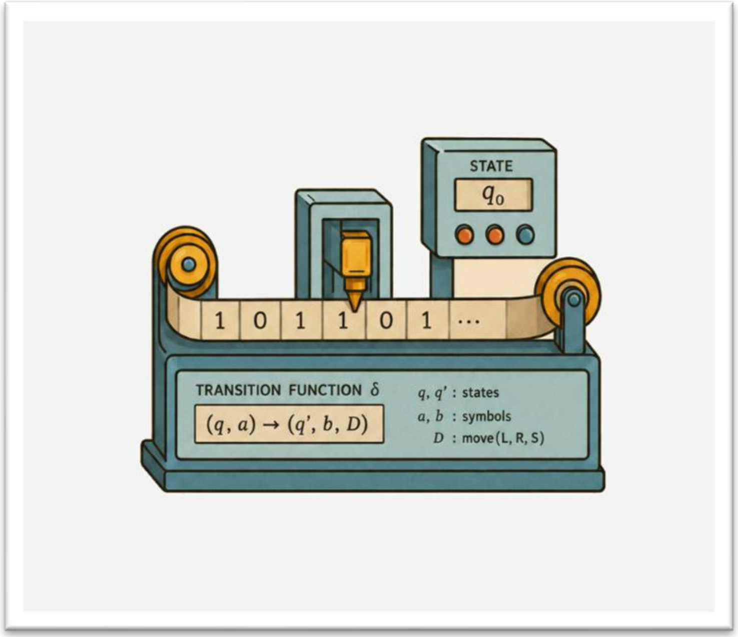
BU CS 332 – Theory of Computation

Lecture 3:

- Deterministic Finite Automata

Check Piazza for Office Hours!



Alexander Poremba

September 10, 2026

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?

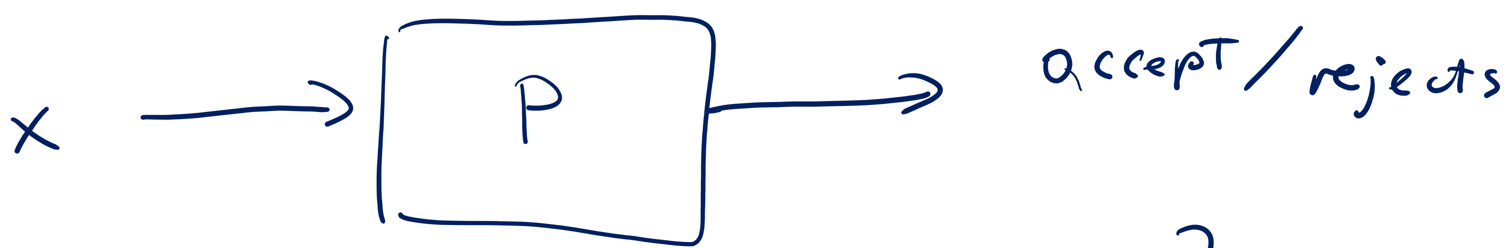
Languages

Languages = computational (decision) problems

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

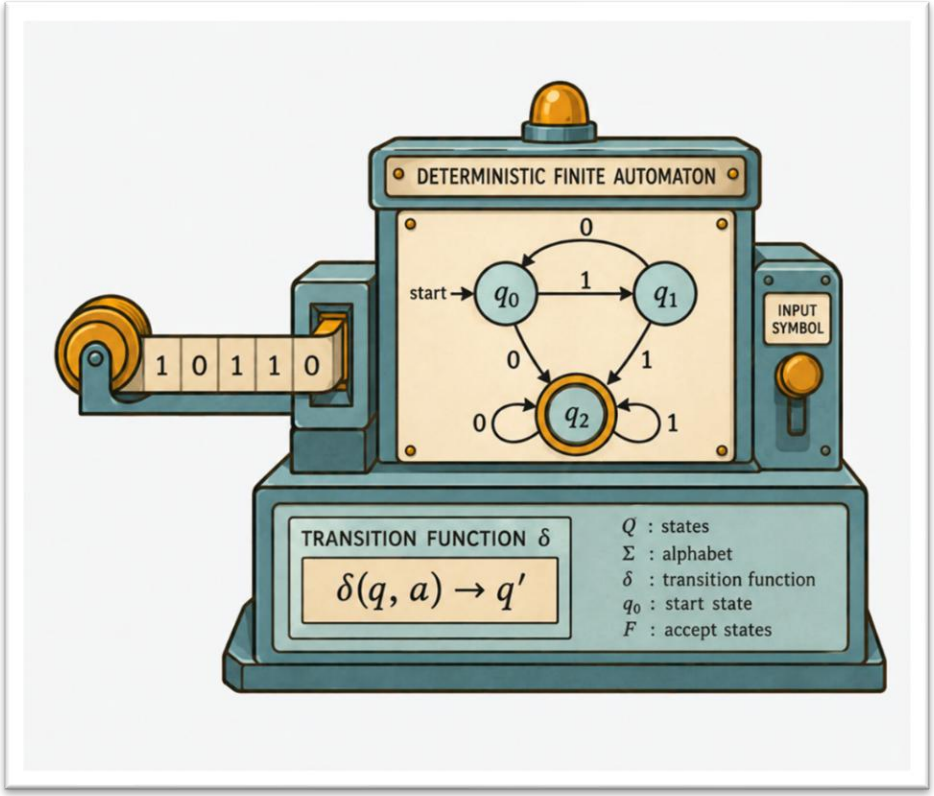
Output: Is $x \in L$? (Yes or No? I.e., Accept or Reject?)

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



$L = \{ x \mid \text{Program } P \text{ accepts string } x \}$

Deterministic Finite Automata



A (Real-Life?) Example

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

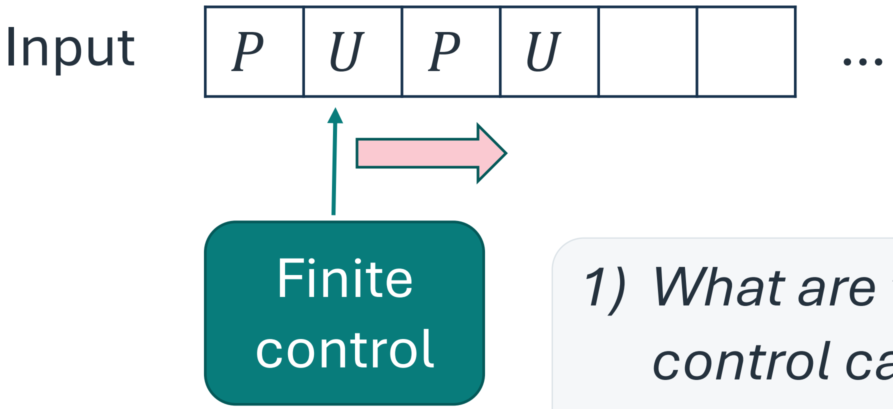
  Only works when scale is ON, but units remembered when scale is OFF
- Starts OFF in g mode

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

Ex. PUUPPUP... Is it in mode ON (oz) ? $\begin{cases} \text{Yes} \\ \text{No} \end{cases}$

Machine Models

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



Control scans left-to-right

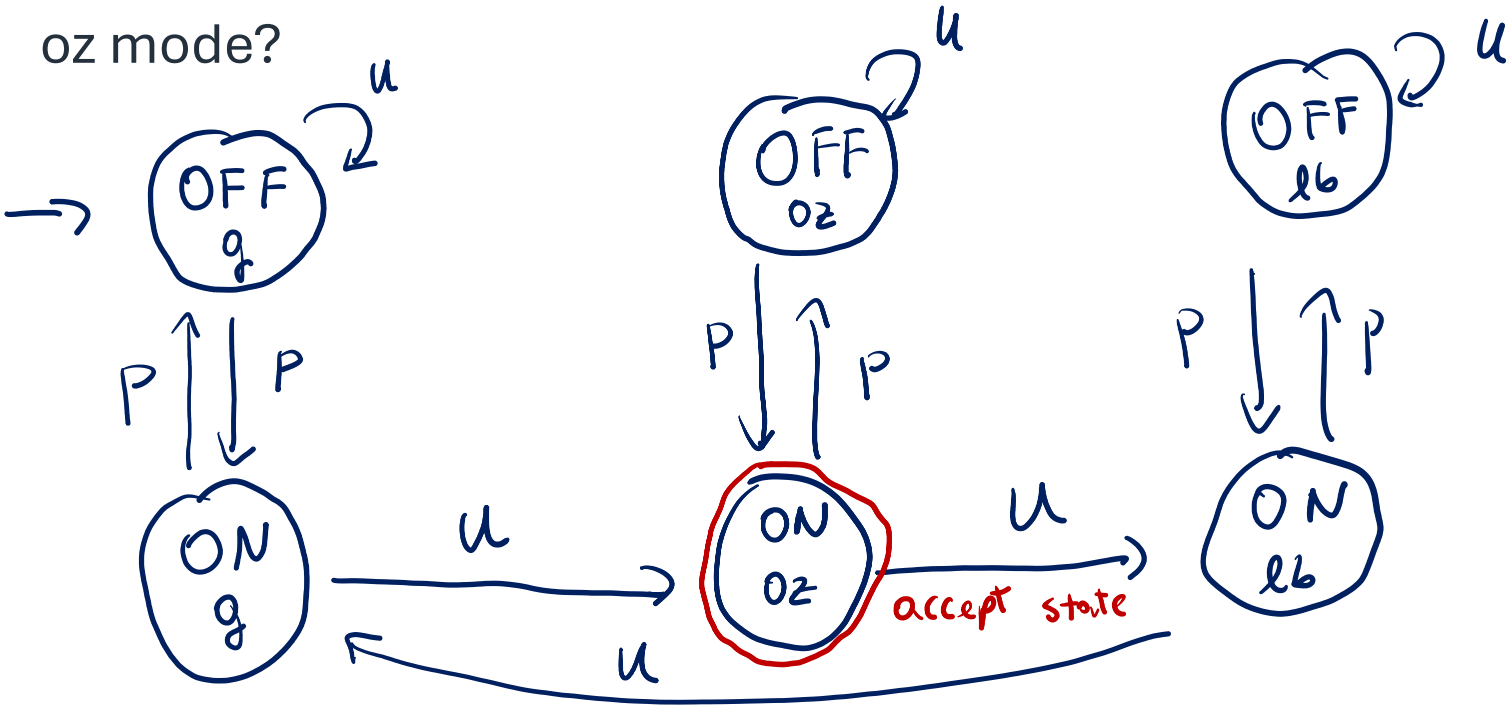
1) *What are the different "states" that the control can be in?*
2) *In what state does the control start?*
3) *When the control reads a new input character, how does it transition to a new state?*
4) *How do I know if I'm in the desired state at the end?*

A DFA for the Kitchen Scale Problem

P = Power button (ON / OFF) (Starts OFF in g mode)

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

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



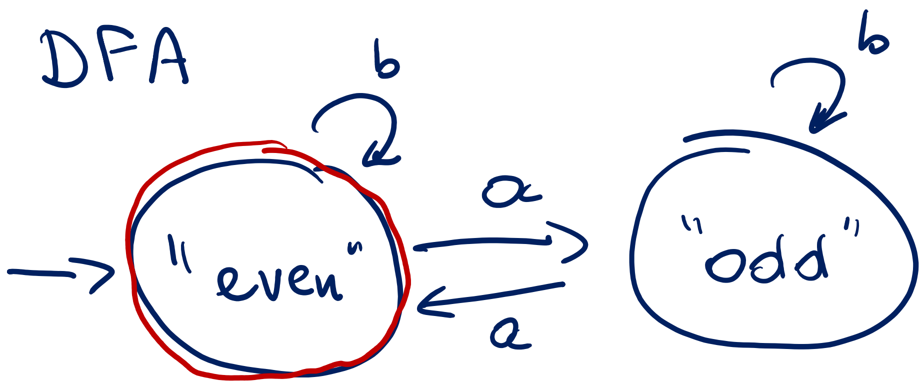
Ex: PUPPU → "no"   PPPU → "yes"

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?

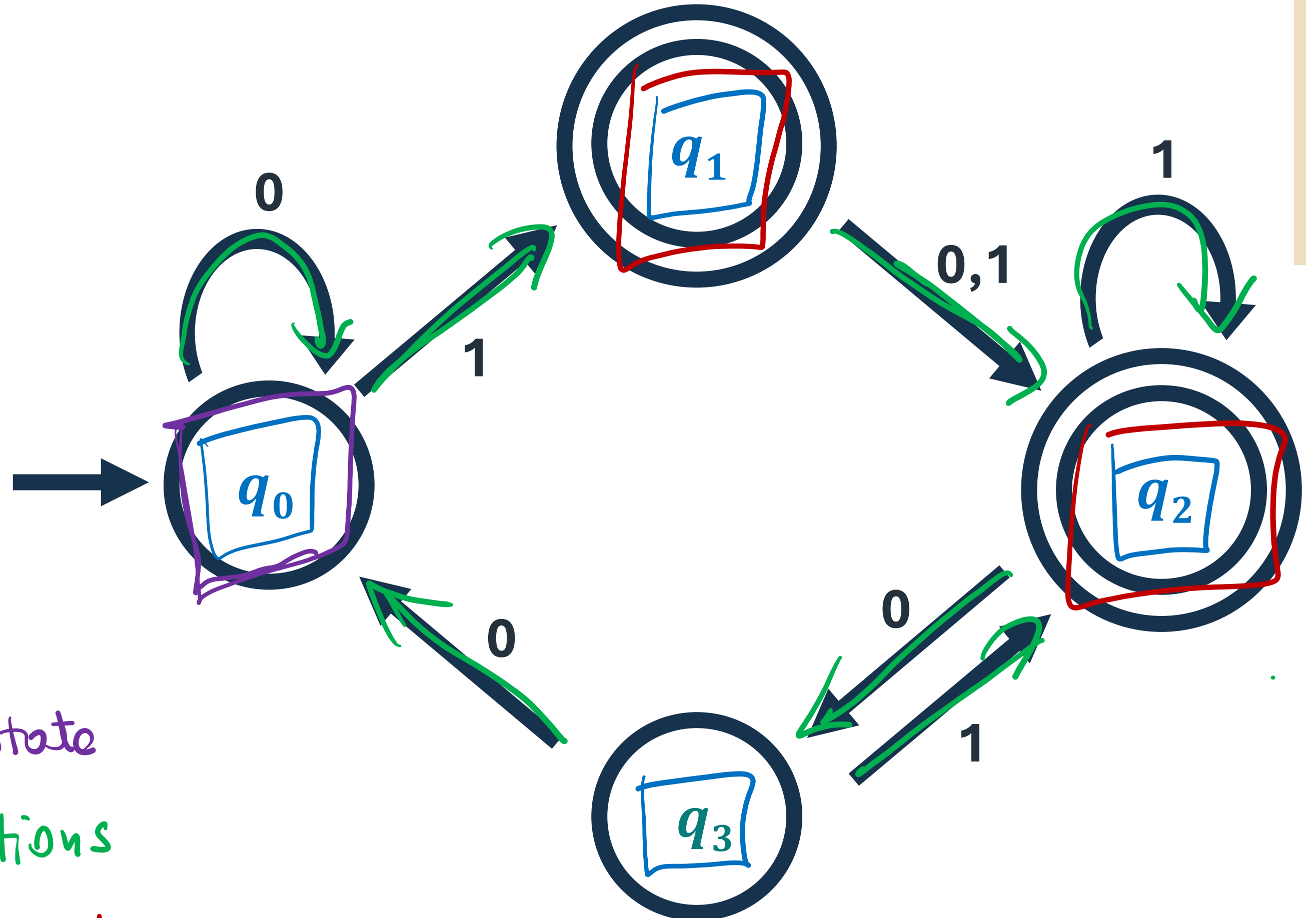
$\Sigma = \{a, b\}$ $L = \{w \mid w \text{ contains an even number of } a\text{'s}\}$

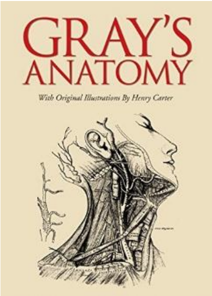


Question: Which state is reached by the parity DFA on input $aabab$?

a) "even"
b) "odd"

Anatomy of a DFA



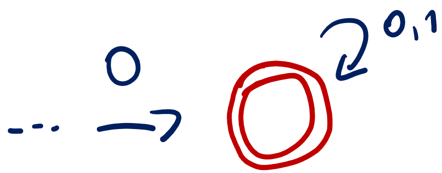


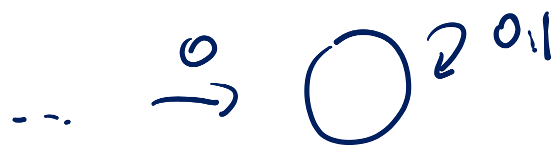
≡ States
≡ start state
≡ transitions
≡ accept states

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?



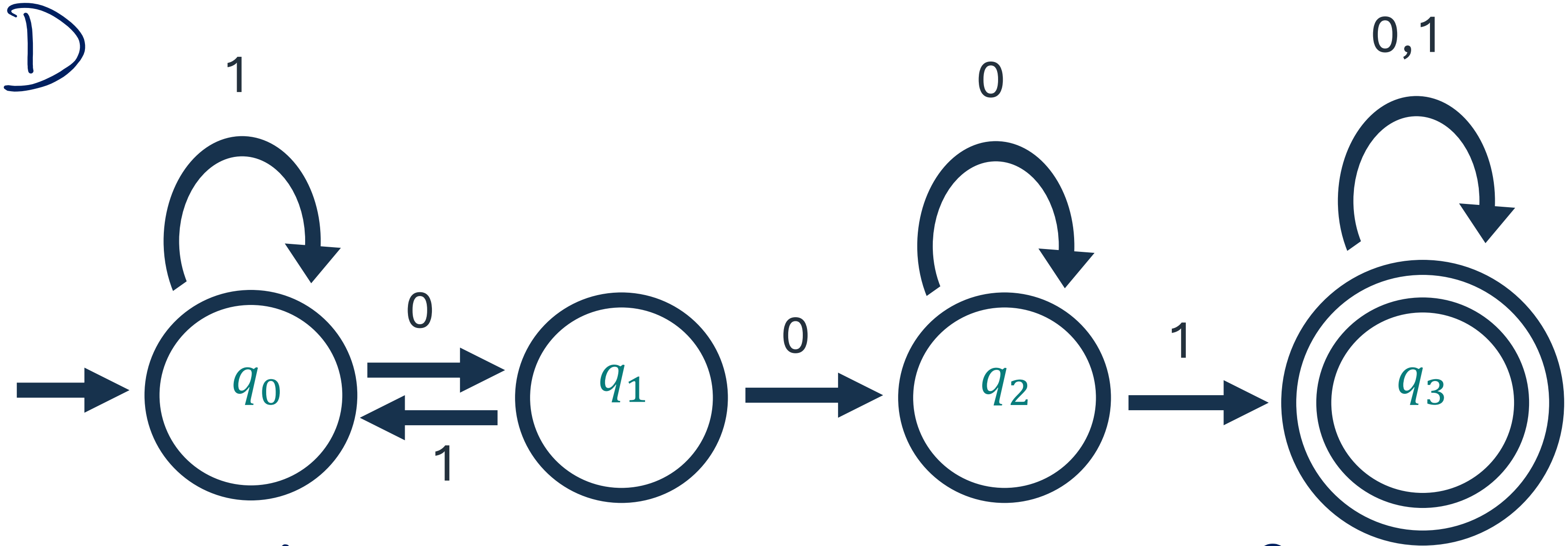


Given a language, construct a DFA recognizing it

- 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?

D



$L(D) = \{ x \mid x$ contains 001 as a substring $\}$

<u>Ex</u>: Input "001"

$q_0 \xrightarrow{0} q_1 \xrightarrow{0} q_2 \xrightarrow{1} q_3 \Rightarrow$ accept!

accept state

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

Σ is the alphabet Ex: $\Sigma = \{a, b\}$

next state

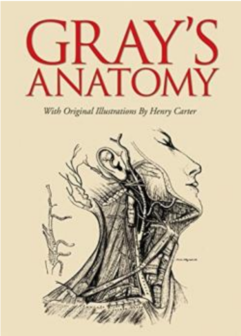
$\delta: Q \times \Sigma \to Q$ is the transition function Ex: $\delta(q_0, a) = q_1$

current state current symbol

$q_0 \in Q$ is the start state

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

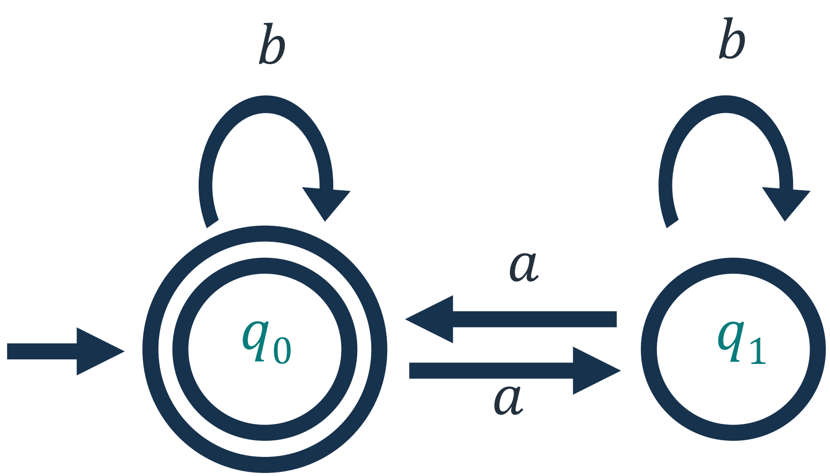
Ex: $F = \{q_3, q_4\}$



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

$\delta(q_0, a) = q_1$

$\delta(q_0, b) = q_0$

$\delta(q_1, a) = q_0$

$\delta(q_1, b) = q_1$

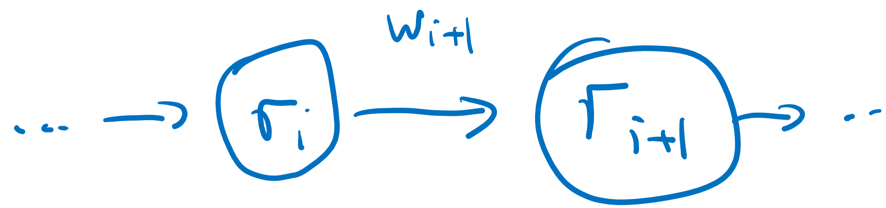
Start state q_0

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

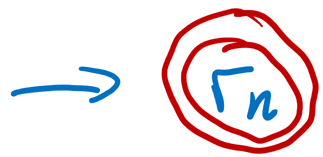
Formal Definition of DFA Computation

A DFA $M = (Q, \Sigma, \delta, q_0, F)$ accepts a string $w = w_1 w_2 \cdots w_n \in \Sigma^*$ (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 $i = 0, \ldots, n-1$, and



accept state

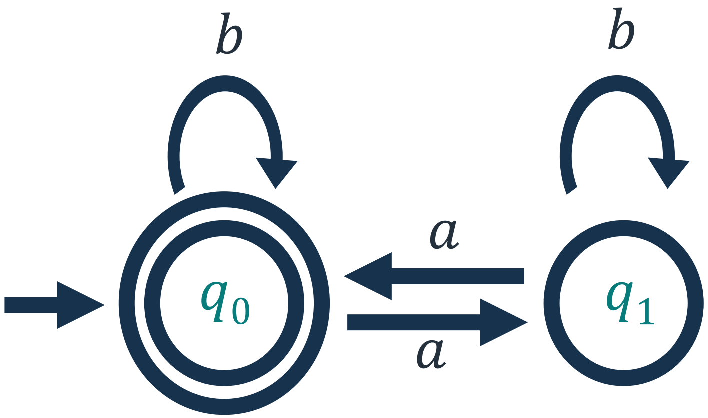
3. $r_n \in F$

$L(M)$ = 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 $w = abba$

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. $r_n \in F$

Input: $w = abba$

$$q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_1 \xrightarrow{b} q_1 \xrightarrow{a} q_0$$

$$r_0 \xrightarrow{w_1} r_1 \xrightarrow{w_2} r_2 \xrightarrow{w_3} r_3 \xrightarrow{w_4} r_4$$

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

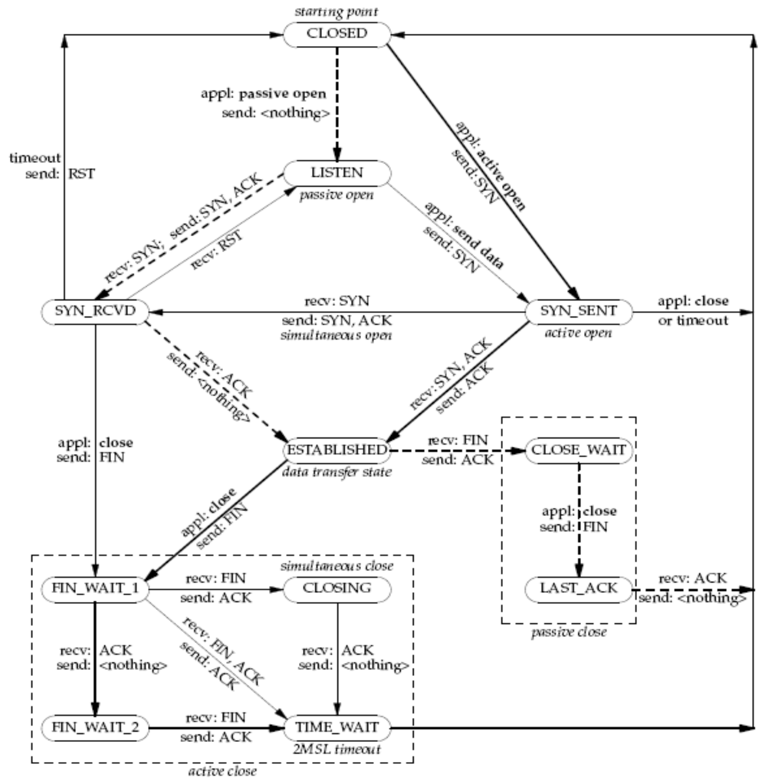
Network Protocols

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

/* comments here */

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

Theorem: COMMENTS is regular

Genetic Testing

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

GATTACA

A gene g is a special substring over this alphabet.

g = TAC

A genetic test searches a DNA sequence for a gene.

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

$$\begin{array}{r} 2 \\ +4 \\ \hline 6 \end{array} \rightarrow \begin{array}{r} 010 \\ +001 \\ \hline 011 \end{array}$$

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

Theorem. ADD is regular.

Ex: String $\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}1\\0\\1\end{bmatrix}, \begin{bmatrix}0\\1\\1\end{bmatrix}$ $\in$ ADD

Next time: Nondeterministic Finite Automata