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*  

\[ \text{A}_{\text{DFA}}, \text{A}_{\text{NFA}}, \text{A}_{\text{REX}}, \text{A}_{\text{CFG}} \]

\[ \text{E}_{\text{DFA}}, \text{E}_{\text{NFA}}, \text{E}_{\text{REX}}, \text{E}_{\text{CFG}} \]

\[ \text{EQ}_{\text{DFA}}, \text{EQ}_{\text{NFA}}, \text{EQ}_{\text{REX}} \]

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

Regular expression that describes $\emptyset$ also $\Sigma^* \emptyset$, but not $\Sigma^* \cup \emptyset$

Idea: convert $R$ to DFA & check if DFA recognizes $\emptyset$

High level description of TM: On input $<R>$, $R$ regex

pf if correct ex.
Undecidable?

- There are many ways to prove that a problem *is* decidable.
- How do we find (and prove) that a problem *is not* decidable?
Before we proved the Pumping Lemma ... 

We proved there was a set that was not regular because all sets of strings are countable, whereas the set of non-regular sets of strings is 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 (finite or same size as $\mathbb{N}$)</th>
<th>Uncountable</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{N}, \mathbb{Z}, \mathbb{Q}$</td>
<td>$\mathbb{R}$</td>
</tr>
<tr>
<td>$\Sigma^*$</td>
<td>${ \text{infinite sequences over } \Sigma }$</td>
</tr>
<tr>
<td>The set of all TMs</td>
<td>$P(\Sigma^*)$</td>
</tr>
</tbody>
</table>

$M = (Q, \Sigma, \Gamma, \delta, q_0, q_f, F)$

$L = \{ \langle M \rangle \in \Sigma^* \mid \text{M accepts } L \}$

Sipser p. 202-204
Counting arguments

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.
Counting arguments

All sets of strings

Countable

All Turing-recognizable sets

Is the set of Turing-decidable sets countable?
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 $A_{DFA} = \{<B,w> | B \text{ is a DFA and } w \text{ is in } L(B)\}$

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

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

Which of the following are in $A_{TM}$?

- $<M_1>$
- $<M_2>$
- $<M_1, \varepsilon>$
- $<M_2, \varepsilon>$
- $<M_1, <M_2>>$
- $<M_2, <M_1>>$

$L(M_1) = \Sigma^*$

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

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 \( A_{\text{TM}} \)
E. More than one of the above.
$A_{\text{TM}}$

$A_{\text{DFA}} = \{<B,w> | \text{B is a DFA and } w \text{ is in } L(B) \}$

Decider for this set simulates arbitrary DFA

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

Bring ID, note card

Check seat maps before coming to class