COMS 3261 - CS Theory

Summer A 2023

Tim Randolph (Tim)

twrand.github.io/3261-sum23.html

__Today:__

1. What is CS theory?
2. Course Nuts & Bolts
3. Automata: Math Machines
4. Discussion: Course Structure & Assessment

---

1. What is CS theory?

Using $\underbrace{\text{math}}_{?}$ to learn about $\underbrace{\text{computational problems}}_{?}$.

Problems vs. Instances

Problems: a well defined task, usually requiring some input.

Instances: a specified input for a problem

Problem: find the sum of two numbers.
Input: 1682, 9387
Output: 11,079

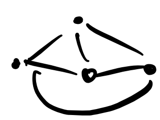
$f_{ADD} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$

$f_{square} : \mathbb{R} \to \mathbb{R}$

P2: find the prime factorization of input
In: 91
Out 7, 13

P3: Sort an input list
In: [18, 3, 12, 9]
Out: [3, 9, 12, 18]

P4: Count all triangles in a graph
In:



Out: ~~3~~ 4

Computation $\tilde{:=}$ solving problems

"how hard is computing solutions to this prob?"

"are these two problems <u>related</u>, and how?"

"does the computer/model of computation matter?"

What math? Whatever it takes.

Science

What's going on?

empirical science

Look at stuff

physics
bio
chem

↓

Generalize observations

↓

Make predictions.
Hope they come true.

formal science

math
philosophy

Invent concepts, symbols, systems, rules

↓

Prove stuff

↓

Hope conclusions apply to the real world.

TCS

1.2 - An impossible program

Q1) Can we write a program that enumerates all the natural numbers $\mathbb{N}$?

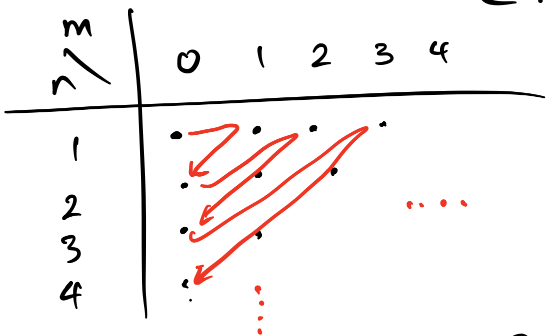
```
i := 1
while true:
    print i
    i := i+1
```

Q2) Can we enumerate $\mathbb{Z}$?

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

Q3) Can we enumerate $\mathbb{Q}$?

m/n



Q4) Can we enumerate $\mathbb{R}$? No.

<u>Theorem:</u> (Cantor, 1891) You can't have a program that enumerates $\mathbb{R}$.

(Given any program that outputs an infinite sequence of real numbers, there is some real not in the sequence.)

Example program output: (on $[0,1]$)

$0.\cancel{1}\,0\,0\,0\,0 \cdots$

$0.\,1\,\cancel{0}\,0\,1\,0 \cdots$

$0.\,1\,3\,\cancel{9}\,7\,5 \cdots$ $\Rightarrow$ $\underline{0.21037}\cdots$

$0.\,2\,1\,6\,\cancel{2}\,8 \cdots$

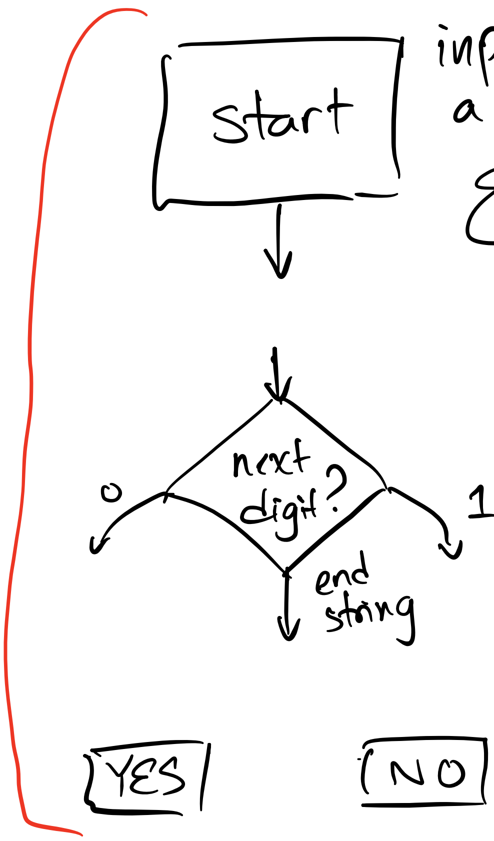
$0.\,2\,3\,8\,4\,\cancel{6} \cdots$

We'll construct a real number not in the output of this program.

Conclusion: given any program that outputs an infinite sequence of reals, there exists a real not in the sequence. $\therefore$

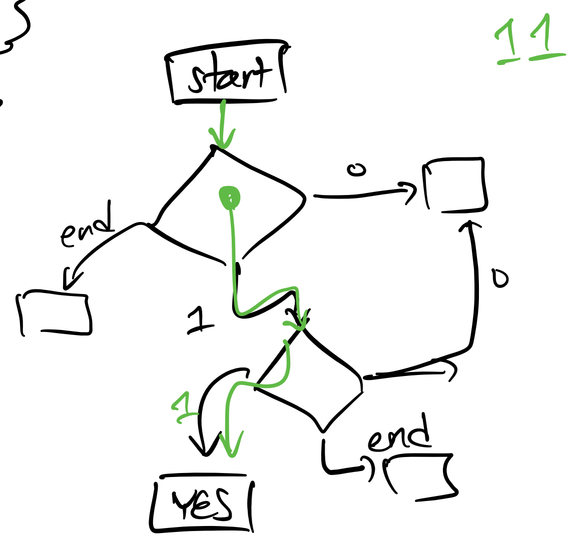
$$\begin{array}{r} \cancel{0} \\ \cancel{0}\,1 \\ \cancel{0}\,0\,2 \\ \cancel{0}\,0\,0\,3 \\ \cancel{0}\,0\,0\,0\,4 \end{array} \Bigg| = 111\cdots$$

Back at 2:15



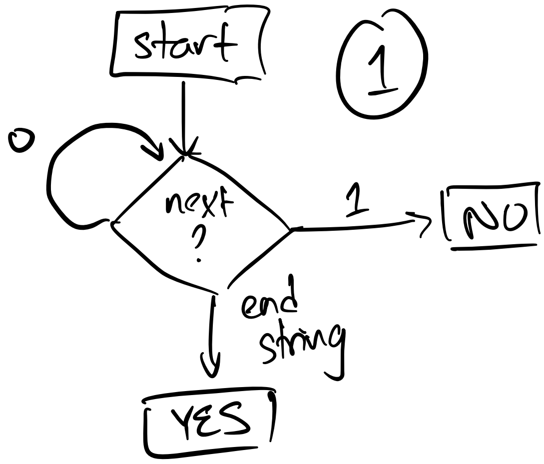
input:
a binary string

10110



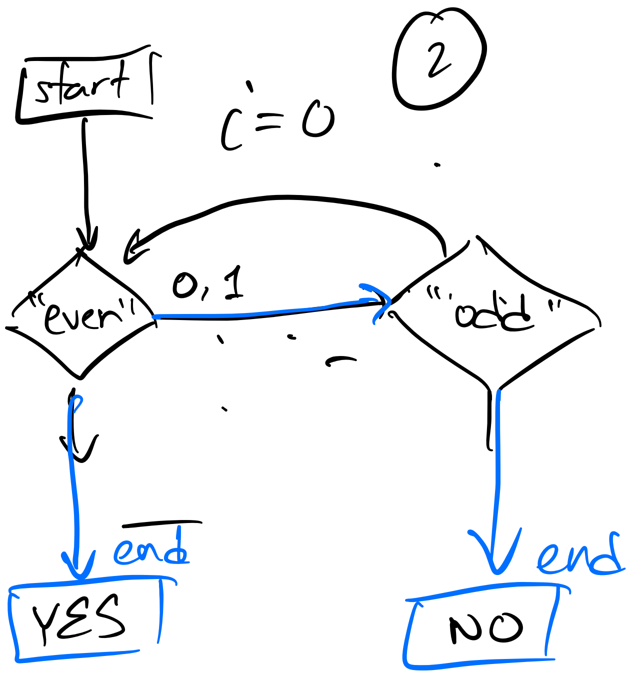
With these pieces:

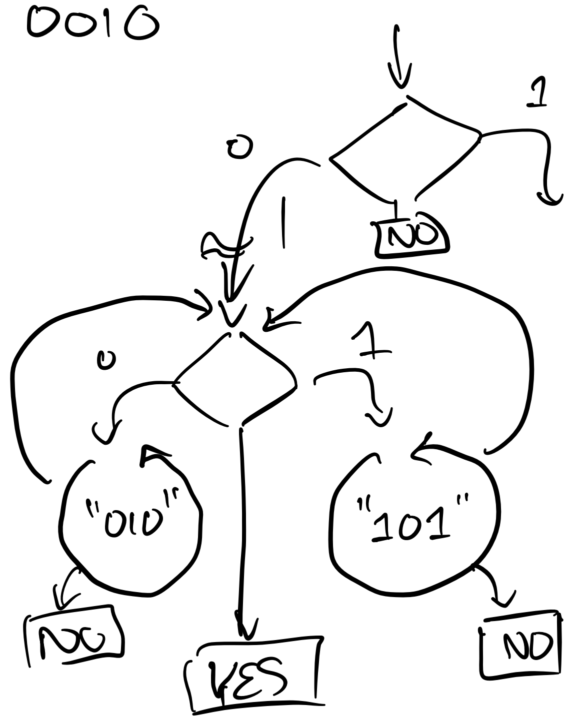
1) Build a flow chart that accepts a binary string if and only if it has no 1's.
2) Build a flow chart that accepts only strings with an even number of digits.
3) Build a flow chart that accepts strings divisible into the substrings "101" and "010" (e.g. 101101 010101)
4) Build a flow chart that accepts palindromes.



00
0000
0

0010





Def. Alphabet = finite, nonempty set (of "characters").

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

$\{0, 1, \ldots, 9, A, \ldots F\}$

$\{a, b, \ldots z\}$

$\{\square, \triangle, ☺\}$

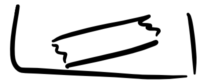
Def. String := finite sequence of characters from/over an alphabet.

* Special: ε is a special symbol for "", empty string.

$\emptyset = \{\} =$ empty set

$\varepsilon =$ empty string

$\{\{\emptyset\}, \emptyset\}$

$\{\varepsilon\} =$ 

$\{\emptyset\} = \{\{\}\}$

$\{\{\varepsilon\}\}$

$\{\emptyset, \varepsilon\}$

ε

$010\varepsilon = 010$

<u>string operators</u>

Let w be some string

$|w|$ is the length (# of characters) in w.

w^R is w "reversed"

$(cat)^R = tac$

For w, x strings, wx is concatenation

011 000

011000

$\{0,1\}^k$ = all the strings consisting of k characters from this alphabet.

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

Def. Language := a (possibly infinite) set of strings.

$\{0, 1, 11, 01100, 101010\}$

$\{x \mid x$ is a string over $\{0,1\}$ of even length$\}$

$\{x \mid x$ is a string over $\{0, 1, \dots 9\}$ and:

- x is prime $\}$
- $x =$ tomorrow's winning lotto number $\}$
- $x = a^n + b^n$ for $n > 2$, a, b, integers $\}$

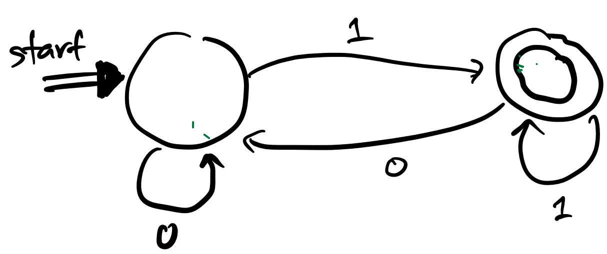
languages $\approx$ concepts

deciding if $x \in$ language L $\approx$ recognizing concept

3.3 DFA (Deterministic Finite Automaton)

A DFA is a machine that takes an input string (from a certain alphabet), reads it one character at a time and accepts / rejects.

On $\{0, 1\}$, an example:



$\Rightarrow$ "start"

◎ "accept state"

1010 111

ε
1011
001
0010

Def. A DFA state diagram has:

- exactly one start state, marked by $\Rightarrow$
- transitions from every state, on every character of a given alphabet

- (optionally) some accept states ◎

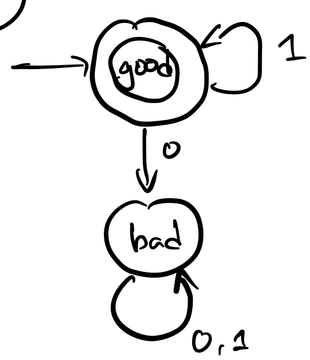
<u>Def.</u> The set of all strings accepted (or "recognized") by a DFA is the language of that DFA.

~ Back at 3:20 ~ (Over $\{0,1\}$)

1) Build a DFA that accepts binary strings with no 0's
2) Build a DFA that accepts strings with length divisible by 3.
3) Build a DFA over the alphabet $\{0,1,2\}$ that accepts if the <u>sum</u> of the digits is 0 mod 3
4) Build a DFA over $\{0,1\}$ that accepts if and only if the string (1) starts and ends with 0
(2) has even length

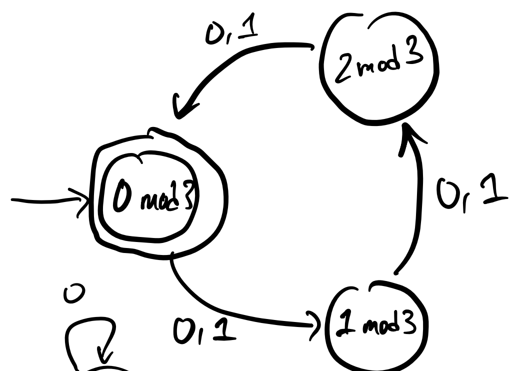
1

ε — no zeroes

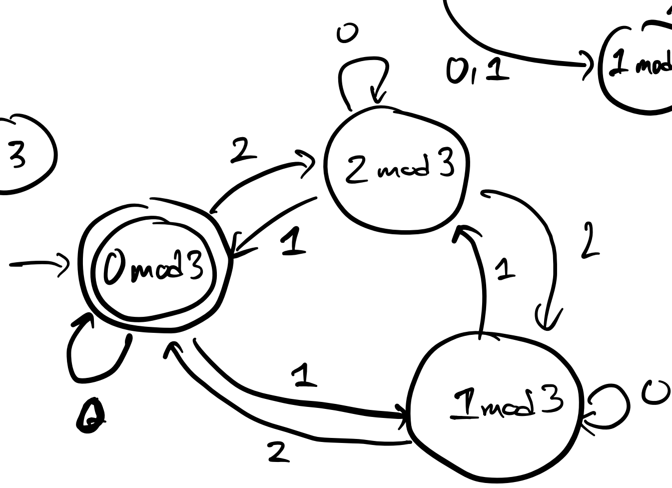


1110

2



3

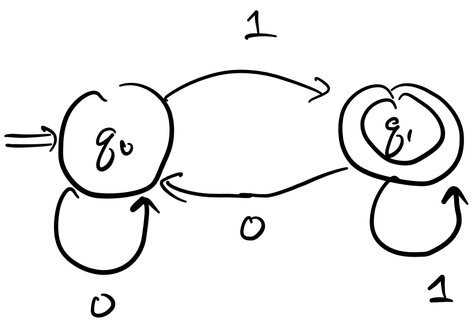


0

1

Def: DFA (math). A DFA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, with the following parts:

- Q is a finite set of states
- Σ is an alphabet
- q_0 is the name of the start state
- $F \subseteq Q$ is the set of accept states
- $\delta : Q \times \Sigma \longrightarrow Q$ is a transition function.



$=$ $\{Q, \Sigma, \delta, q_0, F\}$

$Q = \{q_0, q_1\}$

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

$F = \{q_1\}$

δ: $\delta(q_0, 0) = q_0$
$\delta(q_0, 1) = q_1$
$\delta(q_1, 0) = q_0$
$\delta(q_1, 1) = q_1$

$=$

	0	1
q_0	q_0	q_1
q_1	q_0	q_1

Def.

DFA acceptance: If $M = (Q, \Sigma, \delta, q_0, F)$ is a DFA and $w = \underline{w_0 w_1 \cdots w_{n-1}}$ is an n-digit string, with each $w_i \in \Sigma$, then M accepts w if there is a sequence of states $\underline{r_0, r_1, \cdots r_n} \in Q$ such that 1) $r_0 = q_0$

w_0 w_1

degenerate case: if $|r|=0$, no requirement.

2) $\delta(r_0, w_0) = r_1$, $\delta(r_1, w_1) = r_2, \ldots$
$\delta(r_{n-1}, w_{n-1}) = r_n$

3) $r_n \in F$.

limiting case: accepting ε

$w = \varepsilon$

trivial: "each $w_i \in \Sigma$"

sequence: r_0

$r_0 = q_0$

Def. The set of languages recognized by some DFA is the "regular languages."

To do:

- read course webpage
- fill out survey (today!)
- HW 1 up soon, due Mon.

Solution to DFA puzzle 4, above: