$A = \{ a^i b^j \mid i,j \geq 1$ and i evenly divides $j \}$

$B = \{ \langle M, w \rangle \mid M$ encodes a TM, and w is a string; either (1) $w \in A$ and $M(w)$ accepts or (2) $w \notin A$ and $M(w)$ rejects. $\}$

Puzzle:

- Design a TM that decides A. (high-level OK)
- — " — that ~~decides~~ recognizes B.

(At the "high level", TMs ≈ programs.)

Standard: conversion to a program given directions is unambiguous, reasonal for a competent programmer

Stumbling blocks: missing cases, ∞ loops

Decider for A: M_A = "On input w: (NOT high-level)

★ (1) Check that $w \in a^+b^+$ (using a hardcoded DFA)

★ (2) Shuffle back and forth, crossing out one a for each b.

- if we run out of b's before a's, reject.
- if we run out of a's before b's, uncross all a's and repeat (2)
- if all a's, b's are crossed out, accept."

(High-level):

M_A = "On input w:
Accept if $w = a^i b^j$ for some $i,j \geq 1$ and i divides j."

$B = \{ \langle M, w \rangle \mid M$ encodes a TM, w a string
Either $w \in A$ and $M(w)$ accepts, OR
$w \notin A$ and $M(w)$ rejects. $\}$

function decideB (TM M, string w):
return true/false.

~~($w \in A$, $M(w)$ rejects or loops~~
~~$w \notin A$, $M(w)$ accepts or loops)~~

M_B = "On input x:

(1) Check that x encodes a TM M and a string ω, reject if not.

(2) Simulate $M_A(\omega)$, using a hard-coded copy of M_A.

// % # This tells us if $\omega \in A$ or not, Guaranteed to halt as M_A is a decider/ always halts.

(3) Simulate $M(\omega)$:

- if $M(\omega)$ accepts, and $\omega \in A$, accept. ($\omega \notin A$, reject.)
- if $M(\omega)$ rejects, and $\omega \notin A$, accept. ($\omega \in A$, reject)"

(- if $M(\omega)$ runs forever, we run forever...)

M_B is a recognizer, not a decider, for B.

Today: more time w/ TMs.

1. Example TMs — using regular ops/closure properties.
2. CFLs $\subseteq$ TM-decidable
3. Beyond TMs: an undecidable language.

(4. Review, final exam chat)

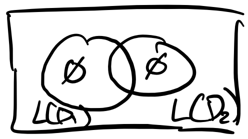
Last time:

$$E_{DFA} = \{\langle D \rangle \mid D \text{ is a DFA that rejects all strings}\}$$

- We have a decider, $M_{E_{DFA}}$, that decides this language.

$$EQ_{DFA} = \{\langle D_1, D_2 \rangle \mid D_1, D_2 \text{ are DFAs and } L(D_1) = L(D_2)\}$$

Fact/Observation: $L(D_1) = L(D_2)$ if and only if $L(D_1) \cap \overline{L(D_2)} = \emptyset$ and $\overline{L(D_1)} \cap L(D_2) = \emptyset$.



$M_{EQ_{DFA}}$ = "On input x:

(1) Check that input encodes DFAs D_1 and D_2

(2) Using our closure construction for complement, write down encoded DFAs for $\overline{L(D_1)}$, and $\overline{L(D_2)}$.

(3) Using our closure construction for intersection (a DFA with states for every pair in $Q_1 \times Q_2$), write down encoded DFAs for $L(D_1) \cap \overline{L(D_2)}$ and $L(D_2) \cap \overline{L(D_1)}$.

(4) Using a hard-coded copy of $M_{E_{DFA}}$, decide if $L(D_1) \cap \overline{L(D_2)}$ and $L(D_2) \cap \overline{L(D_1)}$ are both empty. Accept if so; otherwise, reject."

<u>Proposition.</u> CFLs $\subsetneq$ Turing-Decidable.

<u>Fact:</u> Every CFG can be written in a format called 'Chomsky Normal Form', which guarantees that the derivation of any string w takes $2|w|-1$ steps.

<u>Proof:</u> Consider

$$A_{CFG} = \{\langle G, w\rangle \mid G \text{ is a CFG in Chomsky Normal Form, } w \text{ is a string, and } G \text{ generates } w\}.$$

$M_{A_{CFG}}$ = "On input x:

(1) Check that $x = \langle G, w\rangle$, where G is a CFG in CNF, w a string.

(2) <u>Enumerate</u> every string derivation of length $2|w|-1$.

\# of derivations $\leq |R|^{2|w|-1}$

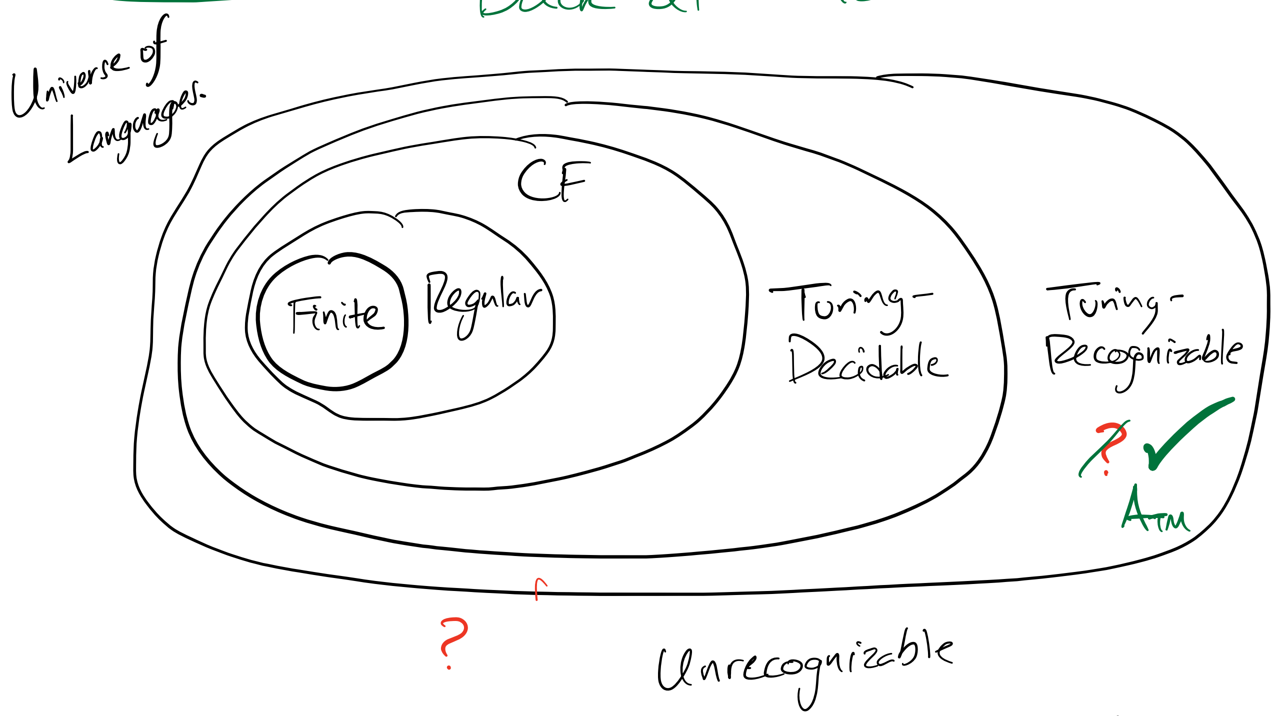
Accept if G generates w in $2|w|-1$ steps; reject otherwise."

Now, let $L(G)$ be some CFL and G be a CFG in CNF that generates $L(G)$. To decide $L(G)$:

$M_{L(G)}$ = "On input w:

(1) Use hard-coded copies of $M_{A_{CFG}}$ and G to simulate $M_{A_{CFG}}(\langle G, w\rangle)$. Accept if and only if the simulation accepts."

Back at 2:15



Paradoxes: Liar's paradox: "This statement is false."

T? F?

Russell's paradox: "The barber shaves everyone who doesn't shave themself."

TM paradox: "If M accepts w, then M rejects w.
If M rejects w, then M accepts w."

Such a machine is impossible. contradiction

Idea: we'll show that a TM that decides A_{TM} would lead to the TM paradox, so deciding A_{TM} is impossible.

$$A_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM, } w \text{ a string, and } M(w) \text{ accepts.}\}$$

Theorem. A_{TM} is undecidable.

Proof. Assume for contradiction that some TM H decides A_{TM}.

$H(\langle M, w\rangle)$ accepts if $M(w)$ accepts, and rejects otherwise — if $M(w)$ rejects or loops.

$\langle M\rangle$ is a string, so we can run $H(\langle M, \langle M\rangle\rangle)$.

$H(\langle M, \langle M\rangle\rangle)$ accepts if $M(\langle M\rangle)$ accepts, and rejects otherwise.

program strLen (str s):
    accept if $|s| = 39$

strLen("program strLen(str s) ... 39") accepts

program selfwriter()
    write ("program selfwriter ... " + recursive)

Define the TM P = " On input $\langle N\rangle$: (N a TM)
(1) Simulate $H(\langle N, \langle N\rangle\rangle)$ using a hard-coded copy of H.
(2) Accept if $H(\langle N, \langle N\rangle\rangle)$ rejects,
Reject if $H(\langle N, \langle N\rangle\rangle)$ accepts. "

$P(\langle N\rangle)$ accepts if $N(\langle N\rangle)$ doesn't accept,
$P(\langle N\rangle)$ rejects if $N(\langle N\rangle)$ accepts.

Now: Run $P(\langle P\rangle)$. $P(\langle P\rangle)$ accepts if $P(\langle P\rangle)$ doesn't accept,
$P(\langle P\rangle)$ rejects if $P(\langle P\rangle)$ accepts. ✗

Our assumption that H decides A_{TM} led to contradiction (TM paradox), so deciding A_{TM} is impossible. ∎

<u>Theorem</u>: $\overline{A_{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM, } w \text{ a string, } M(w) \text{ rejects or loops}\}$ is unrecognizable.

<u>Proof</u>: We'll show that if some TM recognizes $\overline{A_{TM}}$ (call it T), then A_{TM} is decidable, which is a contradiction.

$H_{A_{TM}}$ = " On input $\langle M, w\rangle$:
(1) Simulate $M(w)$ and $T(\langle M, w\rangle)$ "simultaneously."

one step of $M(w)$, then one step of $T(\langle M, w\rangle)$, etc.

(If $M(w)$ accepts, sim. 1 will halt.
If $M(w)$ rejects or loops, $T(\langle M, w\rangle)$ will halt and accept.)

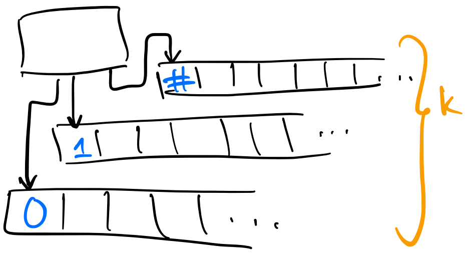
(2) If $M(w)$ accepts, accept
If $T(\langle M, w\rangle)$ accepts, reject." ☐

This decides A_{TM}, which contradicts our proof that A_{TM} is undecidable.

Back at 3:15

TM Variants & Extensions:

Multitape TMs:



Formally: k tapes

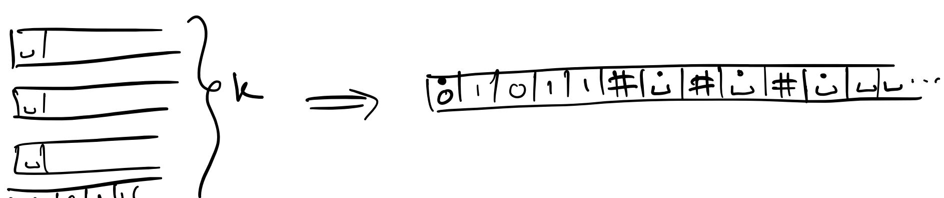
$$\delta : Q \times \underbrace{\Gamma^k}_{\text{k different tape squares}} \to Q \times \underbrace{\Gamma^k}_{\text{write down k symbols}} \times \underbrace{\{L, R\}^k}_{\text{k different movements.}}$$

$k = 3.$ $\quad \delta(q_0, 0, 1, \#) = (q_3, 1, 1, \#, R, L, R)$

Theorem. Every multitape TM has an equivalent single-tape TM.

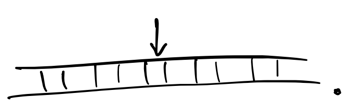
Proof sketch: Given a k-tape TM, we can simulate it with a single-tape TM as follows:

(1) Write down all contents of k tapes on our single tape, separated by delimiters. Use marks ($\dot{\ }$) to keep track of my tape heads.



(2) Simulate one step at a time. Whenever we run out of space on a tape, pause our simulation and shift it all over one square to make room. ☐

Other equivalent variants include:

- TM with two-way infinite tape .
- TM with a "stay put" operation $\{L, R, S\}$.

Today:

- Finite ⊂ Regular ⊂ CF ⊂ decidable ⊂ recognizable.
- A_{TM} undecidable ($A_{TM} = \{\langle M, w\rangle \mid M(w) \text{ accepts}\}$.)
- $\overline{A_{TM}}$ unrecognizable.
- Multitape TMs and other minor variants.

Road Map: course "crunch time."

Today: O hours Zoom 5-6.

Friday: Class same time/same place + video.
Tim O hours AM: 10 - 11:45, in-person

Weekend: Get started on HW5

Monday: Computability ⟹ complexity
P, NP, NP-completeness
$P \stackrel{?}{=} NP$

Tues PM: Review session (+ record)

Wednesday: NP-completeness reductions
Information Theory
In-person review
Course Evals.

not on final

Weds PM: Extended Zoom hours.
Mon, Weds AM: In-person O hours as usual.

Final: Available on Gscope [12:01am Thurs - 11:59pm Fri]
+ up to 11:59pm Sat if you ask me first.

Time: 12 hours, starting w/ download from Gscope.
Similar length, difficulty to p-set
Open-book, open note, open website.