CS 483 - Introduction to the Theory of Computation

Lecture 1

Logistics

Instructor: Wei Zhan

- Email: weizhan@purdue.edu
- Office: DSAI 2055
- Office hour: Tuesday 2:30 - 3:30pm at DSAI 1100

Teaching assistant: None ☹

Announcements: Brightspace, course website

Discussions: Piazza

Logistics

Grading

- Homework: 50%
  - 7 assignments, 10% each, 5 highest
  - there are bonus questions, total capped at 50%
- Midterm exam: 20%
- Final exam: 30%
  - Both exams are in-person and closed-book
  - You can bring a letter-size cheat sheet to the exam.

Guaranteed letter grades

- A+ ≥ 95%, A ≥ 90%, A- ≥ 85%, B+ ≥ 80%, B ≥ 75%, B- ≥ 70%, C+ ≥ 65%, C ≥ 60%

What is Theory of Computation?

A theory for rigorously reasoning about computation:

- How to represent computers as a mathematical model?
- What problems can be computed?
- What problems can be efficiently computed?

What is Theory of Computation?

A theory for rigorously reasoning about computation:

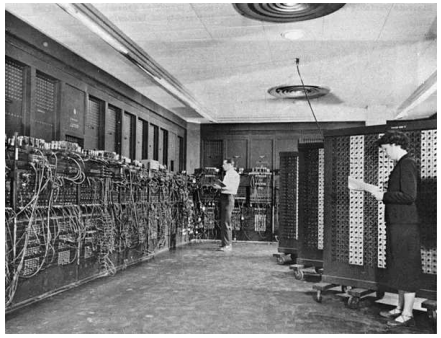
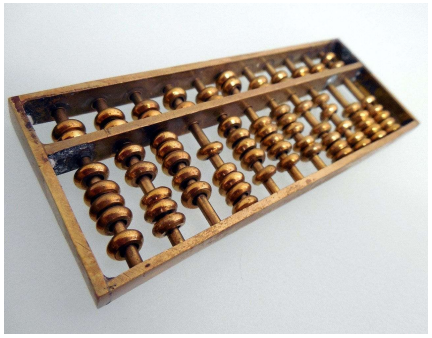
- How to represent computers as a mathematical model?

Input

a finite set of rules

Output




A Simple Task: Counting

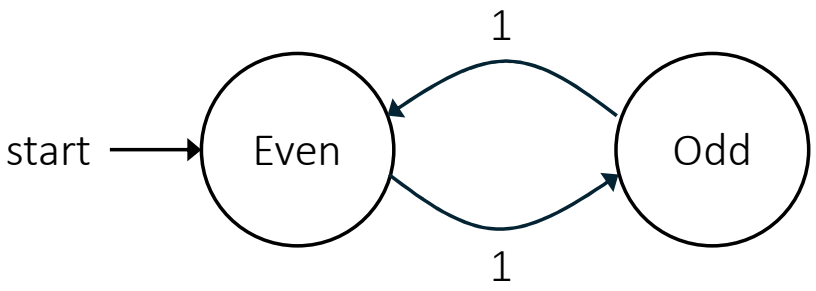
- Input: a sequence of 1’s
- Output: the number of 1’s (in binary)

A Simple Task: Counting

- Input: a sequence of 1's
- Output: the least significant bit of the number of 1's (in binary)

A Simple Task: Counting

- Input: a sequence of 1's
- Output: the least significant bit of the number of 1's (in binary)

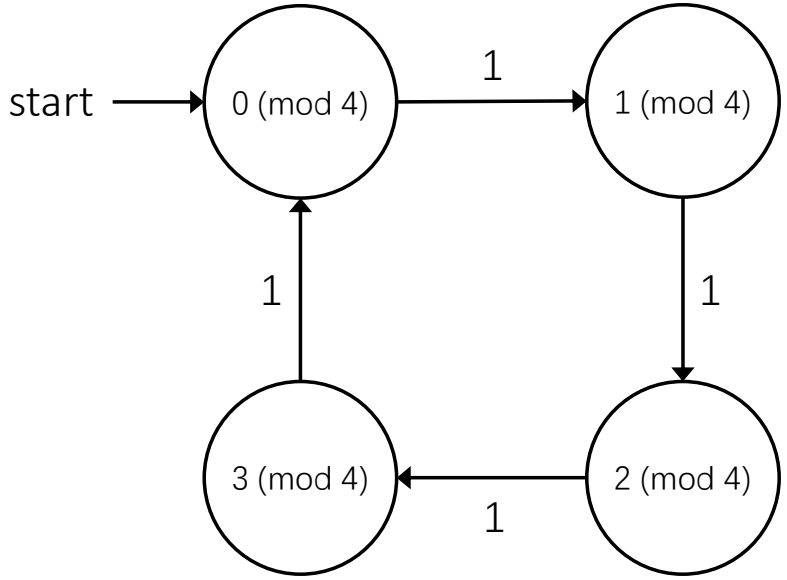


A Simple Task: Counting

- Input: a sequence of 1's
- Output: the second least significant bit of the number of 1's (in binary)

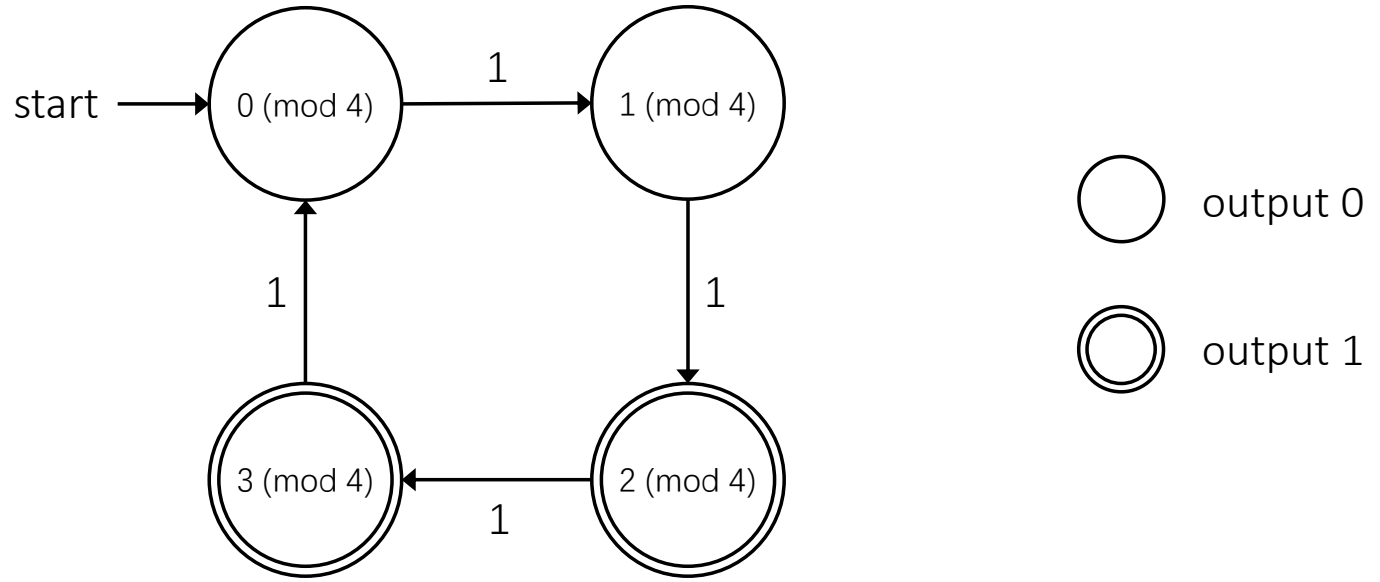
A Simple Task: Counting

- Input: a sequence of 1's
- Output: the second least significant bit of the number of 1's (in binary)



A Simple Task: Counting

- Input: a sequence of 1's
- Output: the second least significant bit of the number of 1's (in binary)

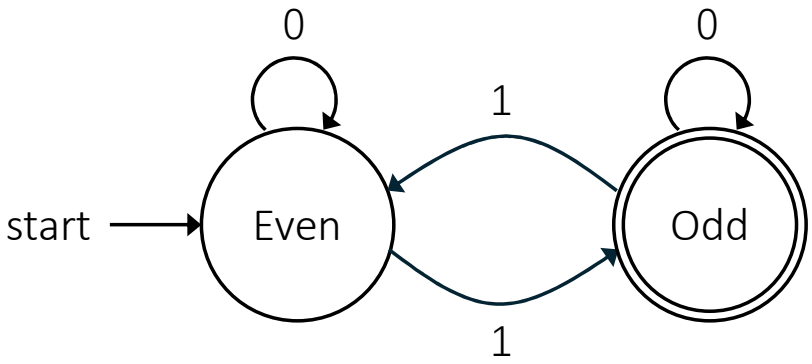


A Simple Task: Counting

- Input: a sequence of 0 and 1's
- Output: the least significant bit of the number of 1's (in binary)

A Simple Task: Counting

- Input: a sequence of 0 and 1's
- Output: the least significant bit of the number of 1's (in binary)



Finite Automaton

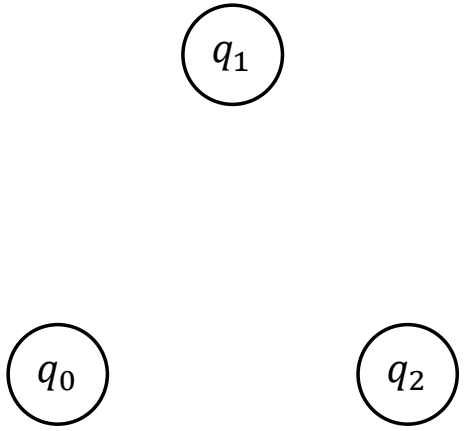
A Deterministic Finite Automaton (DFA) is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$:

Finite Automaton

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

- Q is the finite set of **states**

$$Q = \{q_0, q_1, q_2\}$$

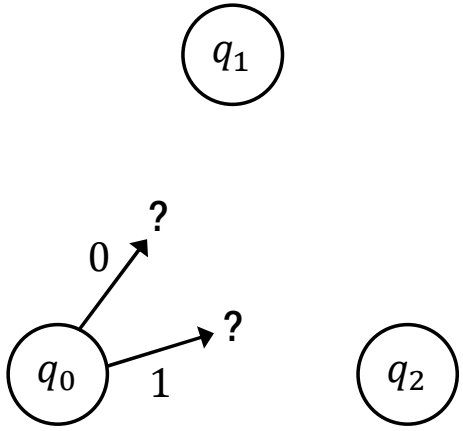


Finite Automaton

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

- Q is the finite set of states
- Σ is the finite set of symbols (**alphabet**)

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

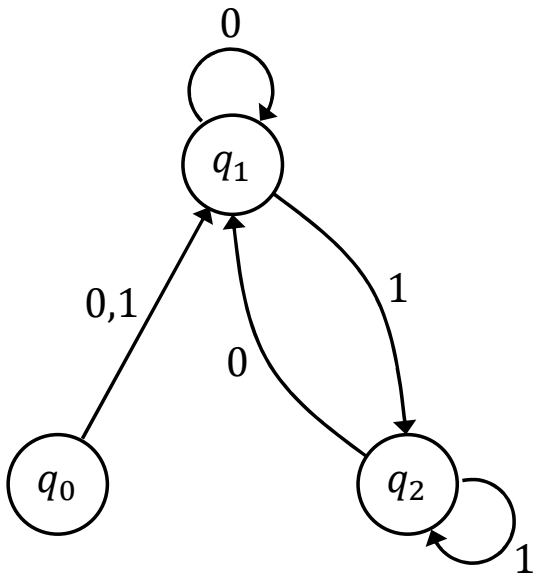


Finite Automaton

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

- Q is the finite set of states
- Σ is the finite set of symbols (alphabet)
- $\delta: Q \times \Sigma \to Q$ is the **transition function**

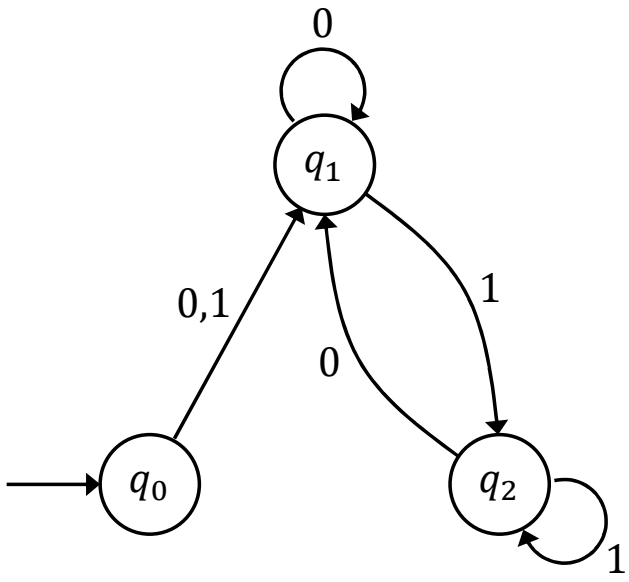
δ	0	1
q_0	q_1	q_1
q_1	q_1	q_2
q_2	q_1	q_2



Finite Automaton

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

- Q is the finite set of states
- Σ is the finite set of symbols (alphabet)
- $\delta: Q \times \Sigma \to Q$ is the transition function
- $q_0 \in Q$ is the **initial state**

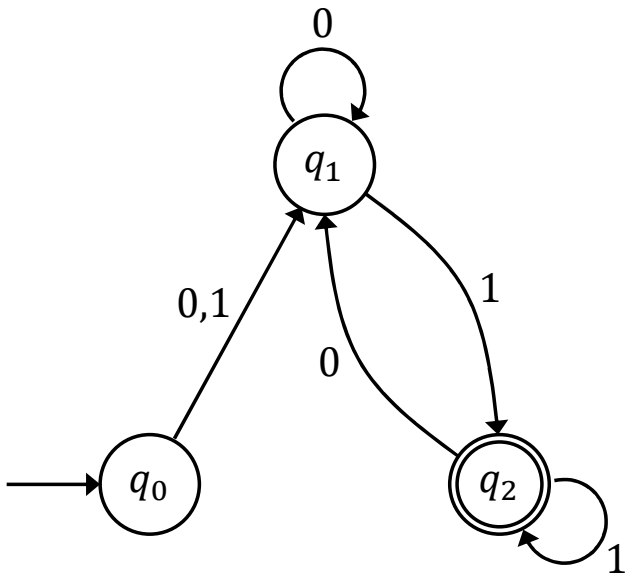


Finite Automaton

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

- Q is the finite set of states
- Σ is the finite set of symbols (alphabet)
- $\delta: Q \times \Sigma \to Q$ is the transition function
- $q_0 \in Q$ is the initial state
- $F \subseteq Q$ is the set of **accept states**

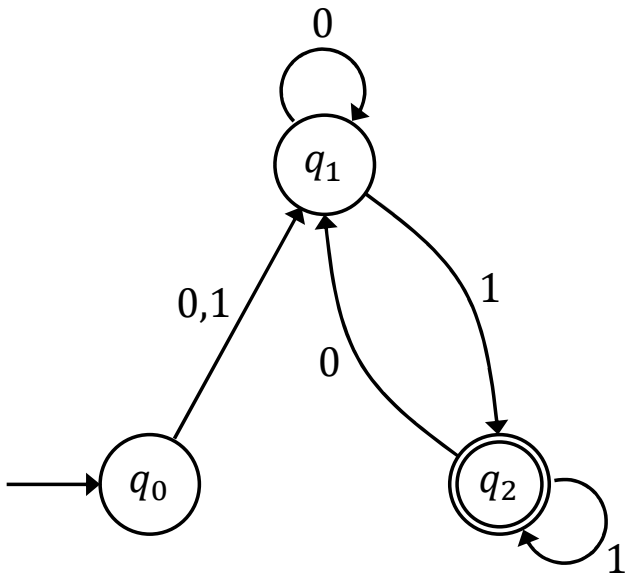
$$F = \{q_2\}$$



Finite Automaton

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

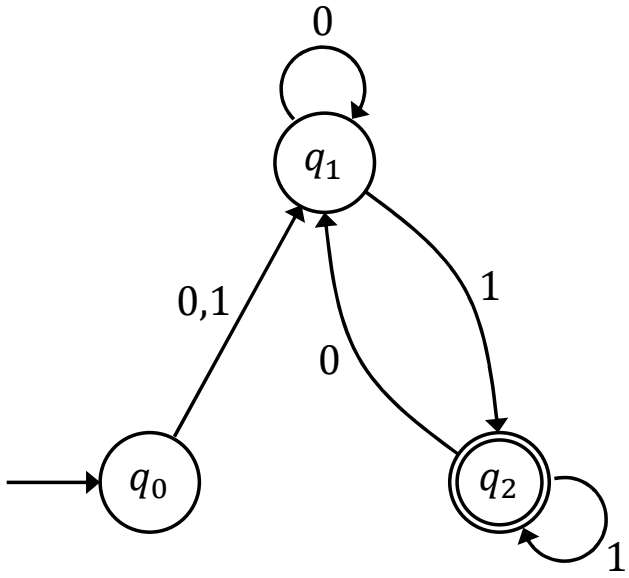
- Q is the finite set of states
- Σ is the finite set of symbols (alphabet)
- $\delta: Q \times \Sigma \to Q$ is the transition function
- $q_0 \in Q$ is the initial state
- $F \subseteq Q$ is the set of accept states



state diagram of M

Execution of Finite Automaton

Input: $0\ 1\ 0\ 0\ 1$

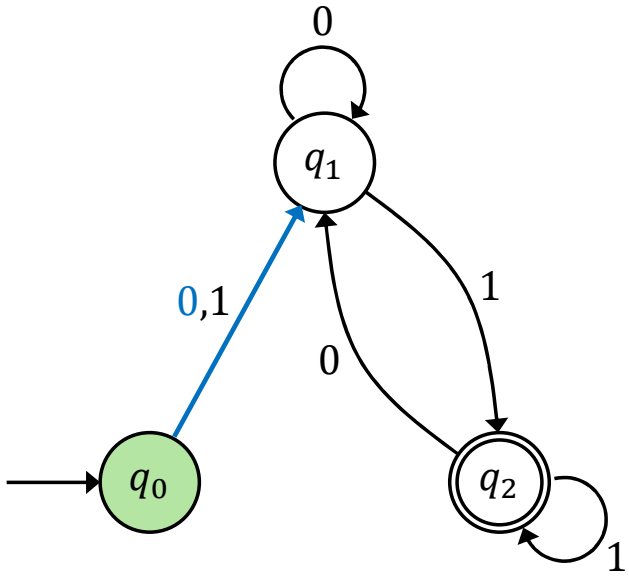


Execution of Finite Automaton

Input: 0 1 0 0 1

q_0

$\delta(q_0, 0) = q_1$

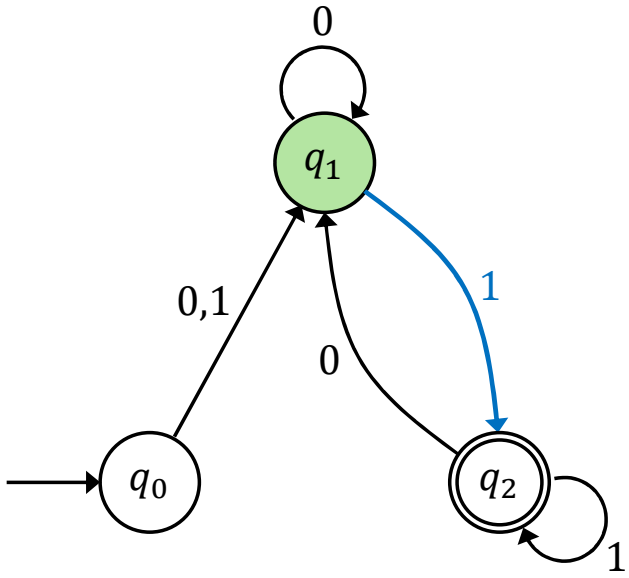


Execution of Finite Automaton

Input: 0 1 0 0 1

q_1

$\delta(q_1, 1) = q_2$

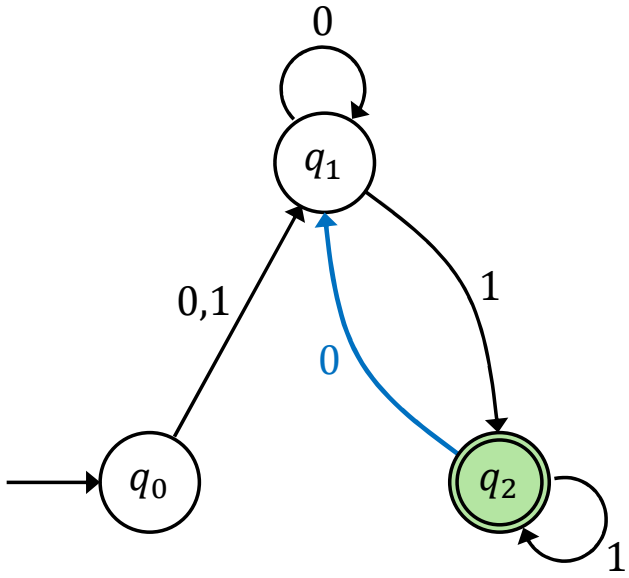


Execution of Finite Automaton

Input: 0 1 0 0 1

q_2

$\delta(q_2, 0) = q_1$

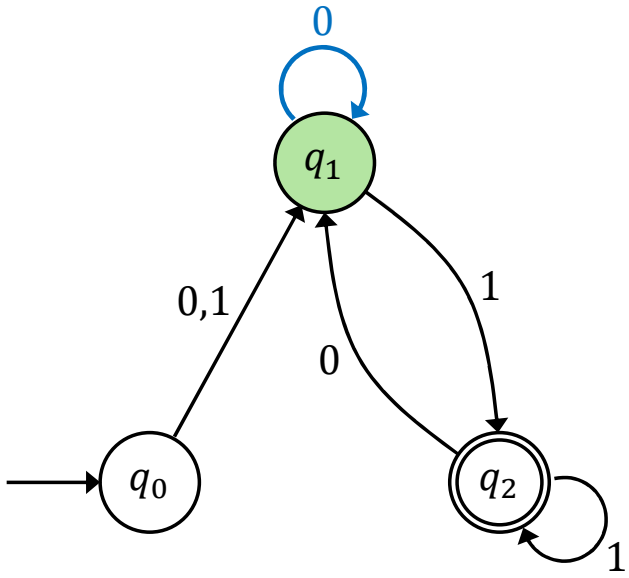


Execution of Finite Automaton

Input: 0 1 0 0 1

q_1

$\delta(q_1, 0) = q_1$

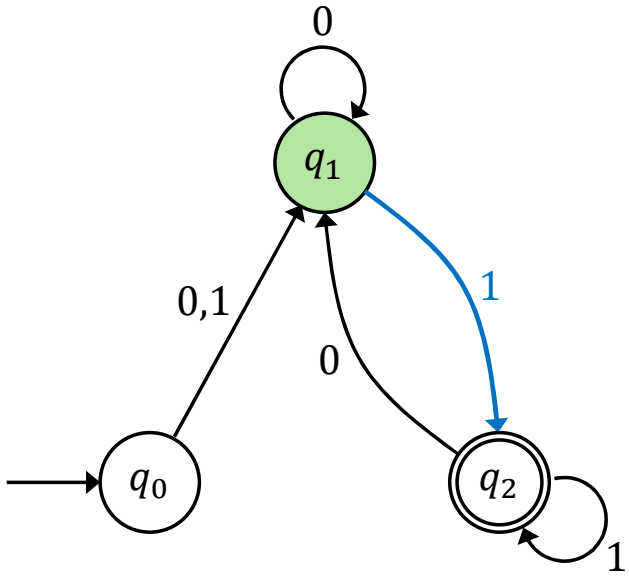


Execution of Finite Automaton

Input: 0 1 0 0 1

q_1

$\delta(q_1, 1) = q_2$

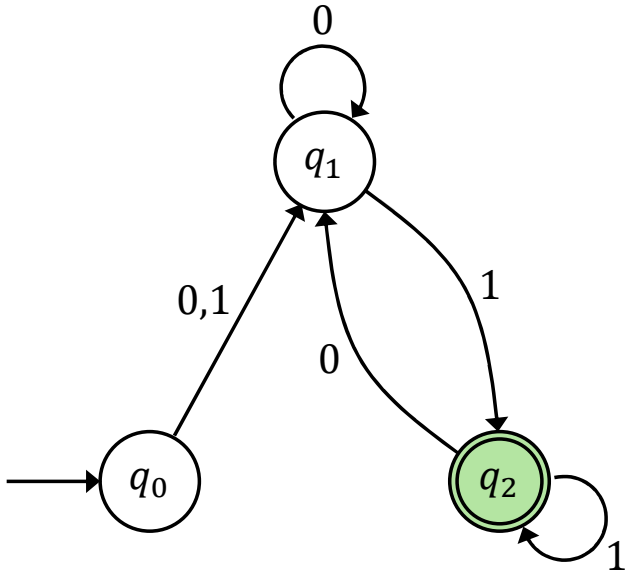


Execution of Finite Automaton

Input: $0\ 1\ 0\ 0\ 1$

$q_2 \in F$

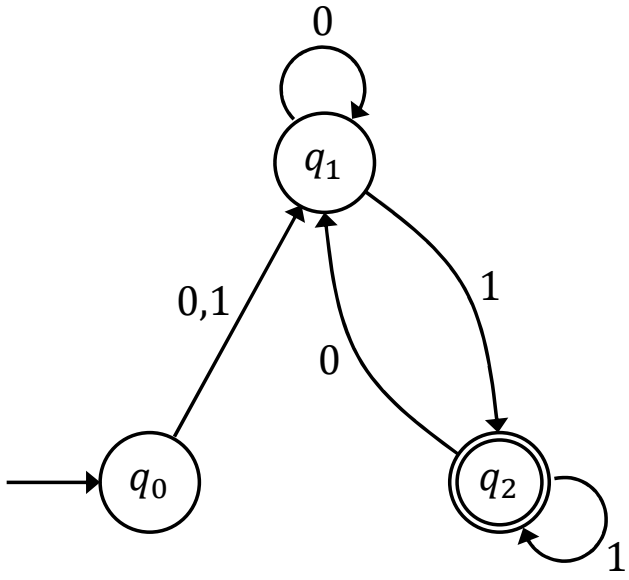
Output: 1 (accept)



Execution of Finite Automaton

Input: $\mathbf{0\ 1\ 0\ 0\ 1\ 0}$ Output: $\mathbf{0}$ (reject)

In general, this DFA M accepts 0,1-strings that has length ≥ 2 and ends with 1.



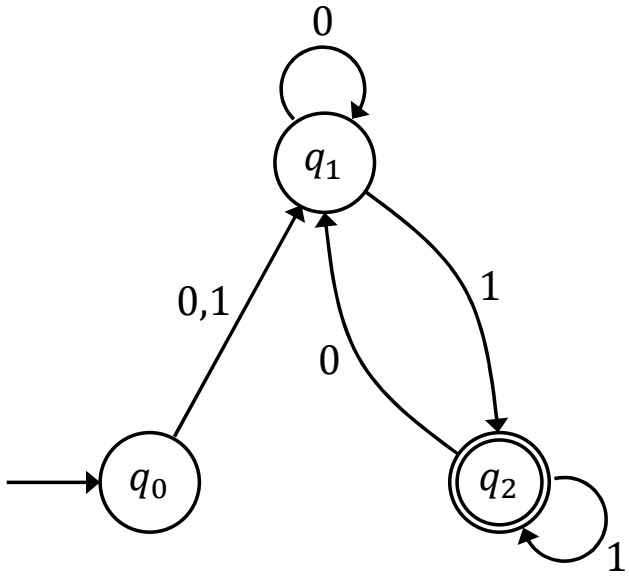
Execution of Finite Automaton

Formally, on input $w = w_1 w_2 \ldots w_n$, the DFA $M = (Q, \Sigma, \delta, q_0, F)$ reaches a sequence of states:

$$r_0 = q_0$$

$$r_i = \delta(r_{i-1}, w_i) \text{ for } i = 1,2, \ldots, n.$$

It accepts w if and only if $r_n \in F$.



Languages

A language is a set of strings (over the alphabet Σ).

The language computed/accepted/recognized by the DFA M, denoted as $L(M)$, is the set of strings accepted by M when given as the input.

A language is regular, if it can be computed by a DFA.

Languages

A language is a set of strings (over the alphabet Σ).

The language computed/accepted/recognized by the DFA M, denoted as $L(M)$, is the set of strings accepted by M when given as the input.

A language is regular, if it can be computed by a DFA.

Example of regular languages on alphabet $\{0,1\}$:

- $\{ w \mid \text{Hamming weight of } w \equiv 0 \pmod 2 \}$
- $\{ w \mid \text{Hamming weight of } w \equiv 0 \pmod k \}$, for every constant k
- $\{w = w_1 \dots w_n \mid n \geq 2, w_n = 1\}$

Languages

Throughout this course, we will model computational tasks as languages.

(Equivalently decision problems: problems that outputs either 0 or 1)

For example,

- Primality, on alphabet $\{0,1,\dots,9\}$: $\{w \mid w \text{ is a prime number}\}$
- Maximum-matching, on alphabet $\{0,1,"(",")",",",\}$:
  $\{ w = (G,k) \mid \text{graph } G \text{ has a matching of size at least } k \}.$
- Cat image, on alphabet $\{0,1,\dots,255\}$: $\{w \mid w \text{ shows the image of a cat}\}$

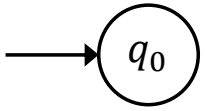
Languages

Regular languages are one of the simplest class of languages, as DFA is a very restricted model of computation:

- It reads each character in the input only once, and in one direction;
  - The input is treated as a one-pass stream
- The size of the DFA does not scale with the input length.

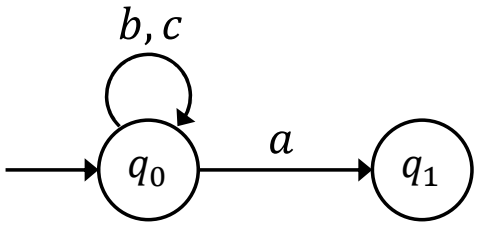
More Examples: Pattern Matching

On alphabet $\{a, b, c\}$, accepts strings that contains the (not necessarily consecutive) substring aba.



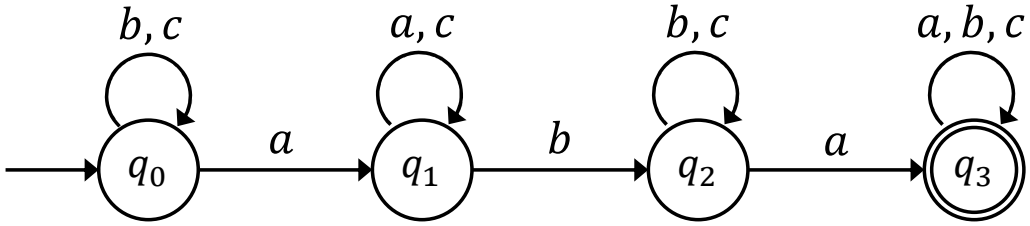
More Examples: Pattern Matching

On alphabet $\{a, b, c\}$, accepts strings that contains the (not necessarily consecutive) substring aba.



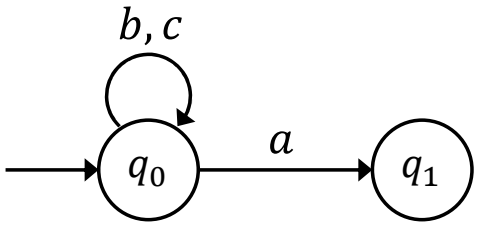
More Examples: Pattern Matching

On alphabet $\{a, b, c\}$, accepts strings that contains the (not necessarily consecutive) substring aba.



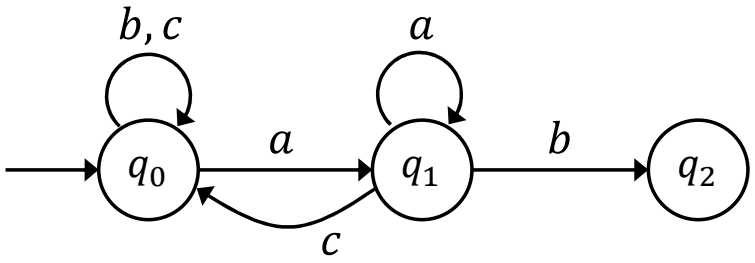
More Examples: Pattern Matching

On alphabet $\{a, b, c\}$, accepts strings that contains the consecutive substring aba.



More Examples: Pattern Matching

On alphabet $\{a, b, c\}$, accepts strings that contains the consecutive substring aba.



More Examples: Pattern Matching

On alphabet $\{a, b, c\}$, accepts strings that contains the consecutive substring aba.