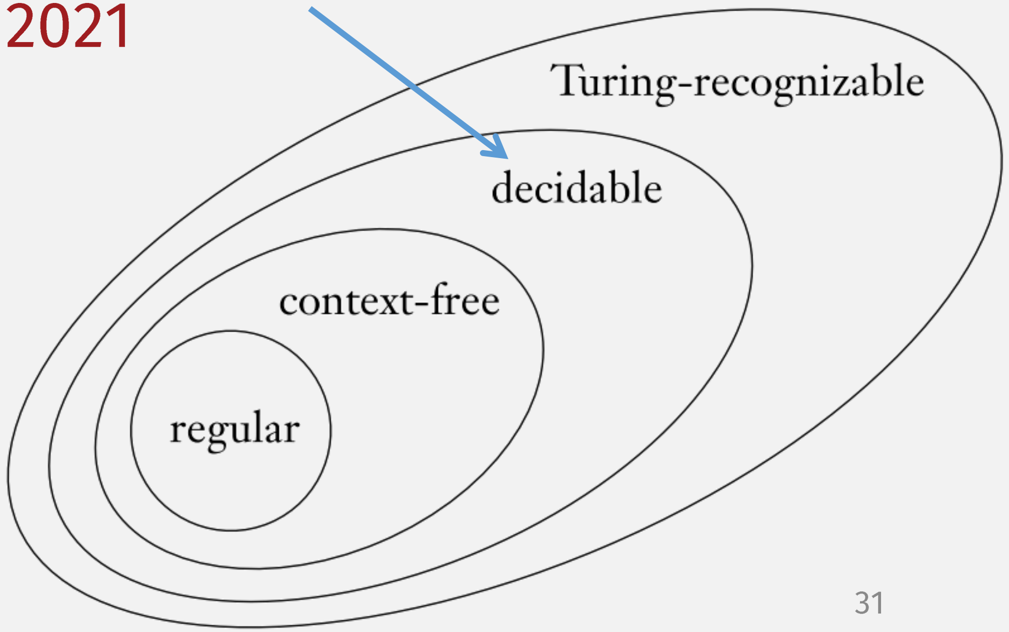
Decidable Problems (i.e., Algorithms) about Context-Free Languages (CFLs)

Monday March 29, 2021

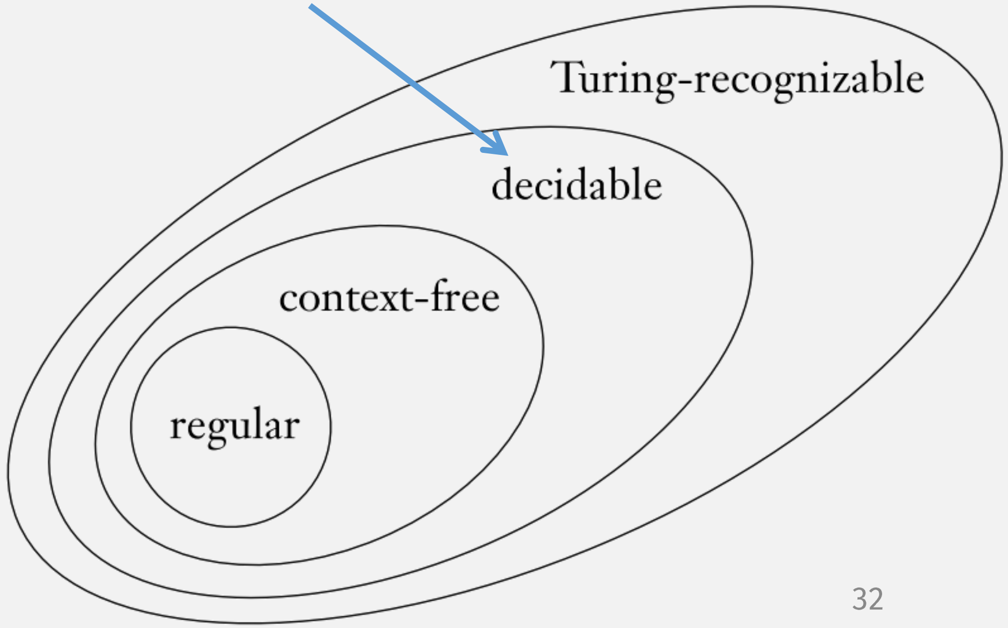


Announcements

- HW 6 due date past

- HW 7 due Sun 4/4 11:59pm EST
  - Remember to use your "library" of theorems

- HW 8 out soon
  - due Sun 4/11 11:59pm EST
  - Covers Ch 4-5 material (starting Wed)



Last time: Decidable DFA Langs (i.e., algorithms)

- $A_{\textsf{DFA}} = \{\langle B, w\rangle \mid B \text{ is a DFA that accepts input string } w\}$
- $A_{\textsf{NFA}} = \{\langle B, w\rangle \mid B \text{ is an NFA that accepts input string } w\}$
- $A_{\textsf{REX}} = \{\langle R, w\rangle \mid R \text{ is a regular expression that generates string } w\}$
- $E_{\textsf{DFA}} = \{\langle A\rangle \mid A \text{ is a DFA and } L(A) = \emptyset\}$
- $EQ_{\textsf{DFA}} = \{\langle A, B\rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}$

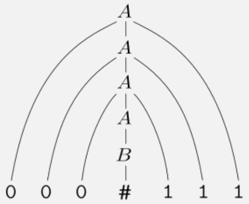
Remember:
TMs = programs
Creating TM = programming
Previous theorems = library

Thm: A_{CFG} is a decidable language

$A_{\mathsf{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG that generates string } w\}$

- This a is very practically important problem …
- … equivalent to:
  - Is there an **algorithm to <u>parse</u> a programming language** with grammar G?

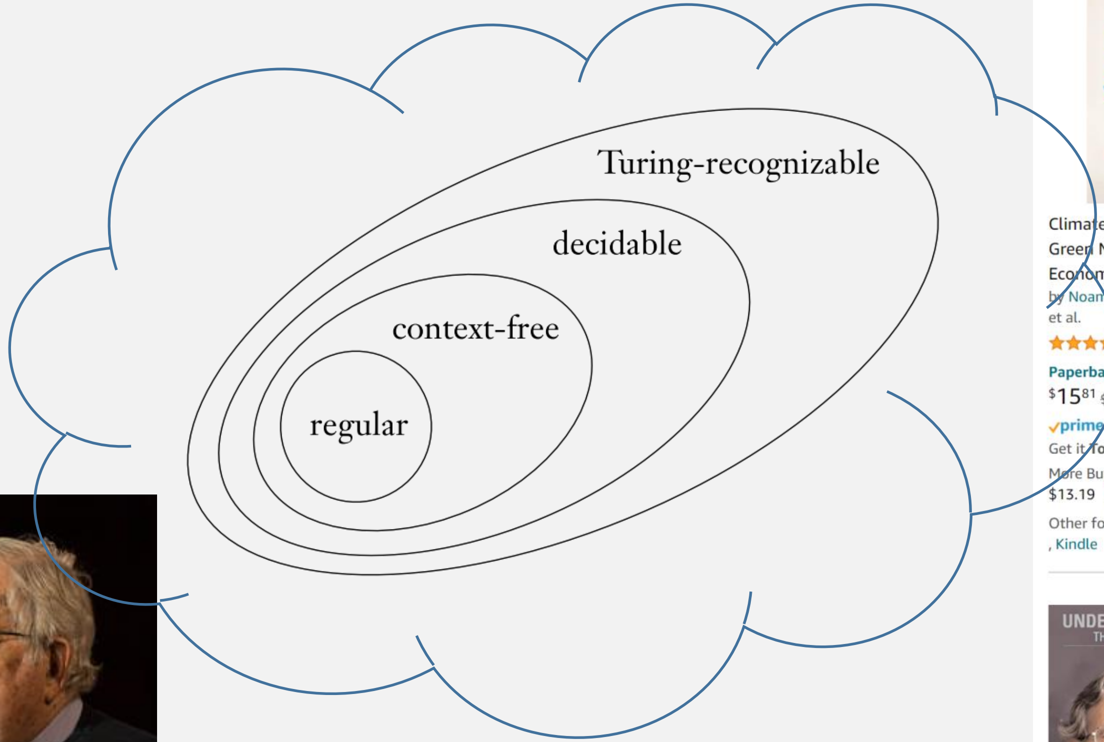
- A Decider for this problem could … ?
  - Try every possible derivation of G, and check if it's equal to w?
  - But this might never halt
    - e.g., if there is a rule like: S -> 0S or S -> S
  - This TM would be a recognizer <u>but not a decider</u>



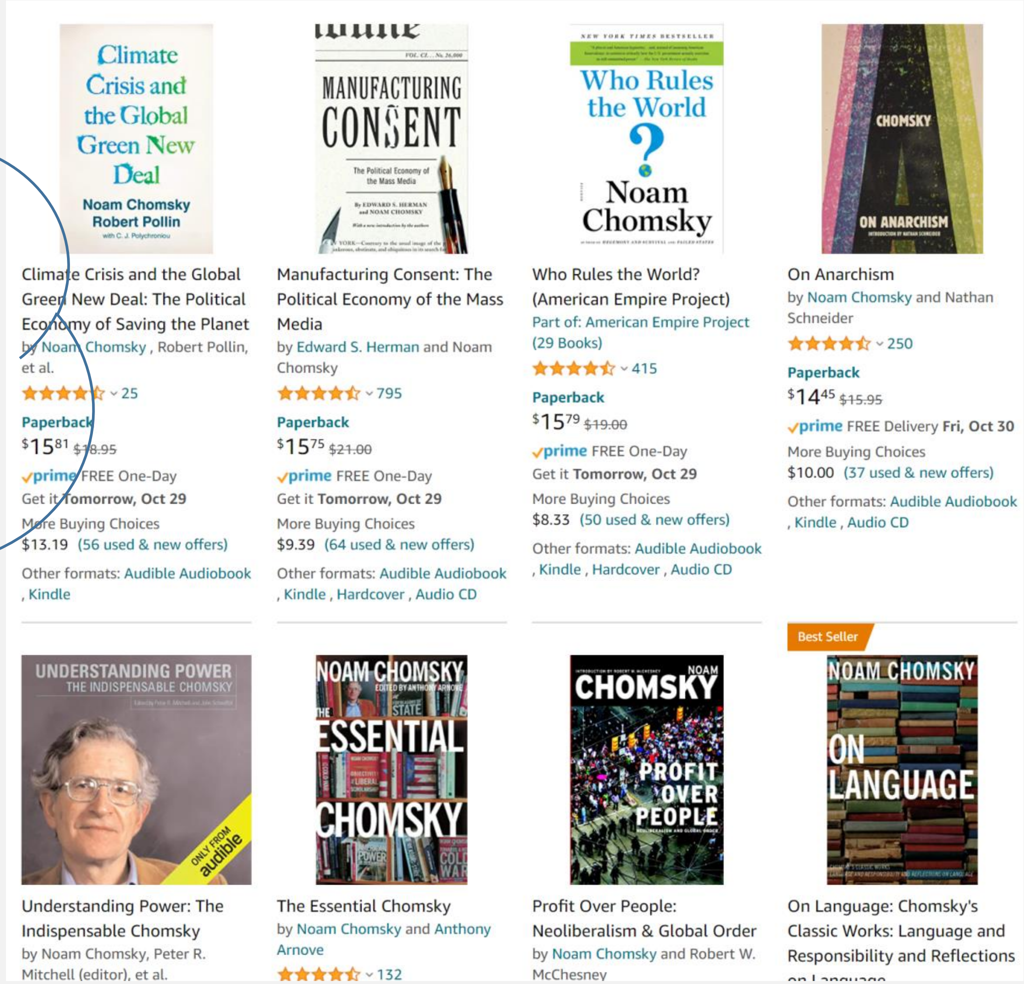
- <u>Idea</u>: can the TM stop checking after some length?
  - i.e., Is there upper bound on the number of derivation steps?

Chomsky Normal Form

Noam Chomsky



Later …



Chomsky Normal Form

DEFINITION 2.8

A context-free grammar is in ***Chomsky normal form*** if every rule is of the form

$$A \to BC$$
$$A \to a$$

Variables only

2 kinds of rules

Terminals only

where a is any terminal and A, B, and C are any variables—except that B and C may not be the start variable. In addition, we permit the rule $S \to \varepsilon$, where S is the start variable.

Chomsky Normal Form: Number of Steps

- To generate a string of length n:
  - $n - 1$ steps: to generate n variables
  - $+ n$ steps: to turn each variable into a terminal
  - Total: $2n - 1$ steps

Chomsky normal form

$$A \to BC$$
$$A \to a$$

Thm: Every CFG has a Chomsky Normal Form

Chomsky normal form

$A \rightarrow BC$
$A \rightarrow a$

1. Add new start variable S_0 that does not appear on any RHS
   - I.e., add rule $S_0 \rightarrow S$, where S is old start var

$$
\begin{aligned}
S &\rightarrow ASA \mid \mathtt{a}B \\
A &\rightarrow B \mid S \\
B &\rightarrow \mathtt{b} \mid \varepsilon
\end{aligned}
\quad \Rightarrow \quad
\begin{aligned}
\boldsymbol{S_0} &\rightarrow \boldsymbol{S} \\
S &\rightarrow ASA \mid \mathtt{a}B \\
A &\rightarrow B \mid S \\
B &\rightarrow \mathtt{b} \mid \varepsilon
\end{aligned}
$$

Thm: Every CFG has a Chomsky Normal Form

Chomsky normal form

$$A \rightarrow BC$$
$$A \rightarrow a$$

1. Add new start variable S_0 that does not appear on any RHS
   - I.e., add rule $S_0 \rightarrow S$, where S is old start var
2. Remove all "empty" rules of the form $A \rightarrow \varepsilon$
   - A must not be the start variable
   - Then for every rule with A on RHS, add new rule with A deleted
     - E.g., If $R \rightarrow uAv$ is a rule, add $R \rightarrow uv$
   - Must cover all combinations if A appears more than once in a RHS
     - E.g., if $R \rightarrow uAvAw$ is a rule, add 3 rules: $R \rightarrow uvAw$, $R \rightarrow uAvw$, $R \rightarrow uvw$

$$\begin{aligned} S_0 &\rightarrow S \\ S &\rightarrow ASA \mid \mathtt{a}B \mid \mathbf{a} \\ A &\rightarrow B \mid S \mid \boldsymbol{\varepsilon} \\ B &\rightarrow \mathtt{b} \mid \varepsilon \end{aligned}$$

Then, add

First, remove

$$\begin{aligned} S_0 &\rightarrow S \\ S &\rightarrow ASA \mid \mathtt{a}B \mid \mathtt{a} \mid \boldsymbol{SA} \mid \boldsymbol{AS} \mid \boldsymbol{S} \\ A &\rightarrow B \mid S \mid \varepsilon \\ B &\rightarrow \mathtt{b} \end{aligned}$$

Then, add

Then, remove

Thm: Every CFG has a Chomsky Normal Form

Chomsky normal form

$A \to BC$

$A \to a$

1. Add new start variable S_0 that does not appear on any RHS
   - I.e., add rule $S_0 \to S$, where S is old start var
2. Remove all "empty" rules of the form $A \to \varepsilon$
   - A must not be the start variable
   - Then for every rule with A on RHS, add new rule with A deleted
     - E.g., If $R \to uAv$ is a rule, add $R \to uv$
   - Must cover all combinations if A appears more than once in a RHS
     - E.g., if $R \to uAvAw$ is a rule, add 3 rules: $R \to uvAw$, $R \to uAvw$, R -> uvw
3. Remove all "unit" rules of the form $A \to B$
   - Then, for every rule $B \to u$, add rule $A \to u$

$$\begin{aligned} S_0 &\to S \\ S &\to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \mid S \\ A &\to B \mid S \\ B &\to \mathtt{b} \end{aligned}$$

Remove, no add (same variable)

$$\begin{aligned} S_0 &\to S \mid \mathbf{ASA} \mid \mathbf{aB} \mid \mathbf{a} \mid \mathbf{SA} \mid \mathbf{AS} \\ S &\to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\ A &\to B \mid S \\ B &\to \mathtt{b} \end{aligned}$$

Remove, then add S RHSs to S_0

$$\begin{aligned} S_0 &\to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\ S &\to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\ A &\to S \mid \mathtt{b} \mid \mathbf{ASA} \mid \mathbf{aB} \mid \mathbf{a} \mid \mathbf{SA} \mid \mathbf{AS} \\ B &\to \mathtt{b} \end{aligned}$$

Remove, then add S RHSs to A

Thm: Every CFG has a Chomsky Normal Form

Chomsky normal form

$A \to BC$

$A \to a$

1. Add new start variable S_0 that does not appear on any RHS
   - I.e., add rule $S_0 \to S$, where S is old start var
2. Remove all "empty" rules of the form $A \to \varepsilon$
   - A must not be the start variable
   - Then for every rule with A on RHS, add new rule with A deleted
     - E.g., If $R \to uAv$ is a rule, add $R \to uv$
   - Must cover all combinations if A appears more than once in a RHS
     - E.g., if $R \to uAvAw$ is a rule, add 3 rules: $R \to uvAw$, $R \to uAvw$, R -> uvw
3. Remove all "unit" rules of the form $A \to B$
   - Then, for every rule $B \to u$, add rule $A \to u$
4. Split up rules with RHS longer than length 2
   - E.g., $A \to wxyz$ becomes $A \to wB$, $B \to xC$, $C \to yz$
5. Replace all terminals on RHS with new rule
   - E.g., for above, add $W \to w$, $X \to x$, $Y \to y$, $Z \to z$

$$\begin{aligned} S_0 &\to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\ S &\to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\ A &\to \mathtt{b} \mid ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\ B &\to \mathtt{b} \end{aligned}$$

$$\begin{aligned} S_0 &\to AA_1 \mid UB \mid \mathtt{a} \mid SA \mid AS \\ S &\to AA_1 \mid UB \mid \mathtt{a} \mid SA \mid AS \\ A &\to \mathtt{b} \mid AA_1 \mid UB \mid \mathtt{a} \mid SA \mid AS \\ A_1 &\to SA \\ U &\to \mathtt{a} \\ B &\to \mathtt{b} \end{aligned}$$

Thm: A_{CFG} is a decidable language

$A_{\text{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG that generates string } w\}$

Proof: create the decider:

$S =$ "On input $\langle G, w\rangle$, where G is a CFG and w is a string:

1. Convert G to an equivalent grammar in Chomsky normal form.
2. List all derivations with $2n-1$ steps, where n is the length of w; except if $n = 0$, then instead list all derivations with one step.
3. If any of these derivations generate w, *accept*; if not, *reject*."

Thm: E_{CFG} is a decidable language

$$E_{\mathsf{CFG}} = \{\langle G \rangle \mid G \text{ is a CFG and } L(G) = \emptyset\}$$

Recall:

$$E_{\mathsf{DFA}} = \{\langle A \rangle \mid A \text{ is a DFA and } L(A) = \emptyset\}$$

T = "On input $\langle A \rangle$, where A is a DFA:

1. Mark the start state of A.
2. Repeat until no new states get marked:
3. Mark any state that has a transition coming into it from any state that is already marked.
4. If no accept state is marked, *accept*; otherwise, *reject*."

"Reachability" (of accept state from start state) algorithm

Thm: E_{CFG} is a decidable language

$E_{\text{CFG}} = \{\langle G \rangle \mid G \text{ is a CFG and } L(G) = \emptyset\}$

- Create decider that calculates reachability for grammar G
  - Except go backwards, start from terminals, to avoid looping

$R =$ "On input $\langle G \rangle$, where G is a CFG:

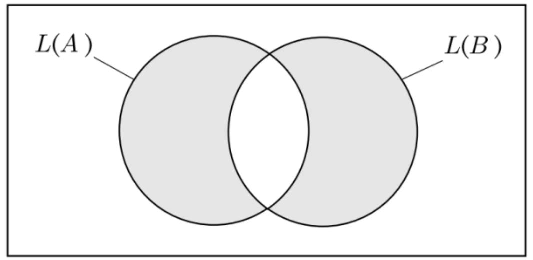
1. Mark all terminal symbols in G.
2. Repeat until no new variables get marked:
3. Mark any variable A where G has a rule $A \rightarrow U_1 U_2 \cdots U_k$ and each symbol $U_1, \ldots, U_k$ has already been marked.
4. If the start variable is not marked, *accept*; otherwise, *reject*."

Thm: EQ_{CFG} is a decidable language?

$EQ_{\mathsf{CFG}} = \{\langle G, H\rangle |\ G \text{ and } H \text{ are CFGs and } L(G) = L(H)\}$

Recall: $EQ_{\mathsf{DFA}} = \{\langle A, B\rangle |\ A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}$

- Used Symmetric Difference



$$L(C) = \emptyset \text{ iff } L(A) = L(B)$$

  - where C = complement, union, intersection of machines A and B
- Can’t do this for CFLs!
  - Intersection and complement are not closed for CFLs!!!

Intersection of CFLs is Not Closed!

- If closed, then intersection of these CFLs should be a CFL:

$$A = \{\mathtt{a}^m \mathtt{b}^n \mathtt{c}^n \mid m, n \geq 0\}$$

$$B = \{\mathtt{a}^n \mathtt{b}^n \mathtt{c}^m \mid m, n \geq 0\}$$

- But $A \cap B$ = $\{\mathtt{a}^n \mathtt{b}^n \mathtt{c}^n \mid n \geq 0\}$

- Not a CFL!
  - See textbook example 2.36

Complement of a CFL is not Closed!

- If CFLs closed under complement:

$$\text{if } G_1 \text{ and } G_2 \text{ context-free}$$

$$\overline{L(G_1)} \text{ and } \overline{L(G_2)} \text{ context-free}$$

$$\overline{L(G_1)} \cup \overline{L(G_1)} \text{ context-free}$$

$$\overline{\overline{L(G_1)} \cup \overline{L(G_1)}} \text{ context-free}$$

$$L(G_1) \cap L(G_2) \text{ context-free}$$

DeMorgan's Law!

Thm: EQ_{CFG} is a decidable language?

$EQ_{\text{CFG}} = \{\langle G, H \rangle | \ G \text{ and } H \text{ are CFGs and } L(G) = L(H)\}$

- No!
  - You cannot decide whether two grammars represent the same lang!

- It's not recognizable either!
  - (But we won't learn how to prove this until Chapter 5)

Decidability of CFGs Recap

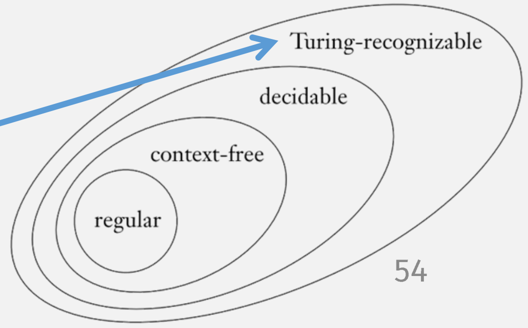
- $A_{\text{CFG}} = \{\langle G, w\rangle |\ G \text{ is a CFG that generates string } w\}$
  - Convert grammar to Chomsky Normal Form
  - Then check all possible derivations of length 2|w| - 1 steps

- $E_{\text{CFG}} = \{\langle G\rangle |\ G \text{ is a CFG and } L(G) = \emptyset\}$
  - Compute “reachability” of start variable from terminals

- $EQ_{\text{CFG}} = \{\langle G, H\rangle |\ G \text{ and } H \text{ are CFGs and } L(G) = L(H)\}$
  - We couldn’t prove that this is decidable!
  - (So you cant use this theorem when creating another decider)

The Limits of Turing Machines?

- So TMs can express any “computation”
  - I.e., any (Python, Java, Racket, …) program you write is a Turing Machine
- So why do we focus on TMs that process other machines?
- Because in CS420, we also want to study the limits of computation
  - And a good way to test the limit of a computational model is to see what it can compute about other computational models …
- So what are the limits of TMs? I.e., what’s here?
  - Or out here?



Next time: A_{TM} is undecidable ???

$$A_{\mathsf{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$$

A_{TM} = the problem of computers simulating other computers, e.g.:

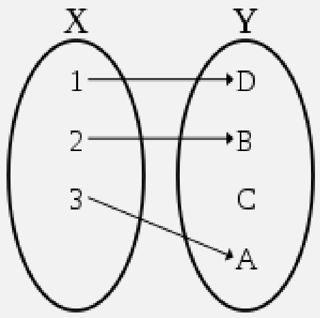
U = "On input $\langle M, w\rangle$, where M is a TM and w is a string:

1. Simulate M on input w.
2. If M ever enters its accept state, *accept*; if M ever enters its reject state, *reject*."

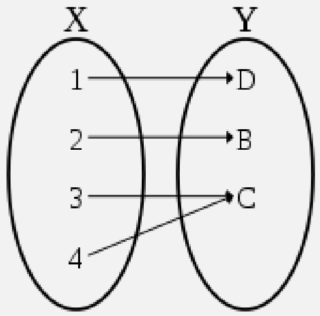
I.e., will machines take over the world?

Kinds of Functions (a fn maps DOMAIN -> RANGE)

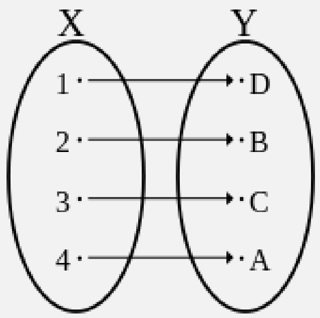
- **Injective**
  - A.k.a., "one-to-one"
  - Every element in DOMAIN has a unique mapping
  - How to remember:
    - DOMAIN is mapped "in" to the RANGE



- **Surjective**
  - A.k.a., "onto"
  - Every element in RANGE is mapped to
  - How to remember:
    - "Sur" = "over" (eg, survey); DOMAIN is mapped "over" the RANGE



- **Bijective**
  - A.k.a., "correspondence" or "one-to-one correspondence"
  - Is both injective and surjective
  - Unique pairing of every element in DOMAIN and RANGE



Countability

- A set is “countable” if it is:
  - Finite
  - Or, there exists a bijection between the set and the natural numbers
    - This set is then considered to have the same size as the set of natural numbers
    - This is called “countably infinite”

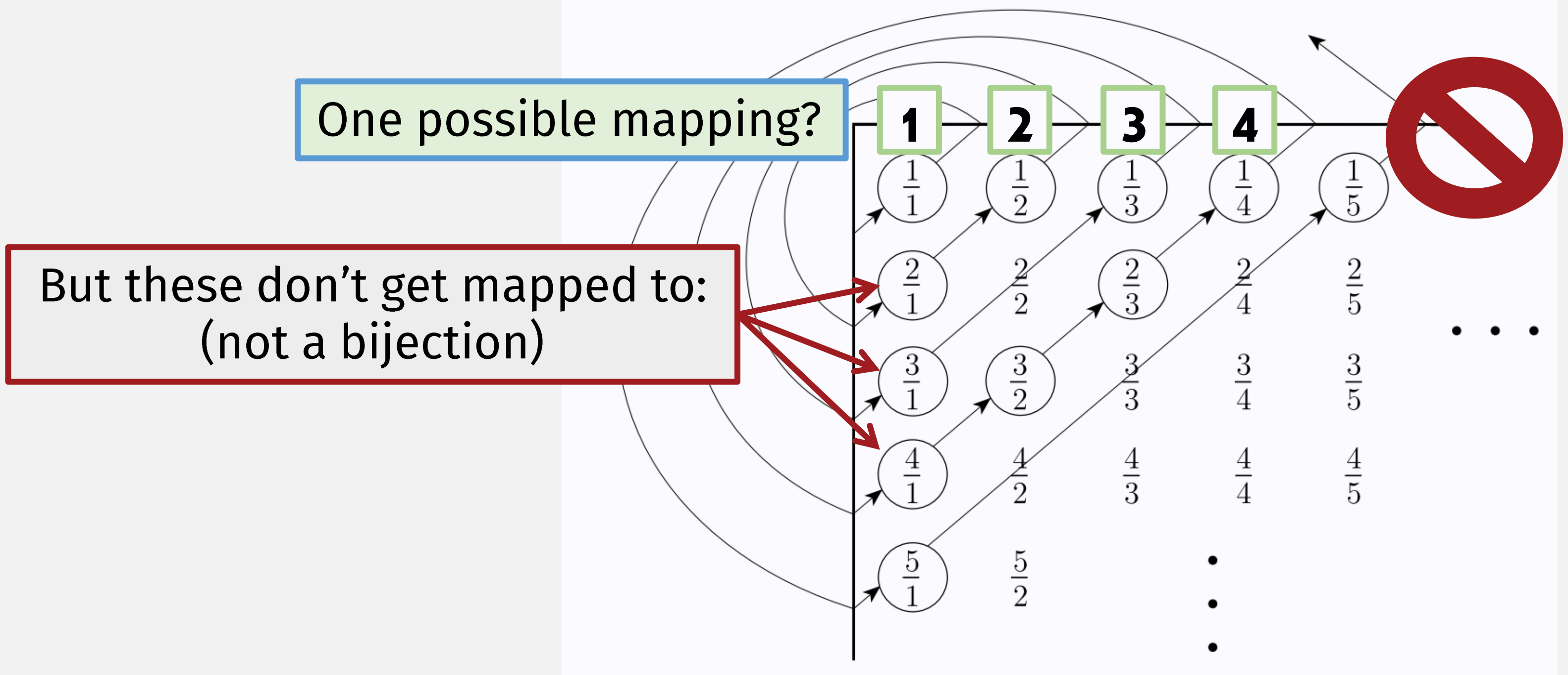
Exercise: Which set is larger?

- The set of:
  - Natural numbers, or
  - Even numbers?
- They are the **same** size! Both are countably infinite
  - Bijection:

n	$f(n) = 2n$
1	2
2	4
3	6
$\vdots$	$\vdots$

Exercise: Which set is larger?

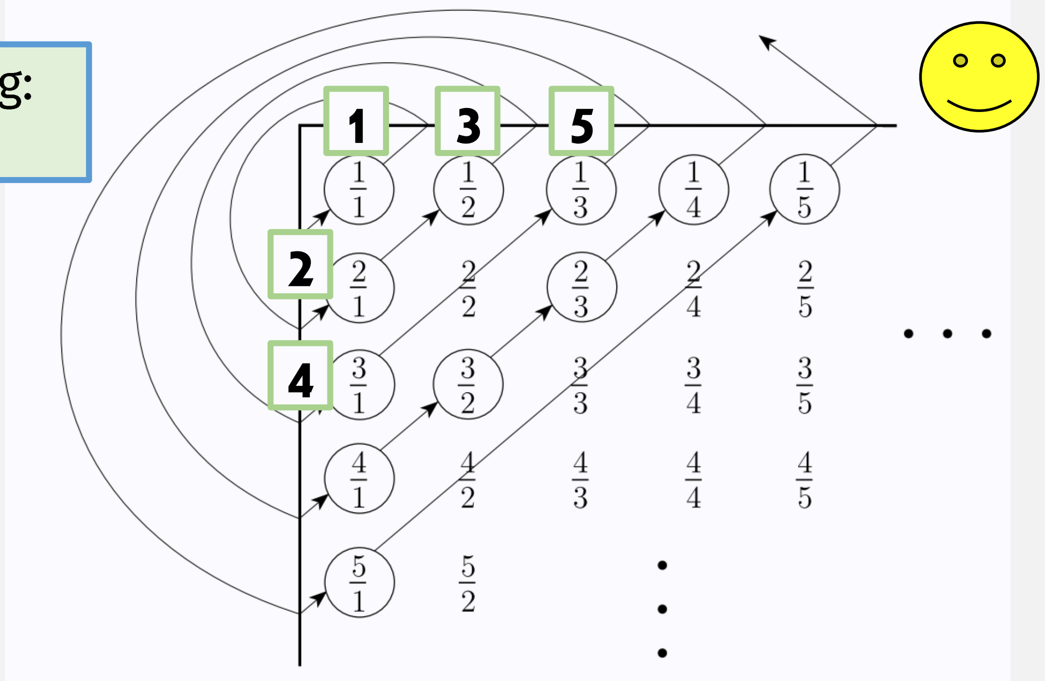
- The set of:
  - Natural numbers $\mathcal{N}$, or
  - Positive rational numbers? $\mathcal{Q} = \{\frac{m}{n} | m, n \in \mathcal{N}\}$
- They are the **same** size! Both are countably infinite



Exercise: Which set is larger?

- The set of:
  - Natural numbers $\mathcal{N}$, or
  - Positive rational numbers? $\mathcal{Q} = \{\frac{m}{n} | m, n \in \mathcal{N}\}$
- They are the **same** size! Both are countably infinite

Another mapping:
(is a bijection)



Exercise: Which set is larger?

- The set of:
  - Natural numbers, or $\mathcal{N}$
  - Real numbers? $\mathcal{R}$
- There are **more** real numbers. It is uncountably infinite.

Proof: next time!

Check-in Quiz 3/29

On gradescope