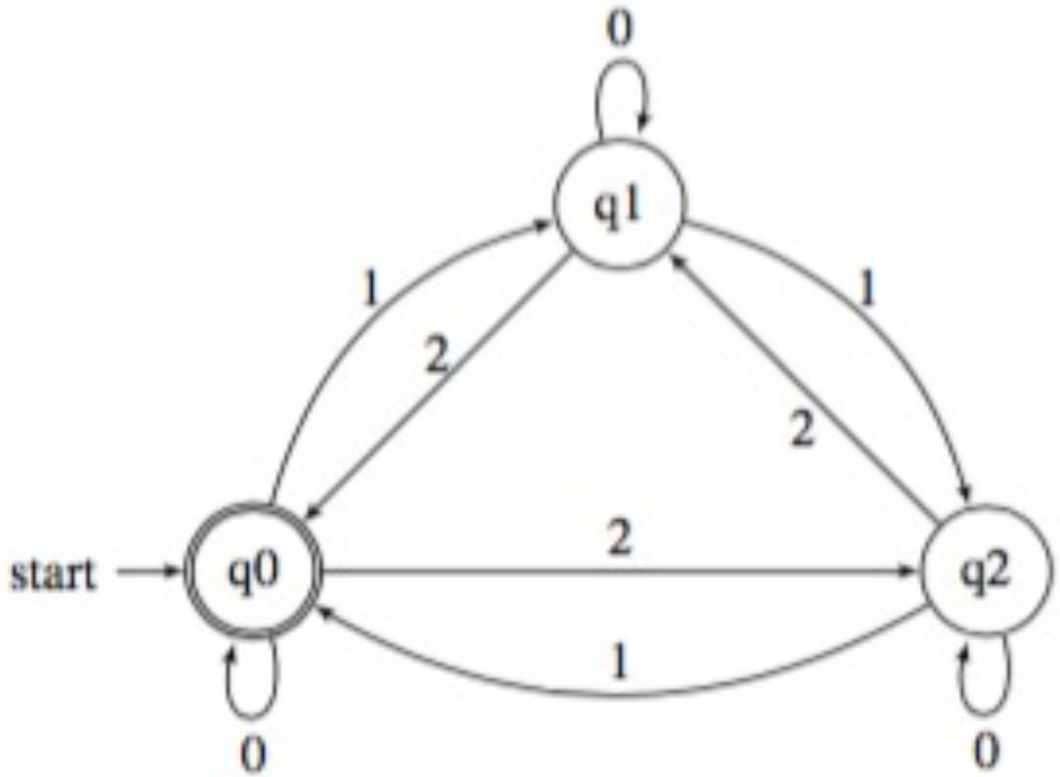
Design an Automata, cont.

- Ex: Design M where the language is:
  - L(M) = {x | x ∈ {0, 1, 2}* such that the sum of digits of x is divisible by 3}
    - How do we build this?
      - Σ = {0, 1, 2}
      - Try numbers: 1 + 1 + 1 = 3 (divisible by 3, so move back to start state: q_0)
      - Try numbers: 2 + 2 = 4 (not divisible by 3, 4 % 3 = 1, so move to state: q_1)
      - Try numbers: 1 + 2 = 3 (divisible by 3, so move to start state: q_0)
      - Adding 0 to any number does not change the value so loop back to state

Design an Automata, cont.

- Ex: Design M where the language is:
  - $L(M) = \{x \mid x \in \{0, 1, 2\}^*$ such that the sum of digits of x is divisible by 3$\}$
    - $Q = \{q_0, q_1, q_2\}$
    - $\Sigma = \{0, 1, 2\}$
    - $q_0 = q_0$
    - $F = \{q_0\}$

δ	**0**	**1**	**2**
$\mathbf{q_0}$	q_0	q_1	q_2
$\mathbf{q_1}$	q_1	q_2	q_0
$\mathbf{q_2}$	q_2	q_0	q_1



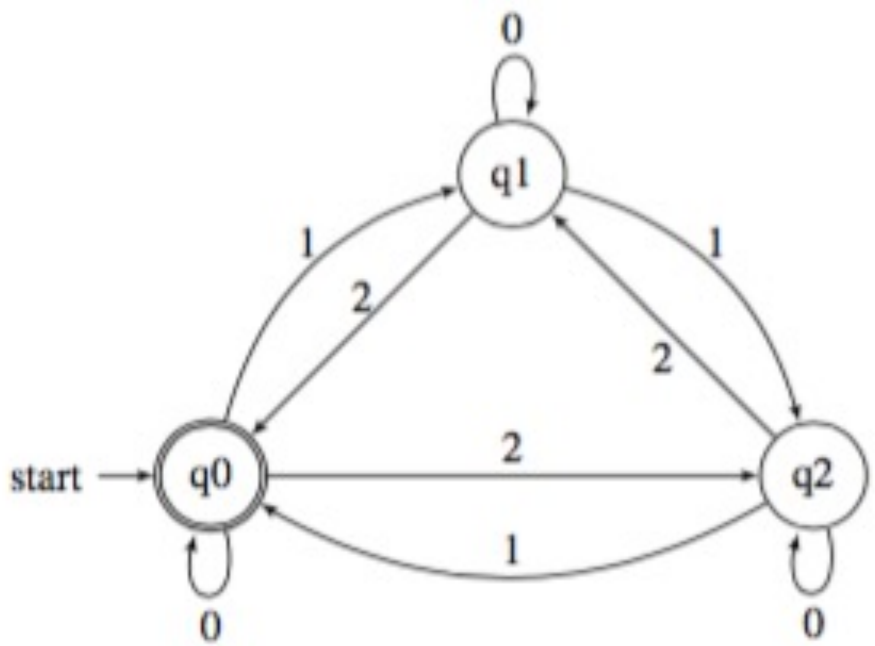
School of Engineering | Computer Science

Formal Definition of Computation

- Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automata (DFA)
  - Let $w = w_1w_2\ldots w_n \in \Sigma^*$ be an input string
  - Then M <u>accepts</u> w if a sequence of states $r_0, r_1, \ldots, r_n$ in Q exists with 3 conditions:
    1. r_0 is the start state q_0 ($r_0 = q_0$)
    2. For $i = 0, \ldots, n-1$, $\delta(r_i, w_{i+1}) = r_{i+1}$
    3. $r_n \in F$ (r_n is an accepting state)
  - Say M recognizes language A if $A = \{w \mid M \text{ accepts } w\}$
- A language is called a <u>regular language</u> if some finite automaton recognizes it.

Formal Definition of Computation, cont.

- Ex: Formally show M where language L(M) = {x | x ∈ $\{0, 1, 2\}^*$ where sum is divisible by 3}

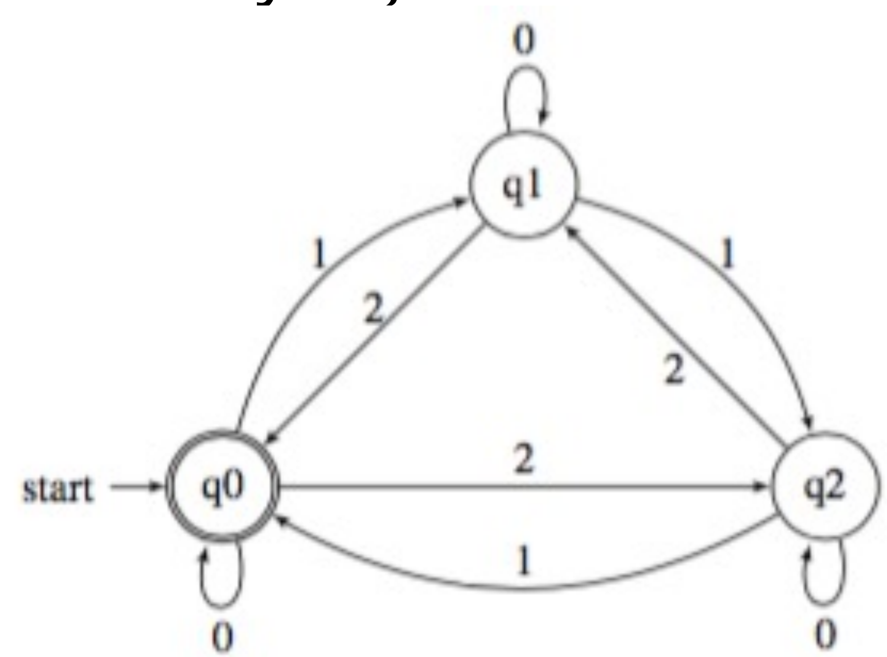


- If w = 1022010 will the machine accept it?
  - States r =

School of Engineering | Computer Science

Formal Definition of Computation, cont.

- Ex: Formally show M where language L(M) = $\{x \mid x \in \{0, 1, 2\}^*$ where sum is divisible by 3$\}$



- If w = 1022010 will the machine accept it?
  - States r = $q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_1 \xrightarrow{2} q_0 \xrightarrow{2} q_2 \xrightarrow{0} q_2 \xrightarrow{1} q_0 \xrightarrow{0} q_0$ (ends at an accept state, so Yes!)

Designing Finite Automata

- Ex: $L(M) = \{x \in \{0, 1\}^* \mid x \text{ has odd number of 1's}\}$
  - How do we design this?

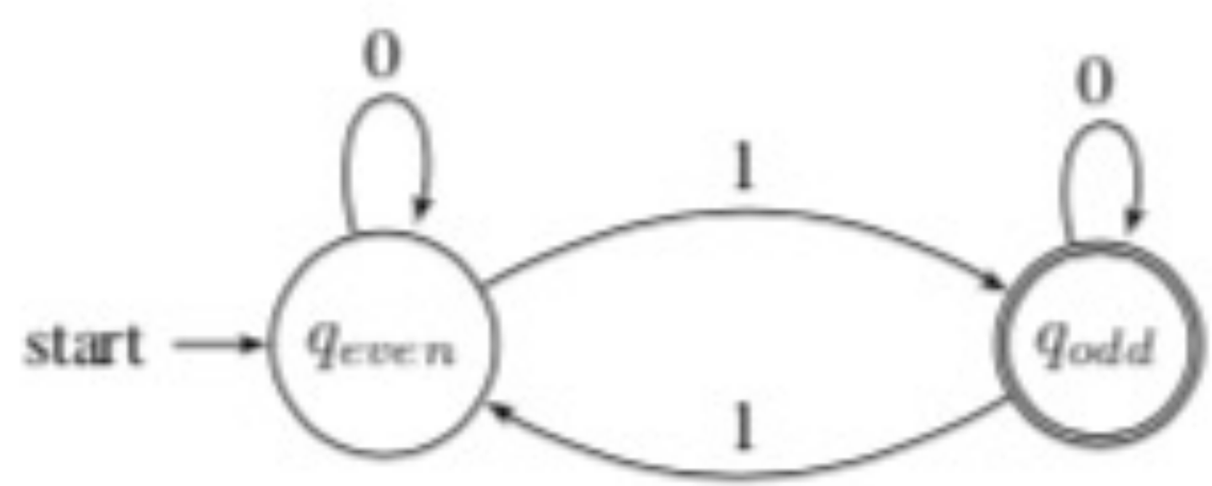
Designing Finite Automata

- Ex: $L(M) = \{x \in \{0, 1\}^* \mid x$ has odd number of 1's$\}$
  - How do we design this?
    - Have to have at least one 1, so start state cannot be accept state, but can make the second state the accept state.
    - Can go back to start state if second 1 is given
    - If another 1 is given can go to the second state, and can continue to go back and forth
    - What about 0's? You should stay at the same state when given a 0 since it does not change the number of 1's.

Designing Finite Automata, cont.

- Ex: $L(M) = \{x \in \{0, 1\}^* \mid x \text{ has odd number of 1's}\}$
  - How do we design this?
    - Two states: $Q = \{q_{even}, q_{odd}\}$
    - Alphabet: $\Sigma = \{0, 1\}$
    - Start state: $q_0 = q_{even}$
    - Final states: $F = \{q_{odd}\}$
    - Transition function:

δ	**0**	**1**
$\mathbf{q_{even}}$	q_{even}	q_{odd}
$\mathbf{q_{odd}}$	q_{odd}	q_{even}



VCU School of Engineering | Computer Science

Designing Finite Automata, cont.

- Ex: $L(M) = \{x \in \{0, 1\}^* \mid x \text{ contains the substring } 001\}$
  - How do we design this?

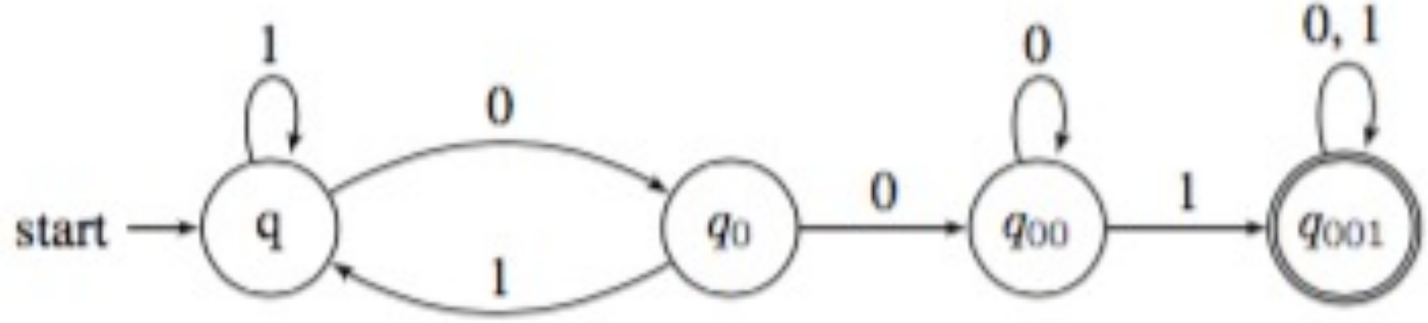
Designing Finite Automata, cont.

- Ex: $L(M) = \{x \in \{0, 1\}^* \mid x \text{ contains the substring } 001\}$
  - How do we design this?
    - Have to have 001 in a sequence.
    - Can start on any input, but only a 0 should move us towards the accept state
    - If enter a 1 after one 0, need to move back to the start, since that is not part of the sequence
    - A second 0 moves us to another state.
    - At this point a 1 takes us to the accept state, another 0 should keep us where we are since it does not change the sequence
    - Any more 0's or 1's do not matter and should keep us where we are

Designing Finite Automata, cont.

- Ex: $L(M) = \{x \in \{0, 1\}^* \mid x \text{ contains the substring } 001\}$
  - How do we design this?
    - Two states: $Q = \{q, q_0, q_{00}, q_{001}\}$
    - Alphabet: $\Sigma = \{0, 1\}$
    - Start state: $q_0 = q$
    - Final states: $F = \{q_{001}\}$
    - Transition function:

δ	**0**	**1**
q	q_0	q
q_0	q_{00}	q
q_{00}	q_{00}	q_{001}
q_{001}	q_{001}	q_{001}



DFA Review

- A finite automata is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where:
  1. Q is a finite set of states
  2. Σ is a finite set called the alphabet
  3. $\delta: Q \times \Sigma \rightarrow Q$ is the transition function
  4. $q_0 \in Q$ is the start state
  5. $F \subseteq Q$ is a set of accept states
- Deterministic means that <u>each</u> state has <u>exactly one</u> transition <u>for each</u> symbol in the alphabet

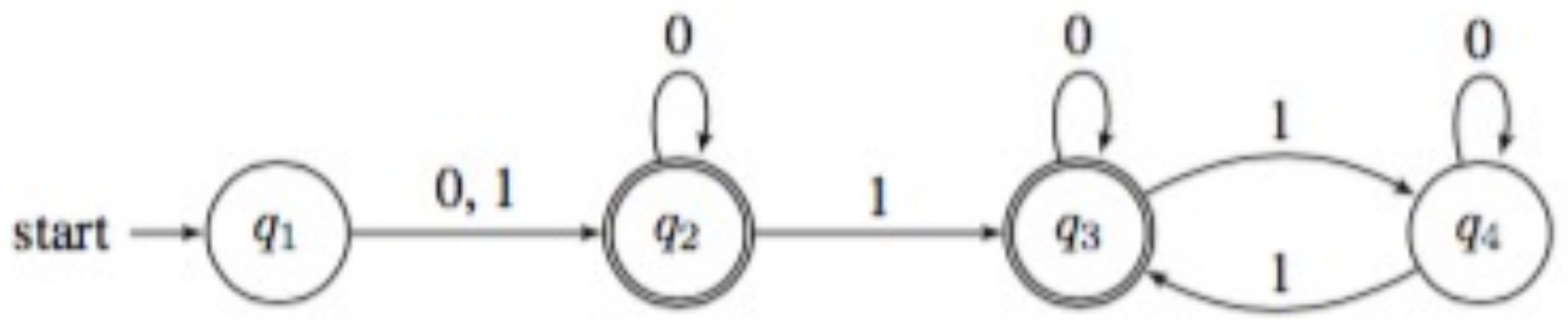
Review, cont.

- Language of Machine M (the DFA):
  - If A is a set of all strings that machine M accepts, say A is the language of M ($L(M) = A$)
  - M recognizes A
  - $L(M) = \{x \mid x \text{ is accepted by } M\}$
    - Build a machine M to accept L(M) or A
  - $A \subseteq \Sigma^*$ (Σ^* means all possible strings formed from symbols in the alphabet Σ)

School of Engineering | Computer Science

Review, cont.

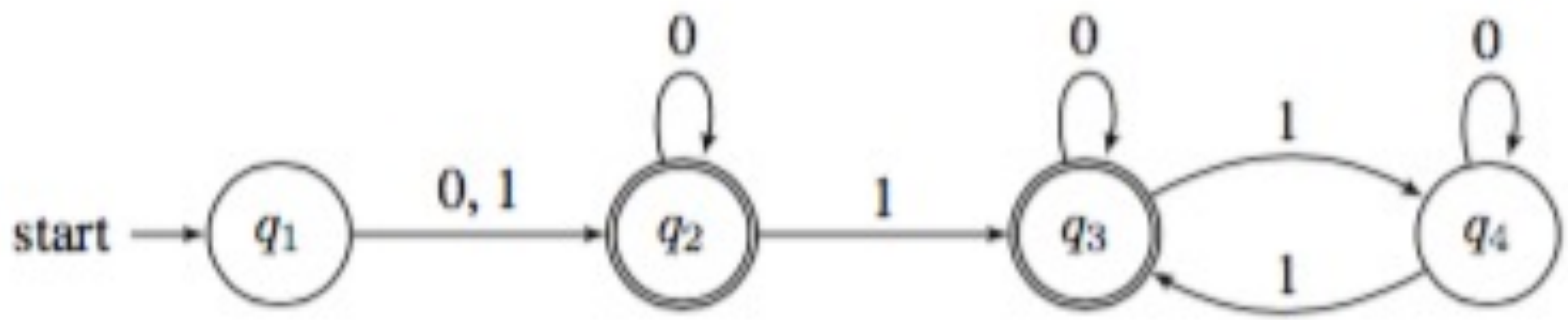
- Formally describe the language of this DFA:



  - Will this DFA accept string 00111?
  - Give the formal description of this machine.

Review, cont.

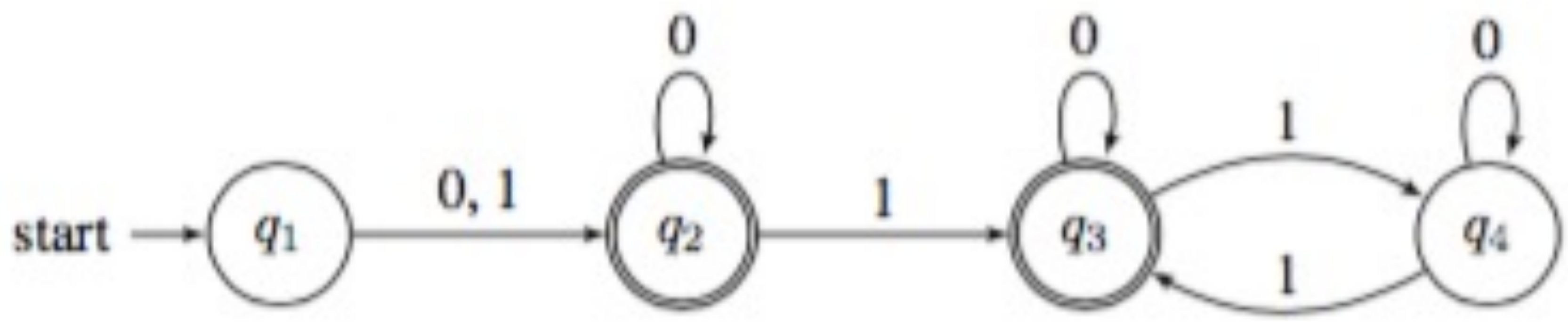
- Formally describe the language of this DFA:



  - Will this DFA accept string 00111?
    - Ans: $q_1 \xrightarrow{0} q_2 \xrightarrow{0} q_2 \xrightarrow{1} q_3 \xrightarrow{1} q_4 \xrightarrow{1} q_3$ (accept state, so Yes!

Review, cont.

- Formally describe the language of this DFA:



  - Give the formal description of this machine.
    - $Q = \{q_1, q_2, q_3, q_4\}$, $\Sigma = \{0, 1\}$, $q_0 = q_1$, $F = \{q_2, q_3\}$

δ	**0**	**1**
$\mathbf{q_1}$	q_2	q_2
$\mathbf{q_2}$	q_2	q_3
$\mathbf{q_3}$	q_3	q_4
$\mathbf{q_4}$	q_4	q_3

Review, cont.

- Given a DFA M = ({q_1, q_2, q_3, q_4, q_5}, {0, 1}, δ, q_1, {q_3, q_4, q_5}) where δ is

δ	**0**	**1**
$\mathbf{q_1}$	q_2	q_2
$\mathbf{q_2}$	q_3	q_1
$\mathbf{q_3}$	q_2	q_4
$\mathbf{q_4}$	q_5	q_4
$\mathbf{q_5}$	q_3	q_5

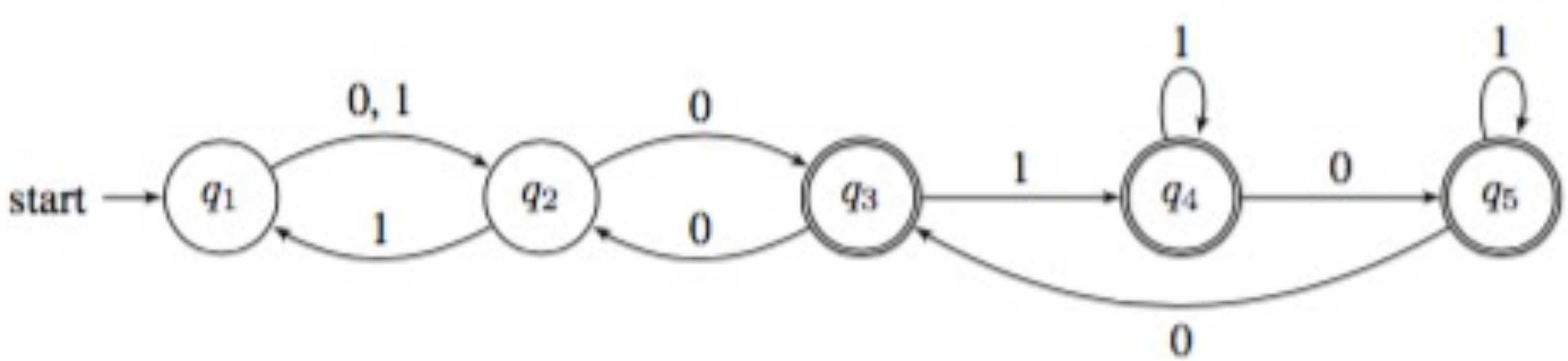
  - Draw the state diagram

Review, cont.

- Given a DFA M = ({q_1, q_2, q_3, q_4, q_5}, {0, 1}, δ, q_1, {q_3, q_4, q_5}) where δ is

δ	**0**	**1**
$\mathbf{q_1}$	q_2	q_2
$\mathbf{q_2}$	q_3	q_1
$\mathbf{q_3}$	q_2	q_4
$\mathbf{q_4}$	q_5	q_4
$\mathbf{q_5}$	q_3	q_5

  - State diagram:



Try It

- Design a finite automata where $L(M) = \{x \in \{0, 1\}^* \mid x$ begins with a 1 and ends with a 0$\}$

Try It

- Design a finite automata where $L(M) = \{x \in \{0, 1\}^* \mid x \text{ begins with a 1 and ends with a 0}\}$