BU CS 332 – Theory of Computation

https://forms.gle/SspWTA9xLbagTzpH6

Lecture 3:

- Deterministic Finite Automata
- Non-deterministic FAs

Reading:

Sipser Ch 1.1-1.2

Mark Bun

January 29, 2025

Last Time

- Parts of a theory of computation: Model for machines, model for problems, theorems relating machines and problems

- 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

<u>Input:</u> String $x \in \Sigma^*$

<u>Output:</u> 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*

Given a program P making accept / reject decisions

$L = \{ x \in \Sigma^* \mid P \text{ accepts } x \}$

is the "language recognized by P"

What Language Does This Program Recognize?

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

On input $x = x_1 x_2 \ldots x_n$:

count = 0

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

    If $x_i = \mathrm{a}$:

      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

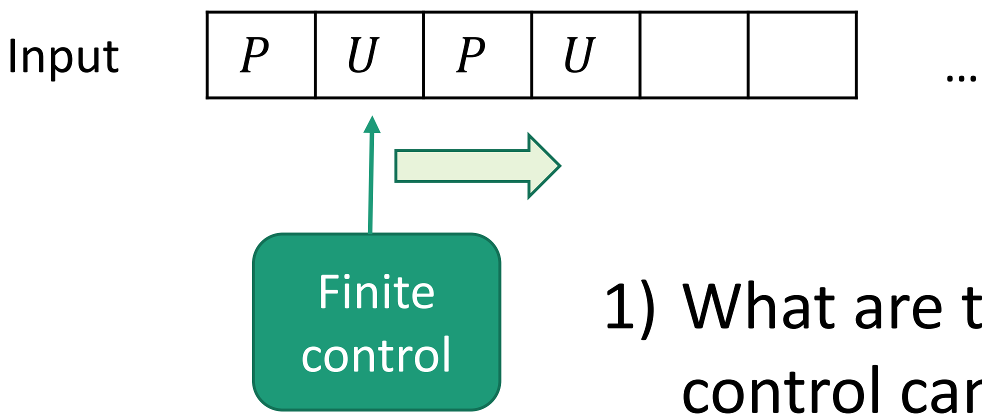
- Example: Kitchen scale
- P = Power button (ON / OFF)
- U = Units button (cycles through g / oz / lb)

  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?

Machine Models

- <u>Finite Automata (FAs):</u> Machine with a finite amount of unstructured memory



Control scans left-to-right

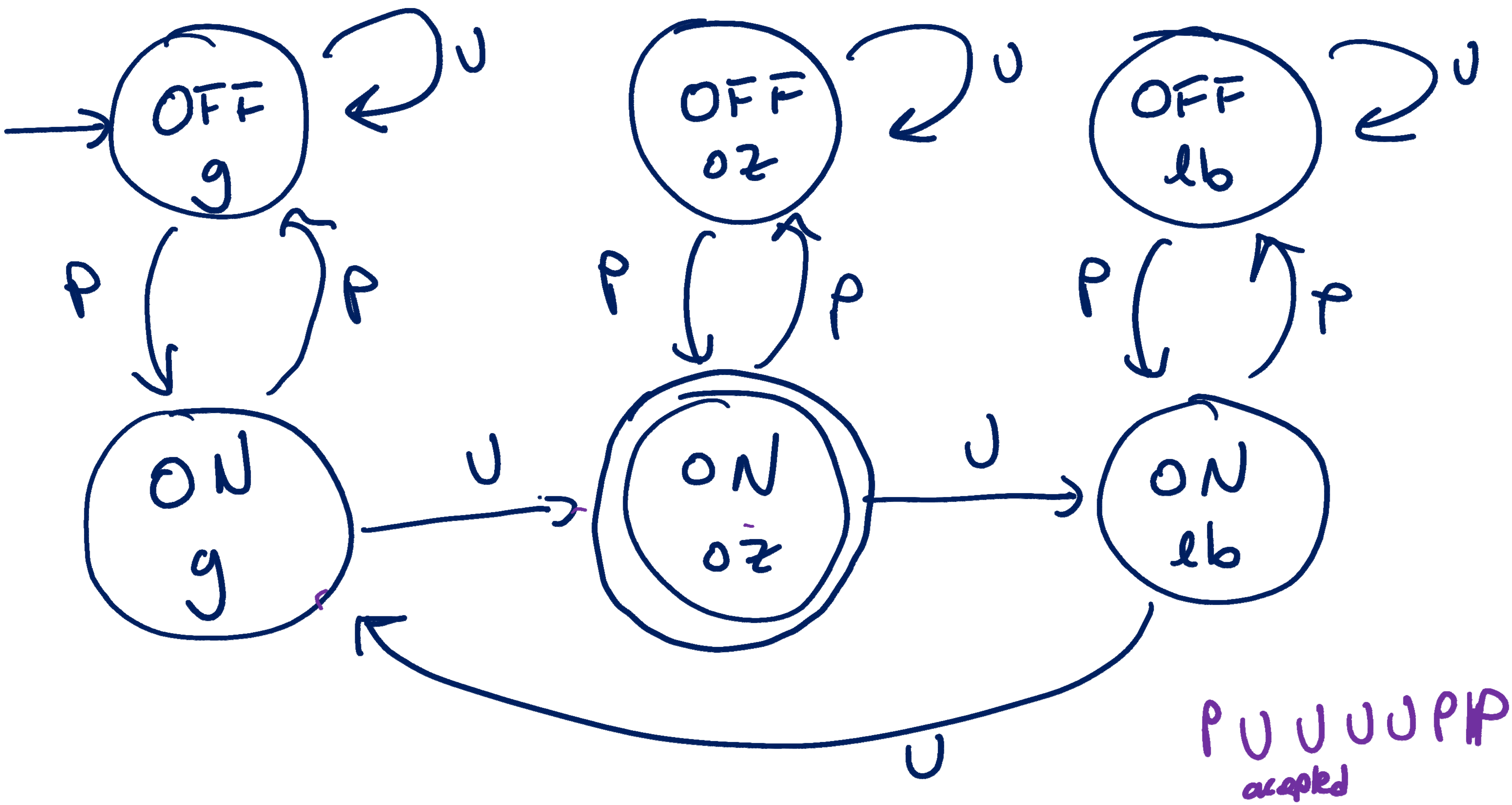
1) What are the different "states" that the control can be in?
2) In what state does the control start?
3) When the control reads a new input character, how does it transition to a new state?
4) How do I know if I'm in the desired state at the end?

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

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

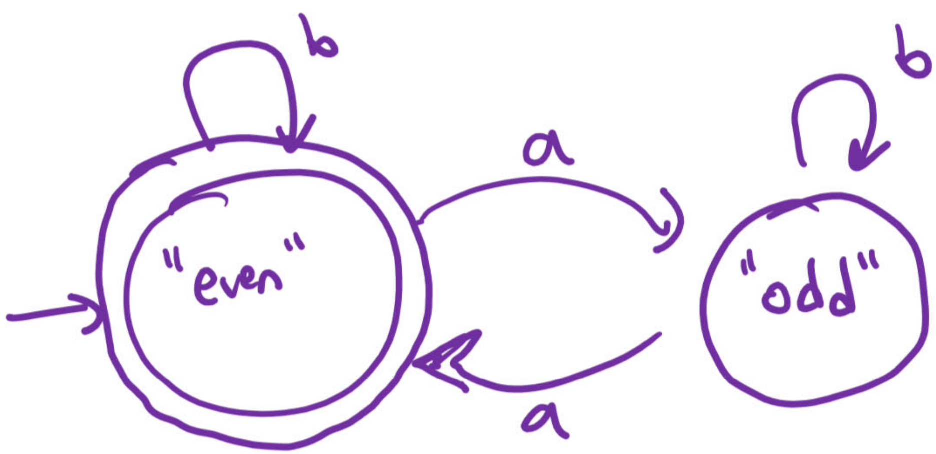


A DFA Recognizing Parity

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?

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

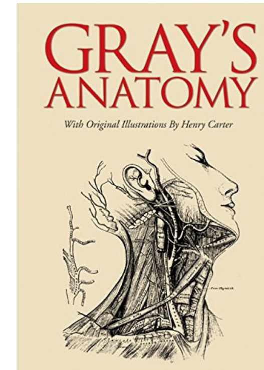


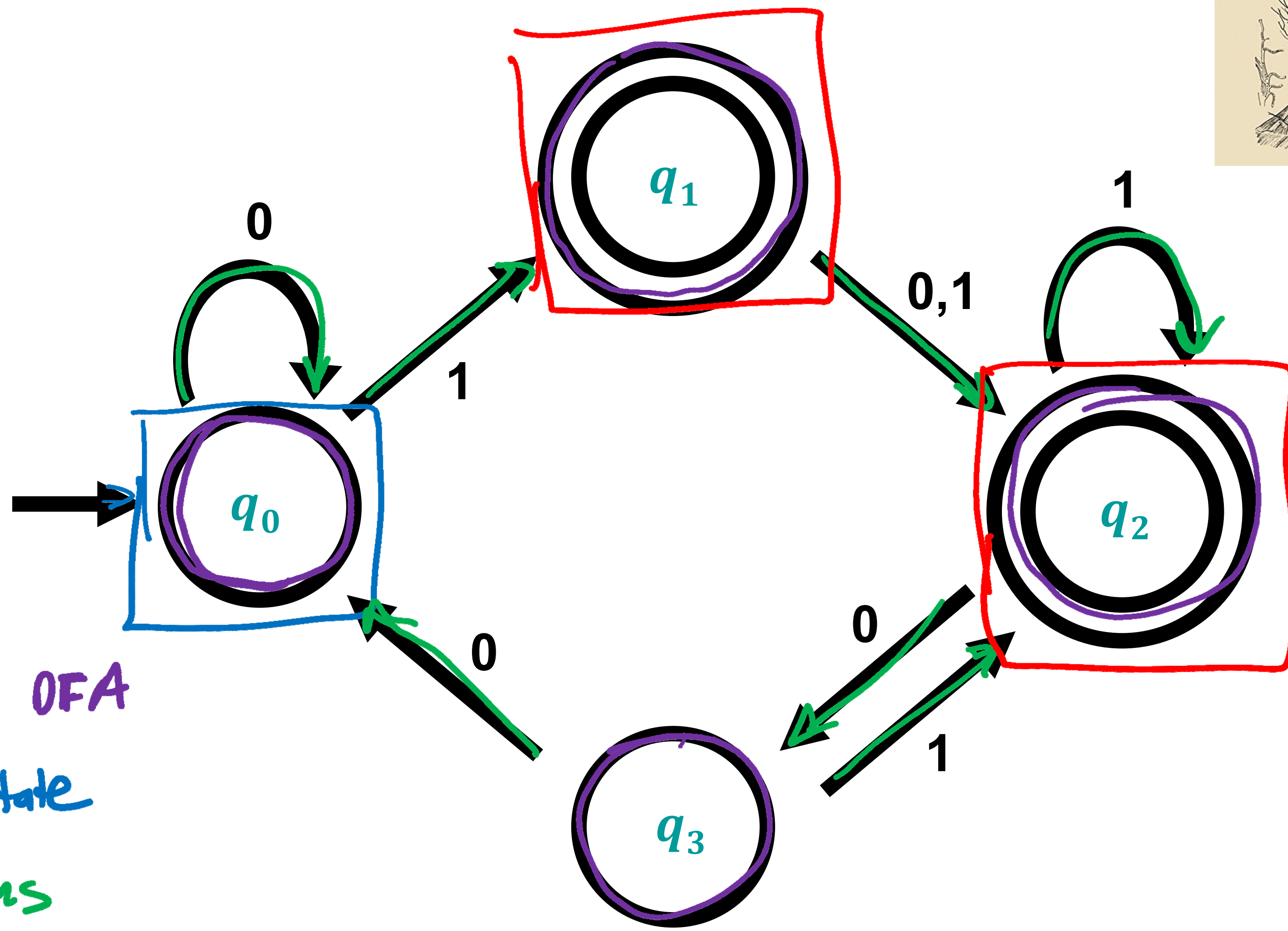
Which state is reached by the parity DFA on input aabab?

a) ~~"even"~~ "odd"

b) ~~"odd"~~ "even"

Anatomy of a DFA





states of DFA

start state

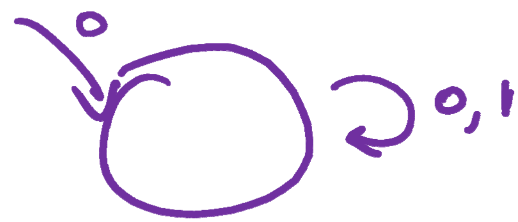
transitions

accept (or "final") states

Some Tips for Thinking about DFAs

Given a DFA, what language does it recognize?

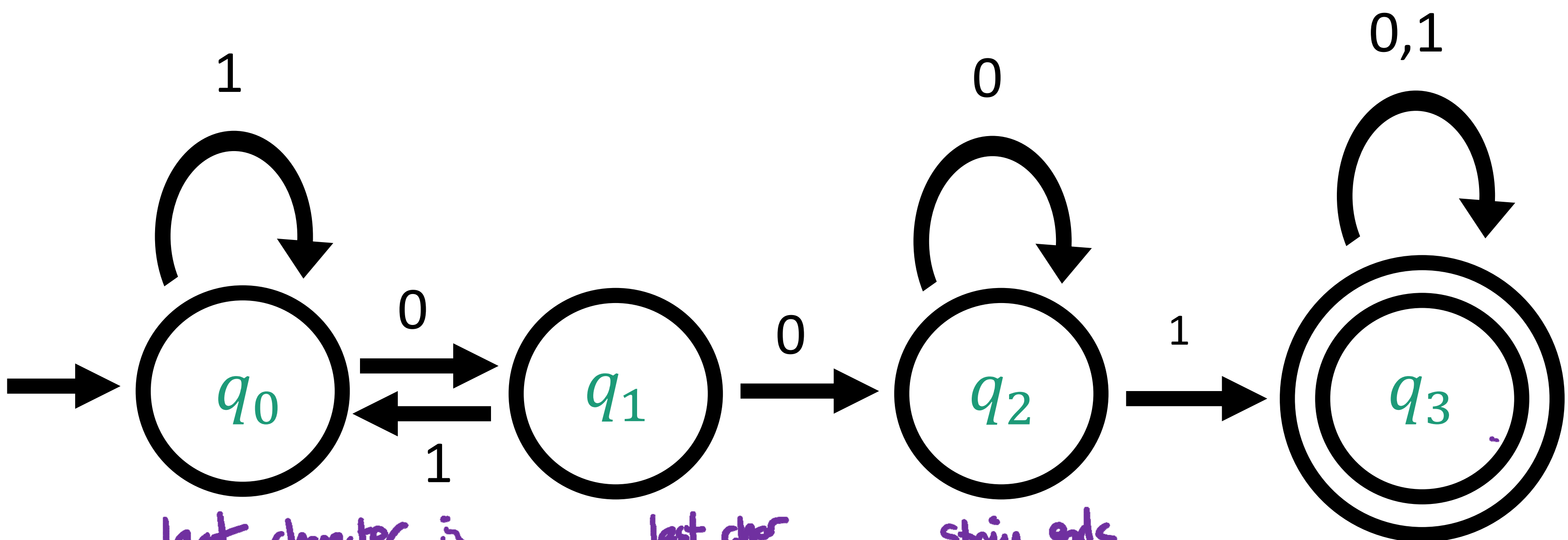
- Try experimenting with it on short strings. Do you notice any patterns?
- What kinds of inputs cause the DFA to get trapped in a state?



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?



last character is a 1 (roughly)

last char is a 0

string ends in at least 2 0's

string contains 001

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

x contains a substring of the form z1 where z is a string of ≥ 2 0's

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

$f(x) = e^x$

$f: \mathbb{R} \to \mathbb{R}_{\geq 0}$

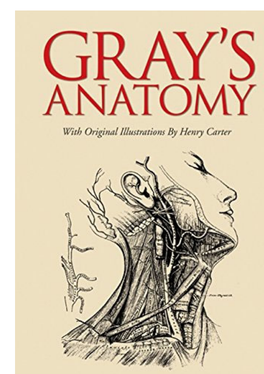
Σ is the alphabet

current state — next symbol — next state

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

$q_0 \in Q$ is the start state

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

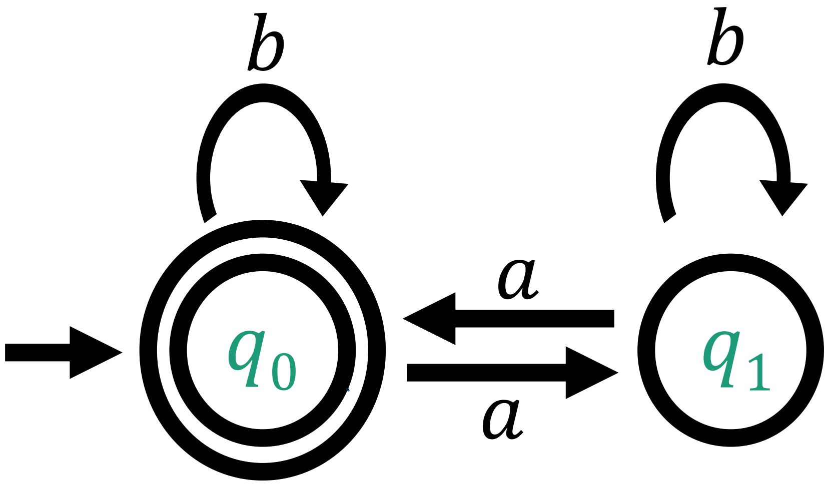


$\delta : Q \times \Sigma \to Q$

A DFA for Parity

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

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



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

Alphabet Σ = $\{a, b\}$

Transition function δ

next symbol

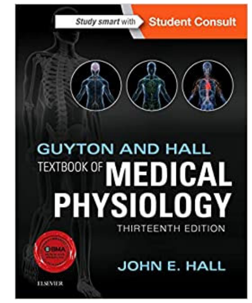
δ	a	b
q_0	$\delta(q_0, a) = q_1$	q_0
q_1	q_0	q_1

Current state

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 $w = w_1 w_2 \cdots w_n \in \Sigma^*$ (where each $w_i \in \Sigma$) if there exist $r_0, \ldots, r_n \in Q$ such that

1. $r_0 = q_0$ starting in start state $\rightarrow q_0$
2. $\delta(r_i, w_{i+1}) = r_{i+1}$ for each $i = 0, \ldots, n-1$, and

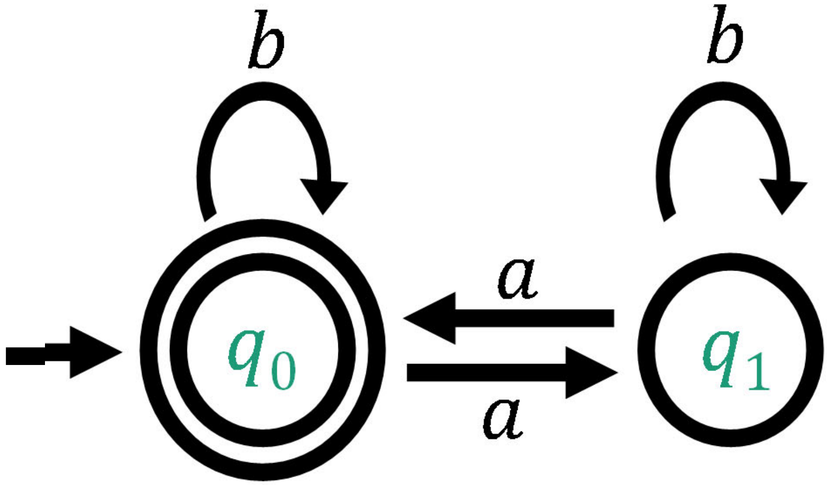
$$r_i \xrightarrow{w_{i+1}} r_{i+1}$$

3. $r_n \in F$ r_n

$L(M)$ = the **language** of machine M
= set of all strings machine M accepts

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

Example: Computing with the Parity DFA



$w_1\ w_2\ w_3\ w_4$

Let $w = abba$

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 $r_0, \ldots, r_n \in Q$ such that

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

$r_0 = q_0$

$r_1 = \delta(r_0, w_1) = \delta(q_0, a) = q_1$

$r_2 = \delta(r_1, w_2) = \delta(q_1, b) = q_1$

$r_3 = \delta(r_2, w_3) = \delta(q_1, b) = q_1$

$r_4 = \delta(r_3, w_4) = \delta(q_1, a) = q_0$

$r_4 \in F$ ✓

Regular Languages

<u>Definition:</u> A language is regular if it is recognized by a DFA

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

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

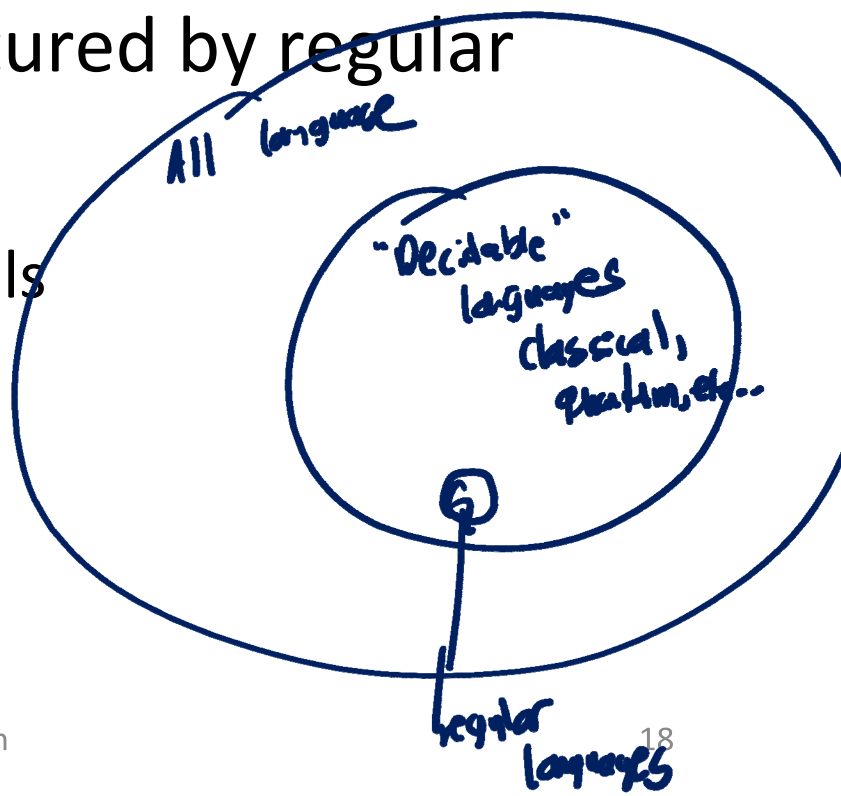
Many interesting problems are captured by regular languages

Network Protocols

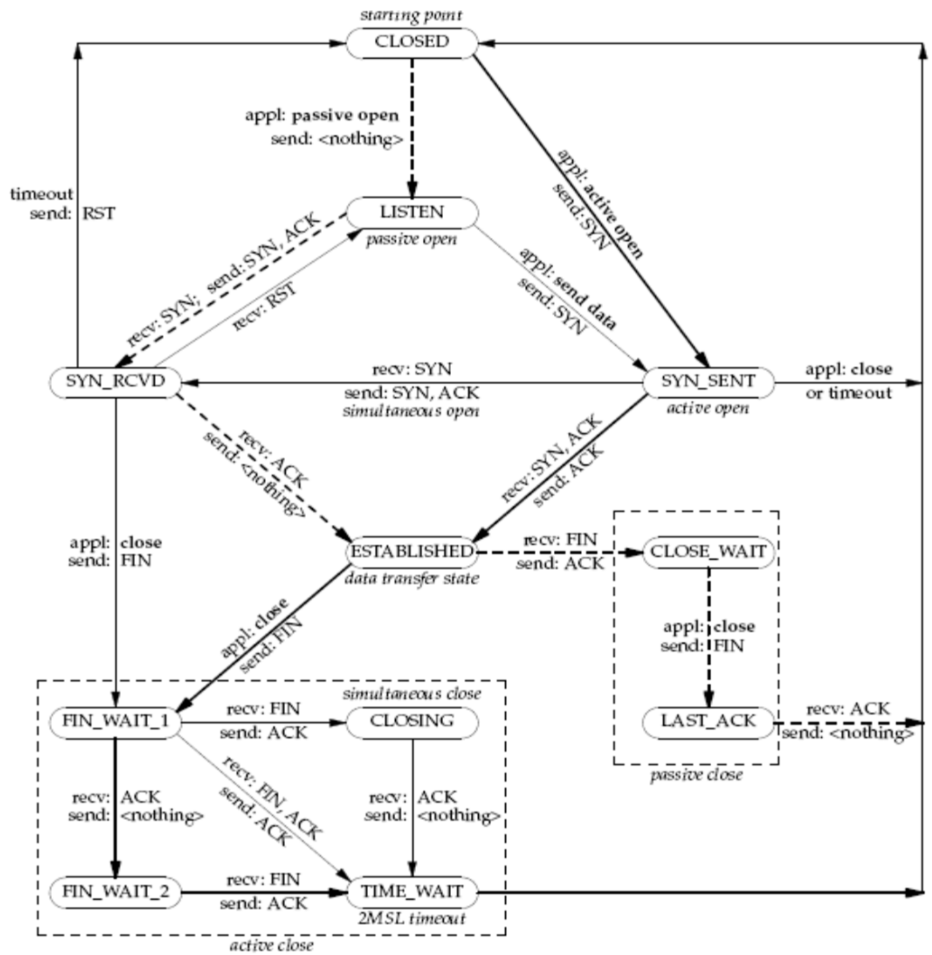
Compilers

Genetic Testing

Arithmetic



Internet Transmission Control Protocol



Let TCPS = { w | w is a complete TCP Session}

Theorem: TCPS is regular

Compilers

Comments :

Are delimited by /* */

Cannot have nested /* */

Must be closed by */

*/ is illegal outside a comment

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

Theorem: COMMENTS is regular

Genetic Testing

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

TAGACAT

A gene g is a special substring over this alphabet.

CAT

A genetic test searches a DNA sequence for a gene.

Does CAT appear as a substring of TAGACAT?

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

Theorem: GENETICTEST$_g$ is regular for every gene g.

Arithmetic

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

- A string over Σ has three ROWS (ROW$_1$, ROW$_2$, ROW$_3$)
- Each ROW $b_0 b_1 b_2 \ldots b_N$ represents the integer

$$b_0 + 2b_1 + \ldots + 2^N b_N.$$

- Let ADD = $\{S \in \Sigma^* \mid \text{ROW}_1 + \text{ROW}_2 = \text{ROW}_3\}$

Theorem. ADD is regular.

Nondeterministic Finite Automata

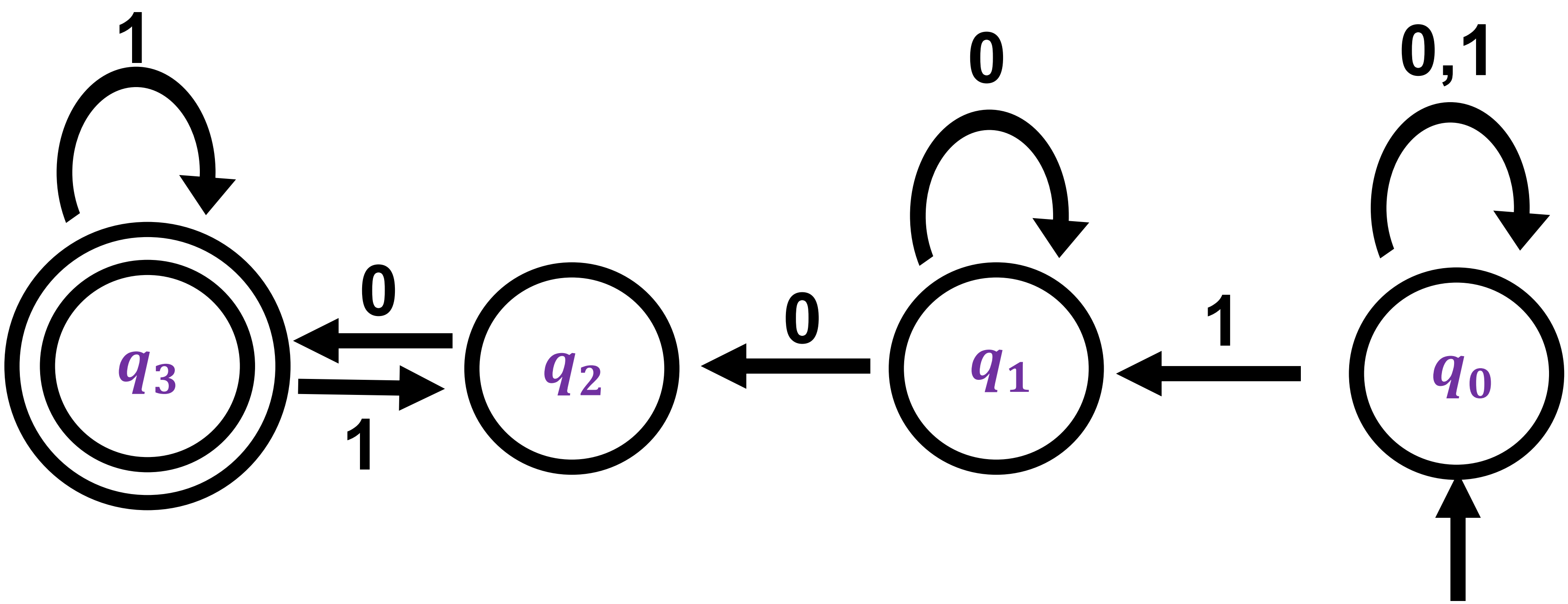
Nondeterminism

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

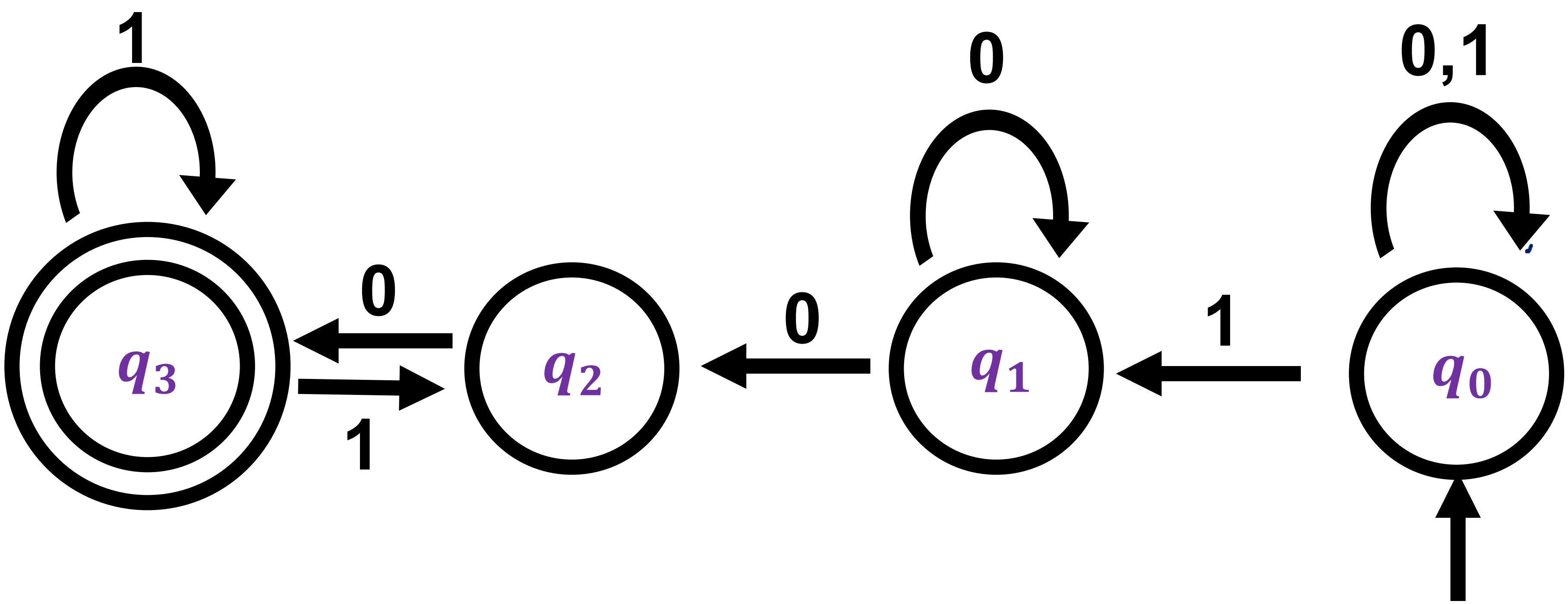
- Next symbol may cause an NFA to “branch” into multiple possible computations
- Next symbol may cause NFA’s computation to fail to enter any state at all

Nondeterminism



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

Nondeterminism



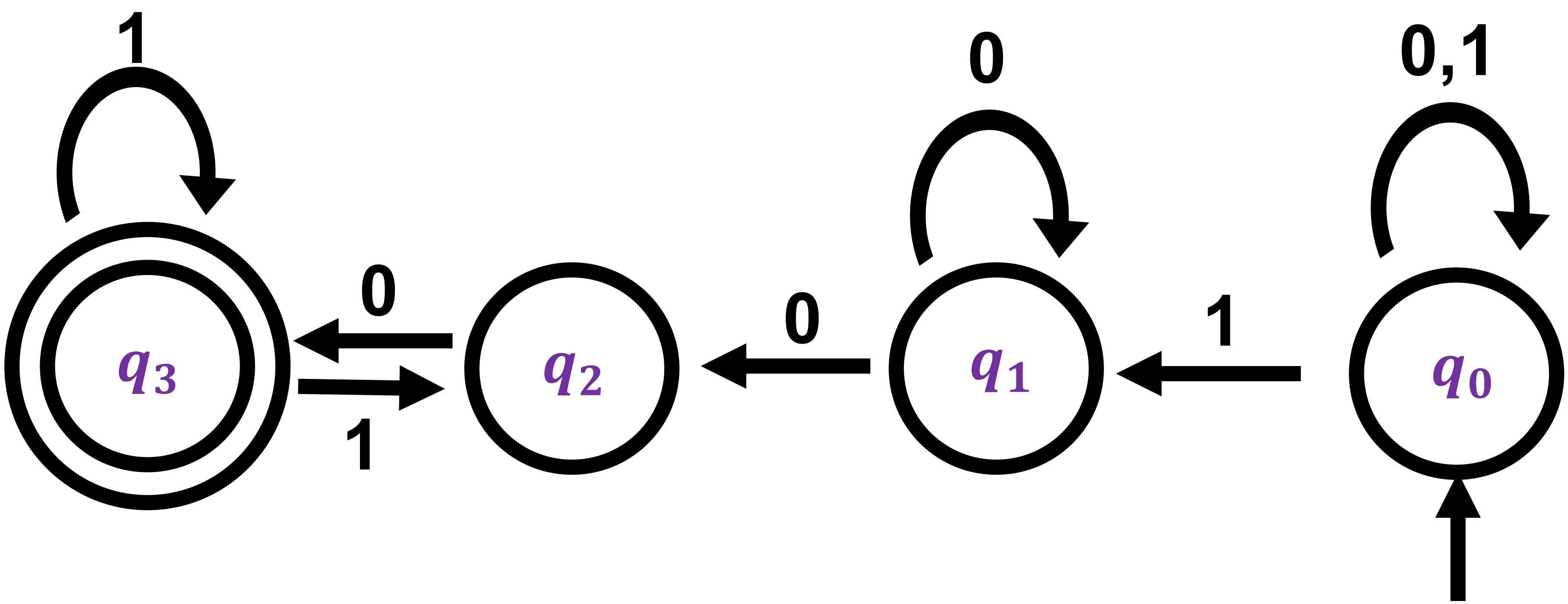
Example: Does this NFA accept the string 1100?

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

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

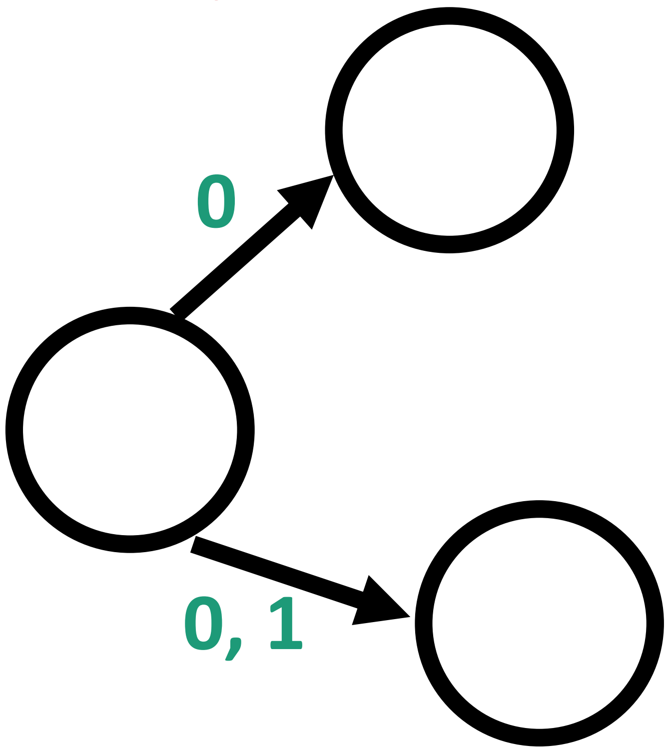
YES, because there <u>exists</u> an accepting computation branch.

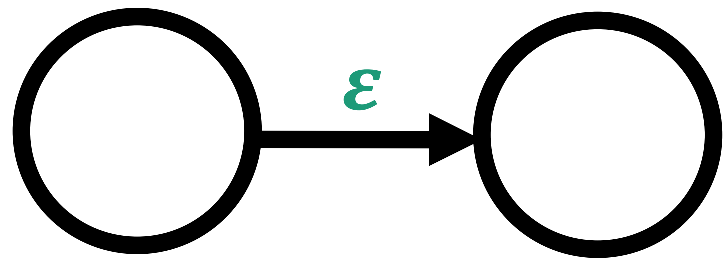
Nondeterminism

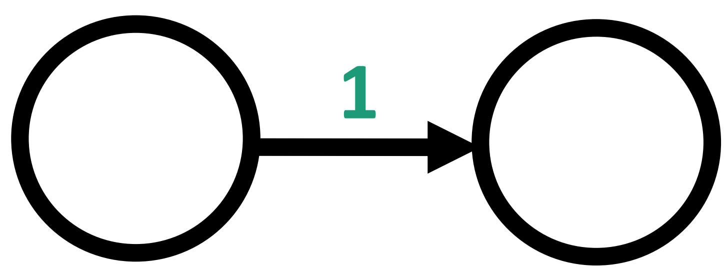


Example: Does this NFA accept the string 11?

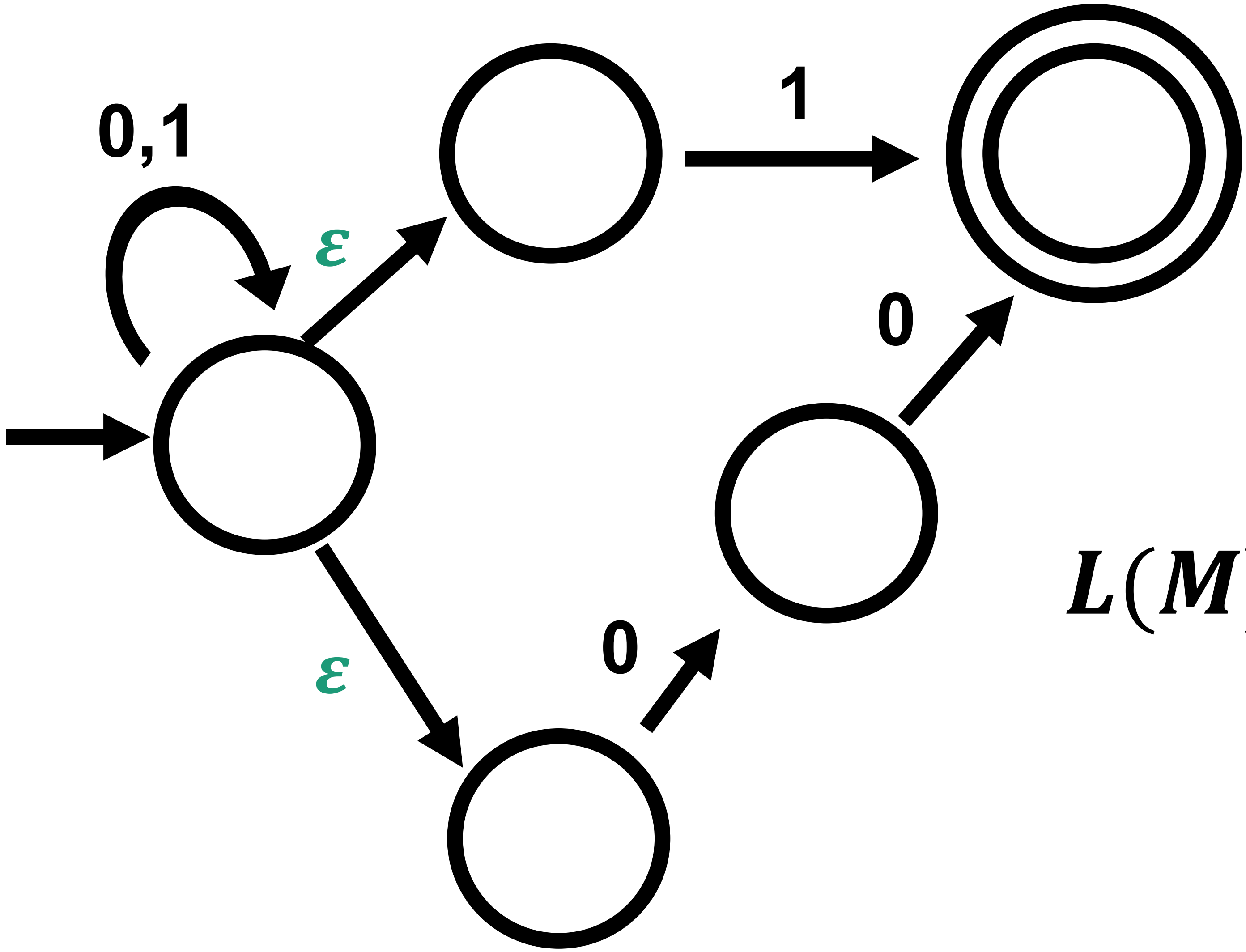
Some special transitions





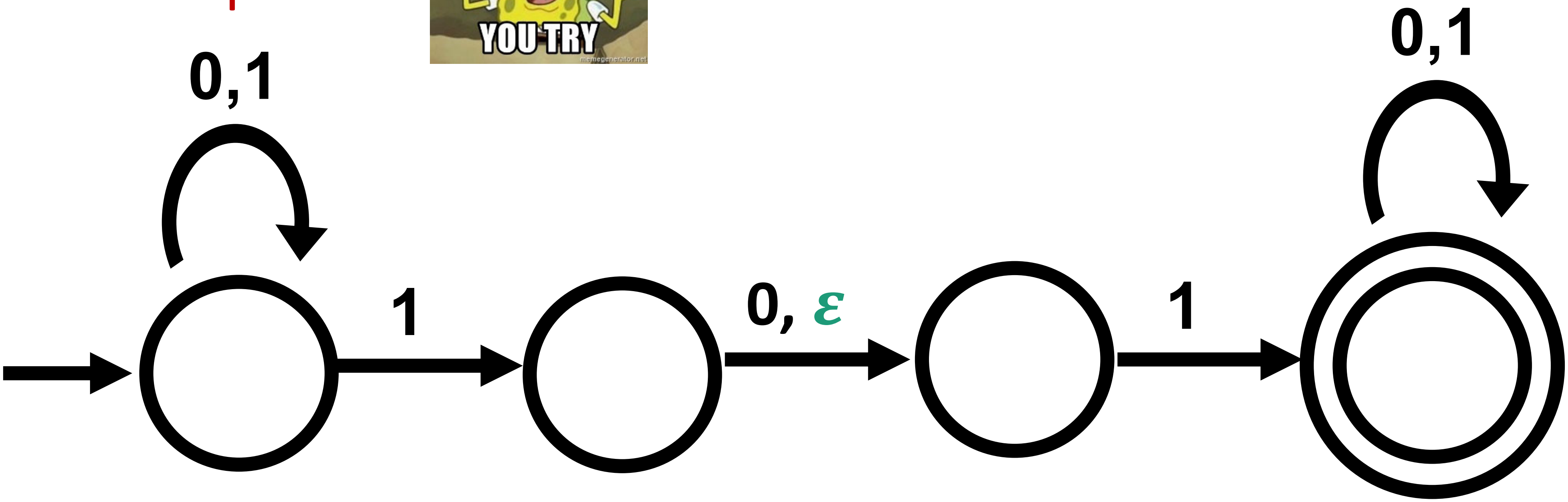


Example



$$L(M) =$$

Example



$\boldsymbol{L(N)} =$

a) $\{w \mid w \text{ ends with } 101\}$
b) $\{w \mid w \text{ ends with } 11 \text{ or } 101\}$
c) $\{w \mid w \text{ contains } 101\}$
d) $\{w \mid w \text{ contains } 11 \text{ or } 101\}$