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*

\[
\begin{align*}
A_{DFA}, A_{NFA}, A_{REX}, A_{CFG} \\
E_{DFA}, E_{NFA}, E_{REX}, E_{CFG} \\
EQ_{DFA}, EQ_{NFA}, EQ_{REX}
\end{align*}
\]

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

• There are many ways to prove that a problem is decidable.
• 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 sets of strings

Uncountable

All Regular Sets

Countable

All sets of strings
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 (finite or same size as N)</th>
<th>Uncountable</th>
</tr>
</thead>
<tbody>
<tr>
<td>N, Z, Q</td>
<td>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>
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

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!
$A_{TM}$

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

From HW5

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

$$<M_1> \quad <M_2> \quad <M_1, \varepsilon> \quad <M_2, \varepsilon> \quad <M_1, <M_2>> \quad <M_2, <M_1>>$$
$A_{TM} = \{<M,w> \mid M \text{ is a TM and } w \text{ is in } L(M) \}$

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

Decider for this set simulates arbitrary DFA

$A_{TM} = \{ <M, w> \mid M \text{ is a TM and } w \text{ is in } L(M) \}$
Diagonalization proof: $A_{TM}$ not decidable Sipser p. 207

Assume, towards a contradiction, that it is.

Call $M_{ATM}$ the decider for $A_{TM}$:

For every TM $M$ and every string $w$,

- Computation of $M_{ATM}$ on $<M,w>$ halts and accepts if $w$ is in $L(M)$.
- Computation of $M_{ATM}$ on $<M,w>$ halts and rejects if $w$ is not in $L(M)$.
Diagonalization proof: $A_{TM}$ not decidable  

Sipser 4.11

Assume, towards a contradiction, that $M_{ATM}$ decides $A_{TM}$

Define the TM $D = "On input <M>:"

1. Run $M_{ATM}$ on $<M, <M>$. 
2. If $M_{ATM}$ accepts, reject; if $M_{ATM}$ rejects, accept."
Diagonalization proof: $A_{TM}$ not decidable  

Sipser 4.11

Assume, towards a contradiction, that $M_{ATM}$ decides $A_{TM}$

Define the TM $D =$ "On input $<M>$:

1. Run $M_{ATM}$ on $<M, <M>>$.
2. If $M_{ATM}$ accepts, reject; if $M_{ATM}$ rejects, accept."

Which of the following computations halt?

A. Computation of $D$ on $<M_1>$
B. Computation of $D$ on $<M_4>$
C. Computation of $D$ on $<D>$
D. All of the above.
E. None of the above.
Diagonalization proof: $A_{TM}$ not decidable \textit{Sipser} 4.11

Assume, towards a contradiction, that $M_{ATM}$ decides $A_{TM}$

Define the TM $D =$ "On input $<M>$:
1. Run $M_{ATM}$ on $<M, <M>>$.
2. If $M_{ATM}$ accepts, reject; if $M_{ATM}$ rejects, accept."

Consider running $D$ on input $<D>$. Because $D$ is a decider:
- either computation halts and accepts …
- or computation halts and rejects …
Assume, towards a contradiction, that \( \mathcal{M} \) \( \mathcal{A}_{TM} \) decides \( \mathcal{A}_{TM} \)

Define the TM \( \mathcal{D} \) = "On input \(<\mathcal{M}>\):
1. Run \( \mathcal{M} \) \( \mathcal{A}_{TM} \) on \(<\mathcal{M}, \langle \mathcal{M} \rangle>\).
2. If \( \mathcal{M} \) \( \mathcal{A}_{TM} \) accepts, reject; if \( \mathcal{M} \) \( \mathcal{A}_{TM} \) rejects, accept."

Consider running \( \mathcal{D} \) on input \(<\mathcal{D}>\). Because \( \mathcal{D} \) is a decider:
- either computation halts and accepts ...
- or computation halts and rejects ...

Diagonalization proof: \( \mathcal{A}_{TM} \) not decidable

Sipser 4.11
Do we have to diagonalize?

- Next time (after exam): comparing difficulty of problems.
Next time: Exam 2

Bring ID, note card

Check seat maps before coming to class
$A_{TM}$

- Recognizable
- Not decidable

**Fact** (from discussion section): A language is decidable iff it and its complement are both recognizable.

**Corollary:** The complement of $A_{TM}$ is **unrecognizable**.