Today's learning goals

• Trace high-level descriptions of algorithms for computational problems.
• Use counting arguments to prove the existence of unrecognizable (undecidable) languages.
• Use diagonalization in a proof of undecidability.
Computational problems

A computational problem is **decidable** iff the language encoding the problem instances is decidable.

In Sipser 4.1: The computational problems below

\(A_{\text{DFA}}, A_{\text{NFA}}, A_{\text{REX}}, A_{\text{CFG}}\)

\(E_{\text{DFA}}, E_{\text{NFA}}, E_{\text{REX}}, E_{\text{CFG}}\)

\(EQ_{\text{DFA}}, EQ_{\text{NFA}}, EQ_{\text{REX}}\)

are all decidable
$E_{\text{REX}}$ decidable

$E_{\text{REX}} = \{ <R> \mid R \text{ is reg exp} \}$

Note: $<\emptyset> \in E_{\text{REX}}$
$<\Sigma^*\emptyset> \in E_{\text{REX}}$

Idea: transform $R \leadsto \text{NFA} \leadsto \text{DFA}$
then use $E_{\text{DFA}}$.

Define TM $M$ = “On input $<R>$, $R$ is reg exp
1. Convert $R$ to equiv NFA, $N$.
2. Convert $N$ to equiv DFA, $D$.
3. Check if $L(D) = \emptyset$, i.e.
   - check if $<D> \in E_{\text{DFA}}$.
   - If yes, accept; if no, reject.”

Proof of correctness...”
Undecidable?

- There are many ways to prove that a problem is decidable. **Goal: Build a decider**
- How do we find (and prove) that a problem is not decidable?
Counting arguments

Before we proved the Pumping Lemma …

We proved there was a set that was not regular because

All regular sets are countable, while all sets of strings are uncountable.
Reminder: countable/ uncountable

Sets A and B have the **same size** \(|A| = |B|\) means there is a function between them that is both one-to-one and onto.

<table>
<thead>
<tr>
<th>Countable</th>
<th>Uncountable</th>
</tr>
</thead>
<tbody>
<tr>
<td>(finite or same size as N)</td>
<td></td>
</tr>
<tr>
<td>N, Z, Q</td>
<td>R</td>
</tr>
<tr>
<td>(\Sigma^*)</td>
<td>{ infinite sequences over (\Sigma})</td>
</tr>
<tr>
<td>The set of all TMs</td>
<td>(P(\Sigma^*))</td>
</tr>
</tbody>
</table>

*Sipser p. 202-204*
Why is the set of Turing-recognizable languages countable?

A. It's equal to the set of all TMs, which we showed is countable.

B. It's a subset of the set of all TMs, which we showed is countable.

C. Each Turing-recognizable language is associated with a TM, so there can be no more Turing-recognizable languages than TMs.

D. More than one of the above.

E. I don't know.
Is the set of Turing-decidable sets countable?  Yes!
Satisfied?

• Maybe not …

• What's a specific example of a language that is not Turing-recognizable? or not Turing-decidable?

• Idea: consider a set that, were it to be Turing-decidable, would have to "talk" about itself, and contradict itself!
Recall $A_{DFA} = \{<B,w> \mid B \text{ is a DFA and } w \text{ is in } L(B) \}$

$A_{TM} = \{<M,w> \mid M \text{ is a TM and } w \text{ is in } L(M) \}$

What is $A_{TM}$?

A. A Turing machine whose input is codes of TMs and strings.
B. A set of pairs of TMs and strings.
C. A set of strings that encode TMs and strings.
D. Not well defined.
E. I don't know.
\[ \sum \text{ fixed to be } \{0,1,2\} \]

\[ A_{TM} = \{ <M,w> | M \text{ is a TM and } w \text{ is in } L(M) \} \]

From HW5

Which of the following are in \( A_{TM} \)?

- \( <M_1> \)
- \( <M_2> \)
- \( <M_1, \varepsilon> \)
- \( <M_2, \varepsilon> \)
- \( <M_1, <M_2>> \)
- \( <M_2, <M_4>> \)

\[ L(M_2) = \{ \varepsilon, 0, 3 \} \]
$A_{TM} = \{ <M, w> \mid M \text{ is a TM and } w \text{ is in } L(M) \}$

Define the TM $N$ = "On input $<M, w>$:
1. Simulate $M$ on $w$.
2. If $M$ accepts, accept. If $M$ rejects, reject."

A. $N$ rejects $<M_1, 0>$
B. $N$ accepts $<M_2>$
C. $N$ rejects $<M_4, 1>$
D. $N$ recognizes $A_{TM}$
E. More than one of the above.
\[ A_{\text{TM}} \]

\[ A_{\text{DFA}} = \{ <B,w> \mid B \text{ is a DFA and } w \text{ is in } L(B) \} \]

Decider for this set simulates arbitrary DFA

\[ A_{\text{TM}} = \{ <M,w> \mid M \text{ is a TM and } w \text{ is in } L(M) \} \]
Next time: Exam 2

Bring ID, note card

Check seat maps before coming to class