UMBCS622
Undecidability
Wednesday, October 20, 2021

The diagram illustrates the hierarchy of computational models: regular languages, context-free languages, decidable sets, and Turing-recognizable sets.
Announcements

• HW 5 due Sun 10/24 11:59pm
**Last Time:** Decidable Algorithms About Regular Languages

- \( A_{\text{DFA}} = \{ \langle B, w \rangle \mid B \text{ is a DFA that accepts input string } w \} \)

- \( A_{\text{NFA}} = \{ \langle B, w \rangle \mid B \text{ is an NFA that accepts input string } w \} \)

- \( A_{\text{REX}} = \{ \langle R, w \rangle \mid R \text{ is a regular expression that generates string } w \} \)

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

- \( EQ_{\text{DFA}} = \{ \langle A, B \rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B) \} \)

---

“Things like even software verification, this has been the Holy Grail of computer science for many decades but now in some very key areas, for example, driver verification we’re building tools that can do actual proof about the software and how it works in order to guarantee the reliability.”  
**Bill Gates, April 18, 2002.**  
[Keynote address at WinHec](https://www.microsoft.com/en-us/events/winhec-2002/keynotes/20020418/bill-gates.aspx)
Last Time: Decidable Algorithms CFLs

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

- \( E_{\text{CFG}} = \{ \langle G \rangle \mid G \text{ is a CFG and } L(G) = \emptyset \} \)
Thm: $E_{CFG}$ is a decidable language.

Recall:

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

$T =$ “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.

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

Now 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.”

If loop marks at least 1 variable on each iteration, then it eventually terminates because there are finite variables; else loop terminates

Termination argument?
Thm: $EQ_{\text{CFG}}$ is a decidable language?

$E_{\text{EQ}_\text{CFG}} = \{ \langle G, H \rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H) \}$

Recall: $E_{\text{EQ}_\text{DFA}} = \{ \langle A, B \rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B) \}$

- Used Symmetric Difference

- where $C = \text{complement, union, intersection of machines } A \text{ and } B$

- 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:

\[ A = \{ a^m b^n c^n \mid m, n \geq 0 \} \]

\[ B = \{ a^n b^n c^m \mid m, n \geq 0 \} \]

• But \( A \cap B = \{ a^n b^n c^n \mid n \geq 0 \} \)

• ... which is not a CFL! (So we have a contradiction)
Complement of a CFL is not Closed!

• If CFLs closed under complement:

\[
\begin{align*}
&\text{if } G_1 \text{ and } G_2 \text{ context-free} \\
&\overline{L(G_1)} \text{ and } \overline{L(G_2)} \text{ context-free (From the assumption)} \\
&\overline{L(G_1)} \cup \overline{L(G_1)} \text{ context-free (Union of CFLs is closed)} \\
&\overline{L(G_1)} \cup \overline{L(G_1)} \text{ context-free (From the assumption)} \\
&L(G_1) \cap L(G_2) \text{ context-free (DeMorgan’s Law!)} \\
\end{align*}
\]

But intersection is not closed for CFLS (prev slide)
Thm: $EQ_{CFG}$ is a decidable language?

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

• No! 🍺
  • There’s no algorithm to decide whether two grammars are equivalent!

• It’s not recognizable either!
Summary of Decidable Algorithms for CFLs

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

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

- \( EQ_{\text{CFG}} = \{ \langle G, H \rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H) \} \)
  - 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 algorithm to determine whether a TM is an algorithm?
Is $A_{TM}$ is undecidable? 

$A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts } w \}$
Thm: \( A_{\text{TM}} \) is Turing-recognizable

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

\( U \) = “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, \( \text{accept} \); if \( M \) ever enters its reject state, \( \text{reject} \).”

\( U \) = Extended delta “run” function for TMs
- Computer that can simulate other computers
- i.e., “The Universal Turing Machine”
- Problem: \( U \) loops when \( M \) loops
**Thm:** $A_{TM}$ is undecidable

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

- ???
Kinds of Functions (a fn maps \textit{Domain} $\rightarrow$ \textit{Range})

- **Injective**, a.k.a., “one-to-one”
  - Every element in \textit{Domain} has a unique mapping
  - How to remember:
    - Entire \textit{Domain} is mapped “in” to the \textit{Range}

- **Surjective**, a.k.a., “onto”
  - Every element in \textit{Range} is mapped to
  - How to remember:
    - “Sur” = “over” (eg, survey); \textit{Domain} is mapped “over” the \textit{Range}

- **Bijective**, a.k.a., “correspondence” or “one-to-one correspondence”
  - Is both injective and surjective
  - Unique pairing of every element in \textit{Domain} and \textit{Range}
Countability

• A set is “countable” if it is:
  • Finite
  • Or, there exists a bijection between the set and the natural numbers
    • This set has 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:

<table>
<thead>
<tr>
<th>$n$</th>
<th>$f(n) = 2n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
</tbody>
</table>
Exercise: Which set is larger?

- The set of:
  - Natural numbers $\mathbb{N}$, or
  - Positive rational numbers? $\mathbb{Q} = \{ \frac{m}{n} \mid m, n \in \mathbb{N} \}$
- They are the same size! Both are countably infinite
Exercise: Which set is larger?

• The set of:
  • Natural numbers $\mathbb{N}$, or
  • Positive rational numbers? $Q = \left\{ \frac{m}{n} \mid m, n \in \mathbb{N} \right\}$

• They are the same size! Both are countably infinite.

Another mapping: (it’s a bijection bc every fraction has a unique mapping)
Exercise: Which set is larger?

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

Proof, by contradiction:
- Assume a bijection between natural and real numbers exists.
  - This means that every real number should get mapped to.
- But we show that in any given mapping, ... e.g.:
  - Some real number is not mapped to ...
  - E.g., a number that has different digits at each position:
    \( x = 0.4641 \ldots \)
- This number cannot be included in mapping ...
- ... So we have a contradiction!
Georg Cantor

• Invented set theory

• Came up with countable infinity in 1873

• And uncountability:
  • And how to show uncountability with “diagonalization” technique
# Diagonalization with Turing Machines

**Diagonal:** Result of Giving a TM its own Encoding as Input

<table>
<thead>
<tr>
<th></th>
<th>(\langle M_1 \rangle)</th>
<th>(\langle M_2 \rangle)</th>
<th>(\langle M_3 \rangle)</th>
<th>(\langle M_4 \rangle)</th>
<th>(&lt;D&gt;)</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>(M_1)</td>
<td>accept</td>
<td>reject</td>
<td>accept</td>
<td>reject</td>
<td>accept</td>
<td>...</td>
</tr>
<tr>
<td>(M_2)</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td>...</td>
</tr>
<tr>
<td>(M_3)</td>
<td>reject</td>
<td>reject</td>
<td>reject</td>
<td>reject</td>
<td>reject</td>
<td>...</td>
</tr>
<tr>
<td>(M_4)</td>
<td>accept</td>
<td>accept</td>
<td>reject</td>
<td>reject</td>
<td>accept</td>
<td>...</td>
</tr>
<tr>
<td>(D)</td>
<td>reject</td>
<td>reject</td>
<td>accept</td>
<td>accept</td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

- **All TMs:**
- **Try to construct “opposite” TM:**
- **TM \(D\) can’t exist!**
- **What should happen here?**
- **It must both accept and reject!**
Thm: $A_{TM}$ is undecidable

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

Proof by contradiction:

1. Assume $A_{TM}$ is decidable. Then there exists a decider:

   $$H(\langle M, w \rangle) = \begin{cases} 
   \text{accept} & \text{if } M \text{ accepts } w \\
   \text{reject} & \text{if } M \text{ does not accept } w 
   \end{cases}$$

2. If $H$ exists, then we can create an “opposite” machine:

   $$D = \text{“On input } \langle M \rangle, \text{ where } M \text{ is a TM:}$$
   1. Run $H$ on input $\langle M, \langle M \rangle \rangle$.
   2. Output the opposite of what $H$ outputs. That is, if $H$ accepts, reject; and if $H$ rejects, accept.”

From the previous slide

Result of giving a TM itself as input
Thm: $A_{TM}$ is undecidable

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

Proof by contradiction:

1. **Assume** $A_{TM}$ is decidable. Then there exists a decider:

   $$H(\langle M, w \rangle) = \begin{cases} 
   \text{accept} & \text{if } M \text{ accepts } w \\
   \text{reject} & \text{if } M \text{ does not accept } w 
   \end{cases}$$

2. If $H$ exists, then we can create an “opposite” machine:

   From the previous slide

   $$D = \text{“On input } \langle M \rangle, \text{ where } M \text{ is a TM:}$$

   1. Run $H$ on input $\langle M, \langle M \rangle \rangle$.

   2. Output the opposite of what $H$ outputs. That is, if $H$ accepts, $\text{reject}$; and if $H$ rejects, $\text{accept}$.

3. But $D$ does not exist! **Contradiction!** So assumption is false.
Easier Undecidability Proofs

- We proved \( A_{\text{TM}} = \{\langle M, w \rangle | M \text{ is a TM and } M \text{ accepts } w \} \) undecidable ...
- ... by contradiction ...
- ... specifically, showing that its decider could be used to implement an impossible decider “\( D \)”!

- Coming up with “\( D \)” was hard (needed to invent diagonalization)

- But then we more easily \textbf{reduced} \( A_{\text{TM}} \) to the “\( D \)”

- Now we can also reduce problems to \( A_{\text{TM}} \)!
The Halting Problem

$$HALT_{TM} = \{ \langle M, w \rangle | M \text{ is a TM and } M \text{ halts on input } w \}$$

Thm: $HALT_{TM}$ is undecidable

Proof, by contradiction:

- Assume $HALT_{TM}$ has decider $R$; use it to create decider for $A_{TM}$:

  - ...  

- But $A_{TM}$ is undecidable and has no decider!
The Halting Problem

\[ \text{HALT}_{TM} = \{ \langle M, w \rangle | M \text{ is a TM and } M \text{ halts on input } w \} \]

Thm: \text{HALT}_{TM} \text{ is undecidable}

Proof, by contradiction:

• Assume \text{HALT}_{TM} \text{ has decider } R; \text{ use it to create decider for } A_{TM}:

\[ S = \text{“On input } \langle M, w \rangle, \text{ an encoding of a TM } M \text{ and a string } w:\]

1. Run TM \( R \) on input \( \langle M, w \rangle \).
2. If \( R \) rejects, reject. \( \rightarrow \) This means \( M \) loops on input \( w \)
3. If \( R \) accepts, simulate \( M \) on \( w \) until it halts. \( \rightarrow \) This step always halts
4. If \( M \) has accepted, accept; if \( M \) has rejected, reject.”
The Halting Problem

Thm: $\text{HALT}_{\text{TM}}$ is undecidable

Proof, by contradiction:

• Assume $\text{HALT}_{\text{TM}}$ has decider $R$; use it to create decider for $A_{\text{TM}}$:

  $S = \text{“On input } \langle M, w \rangle, \text{ an encoding of a TM } M \text{ and a string } w:\n  1. \text{ Run TM } R \text{ on input } \langle M, w \rangle.\n  2. \text{ If } R \text{ rejects, reject.}\n  3. \text{ If } R \text{ accepts, simulate } M \text{ on } w \text{ until it halts.}\n  4. \text{ If } M \text{ has accepted, accept; if } M \text{ has rejected, reject.”}\n
• But $A_{\text{TM}}$ is undecidable!
  • I.e., this decider that we just created cannot exist! So $\text{HALT}_{\text{TM}}$ is undecidable
Easier Undecidability Proofs

In general, to prove the undecidability of a language:
• Use proof by contradiction:

• Assume the language is decidable,

• Show that its decider can be used to create a decider for ...

• ... a known undecidable language ...

• ... which doesn’t have a decider!
Summary: The Limits of Algorithms

- $A_{\text{DFA}} = \{ \langle B, w \rangle | \text{B is a DFA that accepts input string } w \}$ Decidable
- $A_{\text{CFG}} = \{ \langle G, w \rangle | \text{G is a CFG that generates string } w \}$ Decidable
- $A_{\text{TM}} = \{ \langle M, w \rangle | \text{M is a TM and M accepts } w \}$ Undecidable
- $E_{\text{DFA}} = \{ \langle A \rangle | \text{A is a DFA and } L(A) = \emptyset \}$ Decidable
- $E_{\text{CFG}} = \{ \langle G \rangle | \text{G is a CFG and } L(G) = \emptyset \}$ Decidable
- $E_{\text{TM}} = \{ \langle M \rangle | \text{M is a TM and } L(M) = \emptyset \}$ Undecidable
Reducibility: Modifying the TM

Thm: $E_{TM}$ is undecidable

Proof, by contradiction:

- Assume $E_{TM}$ has decider $R$; use to create $A_{TM}$ decider:
  
  $S = \text{"On input } \langle M, w \rangle, \text{ an encoding of a TM } M \text{ and a string } w:\n  
  
  \begin{itemize}
  \item \text{First, construct } M_1
  \item \text{Run } R \text{ on input } \langle M_1 \rangle
  \item \text{If } R \text{ accepts, reject (because it means } \langle M \rangle \text{ doesn’t accept } w
  \item \text{if } R \text{ rejects, then } \text{accept ( } \langle M \rangle \text{ accepts } w
  \end{itemize}$

- Idea: Wrap $\langle M \rangle$ in a new TM that can only accept $w$:

  $$M_1 = \text{"On input } x:\n  
  \begin{itemize}
  \item \text{If } x \neq w, \text{ reject.}
  \item \text{If } x = w, \text{ run } M \text{ on input } w \text{ and accept if } M \text{ does.}"
  \end{itemize}$$
Reducibility: Modifying the TM

Thm: $E_{TM}$ is undecidable

Proof, by contradiction:

• Assume $E_{TM}$ has decider $R$; use to create $A_{TM}$ decider:

  $S = \text{"On input } \langle M, w \rangle, \text{ an encoding of a TM } M \text{ and a string } w:\n
  \begin{align*}
  &\text{First, construct } M_1 \\
  &\text{Run } R \text{ on input } \langle M_1 \rangle \\
  &\text{If } R \text{ accepts, } \textbf{reject} \text{ (because it means } \langle M \rangle \text{ doesn't accept } w) \\
  &\text{if } R \text{ rejects, then } \textbf{accept} \text{ (} \langle M \rangle \text{ accepts } w) \\
  \end{align*}$

• Idea: Wrap $\langle M \rangle$ in a new TM that can only accept $w$:

  $M_1 = \text{"On input } x:\n
  \begin{enumerate}
  \item If } x \neq w, \text{ reject.} \\
  \item If } x = w, \text{ run } M \text{ on input } w \text{ and } \textbf{accept} \text{ if } M \text{ does.} \n  \end{enumerate}$
One more, modify $M$: $\text{REGULAR}_{TM}$ is undecidable

$\text{REGULAR}_{TM} = \{\langle M \rangle | M$ is a TM and $L(M)$ is a regular language$\}$

Proof, by contradiction:

• Assume $\text{REGULAR}_{TM}$ has decider $R$; use to create $A_{TM}$ decider:

  $S =$ “On input $\langle M, w \rangle$, an encoding of a TM $M$ and a string $w$:

  • First, construct $M_2$ (??)
  • Run $R$ on input $\langle M_2 \rangle$
  • If $R$ accepts, accept; if $R$ rejects, reject

  Want: $L(M_2) =$
  • regular, if $M$ accepts $w$
  • nonregular, if $M$ does not accept $w$
Thm: $\text{REGULAR}_{TM}$ is undecidable (continued)

$\text{REGULAR}_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language} \}$

$M_2 = \text{“On input } x:\$

1. If $x$ has the form $0^n1^n$, accept.
2. If $x$ does not have this form, run $M$ on input $w$ and accept if $M$ accepts $w$.

Want: $L(M_2) =$

- regular, if $M$ accepts $w$
- nonregular, if $M$ does not accept $w$
Summary: The Limits of Algorithms

- $A_{DFA} = \{ \langle B, w \rangle | B \text{ is a DFA that accepts input string } w \}$
- $A_{CFG} = \{ \langle G, w \rangle | G \text{ is a CFG that generates string } w \}$
- $A_{TM} = \{ \langle M, w \rangle | M \text{ is a TM and } M \text{ accepts } w \}$
- $E_{DFA} = \{ \langle A \rangle | A \text{ is a DFA and } L(A) = \emptyset \}$
- $E_{CFG} = \{ \langle G \rangle | G \text{ is a CFG and } L(G) = \emptyset \}$
- $E_{TM} = \{ \langle M \rangle | M \text{ is a TM and } L(M) = \emptyset \}$
- $EQ_{DFA} = \{ \langle A, B \rangle | A \text{ and } B \text{ are DFAs and } L(A) = L(B) \}$
- $EQ_{CFG} = \{ \langle G, H \rangle | G \text{ and } H \text{ are CFGs and } L(G) = L(H) \}$
- $EQ_{TM} = \{ \langle M_1, M_2 \rangle | M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2) \}$

Decidable
Decidable
Undecidable
Decidable
Decidable
Undecidable
Decidable
Undecidable
Undecidable
Reduce to something else: $EQ_{TM}$ is undecidable

$EQ_{TM} = \{\langle M_1, M_2 \rangle | M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$

Proof, by contradiction:

• Assume $EQ_{TM}$ has decider $R$; use to create $A_{TM}$ decider:

$S =$ “On input $\langle M \rangle$, where $M$ is a TM:

1. Run $R$ on input $\langle M, M_1 \rangle$, where $M_1$ is a TM that rejects all inputs.

2. If $R$ accepts, accept; if $R$ rejects, reject.”
Reduce to something else: $E_{TM}$ is undecidable

$E_{TM} = \{ \langle M_1, M_2 \rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2) \}$

Proof, by contradiction:

• Assume $E_{TM}$ has decider $R$; use to create $A_{TM}$ decider:

$S = \text{“On input } \langle M \rangle, \text{ where } M \text{ is a TM:}$$

1. Run $R$ on input $\langle M, M_1 \rangle$, where $M_1$ is a TM that rejects all inputs.
2. If $R$ accepts, accept; if $R$ rejects, reject.”

• But $E_{TM}$ is undecidable!
Summary

- \( A_{DFA} = \{ \langle B, w \rangle \mid B \text{ is a DFA that accepts input string } w \} \)  
  Decidable

- \( A_{CFG} = \{ \langle G, w \rangle \mid G \text{ is a CFG that generates string } w \} \)  
  Decidable

- \( A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts } w \} \)  
  Undecidable

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

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

- \( E_{TM} = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset \} \)  
  Undecidable

- \( EQ_{DFA} = \{ \langle A, B \rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B) \} \)  
  Decidable

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

- \( EQ_{TM} = \{ \langle M_1, M_2 \rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2) \} \)  
  Undecidable
Also Undecidable ...

- $\text{REGULAR}_{TM} = \{<M> | M \text{ is a TM and } L(M) \text{ is a regular language}\}$
- $\text{CONTEXTFREE}_{TM} = \{<M> | M \text{ is a TM and } L(M) \text{ is a CFL}\}$
- $\text{DECIDABLE}_{TM} = \{<M> | M \text{ is a TM and } L(M) \text{ is a decidable language}\}$
- $\text{FINITE}_{TM} = \{<M> | M \text{ is a TM and } L(M) \text{ is a finite language}\}$
- ...
- $\text{ANYTHING}_{TM} = \{<M> | M \text{ is a TM and "anything" about } L(M)\}$
Turing Unrecognizable?

Is there anything out here?

Where do these go?

\[ E_{TM} = \{ \langle M \rangle | M \text{ is a TM and } L(M) = \emptyset \} \]

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

\[ EQ_{TM} = \{ \langle M_1, M_2 \rangle | M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2) \} \]
**Thm:** Some langs are not Turing-recognizable

**Proof:** requires 2 lemmas

- **Lemma 1:** The *set of all languages* is *uncountable*
  - **Proof:** Show there is a bijection with another uncountable set ...
    - ... The set of all infinite binary sequences

- **Lemma 2:** The *set of all TMs* is *countable*

- Therefore, some language is not recognized by a TM
Mapping a Language to a Binary Sequence

\[ \Sigma^* = \{ \varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots \} \]

\[ A = \{ 0, 00, 01, 000, 001, \ldots \} \]

\[ \chi_A = \begin{array}{ccccccccccc}
0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & \ldots \\
\end{array} \]

Each digit represents one possible string:
- 1 if lang has that string,
- 0 otherwise
**Thm:** Some langs are not Turing-recognizable

**Proof:** requires 2 lemmas

- **Lemma 1:** The set of all languages is *uncountable*
  - **Proof:** Show there is a bijection with another uncountable set ...
    - ... The set of all infinite binary sequences
      - Now just prove set of infinite binary sequences is uncountable (diagonalization)

- **Lemma 2:** The set of all TMs is *countable*
  - Because every TM $M$ can be encoded as a string $<M>$
  - And set of all strings is countable

- Therefore, some language is not recognized by a TM
Co-Turing-Recognizability

• A language is **co-Turing-recognizable** if ...
• ... it is the **complement** of a Turing-recognizable language.
Thm: Decidable $\Leftrightarrow$ Recognizable & co-Recognizable
**Thm:** Decidable $\iff$ Recognizable & co-Recognizable

$\Rightarrow$ If a language is **decidable**, then it is **recognizable** and **co-recognizable**

- Decidable $\Rightarrow$ Recognizable:
  - A decider is a recognizer, bc decidable langs are a subset of recognizable langs
- Decidable $\Rightarrow$ Co-Recognizable:
  - To create co-decider from a decider ... switch reject/accept of all inputs
  - A co-decider is a co-recognizer, for same reason as above

$\Leftarrow$ If a language is **recognizable** and **co-recognizable**, then it is **decidable**
Thm: Decidable $\iff$ Recognizable & co-Recognizable

$\Rightarrow$ If a language is **decidable**, then it is **recognizable** and **co-recognizable**
- Decidable $\Rightarrow$ Recognizable:
  - A decider is a recognizer, bc decidable langs are a subset of recognizable langs
- Decidable $\Rightarrow$ Co-Recognizable:
  - To create co-decider from a decider ... switch reject/accept of all inputs
  - A co-decider is a co-recognizer, for same reason as above

$\Leftarrow$ If a language is **recognizable** and **co-recognizable**, then it is **decidable**
- Let $M_1 = \text{recognizer for the language}$,
- and $M_2 = \text{recognizer for its complement}$
- Decider $M$:
  - Run 1 step on $M_1$,
  - Run 1 step on $M_2$,
  - Repeat, until one machine accepts. If it’s $M_1$, accept. If it’s $M_2$, reject

Termination Arg: Either $M_1$ or $M_2$ must accept and halt, so $M$ halts and is a decider
A Turing-unrecognizable language

• We’ve proved:

\[ A_{TM} \text{ is Turing-recognizable} \]
\[ A_{TM} \text{ is undecidable} \]

• So:

\[ \overline{A_{TM}} \text{ is not Turing-recognizable} \]

• Because: recognizable & co-recognizable implies decidable
Is there anything out here?

$A_{TM}$

Turing-recognizable

decidable

context-free

regular
Check-in Quiz 10/20

On gradescope