COMS 3261 - Summer B 2021

Lecture 1

Tim Randolph

417 IAB (usually)

twrand.github.io/3261.html
- Ed
- Lectures
- Reading
- Homework
- GradeScope

Today:

1. What is this course about?
2. Course Structure
3. Syllabus
4. Strings, Languages, 'Concept recognition'
5. Deterministic Finite Automata (DFA)
6. Regular operations
7. Reading + Homework.

1. What is this course about?

what questions?

Using math to answer fundamental questions about computation

why math?
what math?

what is computation?

Empirical Science

1. What's going on?
2. Looking at stuff (doing experiments)
3. Organize observations into & explanations
4. Hope these explanations are <u>helpful</u> & <u>predictive</u>

Formal Science

1. What's going on?
2. Inventing concepts, symbols, formal systems.
3. Prove/reason new conclusions
4. Hope conclusions help you in the real world.

TCS := formal science for computers.

How can math teach us about computation? An example.

<u>Theorem</u>. (Cantor, 1891.) You can't <u>enumerate</u> the real numbers $\mathbb{R}$.

describe some finite rule writing them all down in some order.

$\mathbb{N} := \{1, 2, \ldots\}$

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

$\mathbb{Z} := \{\ldots -2, -1, 0, 1, 2 \ldots\}$

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

<u>Proof</u>. (Impossible for $\mathbb{R}$.)

there ~~are~~ exist

Suppose for contradiction $\exists$ some rule for enumerating the reals. Consider the order of reals output by this rule.

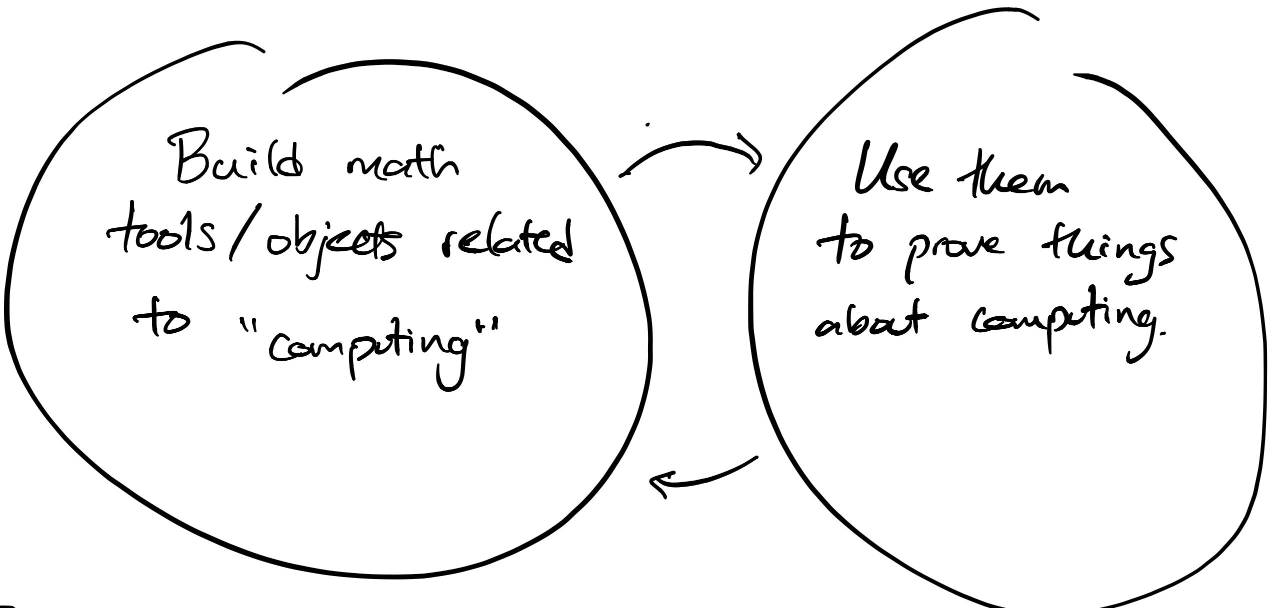
For example:

$$
\begin{array}{l}
\cancel{0}.0100\cdots \\
0.\cancel{9}162\cdots \\
0.0\cancel{3}33\cdots \\
0.00\cancel{0}1\cdots \\
\vdots
\end{array}
\qquad r = 1.041\ldots
$$

Create a new real r by incrementing the n^{th} digit of the n^{th} number for all numbers in our sequence.

Our new number differs from every real in our infinite sequence at at least 1 decimal place. Contradiction $\rightarrow \mathbb{R}$ cannot be enumerated. $\square$

2. Course Structure



Tools: Automata — 'concept recognizers'
- DFA
- NFA
- Pushdown Automata
- Turing Machines

Grammars — 'concept definers'
- Regular Expressions
- CFG

Decidability — "Can a computer answer this sort of YES/NO question?"

Reducibility — "If I can solve problem A, then I can solve problem B."

Complexity — If a problem A is doable, how hard is A?

resources:
time
memory
randomness
quantumness?

3. Nuts & Bolts (Syllabus)

on the website:

People: me, TAs: Eumin, Annie, Quintus, Elena, Brian.

(Office hours, emails)

Dates/time —

Modality — In-person (IAB 417)
Zoom livestream
Asynchronous recordings (YT)

Homework: 6 problem sets — assigned Mondays covering through that Weds
due Mondays @ 11:59 PM EST.

Late policy: 3 non-splittable late days
use whenever up to Friday @ 11:59 PM.
-20% off total grade once late days are used.

Collaboration:

Problem sets: open textbook, notes, reference websites

Wikipedia
NOT Q&A sites.

Collaboration OK (encouraged during office hrs.)
But own write-ups only.
No sharing written solutions/solution notes.

Exam: 1 exam. Probably be any a contiguous 12 hours during a set 48-hour period. (Likely 48-hr period of August 10-11.)

$$\begin{array}{lr} 6 \text{ problem sets} & \times 12\% \\ + \; 1 \text{ exam} & \times 28\% \\ \hline & 100\% \end{array}$$

CSB (Mudd) 522.

↳ Down hall, up the stairs end of hall.

4. 'Concepts' — Strings & Languages

Def. alphabet := nonempty finite set of symbols /characters.

Often — will use symbols Σ or Γ

Examples. $\{0, 1\}$ – Binary

$\{a, b, c, \ldots \quad z\}$ — Roman

$\{0, 1, 2, \ldots 9\}$ — Decimal

$\{\square, \triangle, ☺, \text{WORD}, \quad \}$

Def. string := finite sequence of symbols from a given alphabet.

↳ $(\;, \;, \;, \;)$

010, 00110, 1001111, — Binary

cat, dog, zxxya

ε := "empty string" " "

Let w and x be strings.

$|w|$ — size / length

w^R — reverse. cat^R = tac

wx — concatenation catdog 10, concat. with 3 103

<u>Def.</u> Lexicographic order ≈ dictionary / alphabetical order.

Sort by first ~~letter~~, symbol then second symbol, and so on.

(empty symbol comes first)

$\{\varepsilon$, bat, dog, cat, aa, a, ba$\}$

$\Downarrow$

ε, a, aa, ba, bat, cat, dog.

$\{111, 10, 0, 00, \varepsilon, 1\}$

$\Downarrow$

ε, 0, 0, 1, 111, 10 0 < 1.

$a\varepsilon = a$

aba ($\varepsilon \cup b$) bba

$\Downarrow$

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

Examples. $\{0, 1, 11, 10, 111\}$. ✱ not include 01

$\{0,1\}^k$:= a string of k symbols from the set $\{0,1\}$.

$\{x \mid x$ contains at least one $0\}$ (on $\Sigma = \{0,1\}$)

(such that)

$\{x \mid x$ is in my English dictionary$\}$ Σ = Roman

(*) $\{x \mid x$ is the decimal representation of a prime number$\}$ Σ = decimal

Define languages L_1 and L_2.

$L_4 := L_1 \cup L_2$ $\quad L_3 := \{wx \mid w \in L_1, x \in L_2\}$

$L_1 = \{00, 11\}$
$L_2 = \{1, 11\}$

L_3: $\{001, 0011, 111, 1111\}$

Idea: Languages are 'like' concepts.

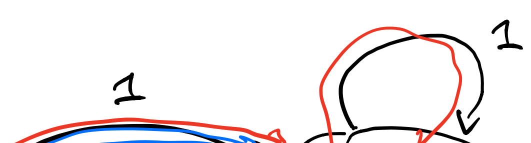
Idea: Being able to tell if a string w is in a language L is $\approx$ being able to recognize the 'concept' L.

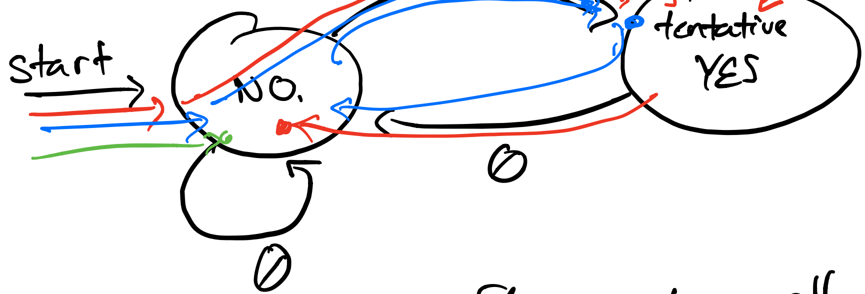
Language-recognizing machines!

Idea: Build a machine that decides whether or not a string is in a language.

↳ Machine will consider an input string symbol by symbol, and say 'YES/NO' (accept/reject) at the end.

Alphabet $\Sigma = \{0,1\}$. Language: $\{w \mid w$ ends in '1'!$\}$





Idea: where we end up after reading all characters is our YES / NO.

Tests: 110 — NO. ε — NO.
101 — YES.

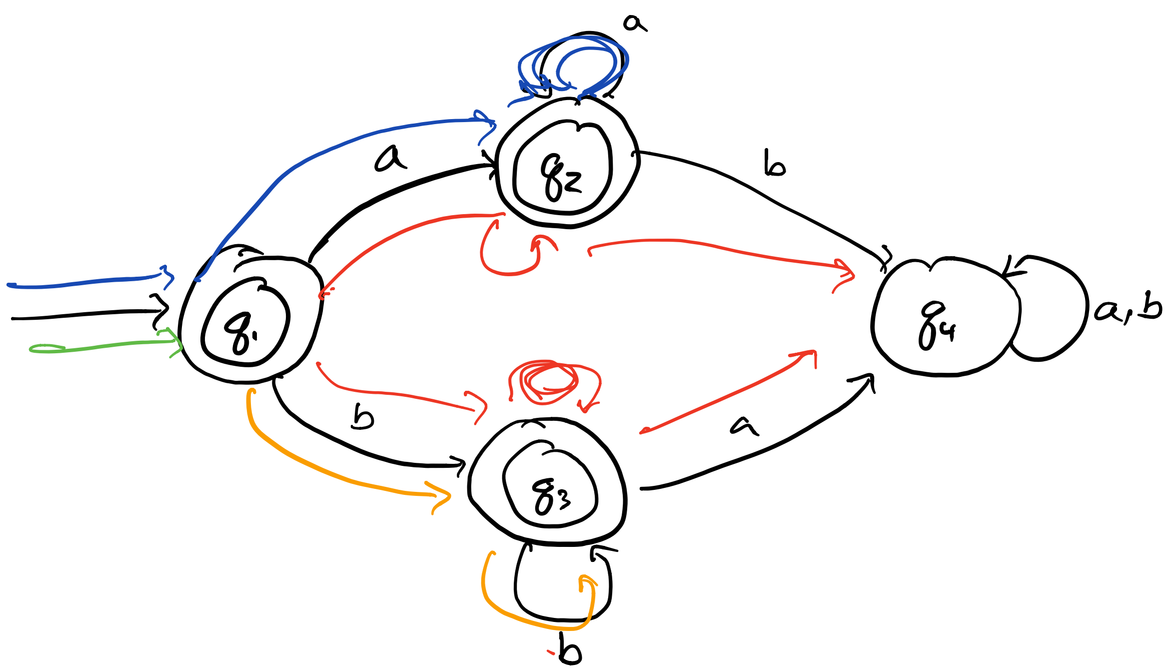
$\{\varepsilon\}$ is a language! not the same as $\{\} = \emptyset$

<u>Def.</u> A <u>state diagram</u> contains:

- start state (mark by an orphan arrow $\to$O)
- zero or more 'YES' or accept states. (inner circle ◎)
- a transition (arrow $\longrightarrow$) indicating what to do at every state, for every symbol.

A state diagram <u>accepts</u> a string iff it is in an accept state after the last symbol is read.

<u>Example 2.</u> $\Sigma = \{a, b\}$



Tests: bbbb — ✓   aaaa — ✓

aab, bbbbab — ✗

ε — ✓

Automaton

Automata

Def. A (Deterministic) Finite Automata (DFAs.)

A DFA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where:

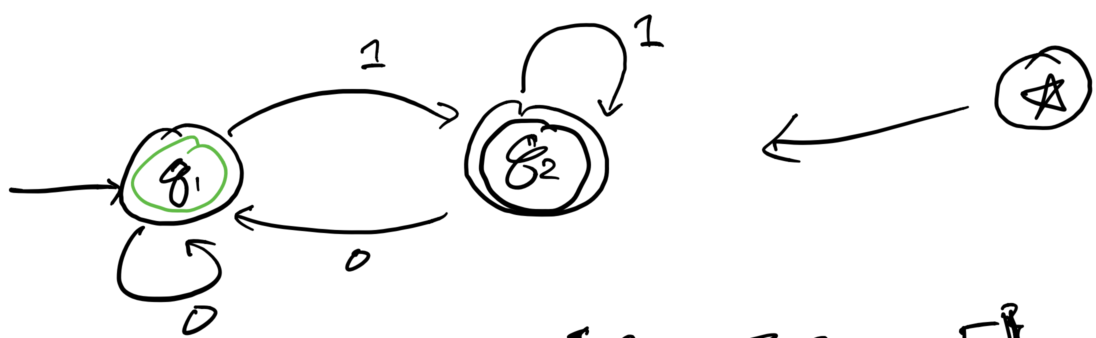
$Q :=$ finite set of states

$\Sigma :=$ finite alphabet of symbols.

$\delta : Q \times \Sigma \longrightarrow Q$. A function that says 'give me a state in Q and a symbol in Σ, and I'll tell you which state in Q to go to next.'

q_0 : a start state

F : A subset of Q, a (possibly empty) set of accept states.



Call this machine $M = \{Q, \Sigma, \delta, q_0, F\}$

$Q = \{q_1, q_2\}$

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

$q_0 = q_1$

$F = \{q_2\} \longrightarrow \{q_1, q_2\}$

$\delta =$

	0	1
q_1	q_1	q_2
q_2	q_1	q_2

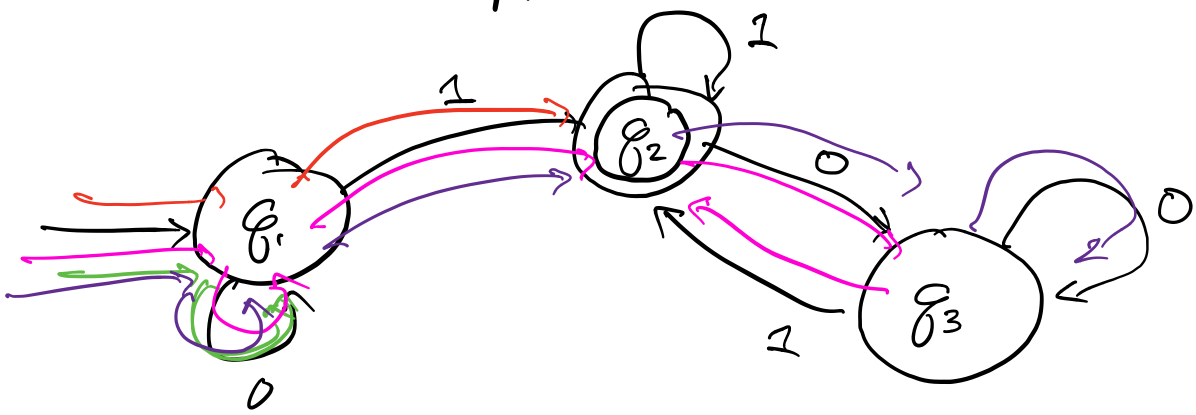
$M_1 = (Q, \Sigma, \delta, q_1, F)$, where

$Q = \{q_1, q_2, q_3\}$ $\Sigma = \{0, 1\}$ $F = \{q_2\}$

$Q = \{q_1, q_2, q_3\}$. $\Sigma = \{0, 1\}$, $F = \{q_2\}$

δ:

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_3	q_2



00 — ✗ 01 — ✓

0101 — ✓ 0100 — ✗

This is still the machine that accepts binary strings that end in 1. (Different machines can do the same thing.)

Def. Let M be a DFA. $L(M)$ is defined to be the set of all strings that M accepts — the <u>language of M</u>.
We also say 'M <u>recognizes</u> the language $L(M)$.'
(M "accepts" $L(M)$).

Def. (Accepting a string – formal.) Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. Let $w_0 w_1 w_2 \cdots w_{n-1}$ be a string where each symbol $w_i \in \Sigma$. Then, M <u>accepts</u> the string $w_0 w_1 \cdots w_{n-1}$ if there exists a sequence of states $r_0, r_1, \ldots, r_n \in Q$ satisfying

$$r_0 = q_0$$

$\delta(r_i, w_i) = r_{i+1}$, for $i = 0, 1, 2, \dots n-1$.

$r_n \in F$.

Zoom out.

Languages are sets of strings.

Languages are $\approx$ mathematical concepts.

DFAs (either specified by a 5-tuple, or by a state diagram) specify a procedure for deciding whether a string is in a language — a way to recognize that language.

Where we're going: complexity of the automata required to recognize language L $\approx$ "complexity" of that concept

$\approx$ difficulty of answering the yes/no question: "Is $w \in L$"?

Def. A language is called regular if some DFA recognizes it. (Not all languages are regular.)