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 Siper 4.1: The computational problems below are all 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?
Before we proved the Pumping Lemma …

We proved there was a set that was not regular because counting arguments.

All sets of strings

All Regular Sets

Countable

Uncountable

$P(\Sigma^*)$

finite sequence of characters in a finite alphabet
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>$\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$</td>
<td>$\mathbb{R}$</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>

Powerset of any infinite set is uncountable *Sipser p. 202-204*
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.

What type of elements are in the set $P(\Sigma^*)$?

A. Strings
B. Regular expressions
C. Languages = sets of strings.
D. Sets of regular expressions
E. I don’t know

Sipser p. 202-204

$\Sigma^*$ is the set of all strings over $\Sigma$. 
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.
All sets of strings

Countable

All Turing-recognizable sets

Uncountable

Is the set of Turing-decidable sets countable?

\[ \text{Turing-decidable } \subseteq \text{Turing-recognizable} \]
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 $A_{\text{DFA}} = \{ <B,w> \mid B \text{ is a DFA and } w \text{ is in } L(B) \}$

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

What is $A_{\text{TM}}$?

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) \} \\

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

Which of the following statements is true?
A. \( N \) decides \( A_{TM} \) 
B. \( N \) recognizes \( A_{TM} \) 
C. \( N \) always halts 
D. More than one of the above. 
E. I don't know
$A_{TM}$

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

Decider for this set simulates arbitrary DFA

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

Decider for this set simulates arbitrary TMs.

What happens when it simulates itself?
Diagonalization proof: $A_{TM}$ not decidable

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 = \text{"On input } \langle M \rangle:\$
1. Run $M_{ATM}$ on $\langle M, \langle M \rangle \rangle$.
2. If $M_{ATM}$ accepts, reject; if $M_{ATM}$ rejects, accept.
Diagonalization proof: $A_{TM}$ not decidable  

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

Define the TM $D = "\text{On input } \langle M \rangle:\n1. \text{ Run } M_{ATM}\text{ on } \langle M, \langle M \rangle \rangle. \n2. \text{ If } M_{ATM}\text{ accepts, reject; if } M_{ATM}\text{ rejects, accept.}"

Which of the following computations halt?
A. Computation of $D$ on $\langle X \rangle$ where $X$ is TM which accepts string $w$ if first character is 0 and loops otherwise.
B. Computation of $D$ on $\langle Y \rangle$ where $Y$ is TM with $L(Y) = \Sigma^*$
C. Computation of $D$ on $\langle D \rangle$
D. All 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 = \text{"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 …
Case 1: $\langle D \rangle \in L(D) \Rightarrow$
when you run D on $\langle D \rangle$ it will halt and accept.

Case 2: $\exists D \text{ will reject } \langle D \rangle.$

Case 3: $\langle D \rangle \notin L(D) \Rightarrow$
when run D on $\langle D \rangle$ it will halt and reject.

$\Rightarrow$ run MATM on $\langle D, \langle D \rangle \rangle$, it will reject.

$\Rightarrow$ run D on $\langle D \rangle$, it will accept.
Diagonalization proof: $A_{TM}$ not decidable

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

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

Consider running $D$ on input $<D>$. Because $D$ is a decider:

- either computation halts and accepts …
- or computation halts and rejects …
$A_{\text{TM}}$

- Recognizable
- Not decidable

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

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