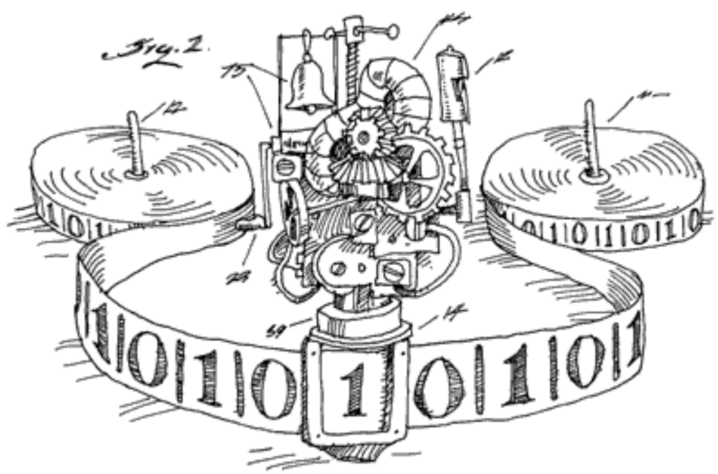
EECS 376: Foundations of Computer Science

Lecture 08 – Introduction to Computability



Introduction to Computability: Deterministic Finite Automata

Techniques and Paradigms in this Course

- Divide-and-conquer, greed, dynamic programming, the power of randomness

Problems that are **easy** for a computer

- Computability

Problems that are **impossible** for a computer

- NP-completeness and approximation algorithms

Problems that are **"probably hard"** for a computer

- Cryptography

Using "probably hard" problems for our benefit (hiding secrets)

What is a computer?

Plan in 5 lectures

Two models of computations (types of "hardware")

Finite Automaton = a person (system whose space **cannot** grow according input size)

- **Q:** What type of problems can a person solve?
- **A:** Very limited!

Turing Machine = a person + papers (as much as they want)

- **Q:** What type of problems can a person with papers solve?
- **A:** Every solvable problem! (Church-Turing thesis)
    - Ignoring efficiency: Nothing is more powerful than Turing Machine

Is every problem solvable? No. Why not?

Problems and Decision Problems

Def

An algorithm solves a problem
if it gives the correct solution
on every instance.

We'll define last 3 terms now.

What is a computational **problem**?

We'll start with an example.

Example problem:

MULTIPLICATION

Instance (also known as ***input***)	***Solution***
`3, 7`	`21`
`610, 25`	`15250`
`50, 610`	`30500`
`15251, 252`	`3843252`
`12345679, 9`	`111111111`

Example problem:

PALINDROME

Instance (*also known as **input***)	***Solution***
`a`	`Yes`
`10101`	`Yes`
`selfless`	`No`
`huh`	`Yes`
`376`	`No`
`emus sail i assume`	`Yes`

Def ⁿ

A problem is a collection of instances and the solution to each instance.

Def

Decision problem:

Problems where the solution is **Yes / No**.

(Also known as **True / False,**

1 / 0,

accept / reject.)

Languages

String notation

① **Alphabet:** **A nonempty finite set Σ of symbols.**

e.g. $\Sigma = \{0,1\}$ or $\Sigma =$ ASCII characters

ex: $\{0,1\}^3 = \{$all binary strings of length 3$\}$

② **String:** **A finite sequence of 0 or more symbols.**
(or "word")

The empty string is denoted ε.

Σ^k 是 finite 的

For any $a \in \Sigma$:

string — a^k means k a's

language — a^* means ≥0 a's

a^+ means ≥1 a's

Σ^k means all strings over Σ of length k.

Σ^* means **all** (finite) strings over Σ.

Σ^+ means all nonempty (finite) strings over Σ

$\Sigma^+ = \Sigma^* \cup \varepsilon$

For any $a,b \in \Sigma$: $a|b$ means a OR b

但是 Σ^* 本身 infinite
(就像 ℝ).

③ **Language:** **A (possibly infinite) set of strings.**

i.e. any subset $L \subseteq \Sigma^*$.

The empty language is denoted Ø.

Examples of Languages

L = a* = {ε, a, aa, aaa, …}

{0, 01, 011, 0111, …}

L = 01* = all strings of one 0 followed by zero or more 1's

L = $\{x^k y^k: k \geq 0\}$ = all strings consisting of some number of x's followed by the same number of y's {ε, xy, xxyy, ..}

Question: L = {ε}. Is L = Ø?

Answer: No. L has 1 element, Ø has 0 elements

Question: How is a*|b* different from (a|b)*

all possible combinations of all possible numbers of a|b

= {ε, a, b, ab, ba, bab, …}

note: a*, b* 是 language

即: {ε, a, aa, …} ∪ {ε, b, bb, …}

Decision Problems $\equiv$ Languages

Representing instances of a problems

The instances of a problem can be:

- numbers
- strings
- pairs of numbers
- lists of strings
- graphs
- images
- …

These can all be conveniently encoded by **strings.**

For an input G, $\langle G \rangle$ denotes its encoding as a string

Representing problems

We can encode instances and solutions as strings.

Thus, we can think of a **problem** as a **function**

$$f : \Sigma^* \rightarrow \Sigma^*$$

mapping instances (string) to solutions (string).

A **decision problem** can be thought of as

$$f : \Sigma^* \rightarrow \{\text{No}, \text{Yes}\}$$

Representing decision problems

A **decision problem** can be thought of as

$$f : \Sigma^* \to \{\text{No, Yes}\}$$

or equivalently as a **language**

$$L \subseteq \Sigma^*$$

$L = \{x \in \Sigma^* : f(x) = \text{Yes}\}$ $\quad$ $f(x) = \begin{cases} \text{Yes if } x \in L \\ \text{No if } x \notin L \end{cases}$

E.g.: $L_{\text{PALINDROME}} = \{x \in \Sigma^* : x \text{ is a palindrome}\}$

all strings

Let us practice this notations

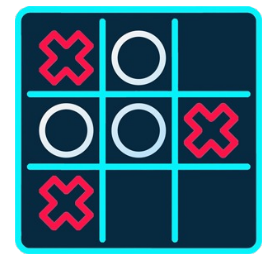
What is a language corresponding to this problem?

Q: Is an input number x even?

- $\langle x \rangle$ = binary string encoding a number x.
- $L_1 = \{\langle x \rangle \in \{0,1\}^* \mid x \bmod 2 = 0\}$

Q: Given a tic-tac-toe board B, can "x" always win?

- $\langle B \rangle$ = encoding of a current board B
- $L_2 = \{\langle B \rangle \in \{0,1,2\}^9 \mid \text{"x" can win in } B\}$



2	1	0
1	1	2
2	0	0

210112200

Upshot

Languages and decision problems

are two ways to think of the same things

First Model of Computation:

Deterministic Finite Automata

capturing any system whose space **cannot** grow according input size

This lecture:

A computational model:

只对 decisive problem 有用

Deterministic Finite Automata (DFA)

Weak model of computation

Capture a very weak type of algorithms

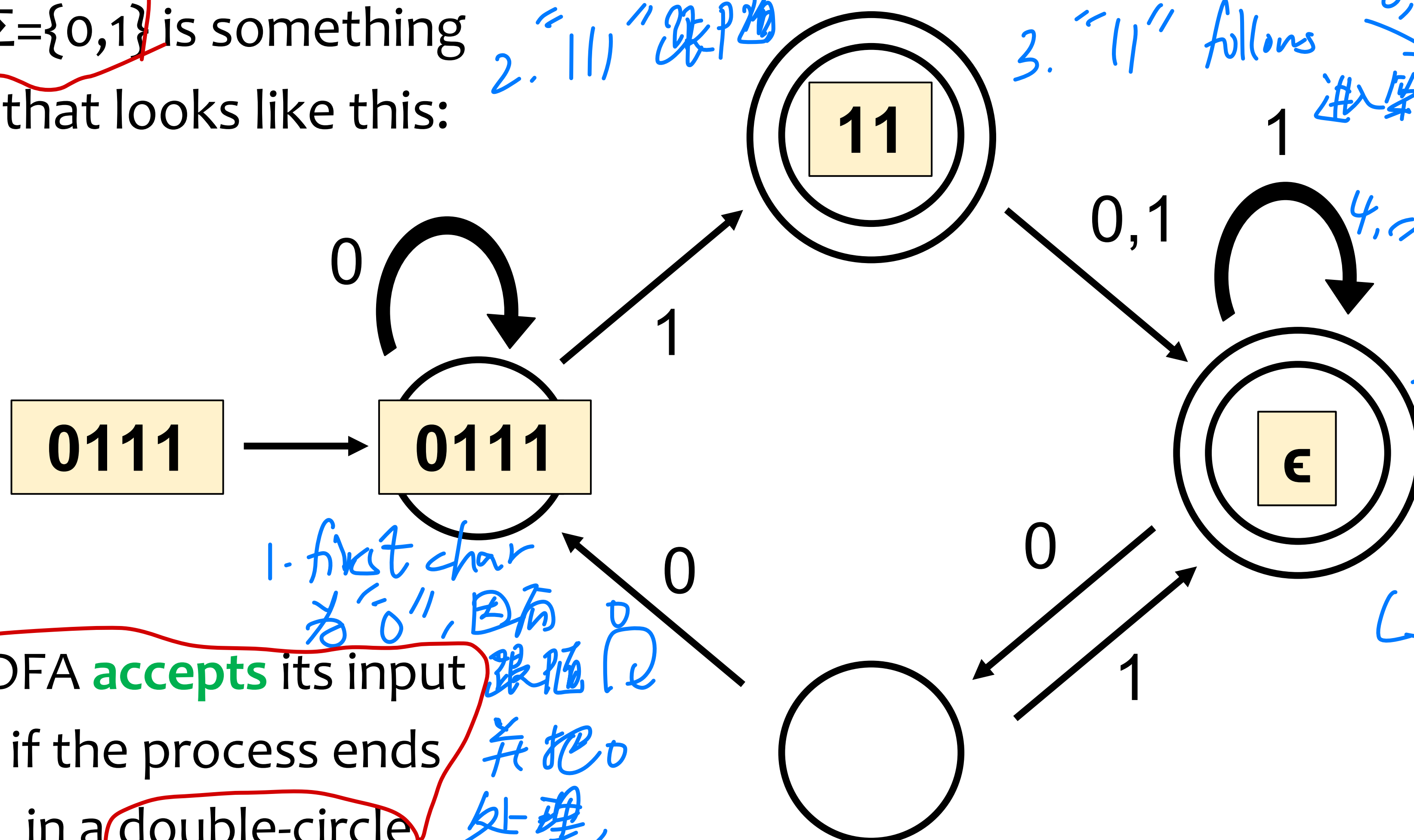
Good warmup before we study Turing Machine (most powerful model of computation)…

Deterministic Finite Automata (DFAs)

A DFA over alphabet $\Sigma=\{0,1\}$ is something that looks like this:

2. "111" 跟随 → 处理掉第一个1并进入第二个state

3. "11" follows "0,1" 进入第三个state



4. "1" 跟随 ↓ 被处理掉 ↓ ε (stop)

1. first char 为"0"，因而跟随 ↺ 并把0处理，变为111

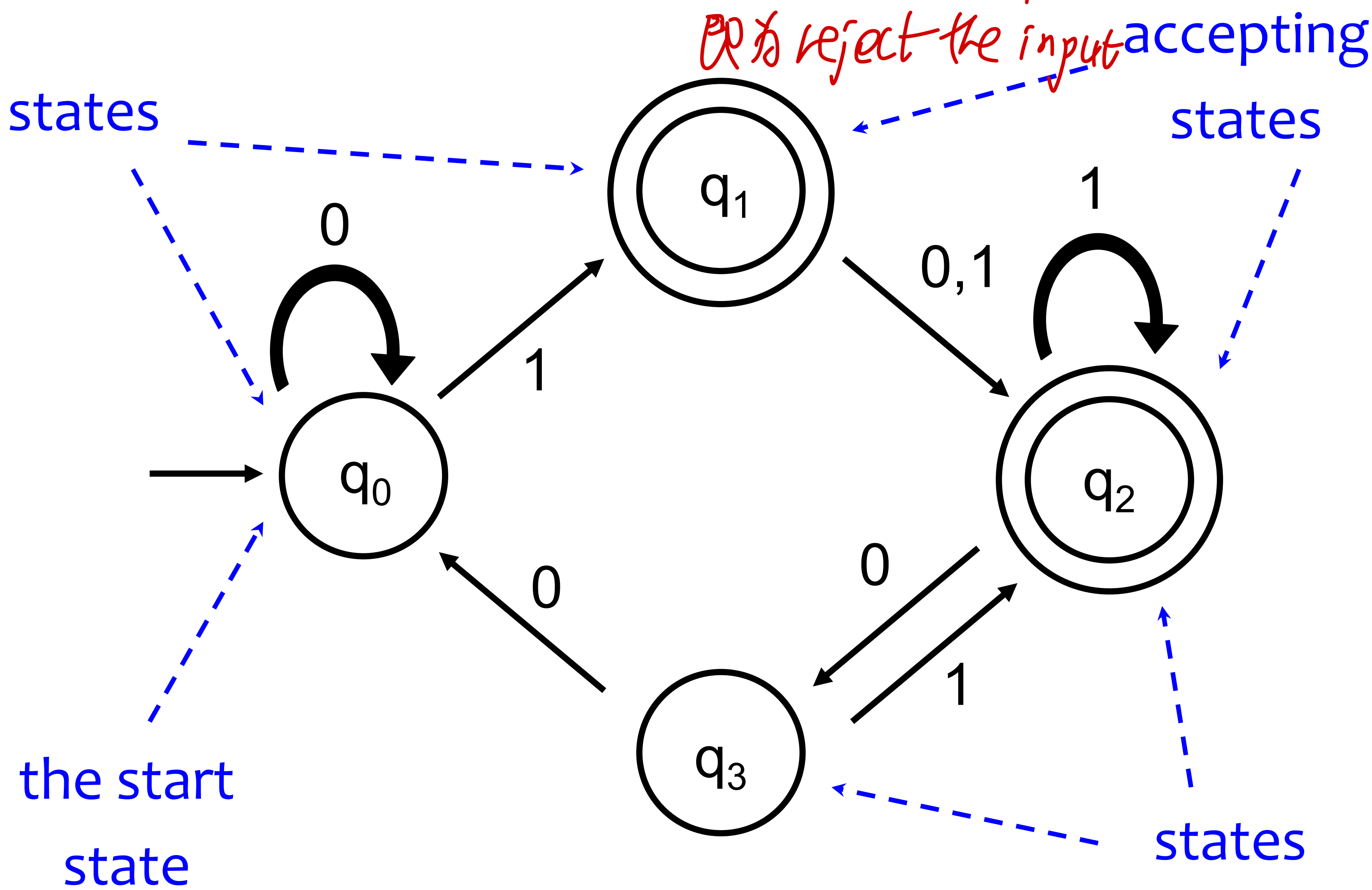
DFA **accepts** its input if the process ends in a double-circle

↓

即 accepting states

如果DFA最后停在accepting states, 即为accept the input; 停在别的地方即为reject the input

Anatomy of a DFA



transition rules: the labeled arrows

Computing with DFAs

Let M be a DFA, using alphabet Σ.

M accepts some strings in Σ^* and rejects the rest.

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

注意!

problem size

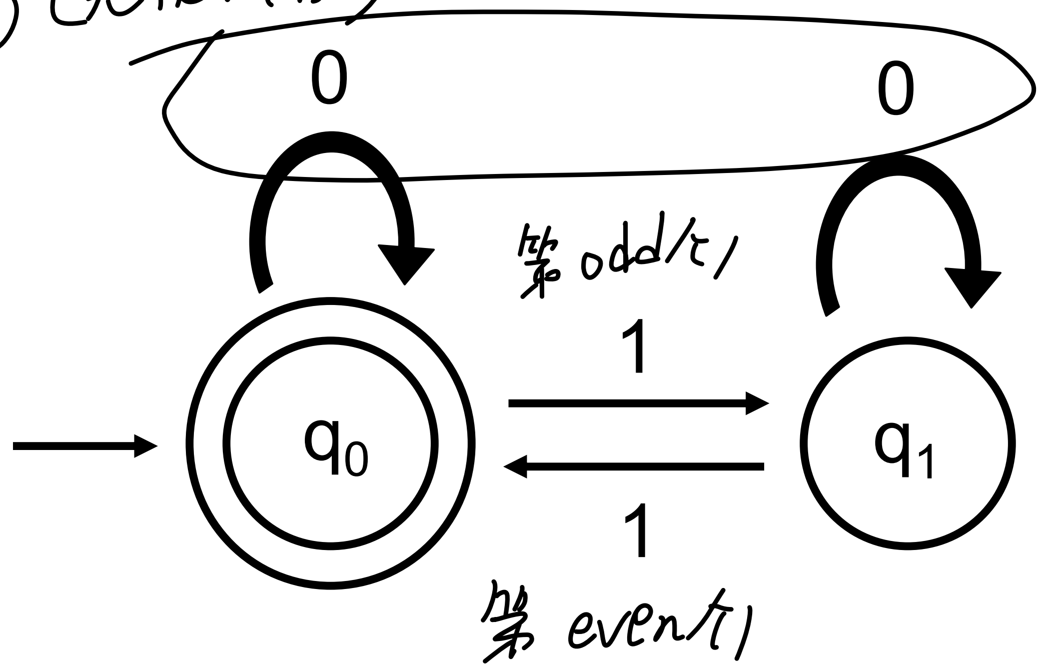
可以 infly large

Called the "language decided by M".

If L is a language,

we say M decides L if $L(M) = L$.

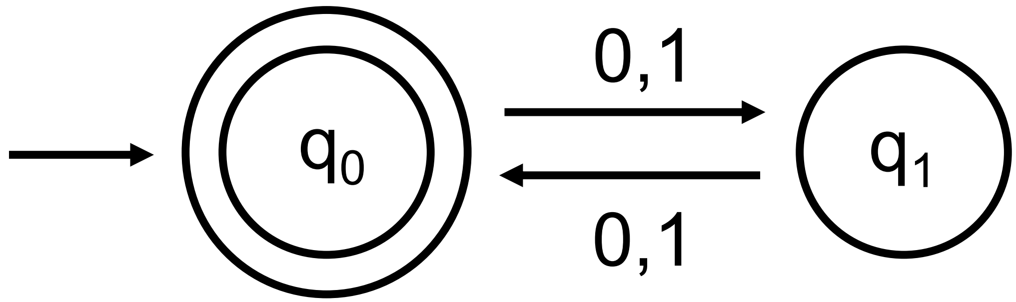
① (无视所有0)



What language does this DFA decide?

binary string with even 个 1

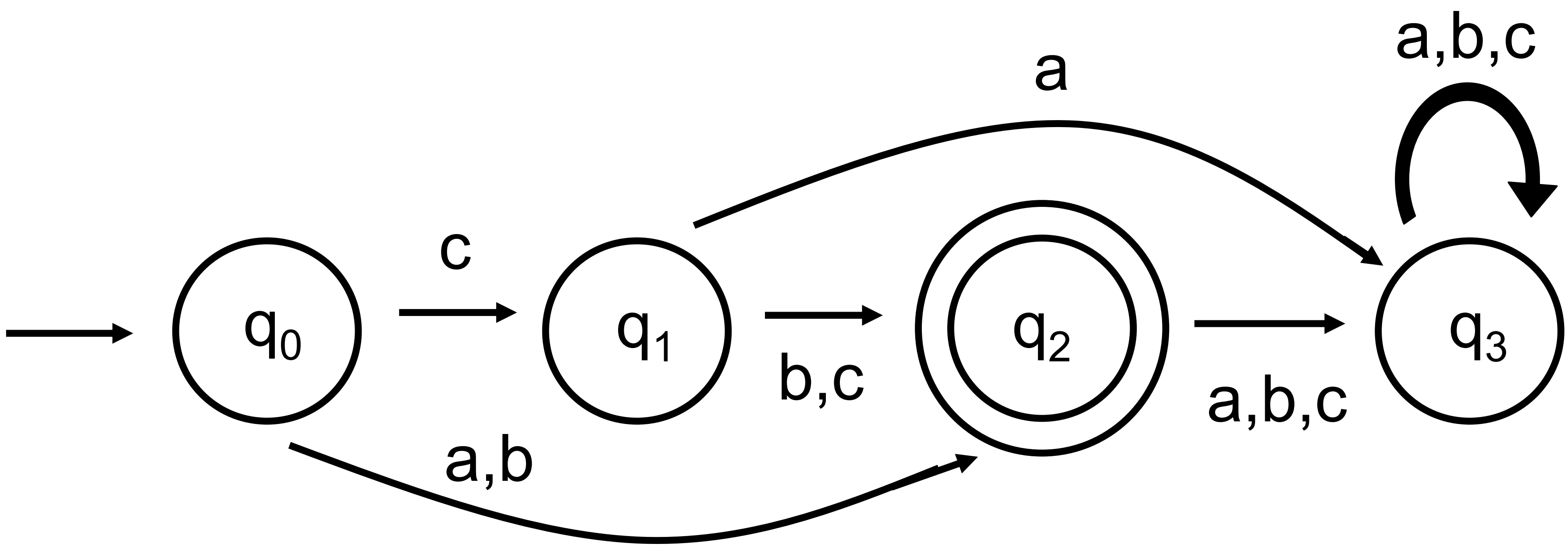
$(aa)^*$



What language does this DFA decide?

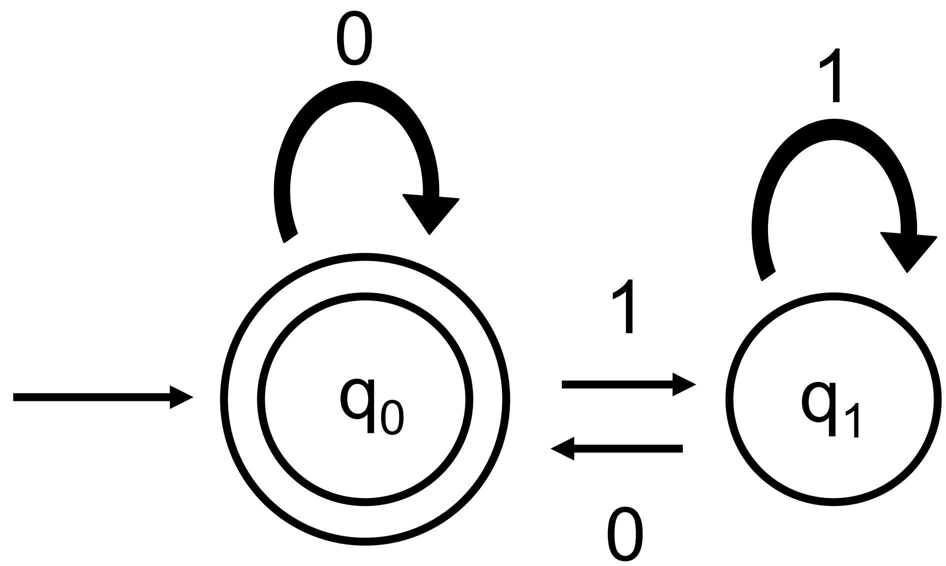
even length binary string

M is the following DFA,
with alphabet Σ={a,b,c}:



L(M) = {a, b, cb, cc}

M:



$L(M) = \{\varepsilon\} \cup \{\text{strings ending with } 0\}$

易证: swap 普通state 和 acc state 即可.

Fact: If there is a DFA that decides a language **L**, then there is also a DFA that decides the **complement of L**. Why?

E.g. **L** = {strings containing “01”}

complement of L = {strings NOT containing “01”}

Formal definition of DFAs

A deterministic finite automaton is a 5-tuple:

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

① Q is a nonempty finite set of states,

② Σ is an alphabet,

③ $\delta : Q \times \Sigma \rightarrow Q$ is the state-transition function,

④ $q_0 \in Q$ is the start state,

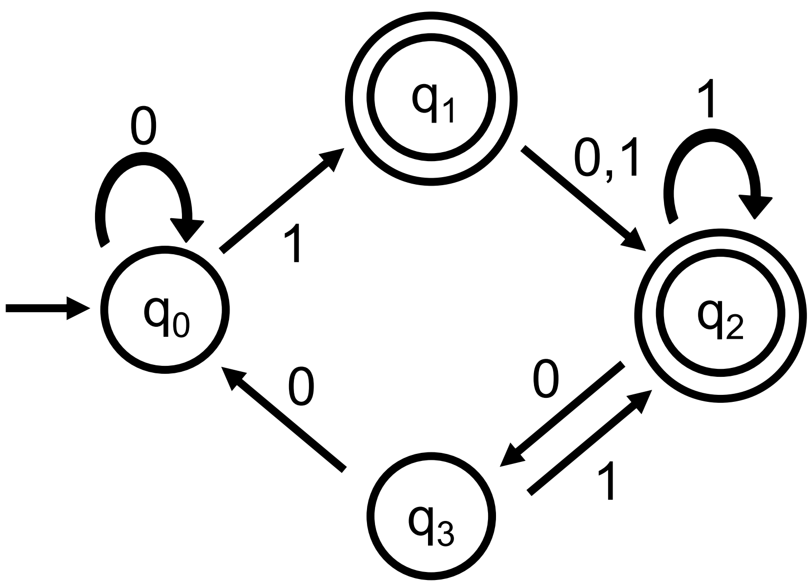
⑤ $F \subseteq Q$ is the set of accepting states.

即: graph 的 abstraction. 有几个 states, 哪个 start, 哪些 acc, 使用什么alphabet, states 间怎么 transit.

Formal definition of DFAs

A deterministic finite automaton is a 5-tuple:

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



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

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

δ we'll come back to

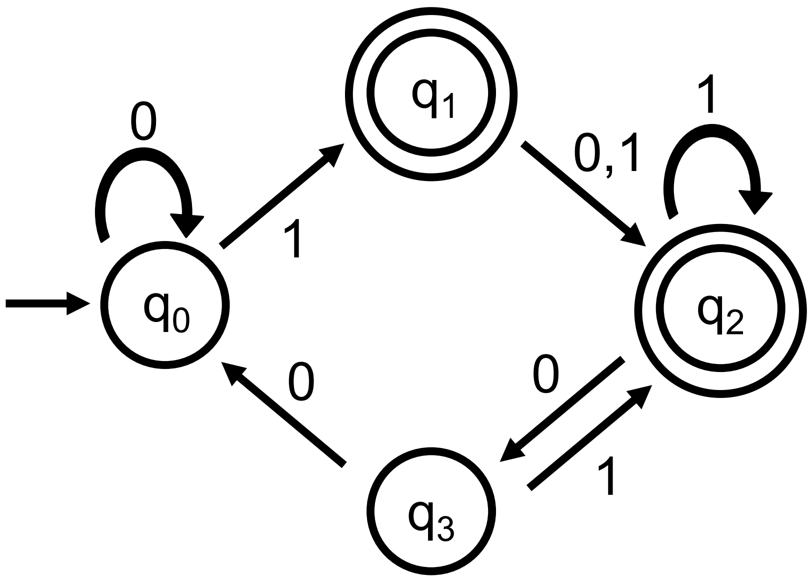
q_0 is the start state

$F = \{q_1,q_2\}$

Formal definition of DFAs

A deterministic finite automaton is a 5-tuple:

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



$\delta : Q \times \{0,1\} \rightarrow Q$ is…

δ	0	1
q_0	q_0	q_1
q_1	q_2	q_2
q_2	q_3	q_2
q_3	q_0	q_2

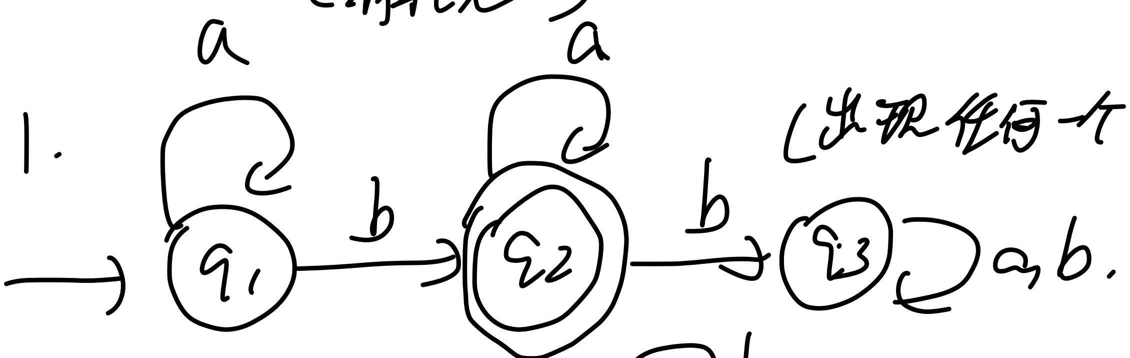
- For $\Sigma = \{a,b\}$, consider the language of strings with exactly one b
    1. Write using string notation a^*ba^*
    2. Draw a DFA

- Draw a DFA for $ab^*|ba^*$ $\{a, ab, abb, \ldots, b, ba, baa, \ldots\} \cup \varepsilon$

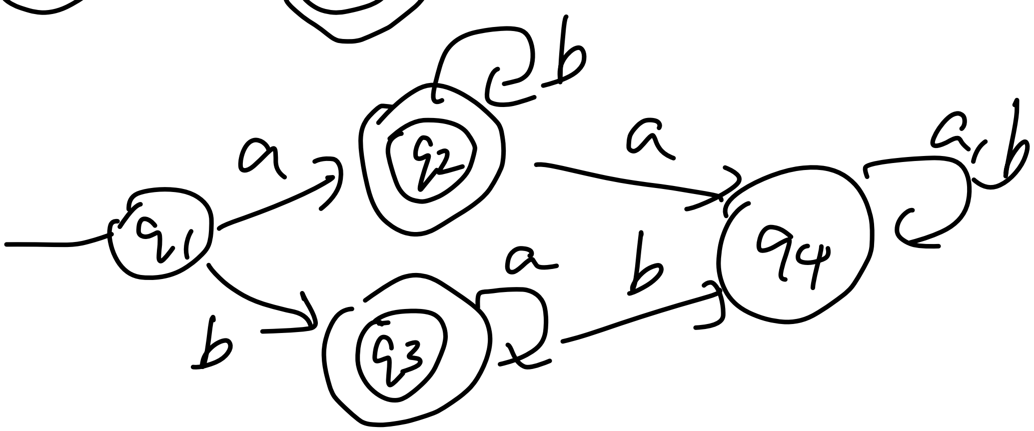
- Draw a DFA for $L = \{x^ky^k : k \geq 0\}$ $\{\varepsilon, xy, xxyy, \ldots\}$

(消耗完a)

1.



2.



Limitation and Characterization of **DFAs**

Thm: No DFA can decide $\{x^k y^k : k \geq 0\}$

Suppose for contradiction there's a DFA **M** that decides $\{x^k y^k : k \geq 0\}$

Let **s** = #states of **M**. 这是核心问题: memory是有限的!

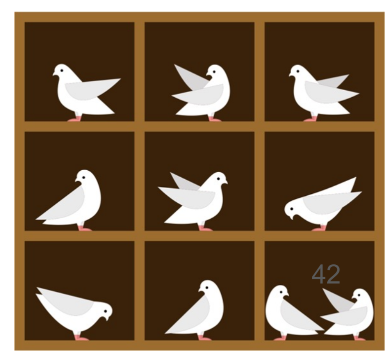
Consider the input string $\mathbf{x^{s+1}y^{s+1}}$.

There are **s states** (pigeonholes) and **s+1 x's** (pigeons), so there must be two values **i , j** ≤ s+1 such that **state**($\mathbf{x^i}$) = **state**($\mathbf{x^j}$).

Claim. The two inputs $\mathbf{x^i y^i}$ and $\mathbf{x^j y^i}$ have the same final state. Why?

But **M** is supposed to accept $\mathbf{x^i y^i}$ and reject $\mathbf{x^j y^i}$.

Contradiction!



Intuitive reason why DFAs cannot decide many languages:

Finite memory!

note: 何为decide?

Decide的意思是：对于无论多大的input，都能弄出结果

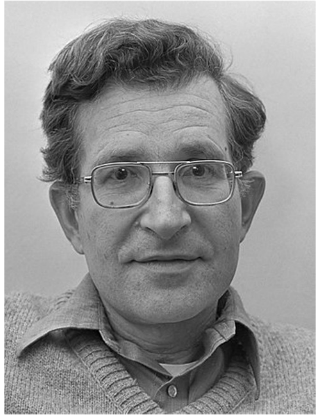
xy^*，实际上只需要两个state，可以在有限memory的DFA中完成；

而 $x^k y^k$，k可以想多大就多大，因而finite memory无法完成

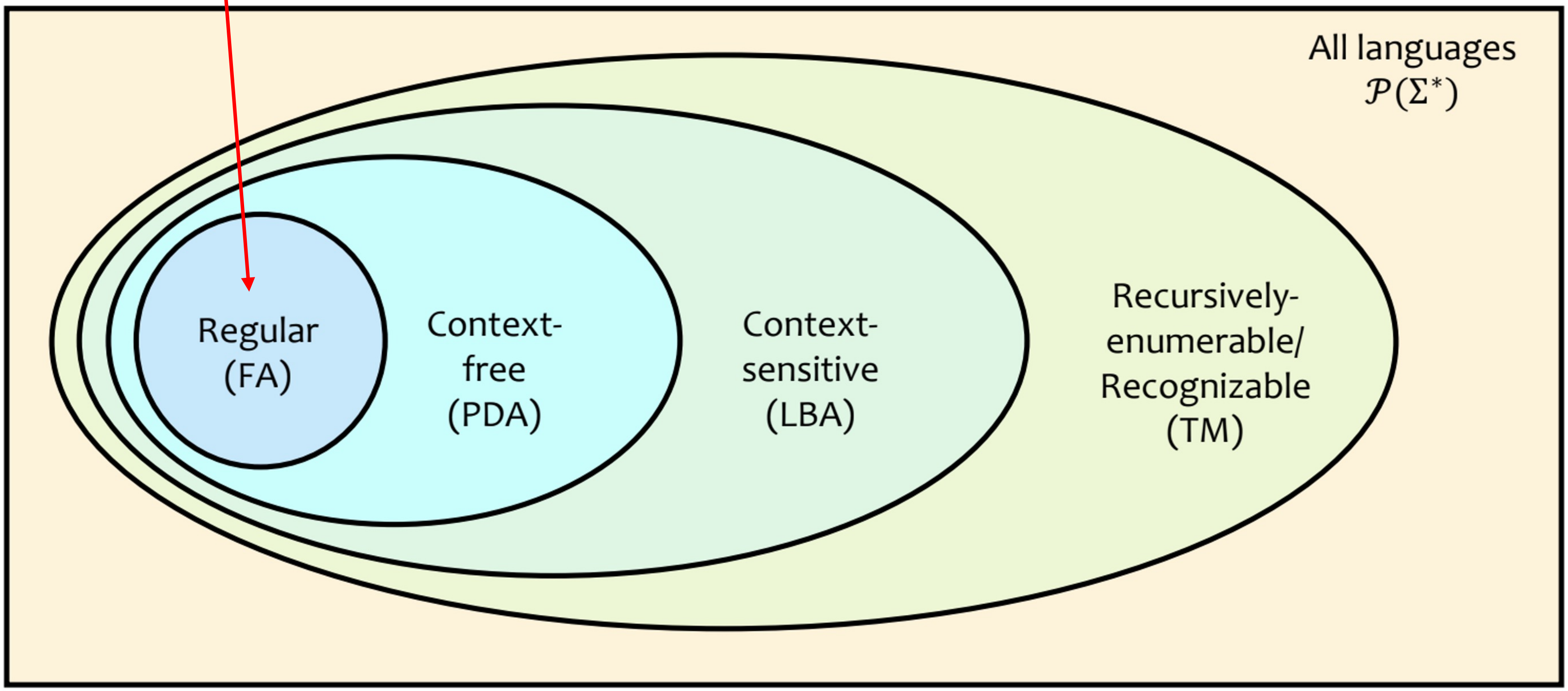
Which language can be decided by DFAs?

- There is an exact characterization!
- **Regular Expression**
  - A regular expression = finite expression using the string notations
    - Start from finite alphabet.
    - Compose them using **concantenation, alternation "|", or Kleen star "*"**
  - Examples:
    - $L(\epsilon) = \{\epsilon\}$
    - $L(\text{ab}*|\text{ba}*) = \{a, ab, abb, \dots\} \cup \{b, ba, baa, \dots\}$
- **Theorem** (RegExp = DFA):
  - L is defined by regular expression $\Leftrightarrow$ L is decided by a DFA
- These languages are called **regular languages.**

The Chomsky Hierarchy (1956)



“Regular Language”: Language decidable by some DFA



More powerful “memory system” →

Noam Chomsky

- **Noam Chomsky** is an American professor and public intellectual known for his work in linguistics, political activism, and social criticism.
- Sometimes called "the father of modern linguistics", Chomsky is also a major figure in analytic philosophy and one of the founders of the field of cognitive science.
- He is a laureate professor of linguistics at the University of Arizona and an institute professor emeritus at the Massachusetts Institute of Technology (MIT).

DFAs/regularity are widely used in practice

- Simple household devices (switches, garage doors, …)
- Software for designing and checking the behavior of digital circuits
- "Lexers," the first stage of compilers
- Flexible string search in text files (logs, latex, …)
  - **Warning:** "Regular exprs" in practice often ***aren't* regular!**
- and much more…

(Bonus) Exercises

Regular Expression Exercises

- All strings over $\{a, b\}$ with an ***even*** number of as.
  - $b^*(b^*ab^*ab^*)^*$
- All strings over $\{a, b\}$ without 2 consecutive as.
  - $(b^*ab)^*\big(b^*(a|\epsilon)\big)$
- All strings over $\{0,1\}$ that begin and end with the same symbol.
  - $(0(0|1)^*0)|(1(0|1)^*1)$
- $N = (0|1|2|\cdots|9)$ $\qquad L = (A|B|\cdots|Z)$
  - Dates: $NN - LLL - NN(NN|\epsilon)$ (E.g., 16-Feb-2023 or 16-Feb-23)
  - Michigan License Plates: $LLL\ NNNN$

DFA Exercises

- Design a DFA to decide $\{x \in \{0,1\}^* \mid (unsigned\ int)\ x\ is\ divisible\ by\ 5\}$
- Design a DFA to decide $\{x \in \{0,1\}^* \mid (unsigned\ int)\ x^R\ is\ divisible\ by\ 5\}$
  - x^R is the *reversal* of x, i.e., least significant digits comes first.

- You need to keep track of the score in a tennis game between A and B. The sequence of points scored is represented by a string $x \in \{A, B\}^*$.
  - The first player to get ≥4 points AND be ahead by 2 wins.
  - For weird historical reasons, 0 pts is “love”, 1 is “15”, 2 is “30”, 3 is “40”.

- Design a DFA to decide $\left\{\begin{matrix} x \in \{A, B\}^* \mid A \text{ has already won after seeing} \\ x \text{ points recorded.} \end{matrix}\right\}$

DFA Impossibility Exercises

- Prove $\mathrm{L}_{\mathrm{Palindrome}} = \{x \in \{0,1\}^* \mid x \text{ is a palindrome}\}$ cannot be decided by a DFA.
  - Let s = #states
  - Hint: consider $0^{s+1}110^{s+1}$

- Prove $\mathrm{L}_{\mathrm{Prime}} = \{x \in \{0,1\}^* \mid\ x = 1^p \text{ and } p \text{ is prime}\}$ cannot be decided by a DFA.