BU CS 332 – Theory of Computation

https://forms.gle/joN5KwcKrauQLFAL7

Lecture 3:

- Deterministic Finite Automata
- Non-deterministic FAs

Reading:

Sipser Ch 1.1-1.2

Mark Bun

September 13, 2022

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

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

Output: Is $x \in L$? (YES or NO?)

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

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

On input $x = x_1 x_2 \dots x_n$:
flag = FALSE
 For $i = 1, \dots, n$:
  If $x_i == \mathrm{b}$:
   flag = !flag
 If flag: accept
 Else: reject

$x = \mathrm{abbab}$

flag = FALSE → F
→ T
→ F
→ F
→ T

Program accepts if and only if input x has an odd # of b's

Language recognized by this program = $\{x \in \{a,b\}^* \mid x \text{ has an odd \# of b's}\}$

Deterministic Finite Automata

A (Real-Life?) Example

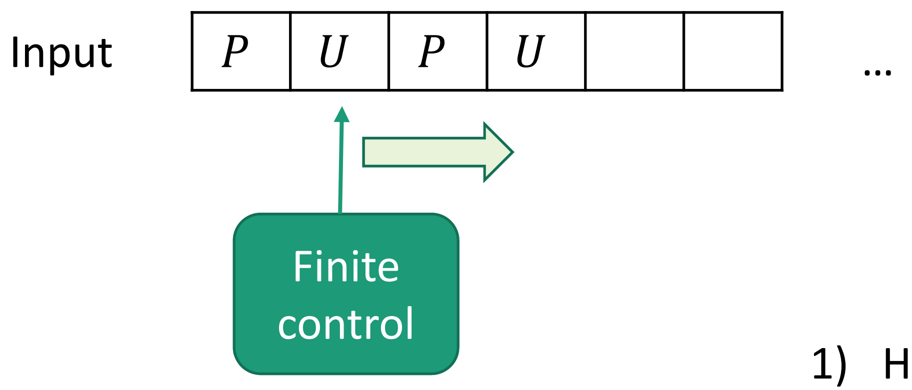
- Example: Kitchen scale
- P = Power button ($\mathtt{ON}$ / $\mathtt{OFF}$)
- U = Units button (cycles through $\mathtt{g}$ / $\mathtt{oz}$ / $\mathtt{lb}$)

  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

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

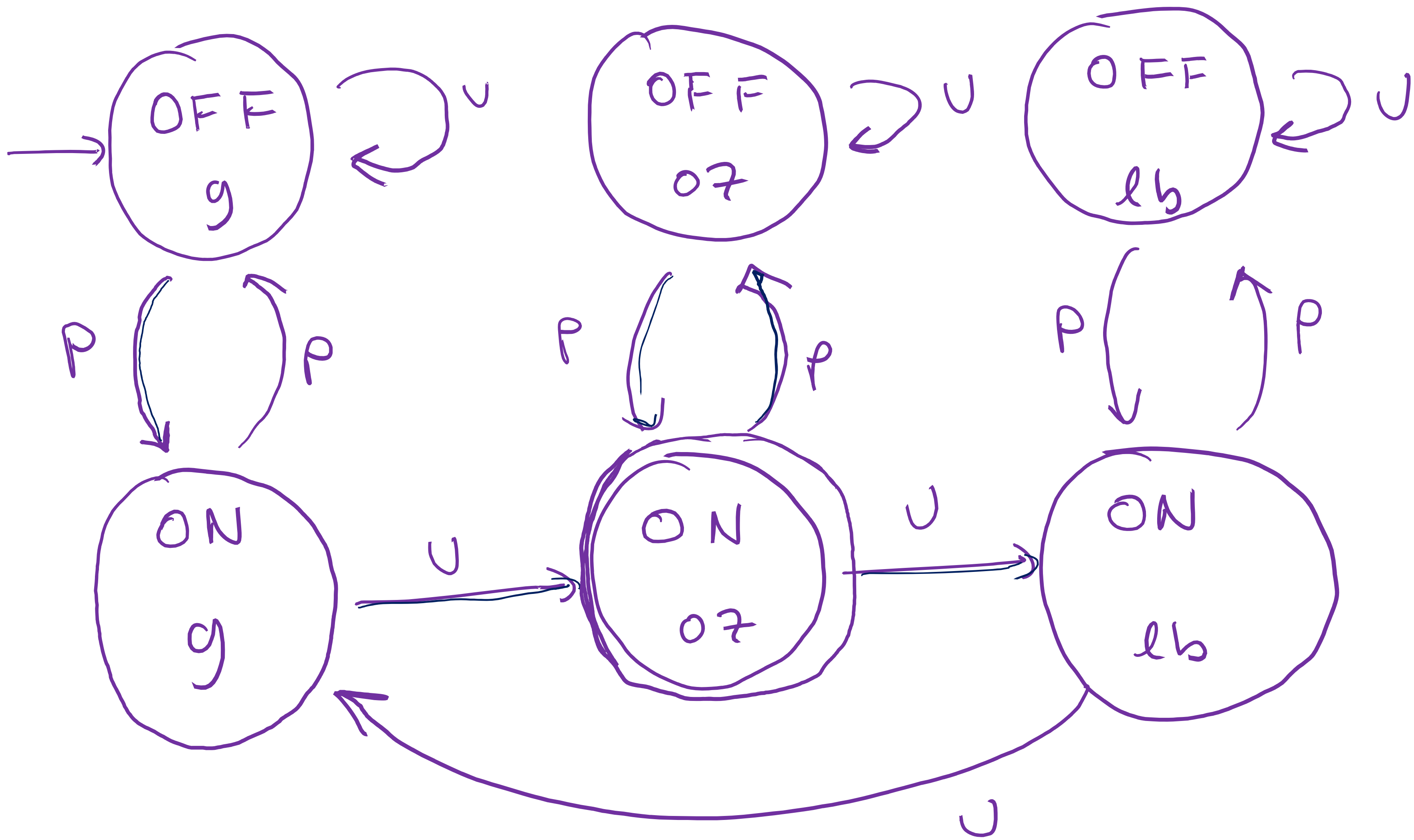


Control scans left-to-right

1) How does the control start?
2) What are the different "states" that the control can be in?
3) When the control reads an 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 u p p u



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?

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

abba

start: even

$\xrightarrow{a}$ odd

$\xrightarrow{b}$ odd

$\xrightarrow{b}$ odd

$\xrightarrow{a}$ even

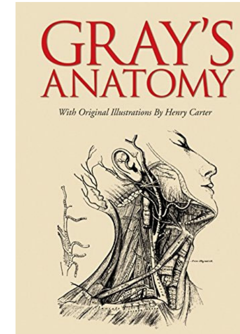
accept

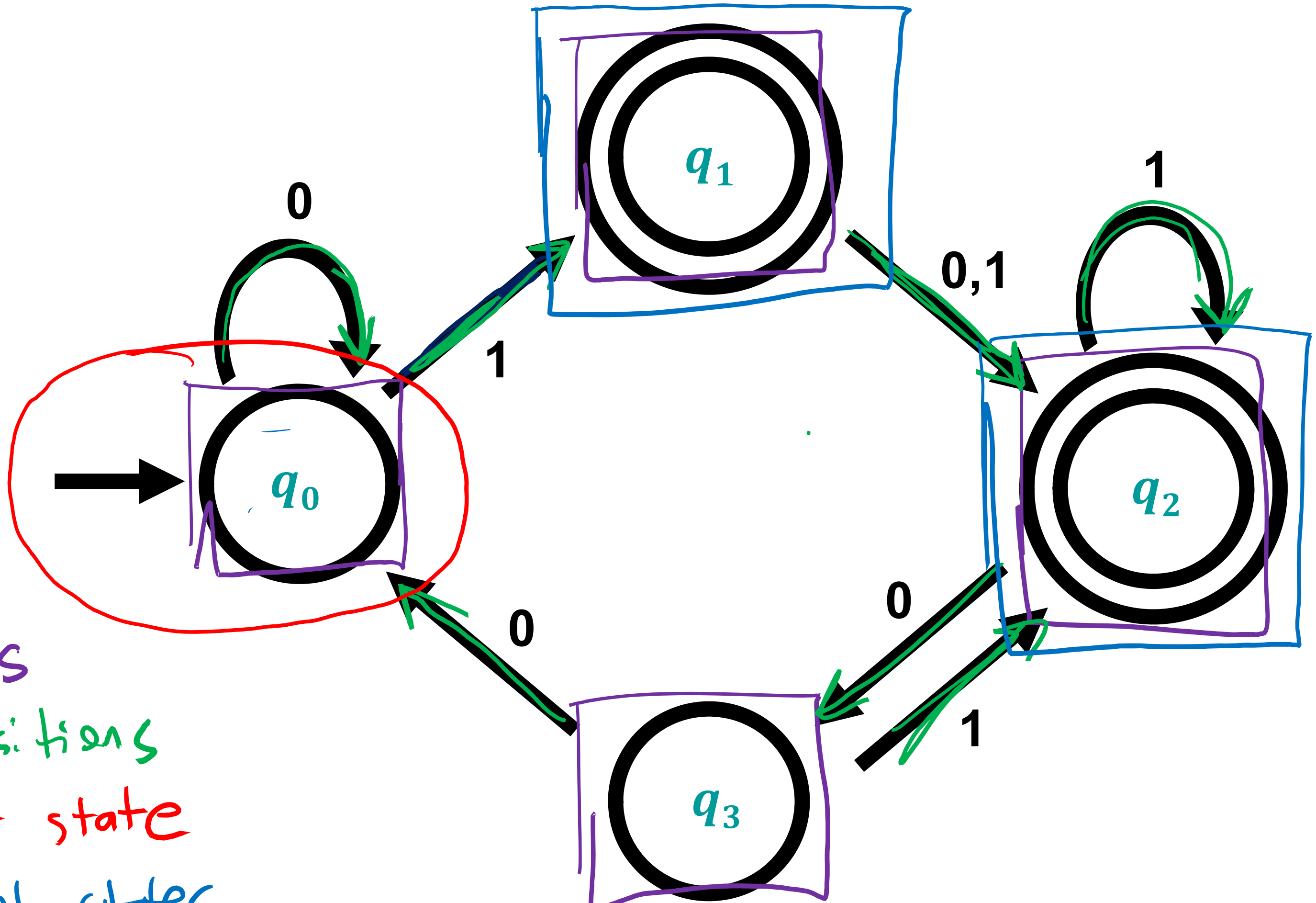
Which state is reached by the parity DFA on input aabab?

a) "even"

b) "odd"

Anatomy of a DFA



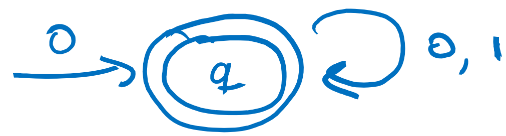


States
transitions
start state
accept states
("final states")

Some Tips for Thinking about DFAs

Given a DFA, what language does it recognize?

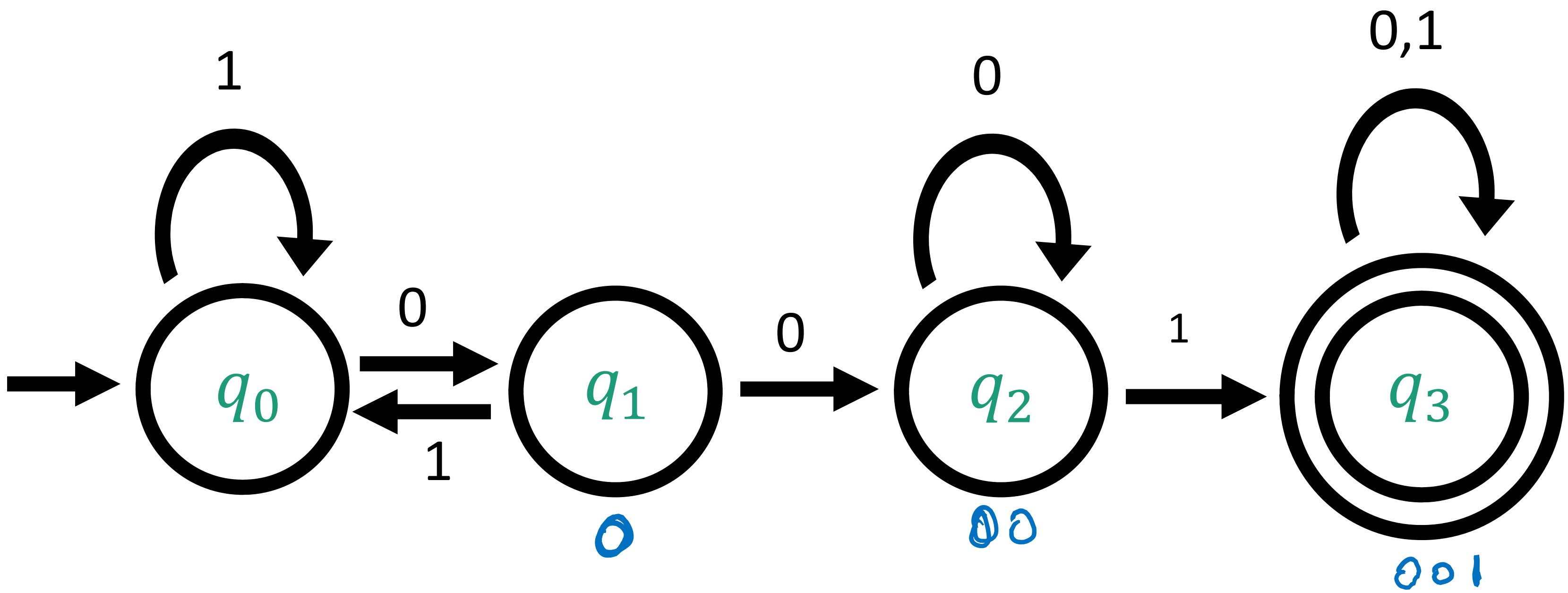
- Try experimenting with it on short strings. Do you notice any patterns?
- What kinds of inputs cause the DFA to get trapped in a state?



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?



$L = \{ w \mid 001 \text{ is a substring of } w \}$

Practice!

- Lots of worked out examples in Sipser
- Automata Tutor: https://automata-tutor.model.in.tum.de/

Formal Definition of a DFA

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

Q is the set of states $\quad Q = \{q_0, q_1, q_2\}$

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

$\delta: \overbrace{Q}^{\text{old state}} \times \overbrace{\Sigma}^{\text{alphabet symbol}} \to \overset{\text{new state}}{Q}$ is the transition function

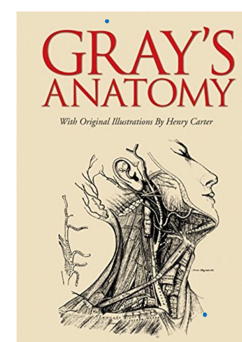
$q_0 \in Q$ is the start state

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

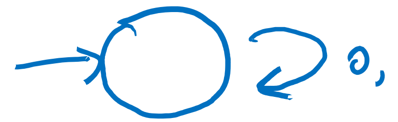
e.g. $F = \{q_1\}$

$F = Q$

$F = \emptyset$

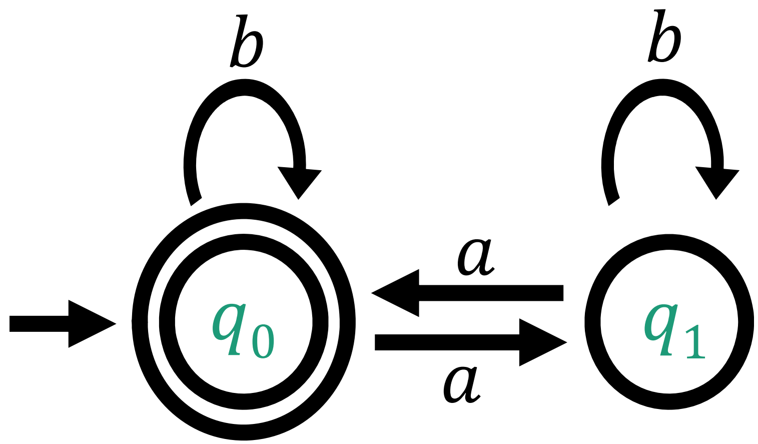


A DFA for Parity



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

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



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

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

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

alphabet symbol

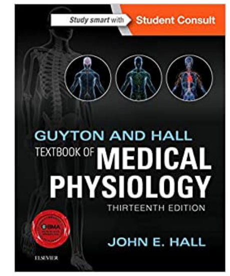
old state

δ	a	b
q_0	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

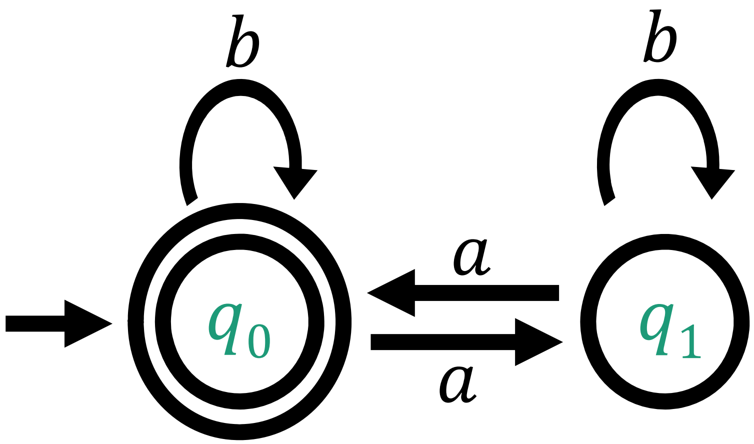
1. $r_0 = q_0$
2. $\delta(r_i, w_{i+1}) = r_{i+1}$ for each $i = 0, \ldots, n-1$, and
3. $r_n \in F$

$L(M)$ = the language of machine M "the language recognized by M"

= set of all strings machine M accepts

M recognizes the language $L(M)$

Example: Computing with the Parity DFA



Let $w = \overset{w_1 w_2 w_3 w_4}{abba}$

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 $i = 0, \ldots, n-1$, and
3. $r_n \in F$

$r_0 = q_0$

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

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

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

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

Regular Languages

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

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

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

Many interesting problems are captured by regular languages

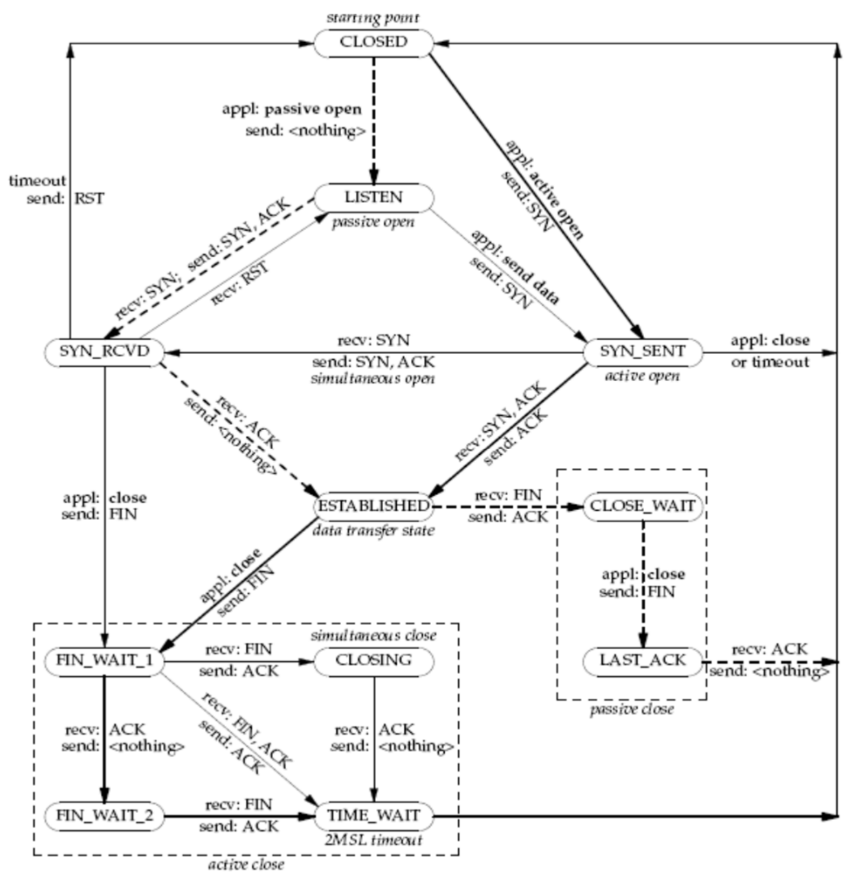
NETWORK PROTOCOLS

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.

e.g. g = CAT

A genetic test searches a DNA sequence for a gene.

e.g. is "CAT" 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 Σ = $\left\{ \begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\1\\1\end{bmatrix}, \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}1\\0\\1\end{bmatrix}, \begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}1\\1\\1\end{bmatrix} \right\}$

$$\begin{array}{cccc} & 0 & 1 & 0 \\ + & 1 & 1 & 0 \\ \hline & 1 & 0 & 1 \\ & \text{lsb} & & \text{msb} \end{array}$$

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

Theorem. ADD is regular.