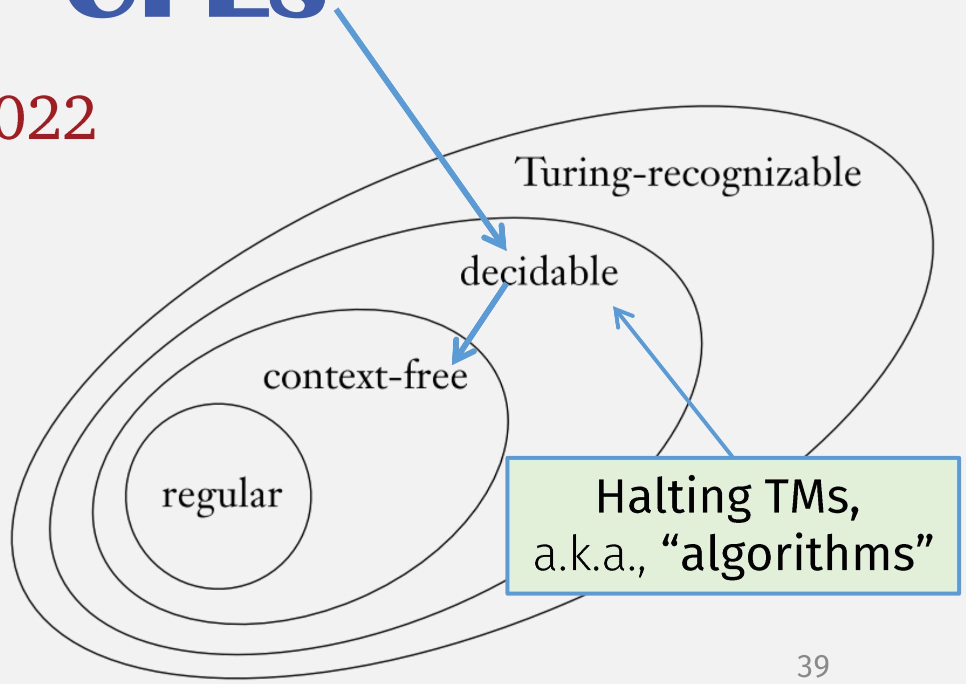
## UMB CS 420

# Decidability for CFLs

Monday, March 21, 2022



# Announcements

- Welcome back!

- HW 6 in
  - Due Sun 3/20 11:59pm

- HW 7 out
  - Due Sun 3/27 11:59pm

- No using Chegg please

# *Last Time:* Algorithms About Regular Langs

- $A_{\mathsf{DFA}} = \{\langle B, w\rangle |\ B \text{ is a DFA that accepts input string } w\}$
  - **Decider**: Implements extended δ function
- $A_{\mathsf{NFA}} = \{\langle B, w\rangle |\ B \text{ is an NFA that accepts input string } w\}$
  - **Decider**: Uses **NFA→DFA** decider + $A_{\mathsf{DFA}}$ decider
- $A_{\mathsf{REX}} = \{\langle R, w\rangle |\ R \text{ is a regular expression that generates string } w\}$
  - **Decider**: Uses **RegExpr→NFA** decider + $A_{\mathsf{NFA}}$ decider
- $E_{\mathsf{DFA}} = \{\langle A\rangle |\ A \text{ is a DFA and } L(A) = \emptyset\}$
  - **Decider**: Reachability algorithm
- $EQ_{\mathsf{DFA}} = \{\langle A, B\rangle |\ A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}$

  - **Decider**: Uses complement and intersection closure construction + $E_{\mathsf{DFA}}$

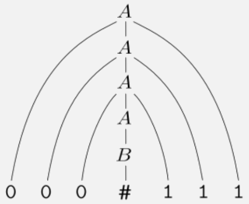
Remember:
**TMs = programs**
**Creating TM = programming**
**Previous theorems = library**

# <u>Thm</u>: $A_{\mathsf{CFG}}$ is a decidable language

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

- This is a very practically important problem …
- … equivalent to:
  - **Algorithm** to <u>**parse**</u> **program** $w$ in **(programming) language** with grammar $G$?

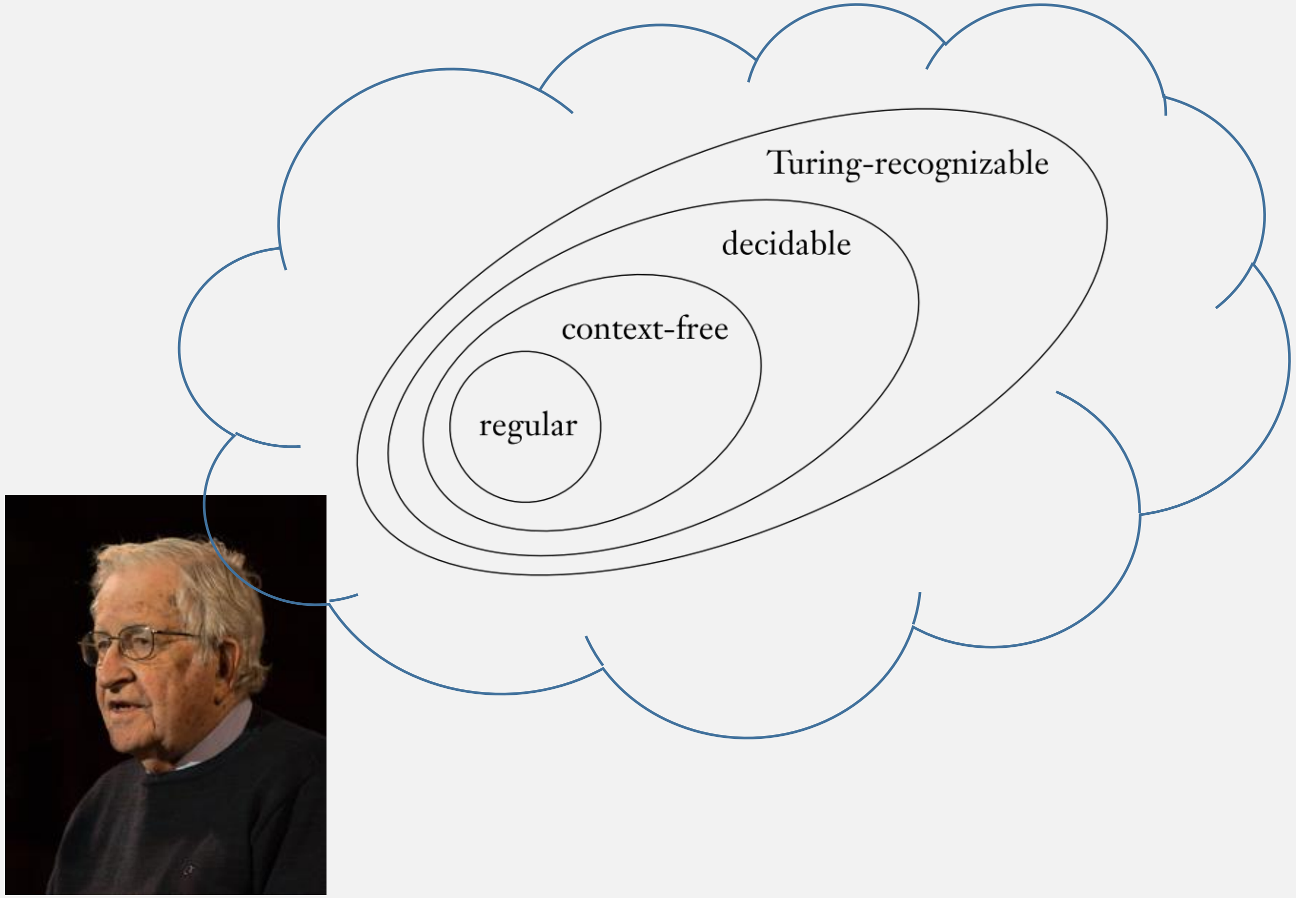
- A Decider for this problem could … ?
  - Try every possible derivation of $G$, and check if it's equal to $w$?
  - But this might never halt
    - E.g., what if there are rules like: $\boldsymbol{S} \rightarrow \boldsymbol{0S}$ or $\boldsymbol{S} \rightarrow \boldsymbol{S}$
  - This TM would be a <u>recognizer 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




He (sort of) invented this course too!

# Chomsky Normal Form

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

(non-start) Variables only

$$A \to BC$$

2 rule shapes

$$A \to a$$

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 Example

- $S \rightarrow AS \mid AB$
- $A \rightarrow \mathtt{a}$
- $B \rightarrow \mathtt{b}$

- **To generate string of length: 2**
  - Use $S$ rule: 1 **time;** $A$ or $B$ rules: 2 **times**
  - $S \Rightarrow AB \Rightarrow \mathtt{a}B \Rightarrow \mathtt{ab}$
  - Derivation total steps: 1 + 2 = 3
- **To generate string of length: 3**
  - Use $S$ rule: 2 **times;** $A$ or $B$ rules: 3 **times**
  - $S \Rightarrow AS \Rightarrow AAB \Rightarrow \mathtt{a}AB \Rightarrow \mathtt{aa}B \Rightarrow \mathtt{aab}$
  - Derivation total steps: 2 + 3 = 5
- **To generate string of length: 4**
  - Use $S$ rule: 3 **times;** $A$ or $B$ rules: 4 **times**
  - $S \Rightarrow AS \Rightarrow AAS \Rightarrow AAAB \Rightarrow \mathtt{a}AAB \Rightarrow \mathtt{aa}AB \Rightarrow \mathtt{aaa}B \Rightarrow \mathtt{aaab}$
  - Derivation total steps: 3 + 4 = 7
- ...

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

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

2 rule shapes

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 $S \rightarrow \varepsilon$, 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

(A **finite** number of steps)

***Chomsky normal form***

$A \to BC$ Use $n$-1 times

$A \to a$ Use $n$ times

# Thm: $A_{\mathsf{CFG}}$ is a decidable language

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

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*."

We first need to prove this is true for all CFGs!

Termination argument?

# Thm: Every CFG has a Chomsky Normal Form

Proof: Create algorithm to convert any CFG into 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

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

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

# 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 $S_0 \rightarrow S$, 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: $R \rightarrow uvAw$, $R \rightarrow uAvw$, $R \rightarrow uvw$

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

Then, add

First, remove

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

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)

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

Remove, then add $S$ RHSs to $S_0$

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

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 $W \to w$, $X \to x$, $Y \to y$, $Z \to z$

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

$$\Downarrow$$

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

# Thm: $A_{\mathsf{CFG}}$ is a decidable language

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

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*."

We first need to prove this is true for all CFGs!

Termination argument:
**Step 1**: any CFG has only a finite # rules
**Step 2**: 2n-1 = finite # of derivations to check
**Step 3**: only 1 step

# Thm: $E_{\mathsf{CFG}}$ is a decidable language

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

Recall:

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

$T$ = "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

Can we compute "reachability" for a CFG?

# Thm: $E_{\mathsf{CFG}}$ is a decidable language

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

Proof: 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 $A \rightarrow U_1U_2\cdots U_k$ and each symbol $U_1, \ldots, U_k$ has already been marked.
4. If the start variable is not marked, *accept*; otherwise, *reject*."

Loop marks 1 new variable on each iteration or stops: it eventually terminates because there are a finite # of variables

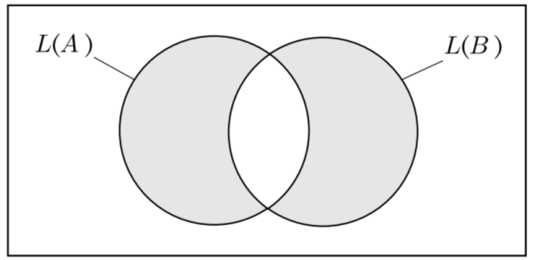
Termination argument?

# <u>Thm</u>: $EQ_{\mathsf{CFG}}$ is a decidable language?

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

<u>Recall</u>: $EQ_{\mathsf{DFA}} = \{\langle A, B\rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}$

- 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!

Proof (by contradiction), Assume intersection is closed for CFLs

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

- … which is not a CFL! (So we have a contradiction)

# Complement of a CFL is not Closed!

- If CFLs closed under complement:

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

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

$$\overline{L(G_1)} \cup \overline{L(G_1)} \text{ context-free}$$ Union of CFLs is closed

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

$$L(G_1) \cap L(G_2) \text{ context-free}$$ DeMorgan's Law!

But intersection is not closed for CFLS (prev slide)

# Thm: $EQ_{\mathsf{CFG}}$ is a decidable language?

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

- No! 
  - There's no algorithm to decide whether two grammars are equivalent!

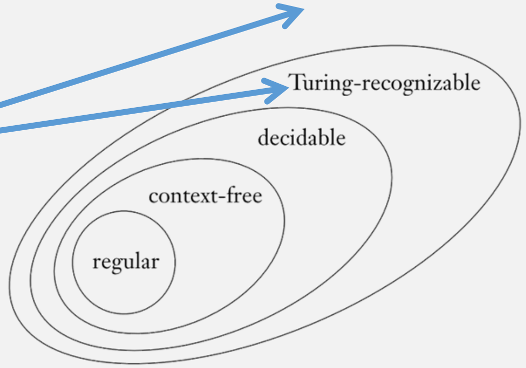
- It's not recognizable either!

# *Summary* Algorithms About CFLs

- $A_{\mathsf{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG that generates string } w\}$
  - **Decider:** Convert grammar to Chomsky Normal Form
  - Then check all possible derivations up to length 2|$w$| - 1 steps
- $E_{\mathsf{CFG}} = \{\langle G\rangle \mid G \text{ is a CFG and } L(G) = \emptyset\}$
  - **Decider:** Compute "reachability" of start variable from terminals
- $EQ_{\mathsf{CFG}} = \{\langle G, H\rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H)\}$
  - **We couldn't prove that this is decidable!**
  - (So you cant use this theorem when creating another decider)

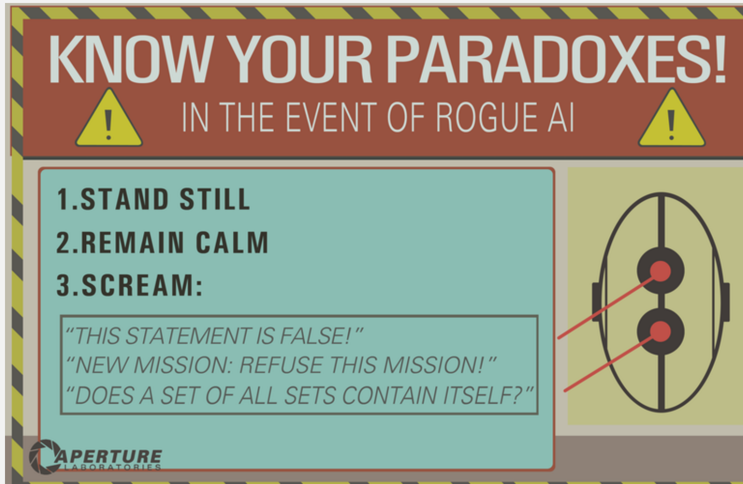
# The Limits of Turing Machines?

- TMs represent all possible "computations"
  - I.e., any (Python, Java, …) program you write is a TM
- So what is **not** computable? I.e., what's here?



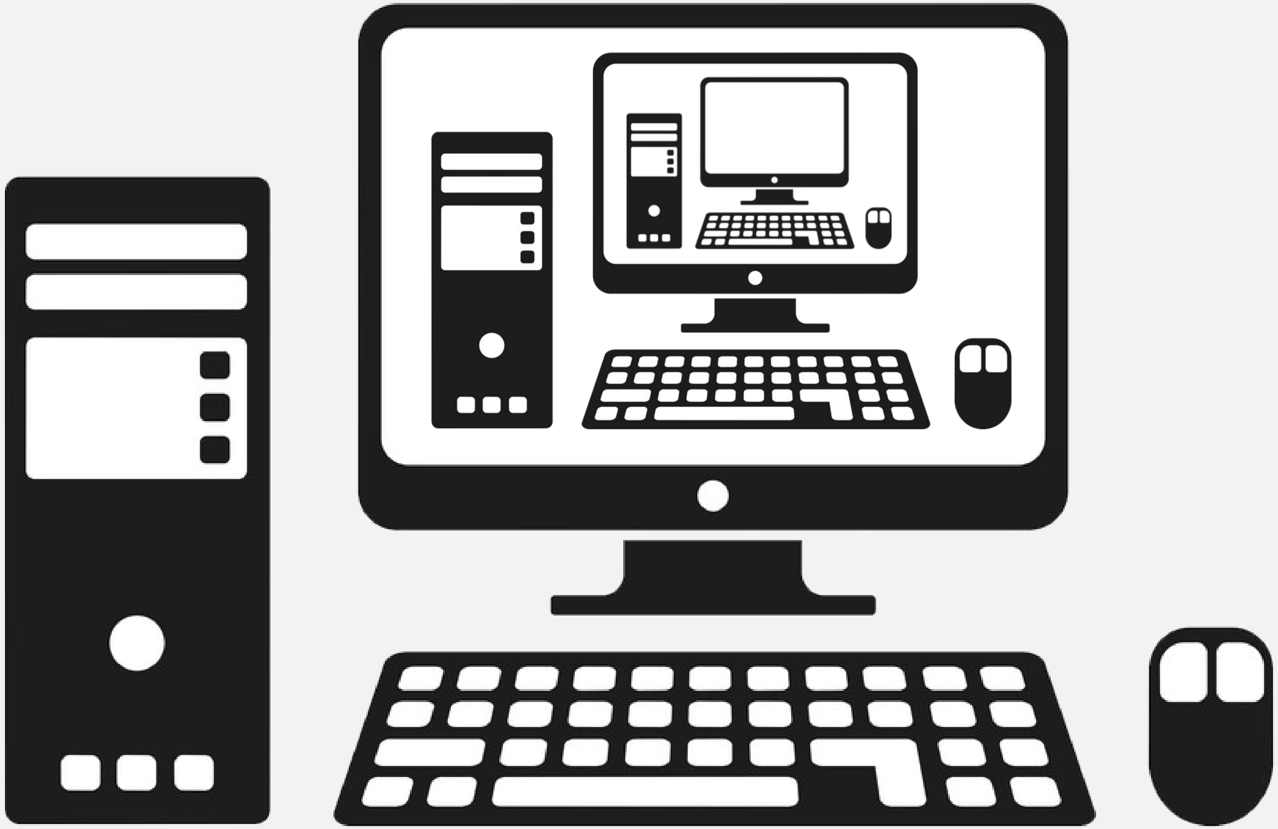

- A way to test the limit of a computational model is to see what it can compute about computational models …
  - **Thought**: Is there an decider (algorithm) to determine whether a TM is an decider?

Hmmm



## *Next time:* Is $A_{\mathsf{TM}}$ decidable?

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



# Check-in Quiz 3/21

On gradescope

# UMBCS622

# Undecidability

Wednesday, October 20, 2021