BU CS 332 – Theory of Computation

Lecture 3:

- Deterministic Finite Automata
- Non-deterministic FAs

Reading:

Sipser Ch 1.1-1.2

Mark Bun

February 1, 2021

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?

Deterministic Finite Automata

A (Real-Life?) Example

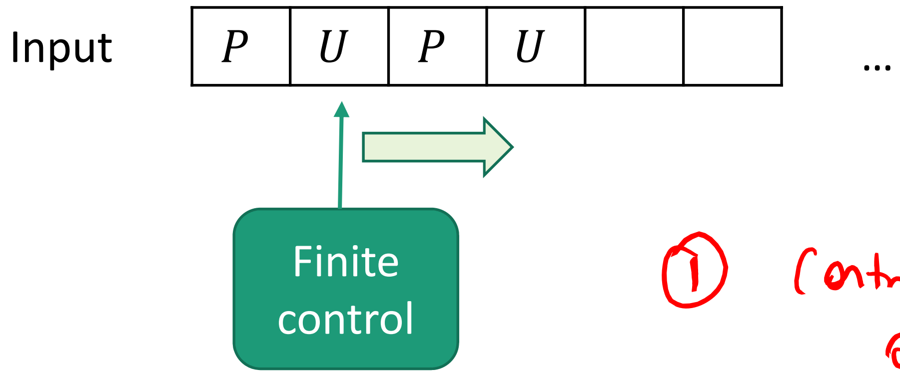
- Example: Kitchen scale
- P = Power button ($\mathtt{ON}$ / $\mathtt{OFF}$)
- U = Units button (cycles through $\mathtt{g}$ / $\mathtt{oz}$ / $\mathtt{lb}$)

  Only works when scale is $\mathtt{ON}$, but units remembered when scale is $\mathtt{OFF}$
- Starts $\mathtt{OFF}$ in $\mathtt{g}$ mode

- A computational problem: Does a sequence of button presses in $\{P, U\}^*$ leave the scale $\mathtt{ON}$ in $\mathtt{oz}$ mode?

Machine Models

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



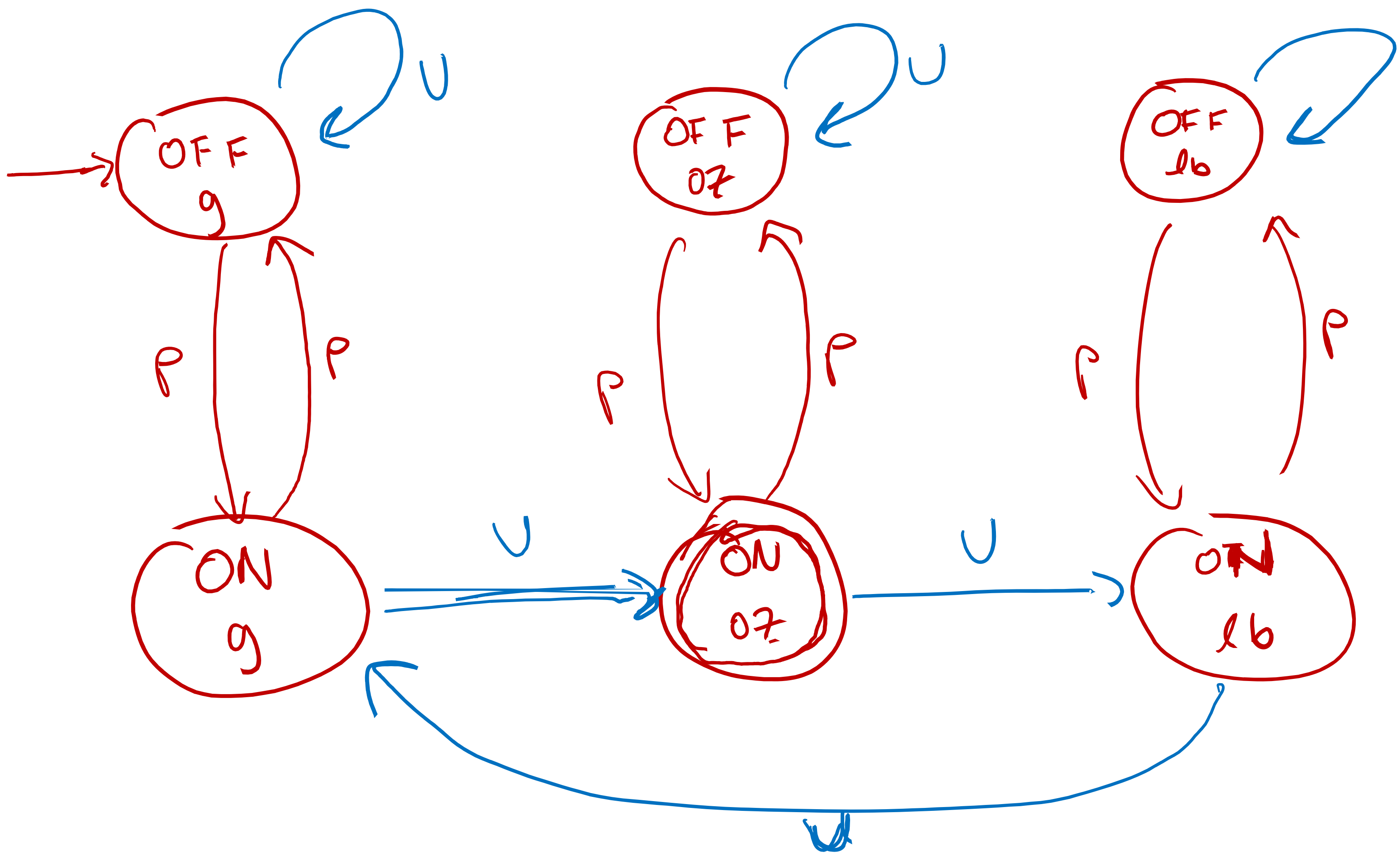
Control scans left-to-right

① Control can take on finite # of "states"

② Control "transitions" between states

"state diagram"

A DFA for the Kitchen Scale Problem



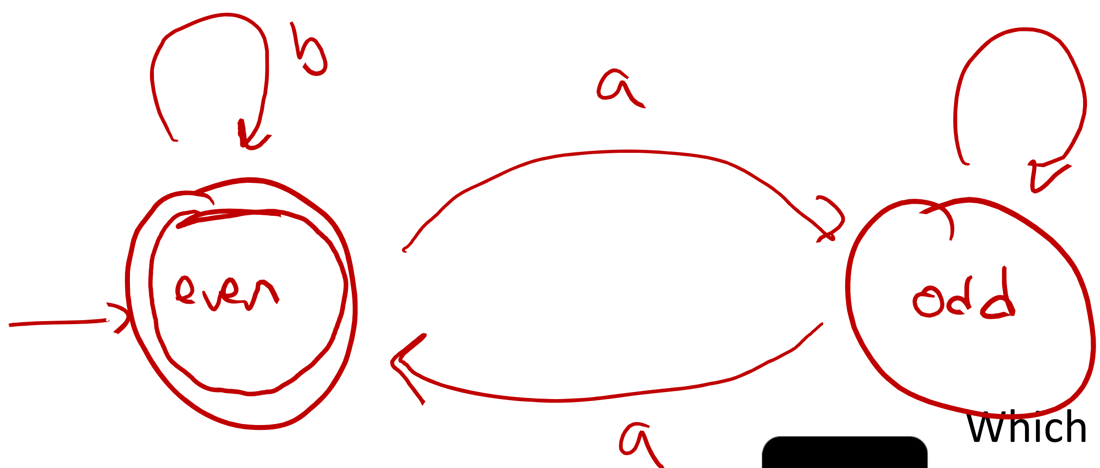
A DFA Recognizing Parity

"deterministic finite automaton"

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?

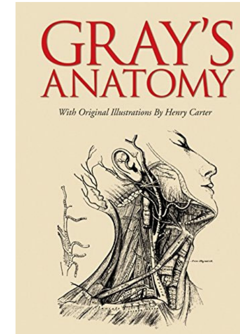
$\Sigma = \{a, b\}$ $L = \{w \mid w \text{ contains an even number of } a\text{'s}\}$

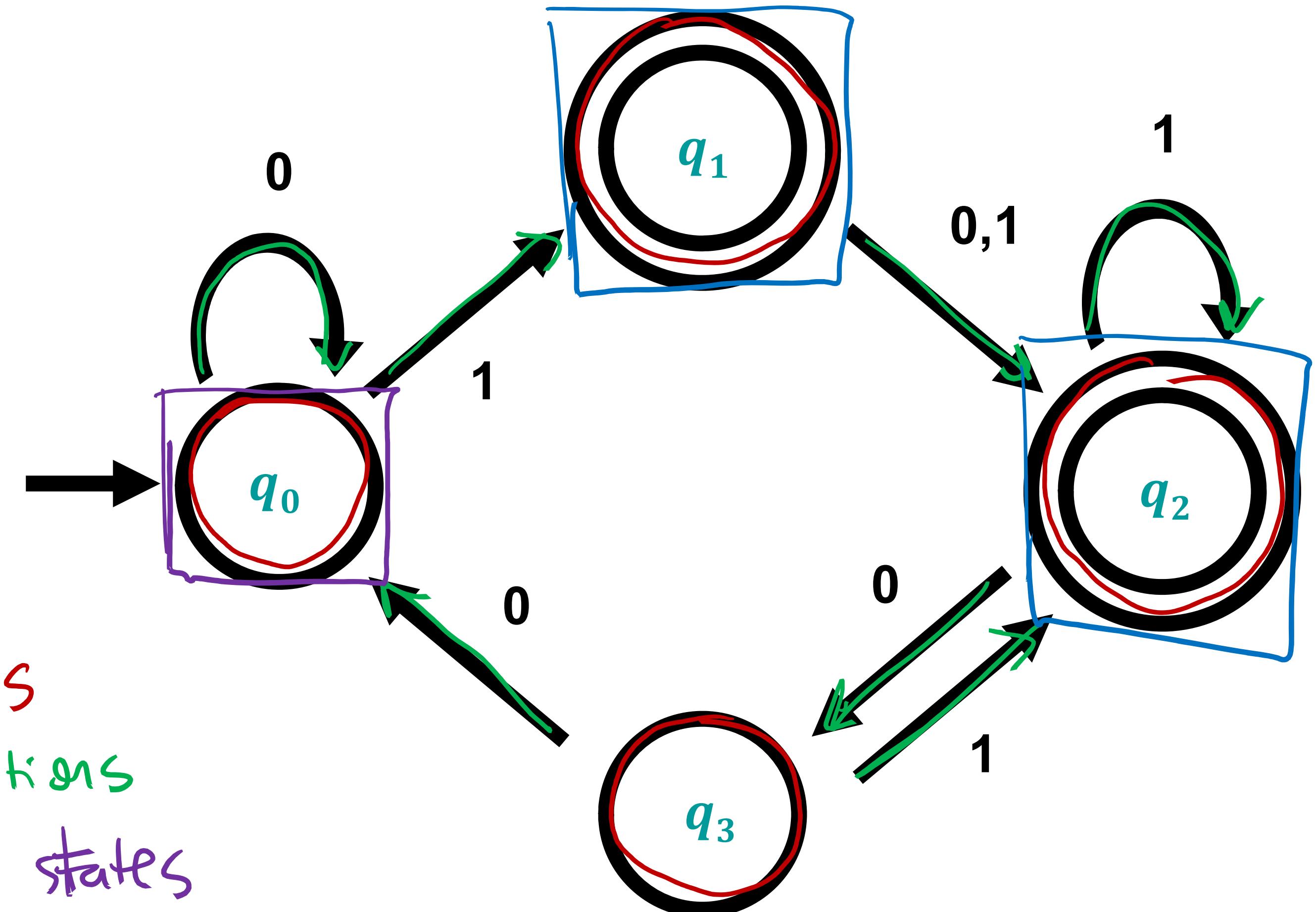


Which state is reached by the parity DFA on input aabab?

a) "even"
b) "odd"

Anatomy of a DFA





States

transitions

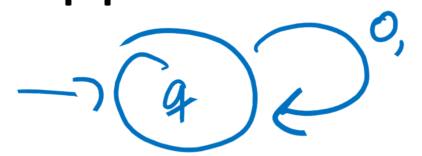
start states

accept states (final states)

Some Tips for Thinking about DFAs

Given a DFA, what language does it recognize?

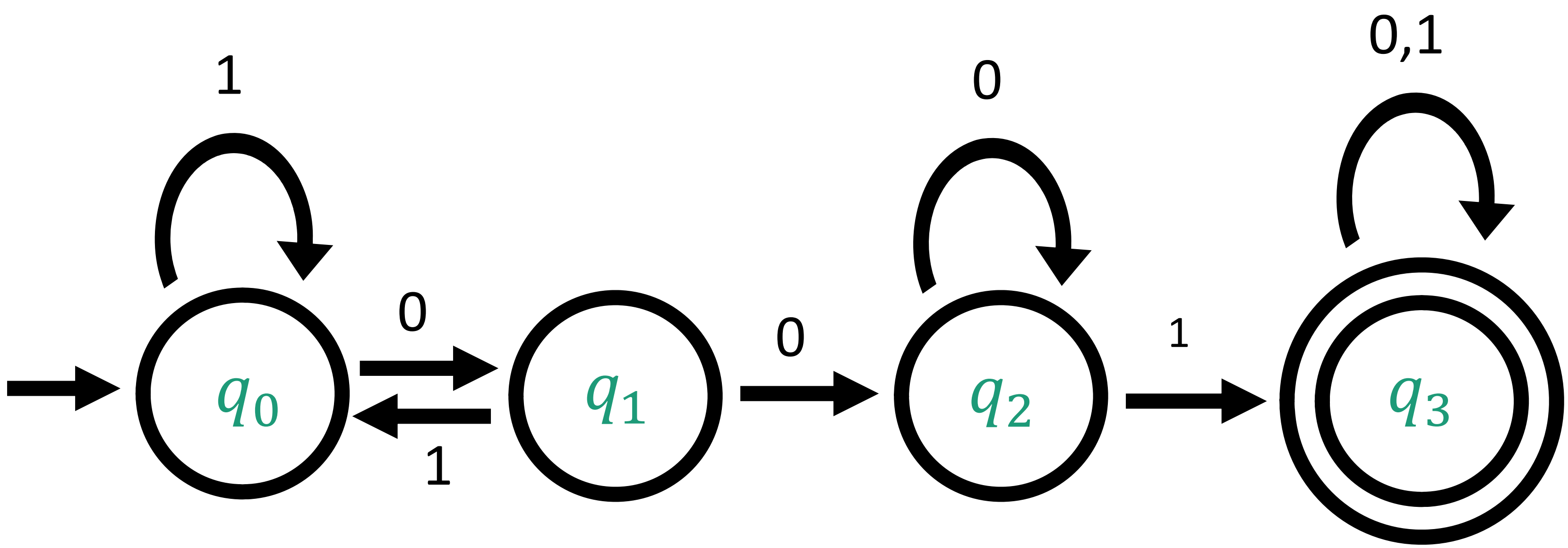
- Try experimenting with it on short strings. Do you notice any patterns?
- What kinds of inputs cause the DFA to get trapped in a state?



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?



$\{ w \mid w$ contains substring 001 $\}$

Practice!

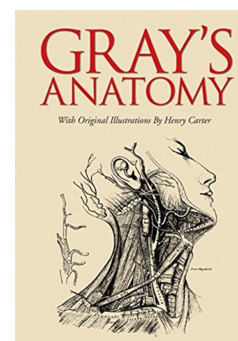
- Lots of worked out examples in Sipser
- Tomorrow's discussion section
- Automata Tutor: https://automata-tutor.model.in.tum.de/

Formal Definition of a DFA

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

input to δ: (state, next symbol)
output of δ: next state

- Q is the set of states
- Σ is the alphabet
- $\delta: Q \times \Sigma \rightarrow Q$ is the transition function
- $q_0 \in Q$ is the start state
- $F \subseteq Q$ is the set of accept states



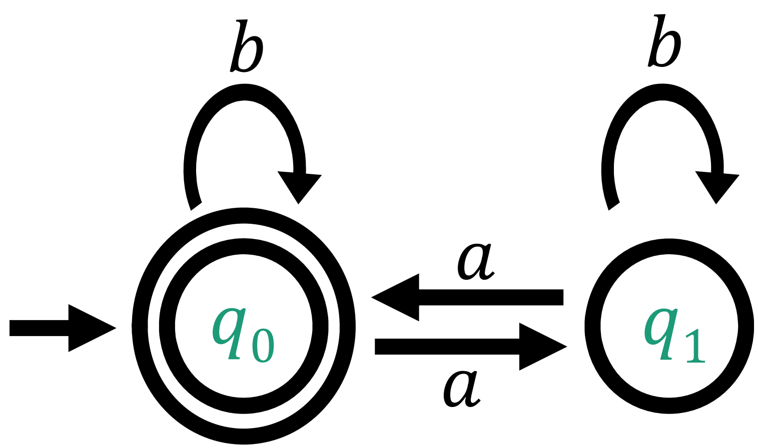
A DFA for Parity

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

$\delta(q, \sigma)$

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

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



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

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

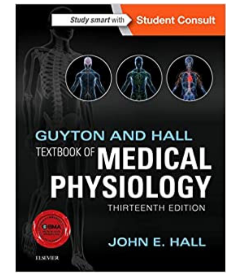
Transition function δ

δ	a	b
q_0	$\delta(q_0, a) = q_1$	q_0
q_1	q_0	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 $w = w_1 w_2 \cdots w_n \in \Sigma^*$ (where each $w_i \in \Sigma$) if there exist $r_0, \ldots, r_n \in Q$ such that

(1) $\to q_0$

(2) $r_i \xrightarrow{w_{i+1}} r_{i+1}$ or $r_i = r_{i+1}$, w_{i+1}

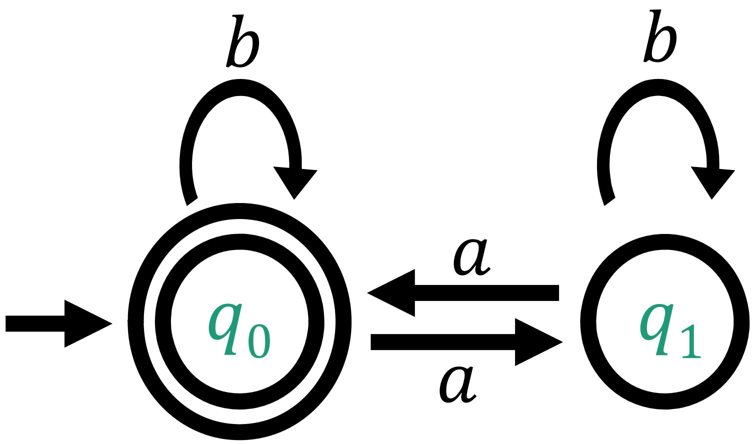
(3) r_n

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



$w_1 \; w_2 \; w_3 \; w_4$

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

$\boxed{r_0} = q_0$

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

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

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

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

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

$r_4 \in F$

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**

Not regular: $\{ a^n b^n \mid n \geq 0 \}$

Many interesting programs recognize regular languages

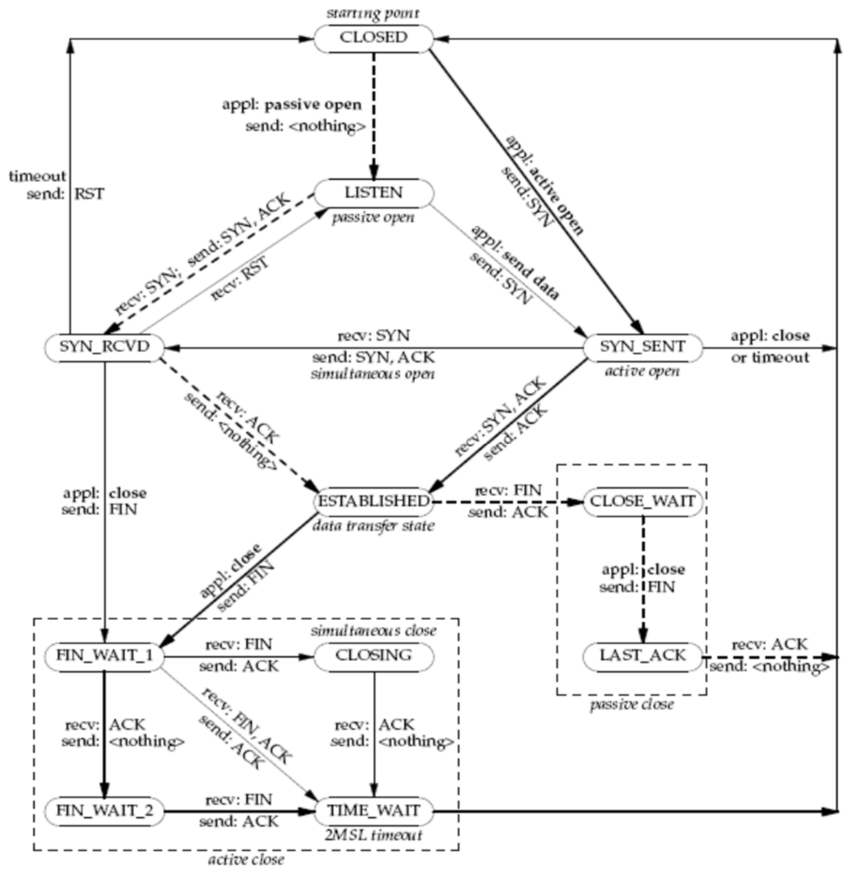
NETWORK PROTOCOLS

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**

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

Theorem. COMMENTS is regular.

Genetic Testing

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

w = AAGCTGCT

A gene g is a special substring over this alphabet.

g = GCT

A genetic test searches a DNA sequence for a gene.

Is g a substring of w?

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

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

Arithmetic

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

← bit of x
← bit of y
← bit of z

check if $x + y = z$

- **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

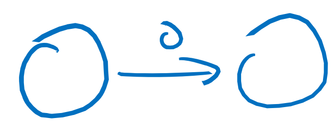
In a DFA, the machine is always in exactly one state upon reading each input symbol

(NFA)

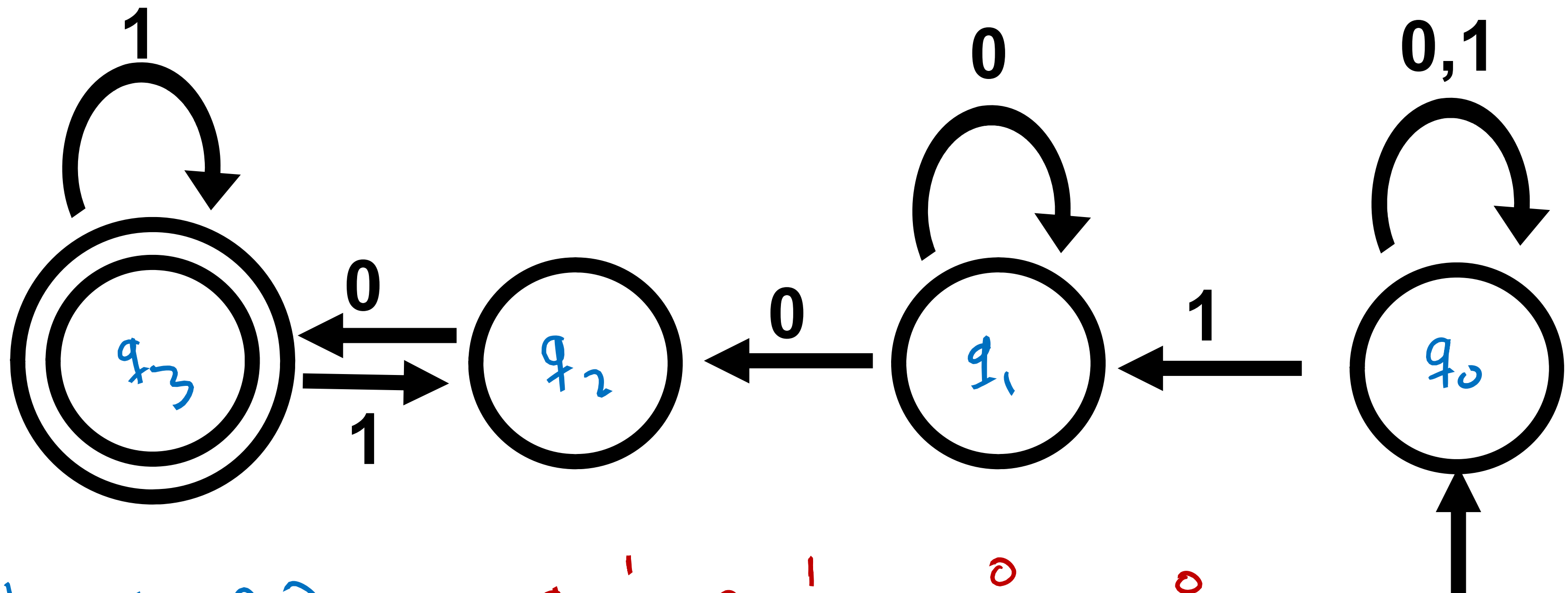
In a nondeterministic FA, the machine can try out many different ways of reading the same string

- 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



$w = 1100$

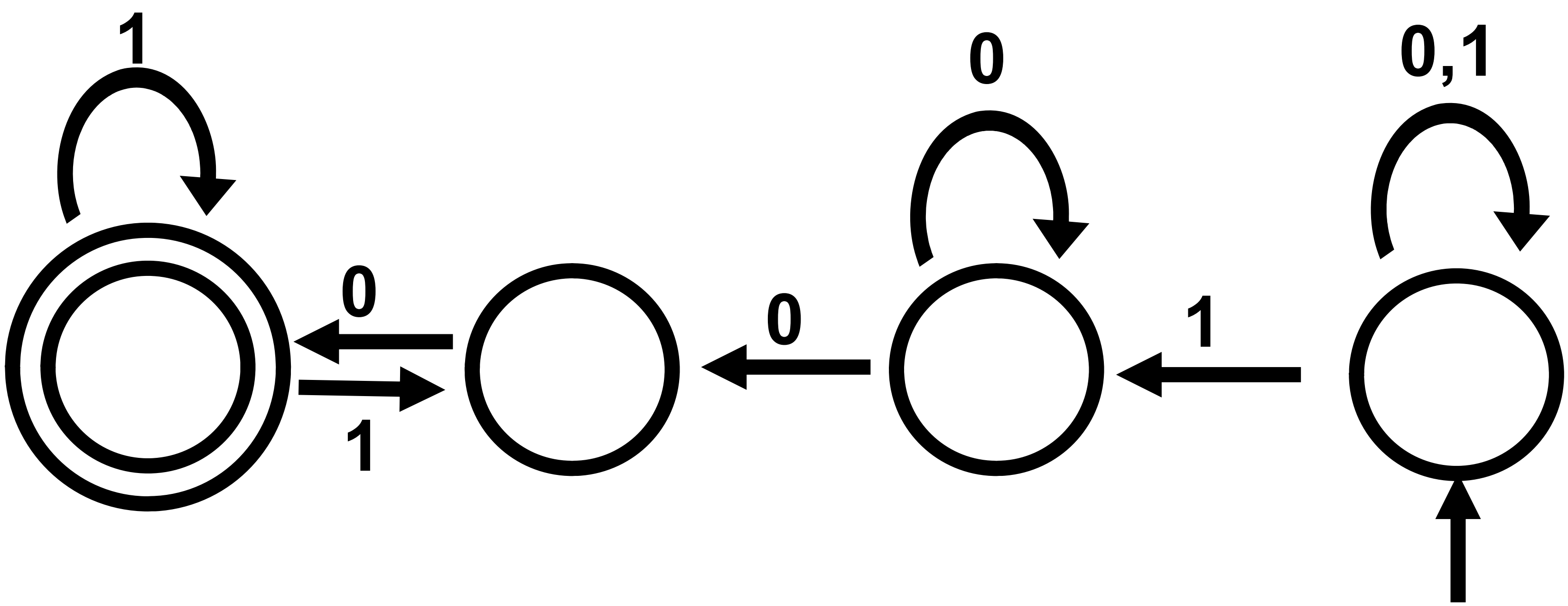
$q_0 \xrightarrow{1} q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_0$

$q_0 \xrightarrow{1} q_1$

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

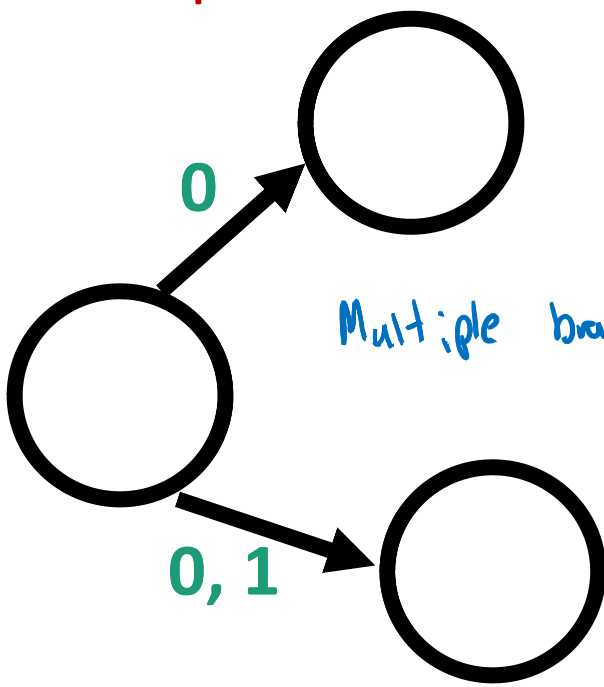
$q_0 \xrightarrow{1} q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_2 \xrightarrow{0} q_3$ (accept)

Nondeterminism

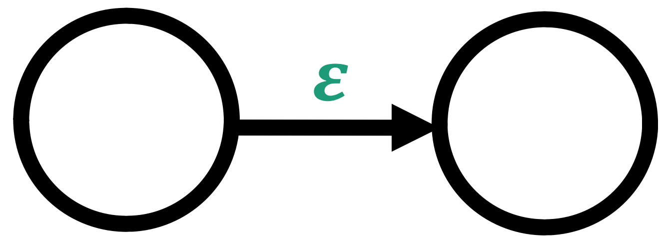


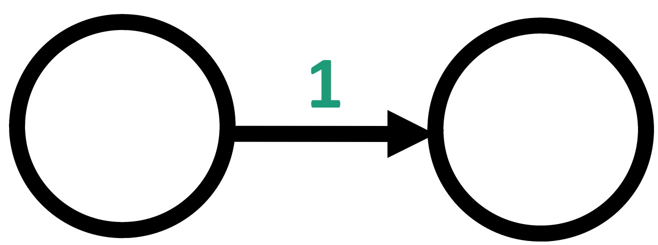
Example: Does this NFA accept the string 1100? YES

Some special transitions



Multiple branching





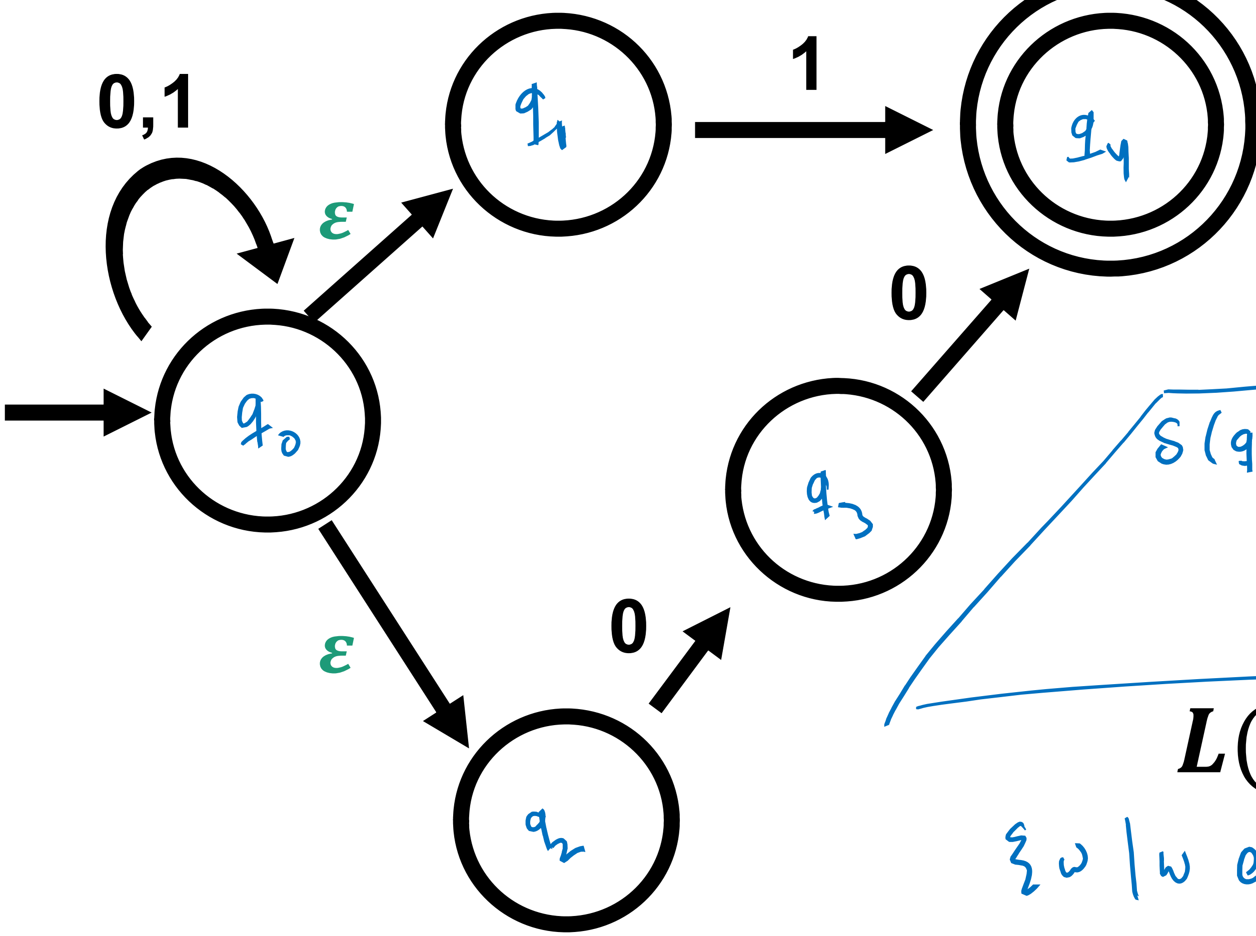
No branching
(a terminates
any comp. path)

Example

$w: 0011$

don't read any thing

$$q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_0 \xrightarrow{\epsilon} q_1 \xrightarrow{1} q_4$$



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

$(\Sigma_\epsilon = \{\epsilon, 0, 1\})$

$\delta(q_0, \epsilon) = \{q_1, q_2\}$

$$L(M) =$$

$\{w \mid w$ ends in 1 or in $00\}$