06 Jan 2025 - Theory of Computation - Week 01

What are the fundamental limits and capabilities of computers?

→ different interpretations in areas of automata, computability & complexity

computer —abstraction→ mathematical device

P vs. NP problem

input ←
size : n
(∈ log n)

non-efficient algorithms
2^n, n^n, $n^{\log n}$

efficient
⇒ polynomial time
n^2, n^3, n^{100}

factorization

input : int m

output : int a, b ≠ 1 ; s.t
$m = a \times b$

Claimed solution : a_1, b_1

claim : $a_1 \times b_1 = m$

How to check? multiply and check easily

Solution? Check one by one : time required $= O(n)$

$= 2^{\log n}$

solve efficiently $\Rightarrow$ verify efficiently

verify efficiently $\overset{(?)}{\Rightarrow}$ solve efficiently

→ We do not yet know how to factorize efficiently

(basis for credit card pins / secure transaction / encryption)

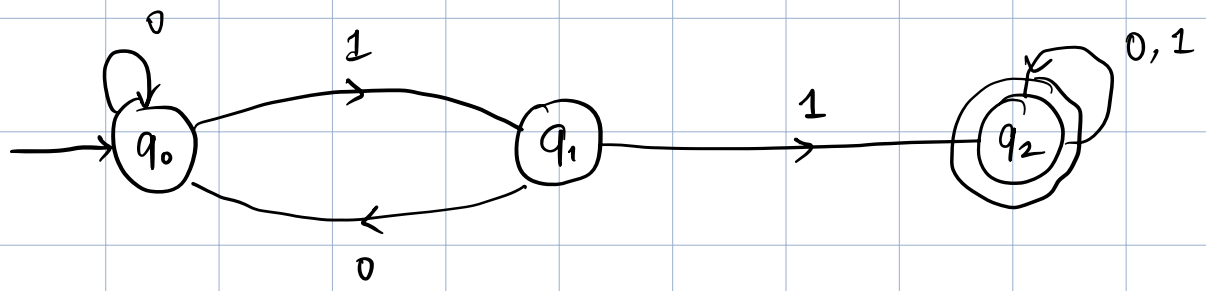
Computability theory (1900s) "computation in principle" → do not care about time, space can you come up with a solution

Complexity theory "computation in practice"

Computability theory → classification of problems as solvable or not ← complexity theory : easy or hard

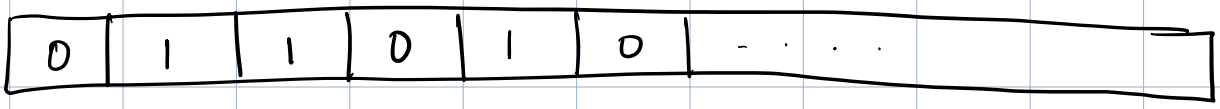
Finite Automata ~ switches / machine

M_1



nodes are called states

input tape



$Q = \{q_0, q_1, q_2\}$ finite set of states

q_0 : start state

q_2 : final state

Σ : finite set of symbols (0, 1)

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

$\delta : Q \times \Sigma \longrightarrow Q$ } transition function

$\delta(q_0, 0) = q_0$

$\delta(q_0, 1) = q_1$

M_1 accepts a string $w = w_1, w_2, \ldots$

if the run of w on M_1 ends at a final state.

(There can be multiple final states)

0	1	0	1	0	1	0	1	
q_0	q_0	q_1	q_0	q_1	q_0	q_1	q_0	q_1

$$L(M_1) = \{ w \in \{0,1\}^* \mid M_1 \text{ accepts } w \}$$

- $L(M_1)$: set of strings — language
- $\{0,1\}^*$: all finite binary strings

→ a language $L \subseteq \Sigma^*$

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

- Q: finite set of states
- Σ: finite set of alphabets
- δ: transition func
- q_0: start state
- $F \subseteq Q$: set of final states

→ You cannot build an automata that only accepts a particular $L \subseteq \Sigma^*$??

Doubts:

① Though we do not know an efficient algorithm to factorise a number, we still can do it in finite time. How does encryption work?

② Should the set F be non-empty? Not neccessarily

③ How does a machine behave as: ~~ finite states, or finite alphabets?

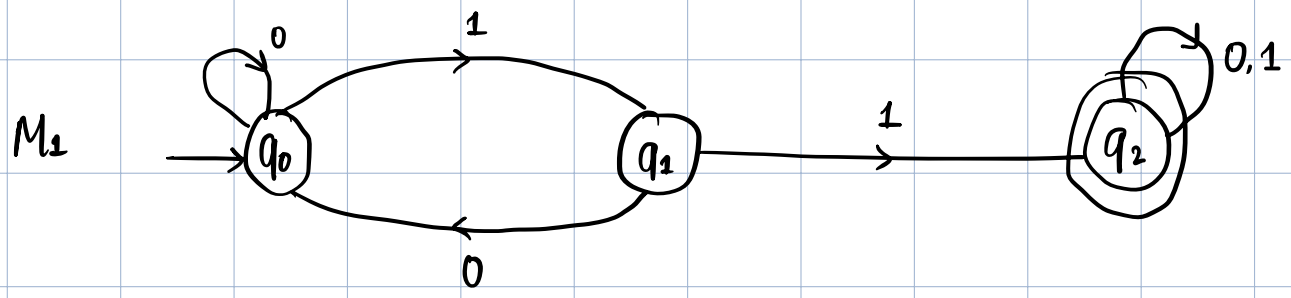
→ $Q \to \infty$ and finite Σ

→ $\Sigma \to \infty$ and finite Q

→ $Q, \Sigma \to \infty$

8 Jan 2025

Finite Automata



Def^n:

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

finite automata

- Q : finite set of states
- Σ : finite set of alphabets
- $\delta : Q \times \Sigma \longrightarrow Q$
- $q_0 \in Q$ is the start state
- $F \subseteq Q$ is the set of final states
  - can be empty

alphabets : finite set of symbols
- atomic

$\Sigma = \{0, 1\}, \{a, b, c\}, \{0, 1, \ldots, 9\}$

Σ^* := set of all finite strings over Σ

$$\Sigma^* = \bigcup_{n \geq 0} \Sigma^n$$

(Σ^n: all strings of length n)

string = "bunch" (finite) of input alphabet symbols concatenated together

→ Σ^* is infinite, each element in Σ^* has finite length

$x \in \Sigma^*$

$|x|$:= length of string x

ε := empty string

$\varepsilon \in \Sigma^*$

$x_1 \; x_2 \; x_3 \; \cdots \; x_n$, each $x_i \in \{0,1\}$

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

$x = x_1 \; x_2 \; \cdots \; x_n$

M accepts x iff $\exists$ a sequence of states $r_0, \ldots, r_n$ s.t the following holds:

(1) $r_0 = q_0$

(2) $r_1 = \delta(q_0, x_1)$

$r_i = \delta(r_{i-1}, x_i) \quad \forall \; i \in [n]$

(3) $r_n \in F$

Language accepted by M

$$L(M) := \{ x \in \Sigma^* \mid M \text{ accepts } x \}$$

M recognizes $A \subseteq \Sigma^*$ iff $L(M) = A$

Any $A \subseteq \Sigma^*$ is a language (empty strings too)

$$L(M_2) = \{ x \in \{0,1\}^* \mid x \text{ contains } 11 \text{ as a substring} \}$$

$L(M_2)$ → B, $\{0,1\}^*$ → D

Proof: We need to prove ① $B \subseteq D$ ② $D \subseteq B$

Def^n: A language $A \subseteq \Sigma^*$ is called "regular", iff $\exists$ a finite automaton M such that $L(M) = A$

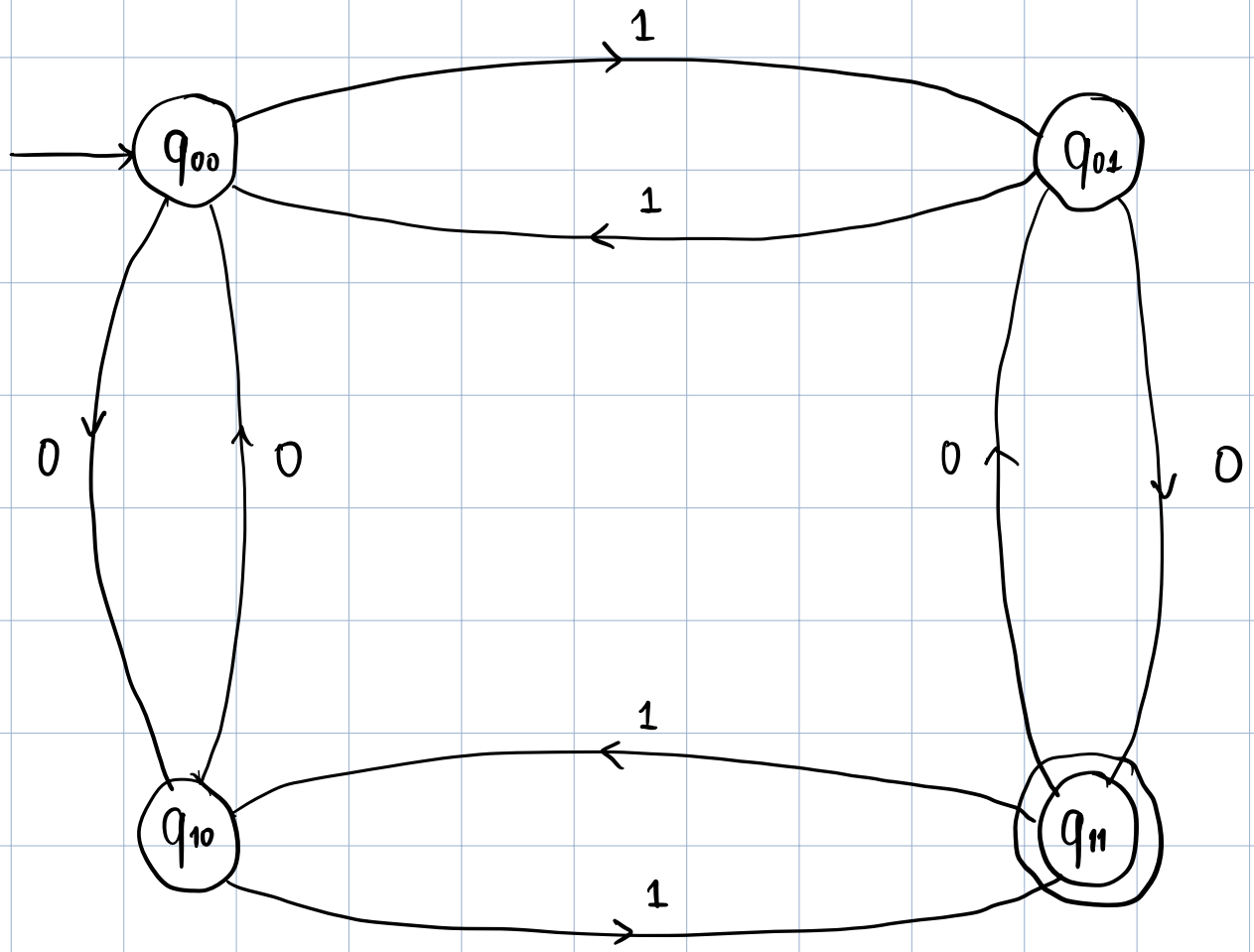
empty string $\neq$ empty set

string $\neq$ set

$\rightarrow \{\varepsilon\}$ is a language

$\rightarrow \phi$ is a language

$B = \{ x \in \{0,1\}^* \mid x$ contains odd no. of 0's and odd no. of 1's $\}$



0 (#0's = even)

0 (#1's = even)

1 (#1

1 q_{01}

0

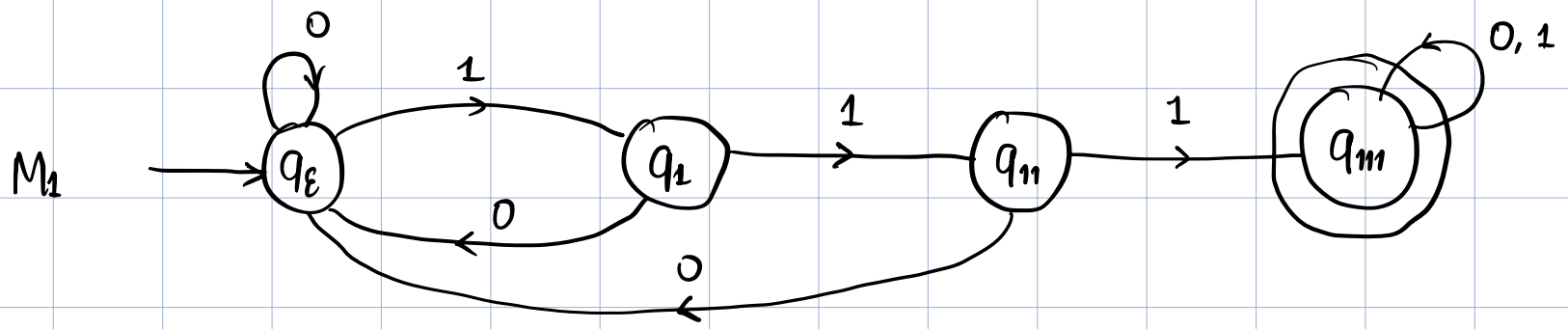
1 (# 0's = odd)

0 (#1's = even)

q_{10}

9 Jan 2025

$A = \{ x \in \{0,1\}^* \mid$ it contains 111 as a substring



To prove: $L(M_1) = A$

M_1 accepts every string in A
does not accept any string not in A.

Let $B = \{ x \in \{0,1\}^* \mid int(x) \text{ is a multiple of } 3 \}$

$11 \in B$? ✓

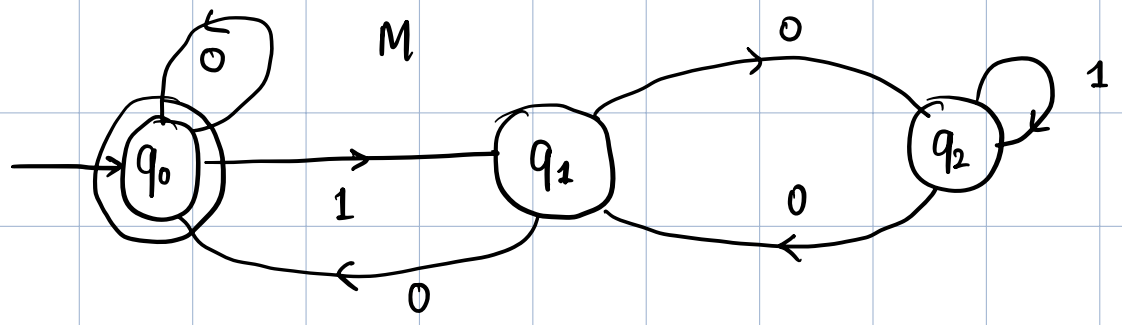
$0 \in B$? ✓

$1 \in B$? ✗

$10 \in B$? ✗

$x \in B$ iff $int(x) \equiv 0 \pmod 3$

q_0 q_1 q_2



$q_i \xrightarrow{c \in \{0,1\}} q_{(2i+c) \bmod 3}$

$x \rightarrow x0 \quad int(x0) = 2\, int(x)$

$x \rightarrow x1 \quad int(x1) = 2\, int(x) + 1$

$int(\varepsilon) = 0$

Prove: $L(M) = B$

1) $L(M) \subseteq B$

$x \in L(M)$

$\Downarrow$

$x \in B$

2) $B \subseteq L(M)$

$y \in B$

$\Downarrow$

$y \in L(M)$

Let $q \in Q$, $x \in \Sigma^*$, and

$\hat{\delta}(q, x)$:= the state where M will be after processing x starting at q.

$\hat{\delta} : Q \times \Sigma^* \longrightarrow Q$

→ $\hat{\delta}(q, \varepsilon) = q \qquad \forall\, q \in Q$

→ $\hat{\delta}(q, x) = \delta(\hat{\delta}(q, y), a)$

where $x = ya$ and $a \in \Sigma$

Claim: For every x,

$$\hat{\delta}(q_0, x) = q_j \quad , \quad \text{where} \quad j = \text{int}(x) \bmod 3$$

$x \in L(M) \iff \hat{\delta}(q_0, x) = q_0$

$\Updownarrow$ induction

$0 \equiv \text{int}(x) \pmod 3$

$\Updownarrow$

$x \in B$

Proves both 1) and 2)

Proof of the claim: Induction over $|x|$

Base case: $|x| = 0$ $(x = \varepsilon)$: $\hat{\delta}(q_0, \varepsilon) = q_0$

Induction hypothesis: True for all strings of length almost l.

Induction step: x s.t $|x| = l + 1$

$x = ya$

$$\hat{\delta}(q_0, x) = \delta(\hat{\delta}(q_0, y), a)$$

$$= \hat{\delta}(q_j, a) \quad \text{where} \quad j = \text{int}(y) \bmod 3$$

$$= q_{(2j+a) \bmod 3}$$

// To prove

$x \bmod 3$

$$(2j + a) \bmod 3 = (2\ \text{int}(y) + a) \bmod 3$$

$$= \text{int}(ya) \bmod 3 = \text{int}(x) \bmod 3.$$

Main idea of a finite automaton → finite memory requirement.