

BU CS 332 – Theory of Computation

<https://forms.gle/5V2QAntsehE78St19>



Lecture 3:

- Deterministic Finite Automata
- Non-deterministic FAs

Reading:

Sipser Ch 1.1-1.2

Alexander Poremba & Mark Bun

January 27, 2025

Last Time

- Strings: Finite concatenations of symbols
- Languages: Sets L of strings
- Computational (decision) problem: Given a string x , is it in the language L ?

Languages

Languages = computational (decision) problems

Input: String $x \in \Sigma^*$

Output: Is $x \in L$? (Yes or No? I.e., Accept or Reject?)

The language **recognized** by a program is the set of strings $x \in \Sigma^*$ that it *accepts*

Language recognized by program P is:

$$L(P) = \{ x \in \Sigma^* \mid \begin{array}{l} P \text{ accepts } x \\ \text{i.e. } P \text{ answers "yes" with run on } x \end{array} \}$$

What Language Does This Program Recognize?

Alphabet $\Sigma = \{a, b\}$

On input $x = x_1x_2 \dots x_n$:

count = 0

For $i = 1, \dots, n$:

If $x_i = a$:

count = count + 1

If count ≤ 4 : **accept**

Else: **reject**

- a) $\{x \in \Sigma^* \mid |x| > 4\}$
- b) $\{x \in \Sigma^* \mid |x| \leq 4\}$
- c) $\{x \in \Sigma^* \mid |x| = 4\}$
- d) $\{x \in \Sigma^* \mid x \text{ has more than 4 a's}\}$
- e)** $\{x \in \Sigma^* \mid x \text{ has at most 4 a's}\}$
- f) $\{x \in \Sigma^* \mid x \text{ has exactly 4 a's}\}$

Deterministic Finite Automata

A (Real-Life?) Example



- **Example:** Kitchen scale
- P = Power button (ON / OFF)
- U = Units button (cycles through g / oz / lb)
 - Only works when scale is ON, but units remembered when scale is OFF
- Starts OFF in g mode
- **A computational problem:** Does a sequence of button presses (describable as a string in $\{P, U\}^*$) leave the scale ON in oz mode?

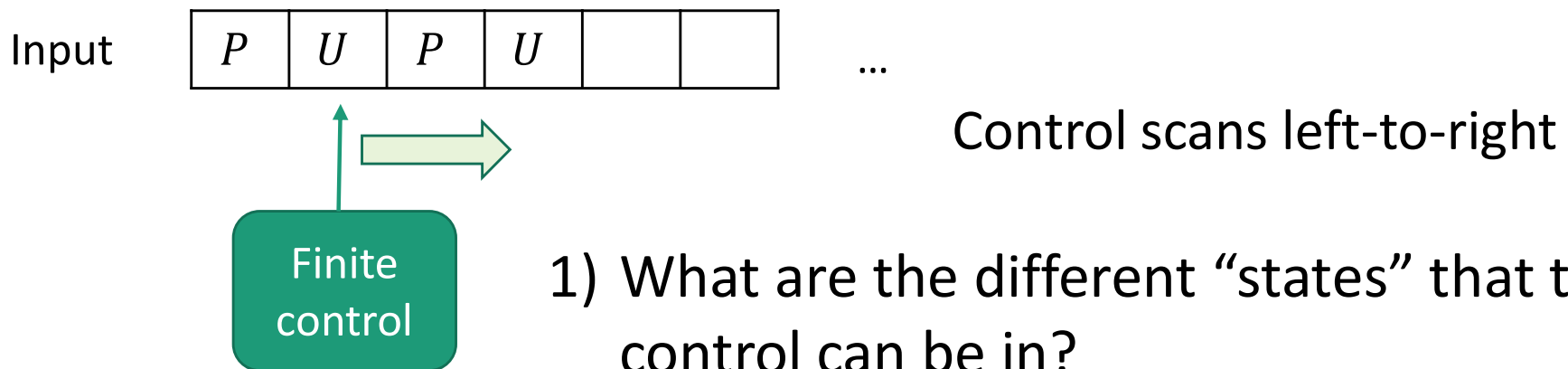
For alphabet Σ

$\Sigma^* = \{ \text{all strings comprised of characters from } \Sigma \}$

$\{P, U\}^* = \{ \epsilon, P, U, PP, PU, \dots \}$

Machine Models

- Finite Automata (FAs): Machine with a finite amount of unstructured memory



- 1) What are the different “states” that the control can be in?
- 2) In what state does the control start?
- 3) When the control reads a new input character, how does it transition to a new state?
- 4) How do I know if I’m in the desired state at the end?

A DFA for the Kitchen Scale Problem

P = Power button (ON / OFF)

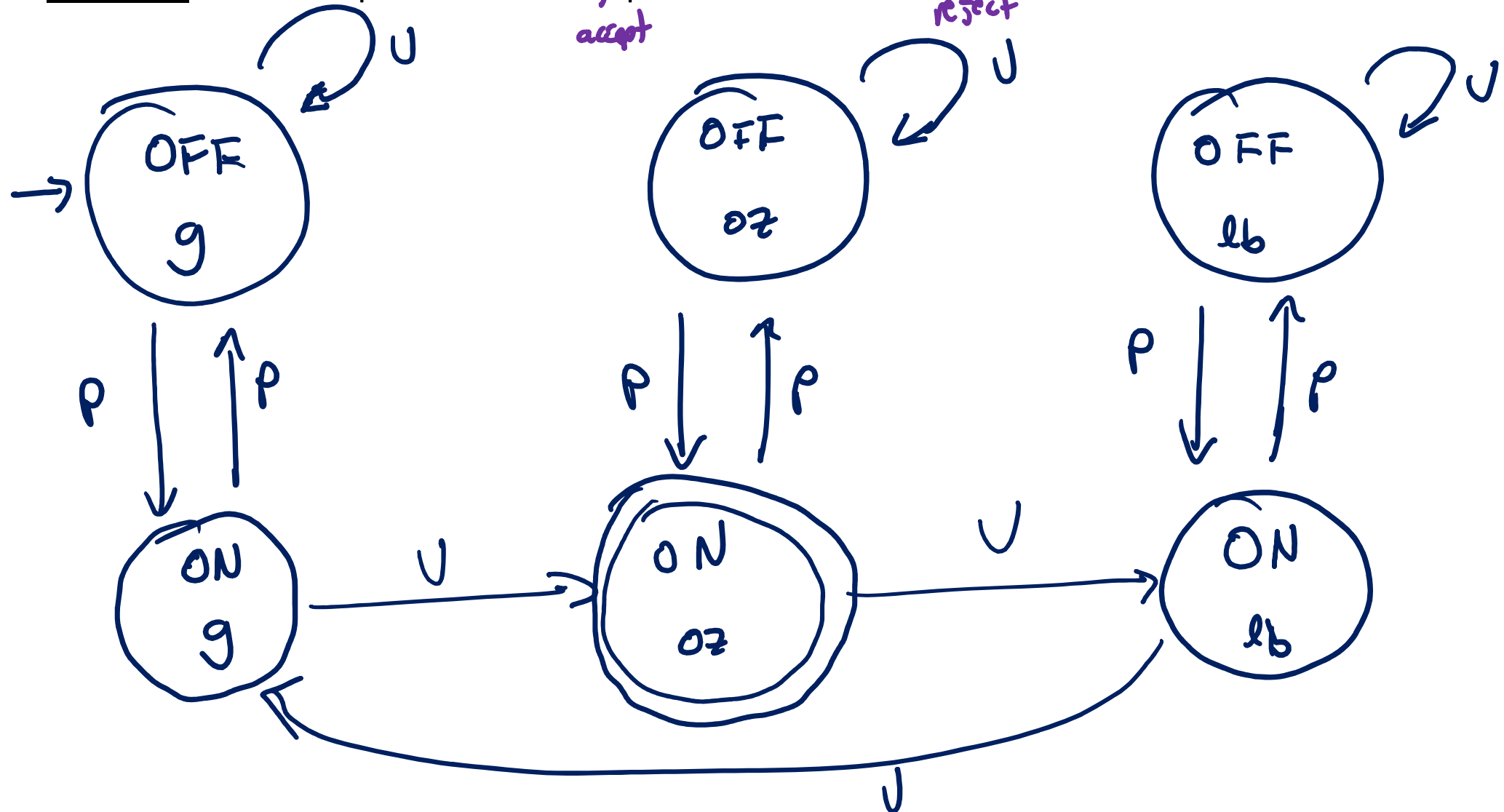
U = Units button (cycles through g / oz / lb)

Starts OFF in g mode

Problem: Does a sequence of button presses leave the scale ON in oz mode?

PPUUPU
accept

PUPP
reject

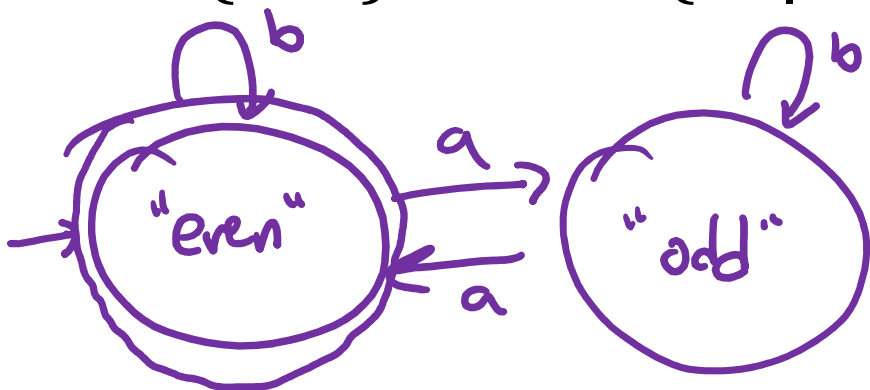


A DFA Recognizing Parity

The **language** recognized by a DFA is the set of inputs on which it ends in an “accept” state

Parity: Given a string consisting of a 's and b 's, does it contain an even number of a 's?

$\Sigma = \{a, b\}$ $L = \{w \mid w \text{ contains an even number of } a\text{'s}\}$



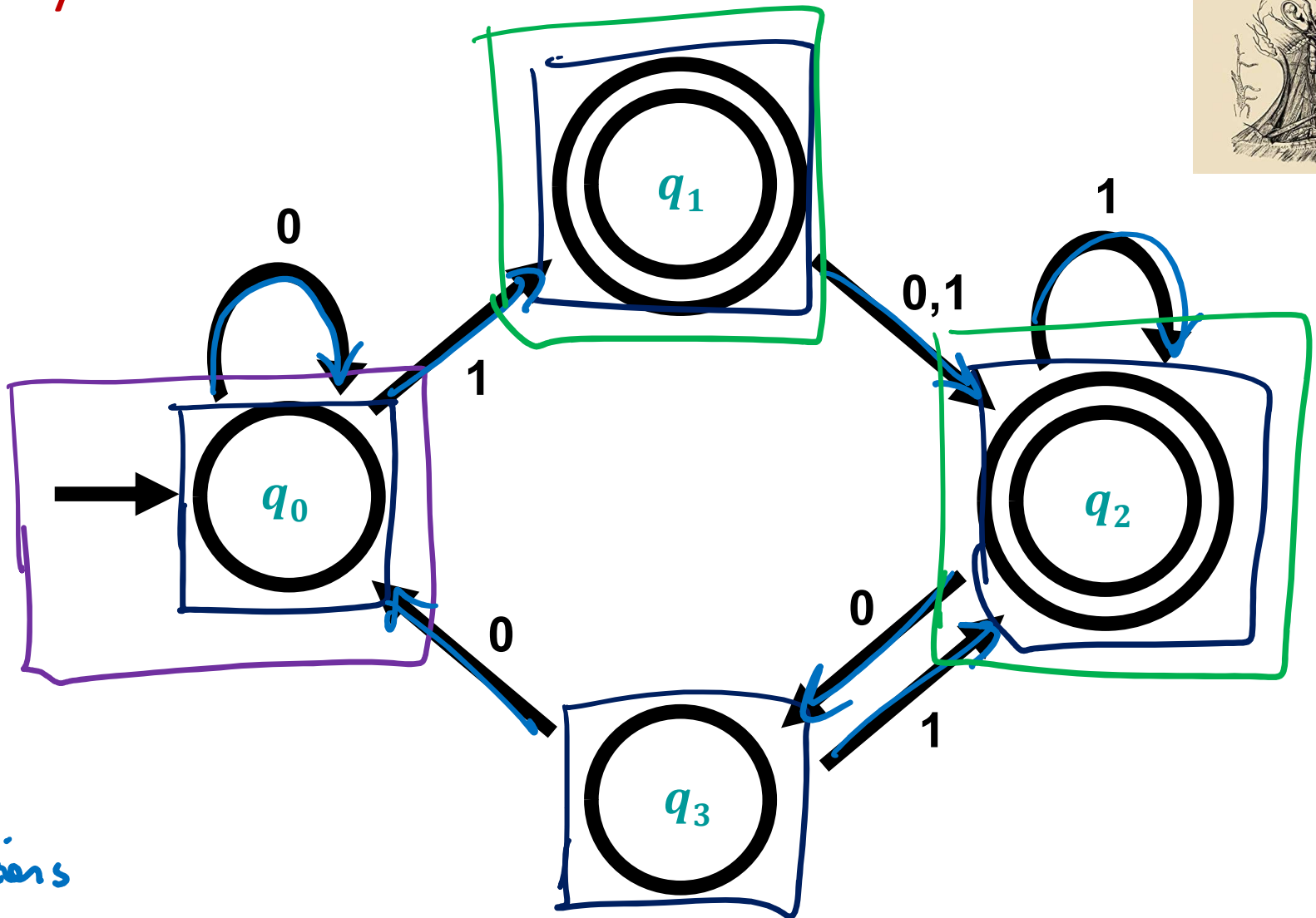
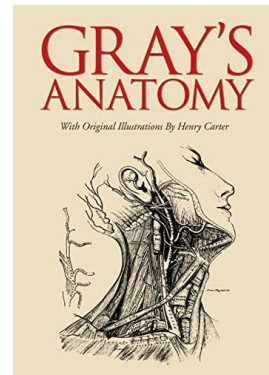
"even" \xrightarrow{a} "odd" \xrightarrow{a} "even" \xrightarrow{b} "even"
 \xrightarrow{a} "odd" \xrightarrow{b} "odd"



Which state is reached by the parity DFA on input aabab?

- a) "even"
- b) "odd"

Anatomy of a DFA



☐ = states

→ = start state

→ = transitions

⊖ = accept states or "final" states

Some Tips for Thinking about DFAs

Given a DFA, what language does it recognize?

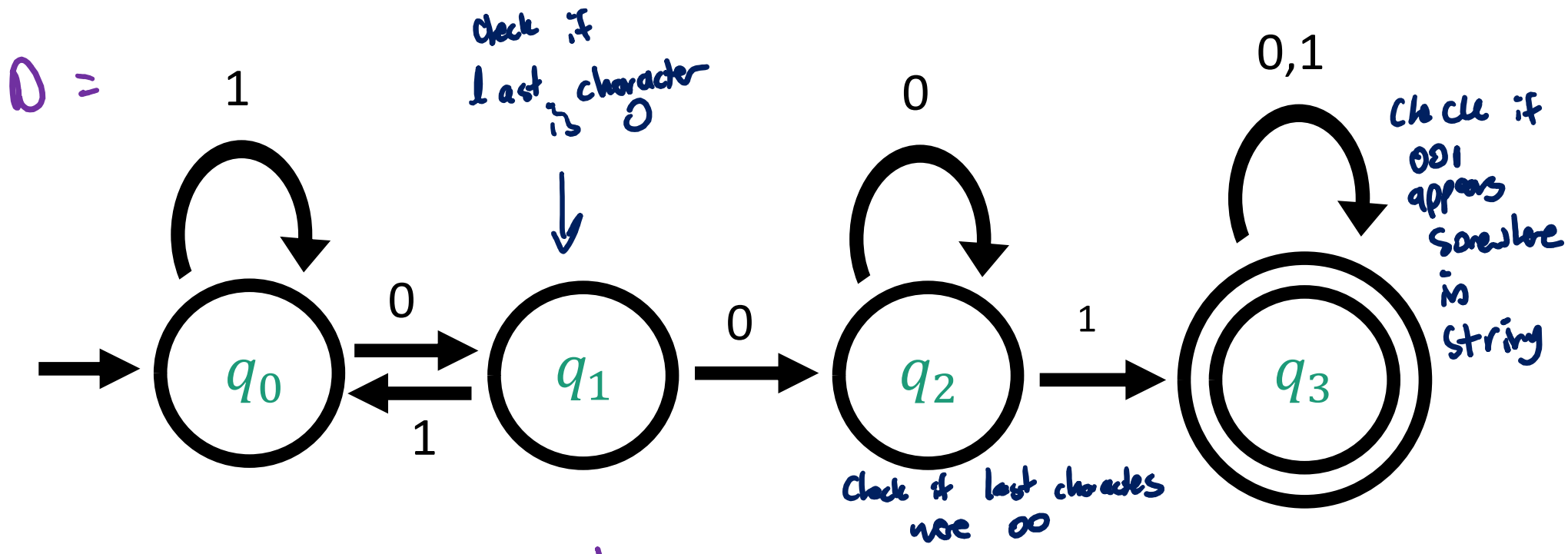
- Try experimenting with it on short strings. Do you notice any patterns?
- What kinds of inputs cause the DFA to get trapped in a state?



Given a language, construct a DFA recognizing it

- Imagine you are a machine, reading one symbol at a time, always prepared with an answer
- What is the essential information that you need to remember? Determines set of states.

What language does this DFA recognize?



$$L(D) = \{ x \in \{0,1\}^* \mid x \text{ contains } 001 \text{ as a substring} \}$$

Practice!

- Lots of worked out examples in Sipser
- Automata Tutor: <https://automata-tutor.live-lab.fi.muni.cz/>

Formal Definition of a DFA

A **finite automaton** is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$

Q is the set of states

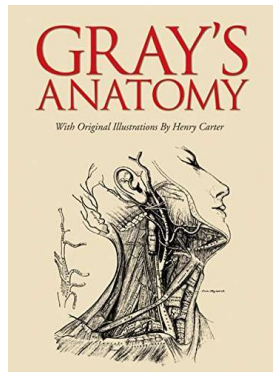
Σ is the alphabet

$\delta: Q \times \Sigma \rightarrow Q$ is the transition function

Handwritten annotations above the transition function:
- "current state" with an arrow pointing to the first Q
- "next symbol" with an arrow pointing to Σ
- "next state" with an arrow pointing to the second Q

$q_0 \in Q$ is the start state

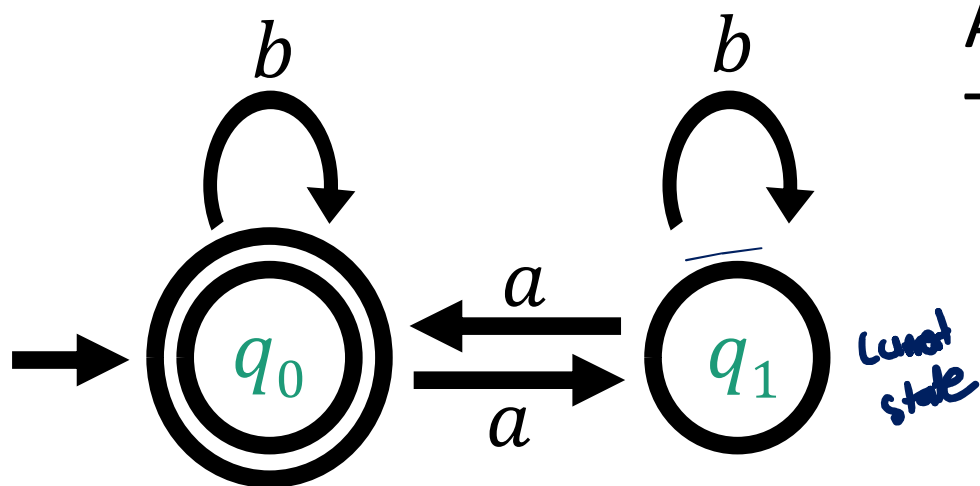
$F \subseteq Q$ is the set of accept states



A DFA for Parity

Parity: Given a string consisting of a 's and b 's, does it contain an even number of a 's?

$\Sigma = \{a, b\}$ $L = \{w \mid w \text{ contains an even number of } a\text{'s}\}$



State set $Q = \{q_0, q_1\}$

Alphabet $\Sigma = \{a, b\}$

Transition function δ

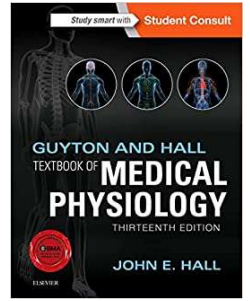
δ	a	b
q_0	q_1	q_0
q_1	q_0	q_1

next symbol

$\delta(q_0, a) = q_1$
 $\delta(q_0, b) = q_0$
 $\delta(q_1, a) = q_0$
 $\delta(q_1, b) = q_1$

Start state q_0

Set of accept states $F = \{q_0\}$



Formal Definition of DFA Computation

A DFA $M = (Q, \Sigma, \delta, q_0, F)$ **accepts** a string $w = w_1w_2 \cdots w_n \in \Sigma^*$ (where each $w_i \in \Sigma$) if there exist $r_0, \dots, r_n \in Q$ such that

1. $r_0 = q_0$



2. $\delta(r_i, w_{i+1}) = r_{i+1}$ for each $i = 0, \dots, n - 1$, and



3. $r_n \in F$

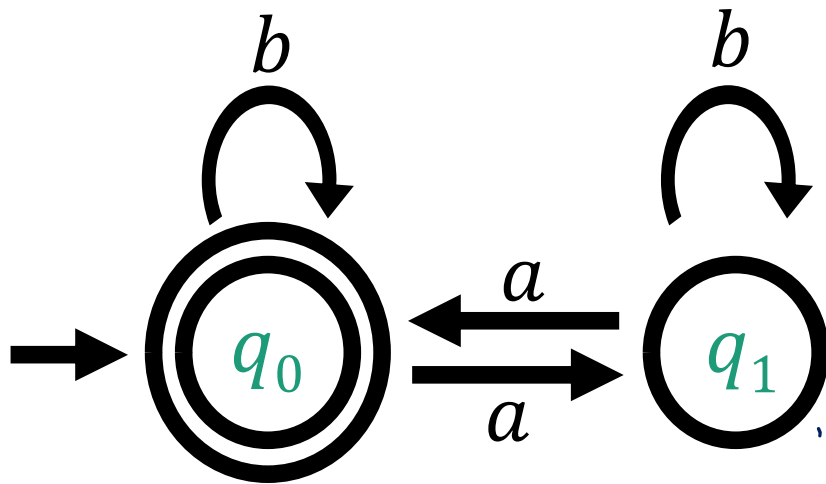


$L(M)$ = the **language** of machine M

= set of all strings machine M accepts

M **recognizes** the language $L(M)$

Example: Computing with the Parity DFA



Let $w = \overset{w_1}{a}\overset{w_2}{b}\overset{w_3}{b}\overset{w_4}{a}$
 Does M accept w ?

What is $\delta(r_2, w_3)$?

- a) q_0
- b) q_1



A DFA $M = (Q, \Sigma, \delta, q_0, F)$ **accepts** a string $w = w_1 w_2 \cdots w_n \in \Sigma^*$ (where each $w_i \in \Sigma$) if there exist $r_0, \dots, r_n \in Q$ such that

1. $r_0 = q_0$
2. $\delta(r_i, w_{i+1}) = r_{i+1}$
for each $i = 0, \dots, n - 1$
3. $r_n \in F$

$$\begin{aligned}
 r_0 &= q_0 \\
 r_1 &= \delta(r_0, w_1) = \delta(q_0, a) = q_1 \\
 r_2 &= \delta(r_1, w_2) = \delta(q_1, b) = q_1 \\
 r_3 &= \delta(r_2, w_3) = \delta(q_1, b) = q_1 \\
 r_4 &= \delta(r_3, w_4) = \delta(q_1, a) = q_0 \\
 r_4 &= q_0 \in F \quad \checkmark
 \end{aligned}$$

Regular Languages

Definition: A language is **regular** if it is recognized by a DFA

$L = \{ w \in \{a, b\}^* \mid w \text{ has an even number of } a\text{'s} \}$ is regular

$L = \{ w \in \{0, 1\}^* \mid w \text{ contains } 001 \}$ is regular

Many interesting problems are captured by regular languages

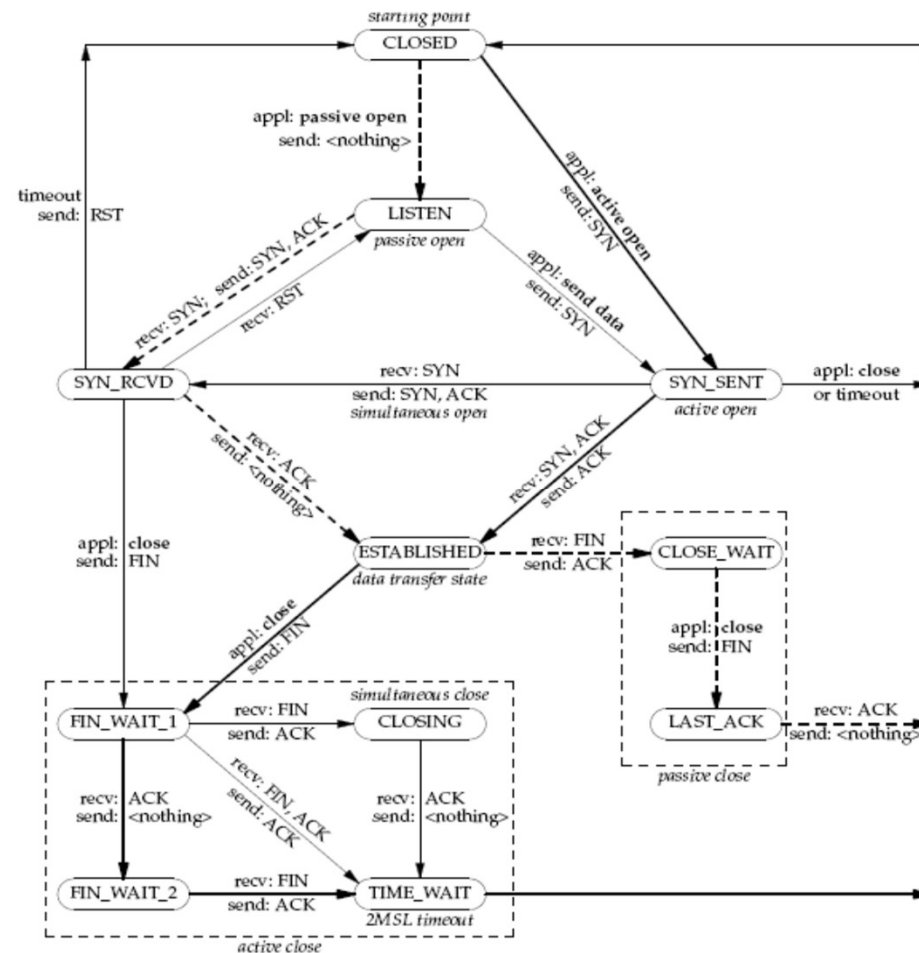
Network Protocols

Compilers

Genetic Testing

Arithmetic

Internet Transmission Control Protocol



Let $TCPS = \{ w \mid w \text{ is a complete TCP Session} \}$

Theorem: TCPS is regular

Compilers

Comments :

/ this is a comment */*

Are delimited by `/* */`

Cannot have nested `/* */`

Must be closed by `*/`

`*/` is illegal outside a comment

COMMENTS = {strings over $\{0,1, /, *\}$ with legal comments}

Theorem: COMMENTS is regular

Genetic Testing

DNA sequences are strings over the alphabet $\{A, C, G, T\}$.

TAGACAT

A gene g is a special substring over this alphabet.

$g = AGA$

A genetic test searches a DNA sequence for a gene.

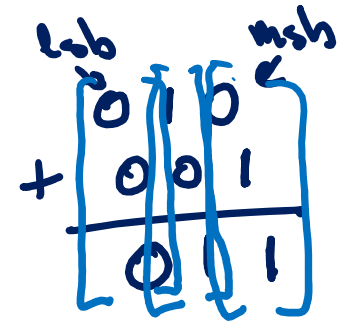
Is AGA a substring of TAGACAT?

$\text{GENETICTEST}_g = \{\text{strings over } \{A, C, G, T\} \text{ containing } g \text{ as a substring}\}$

Theorem: GENETICTEST_g is regular for every gene g .

Arithmetic

$$\begin{array}{r} 2 \\ + 4 \\ \hline 6 \end{array}$$



$$\text{LET } \Sigma = \left\{ \begin{array}{l} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \\ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \end{array} \right\}$$

- A string over Σ has three ROWS ($\text{ROW}_1, \text{ROW}_2, \text{ROW}_3$)
- Each ROW $b_0b_1b_2 \dots b_N$ represents the integer

$$b_0 + 2b_1 + \dots + 2^N b_N.$$
- Let $\text{ADD} = \{S \in \Sigma^* \mid \text{ROW}_1 + \text{ROW}_2 = \text{ROW}_3\}$

Theorem. ADD is regular.

Nondeterministic Finite Automata

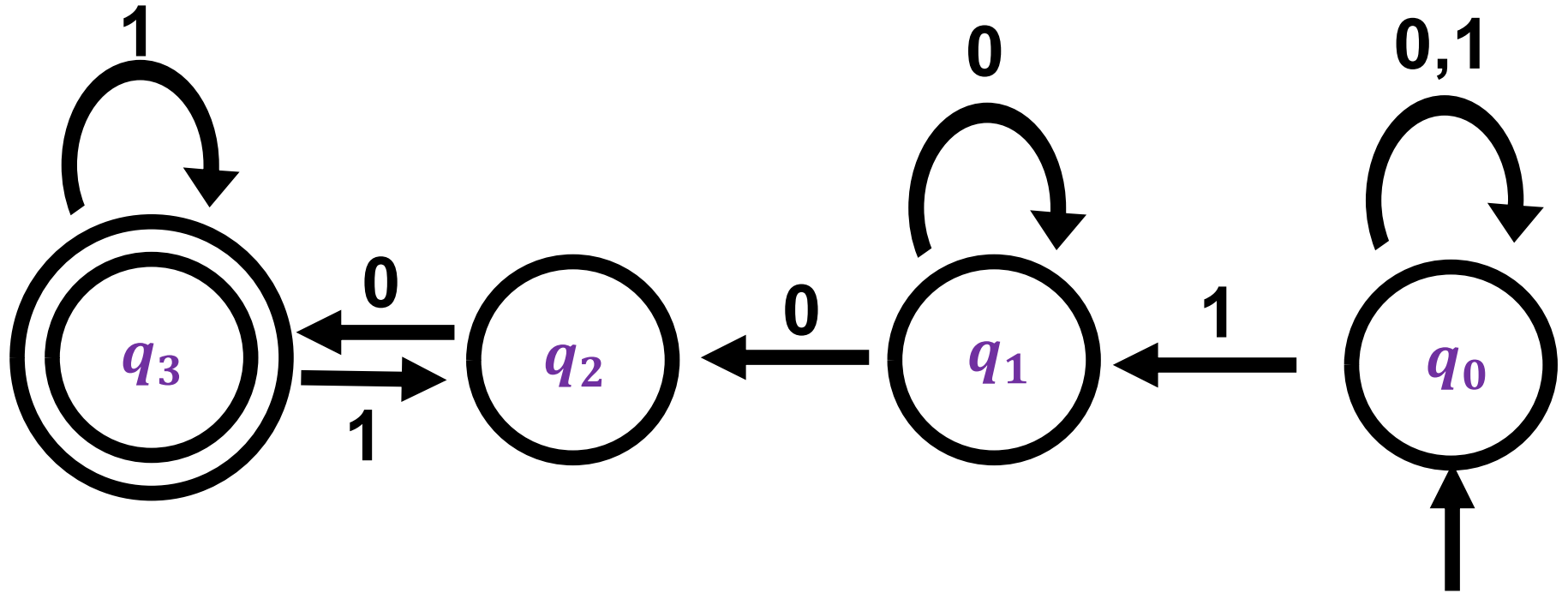
Nondeterminism

In a DFA, the machine is always in exactly one state upon reading each input symbol

In a **nondeterministic** FA, the machine can “try out” many different ways of reading the same string

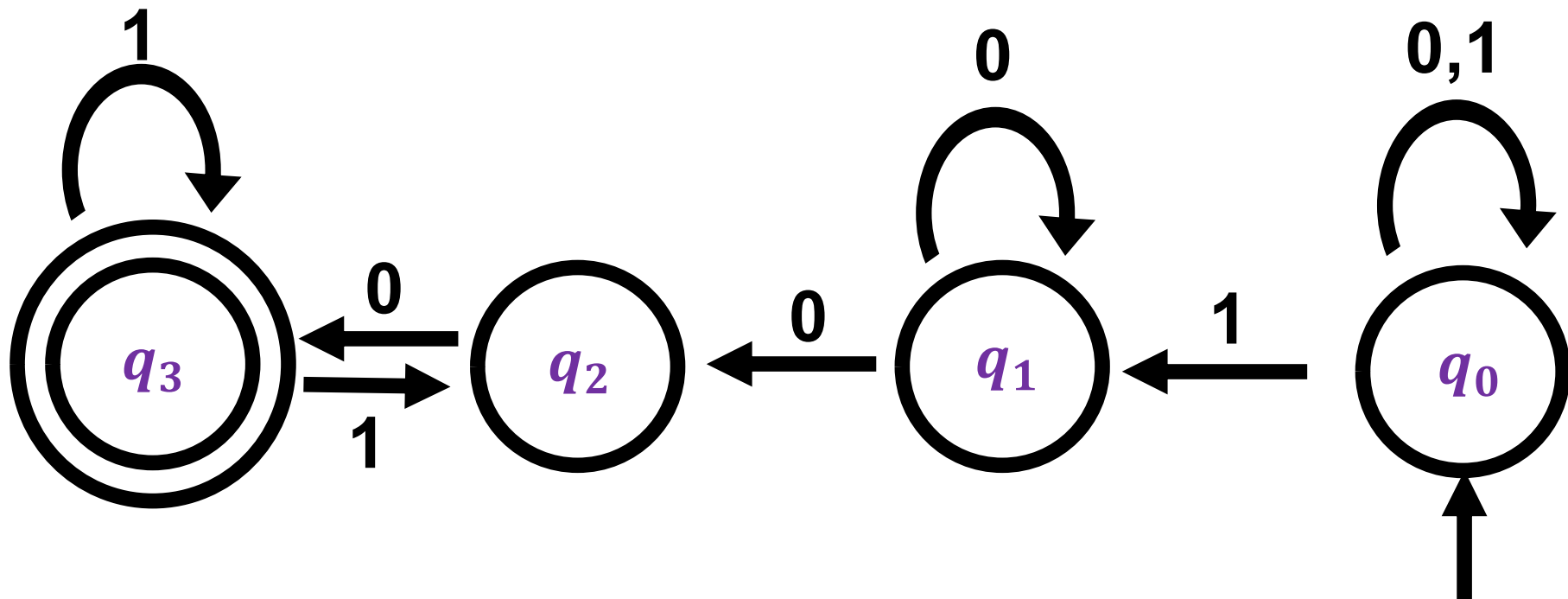
- Next symbol may cause an NFA to “branch” into multiple possible computations
- Next symbol may cause NFA’s computation to fail to enter any state at all

Nondeterminism



A **Nondeterministic Finite Automaton** (NFA) accepts if there **exists** a way to make it reach an accept state.

Nondeterminism



Example: Does this NFA accept the string 1100?

YES because
there exists
a computation path
leading to acceptance

$q_0 \xrightarrow{1} q_1 \xrightarrow{1} \perp$ "branch of computation fails"

$q_0 \xrightarrow{1} q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_2 \xrightarrow{0} q_3$ ✓

$q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_0$ ✗

$q_0 \xrightarrow{1} q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_1 \xrightarrow{0} q_1$ ✗