Coms 3261 – Sum '22 – Lecture 1

twrand.github.io/3261-sum22.html

TIM

Today:

1. What is CS Theory?
2. Nuts & Bolts
3. Building Blocks: Mathmatical Primitives
4. Automata: Math machines

---

1. What is CS Theory?

Using math to learn about computation

what math?
why math?

what is computation, really?

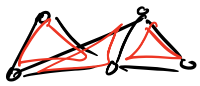
Math problems:

Input: 1682, 9837
Question: What's the sum of these numbers?
Output: 11,079

Input: 91

Question: Is the input prime?

Output: $91 = 7 \cdot 13$. No

Input: 

Question: How many triangles in the input

Output: 3

Input: [19, 3, 9, 11]

Question: What does the input look like in sorted order?

Output: [3, 9, 11, 19]

What is computation?

- Answering well-defined questions
- Computing functions
- Solving problems

CS theory: how hard is the question?

↳ refers to some "computer."

↳ use math to formally define some "computers."

Computability — can we write a program that solves this question?

Complexity — If a problem is computable, how complicated is the best/simplest program to solve it?

(- length? - time (steps)? memory?)

What math? Whatever it takes to get interesting answers.

1.1 Digression on Formal Science & "Tinkering"

Science

"What's going on?"

empirical ↙ ↘ formal

Look at stuff	Invent concepts, symbols, rules that seem useful
↓	↓
generalize observations explain	Prove new conclusions
↓	↓
hope reality matches our predictions.	hope conclusions hold up in analogous real world structures.

<u>CST:</u> formal science applied to computation.

1.2 An impossible program

<u>Question:</u> can we write a program that <u>enumerates</u> all the numbers in a certain set S?

runs (potentially forever), eventually writes down any number you care to choose from the set.

$S = \{a, b, c\}$

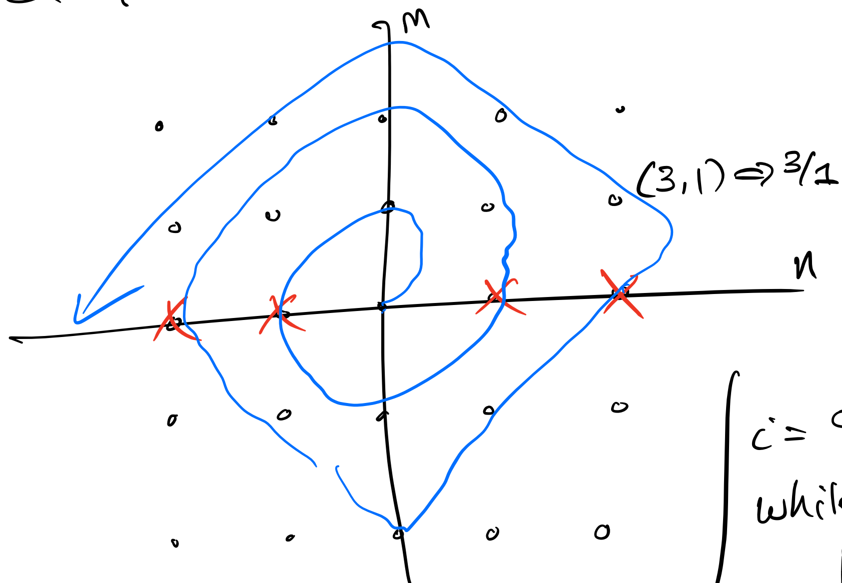
Can we enumerate $\mathbb{N} = \{1, 2, 3 \dots\}$?

```
i := 0
while true:
    print i, -i
    i := i+1
```

Can we enumerate $\mathbb{Z} = \{\dots -2, -1, 0, 1, 2 \dots\}$?

_____ " _____ $\mathbb{Q}$?

$\mathbb{Q}: \{m/n,\ m, n \in \mathbb{Z},\ n \neq 0\}$




```
i = 0/1
while true:
    print i
    i = spiral(i)
```

Theorem: (Cantor 1891)

You can't enumerate the reals $\mathbb{R}$ (on $[0,1]$.)

Proof: suppose for contradiction that some program P enumerates $[0, 1]$.

Example output (assume that printing infinite decimals ok).

$S_1 = 0.5000000\cdots$
$S_2 = 0.111111\cdots$
$S_3 = 0.1234512\cdots$
$S_4 = 0.7770000\cdots$
$S_5 = 0.182596$
$\vdots$

Define r as follows:
- concatenate the n^{th} digit of S_n, for all n

$$0.1309\cdots$$

- change each digit.

$$r = 0.2410\cdots$$

Do we ever print r?
For all m, $S_m \neq r$, because r and m differ on the m^{th} digit.

$\therefore$ r is not printed by our program P. (contradiction)
$\therefore \nexists$ any program P that enumerates the reals.

---

2. Nuts & Bolts.

Goals / Learning Objectives:

- how do I formally encode a concept / pattern / etc?

- how complicated is an objects procedure, or a problem?

Skills:

- discrete math/TCS primitives.
- building/using automata
- proof work.

Syllabus.

{we looked over the syllabus/website}

How I'd navigate the course:

- come to class, take notes
- take a stab at problem sets.
- use videos/textbook as necessary
- (Ed, TAs, O. hours)

- Emergency/big failure/something else
  Talk to course staff ASAP!

— 5 min —— 2:17 pm.

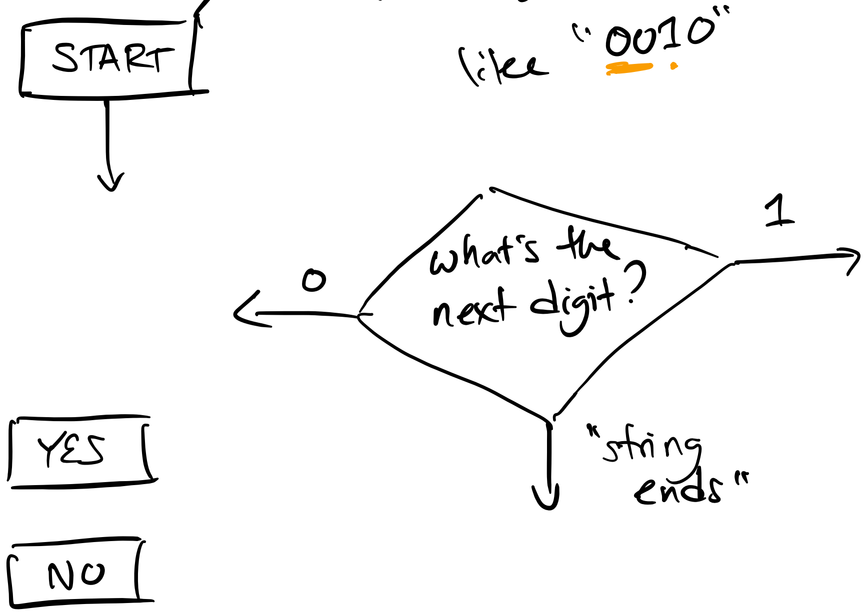
$\forall$ enumerators P, define $s^P := (s_1^P, s_2^P \dots)$.

$\forall P, \exists r_P$ s.t. $r_P \neq s_m^P$ for any $m \in \mathbb{N}$.

Math machines

Puzzle.

// you should be thinking of a binary string like "0010"



1: build a flowchart that says "YES" if and only if it has no 1's.

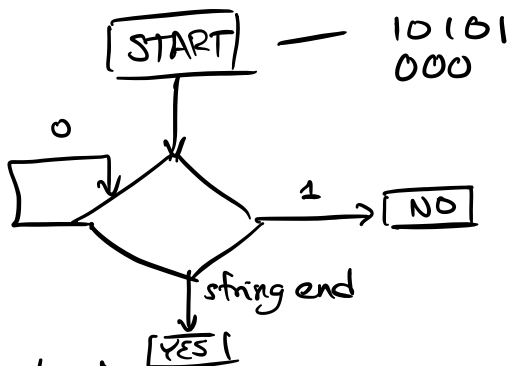
2: —— " —— YES if and only if the string has an even # of digits.

3: —— " ———— YES if the string can be made by concatenating copies of the strings "010" and "101"

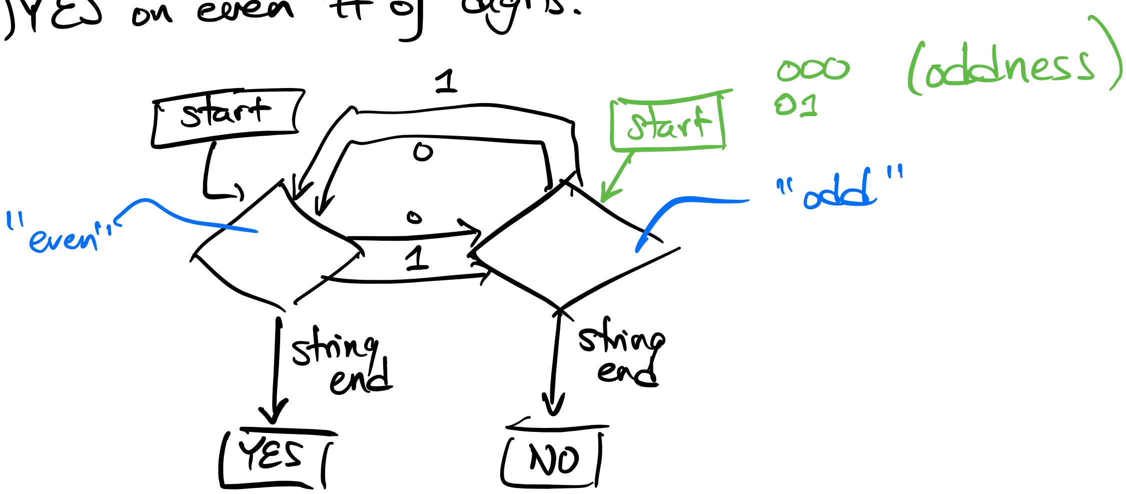
(ex 010, 101101, 010101 ...)

4: YES on palindromes (0110, 111, 01010...)

1) YES if no ones



2) YES on even # of digits.



3) skipped.

4) provably impossible.

3.2 Primitives

Def. Alphabet := non-empty, finite set.

$\{0, 1\}$ "symbol" "character"

$\{a, b, c\}$

$\{a, \dots, z\}$ $\{☺, □, ○\}$

$\{0, 1, \dots, 9\}$

Def. String := finite sequence of symbols from/over an alphabet.

01101, 0, 11

937, 86

cat, dog, ...

Def. ε (ϵ) is a special symbol for the empty string "".

String operations — $|w| \to$ string length

$w^R \to w$ reversed

$wx \to w$ concatenated with x $(w \circ x)$

$\{0,1\}^k \to$ all strings of k concatenated characters from this alphabet.

$\{0,1\}^3 = \{000, 001, 010, 100, 110, 101, 011, 111\}$

$1^3 = 111$

$a^3 = aaa$

$w = 0110$, $wx = 0110101$
$x = 101$, $xw = 1010110$

$\{a, b, c\} = \{c, b, a\}$ set
vs.
$(a, b, c) \neq (c, b, a)$ sequence.

<u>cartesian product $\times$</u>

sets A, B. $A \times B$ = the set of all tuples containing one from A, one from B

$A = \{0, 1\}$
$B = \{a, b, c\}$
$A \times B = \{(0,a), (0,b), (0,c), (1,a), (1,b), (1,c)\}$

$\mathbb{R}^2$ $\mathbb{Z}^2$ $\mathbb{R} \times \mathbb{R}$

<u>Def.</u> A language is a (possibly infinite) set of strings.

$\{0, 1, 11, 010\}$
$\{0,1\}^{10}$

$\{x \mid x$ is a string over $\{0,1\}$ that has two 1's$\}$

$\{x \mid x \in \underbrace{\{0,1\}^*}_{\text{"all binary strings"}}$ and $|x|$ even$\}$

$\{x \mid x$ is over $\{a, b, \ldots z\}$ and x is in my dictionary$\}$

$\{x \mid x$ is over $\{0, 1, \ldots 9\}$ and x is a Senator's phone #$\}$

$\{x \mid x$ is a string over UNICODE and x is a syntactically correct C program$\}$

$\{a, b, c, n \mid$ — " — $a^n + b^n = c^n, n > 2\}$

$\{x \mid x$ is a "complete proof" that the preceding language is empty$\}$

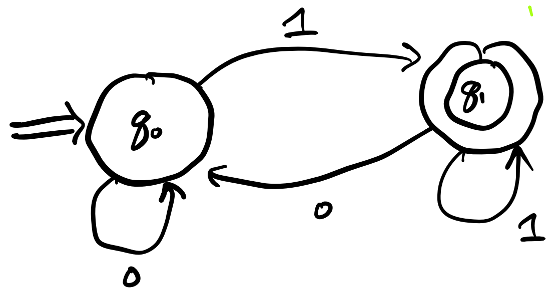
——— 3 min ———

3:14 pm

3.3 Deterministic Finite Automaton (DFA)

DFAs read in strings, char by char, from a certain alphabet, and accept or reject.

Example: alphabet: $\{0, 1\}$



- start at $\Rightarrow$
- accept if we finish ◎

test:

input	output
01001	✓
111	✓
000	✗
ε	✗

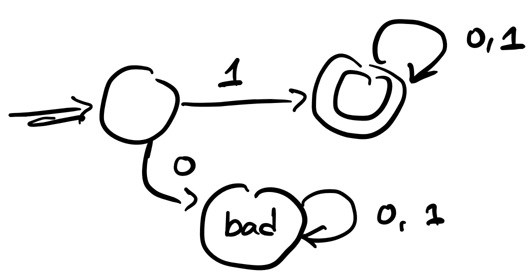
rule: accept strings that end in 1.

D recognizes the language $\{x \mid x \in \{0,1\}^*$ and x ends in $1\}$

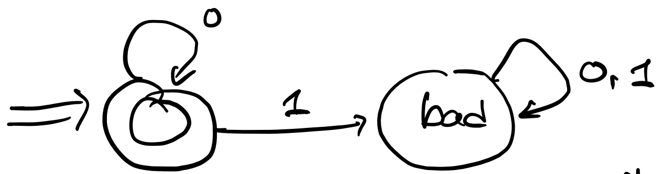
Def. rules for DFA state diagrams.

- must have a start state ($\Rightarrow$O)
- ≥ 0 accept states.
- an arrow/transition for every alphabet symbol, from every state.

A DFA state diagram accepts if and only if it is in an accept state after reading in all chars one by one, left to right.



"all strings starting with 1"

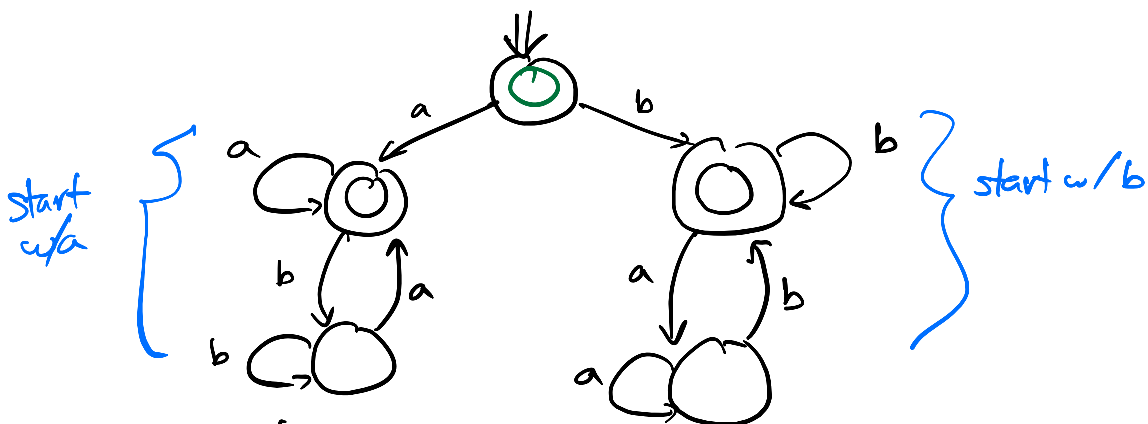


"all strings with no ones"

input	output
01	✗
00	✓
1	✗
ε	✓

Example:

Alphabet: $\{a, b\}$



in	out
abba	✓
ε	✗ / ✓

Puzzles

Goal: Build a DFA over $\{0,1\}$ that accepts strings of odd length

(*) Goal: Build a DFA over $\{0, 1, 2\}$ that accepts if all digits sum to 0 mod 3.

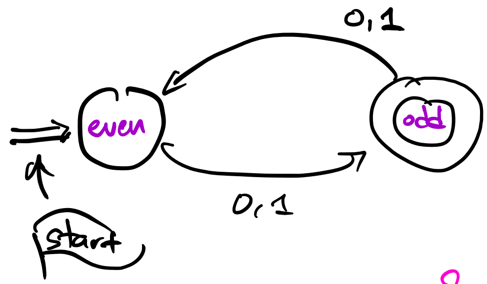
$\varepsilon \to$ yes or your choice

divisible by 3 $\{0, 3, 6, 9 \ldots\}$

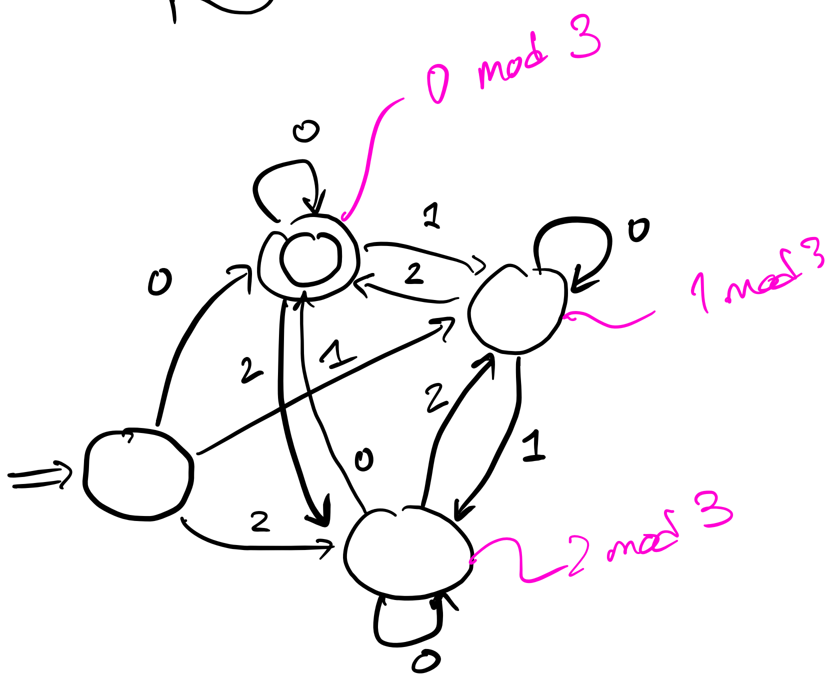
(**) Goal 3: DFA over $\{0,1\}$ that accepts if

- the string starts, ends in 0
- AND the string has even length.

"odd length"



"0 mod 3"



Goal 3: exercise.

Def. { DFA. A Deterministic Finite Automaton is a 5-tuple as follows:

$(Q, \Sigma, q_0, F, \delta)$

Q is a set of states $\{q_0, q_1\}$

start state

Σ is an alphabet $\{0, 1\}$

set of accept states $\emptyset, \{q_1\}, \{q_0, q_1\}$

δ: transition function

$\delta : Q \times \Sigma \longrightarrow Q$ that tells you where to go

$(q_0, 0) \longrightarrow q_1$

	0	1
q_0	q_1	q_1
q_1	q_0	q_1

"Let D be a DFA $(Q, \Sigma, q_0, F, \delta)$, with $Q = \{ \ldots \}$, $\Sigma = \ldots$

Def. DFA acceptance.

Let $D = (Q, \Sigma, q_0, F, \delta)$ be a DFA

D accepts the string $w = w_1 w_2 \cdots w_{n-1}$, if there is some sequence of states $r_0, r_1, \ldots, r_n$

s.t. $r_0 = q_0$,

$\delta(r_0, w_0) = r_1$, $\delta(r_1, w_1) = r_2$,

and $r_n \in F$.

<u>Def.</u> Any language recognized by some DFA is <u>regular</u>.