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

\[ A_{\text{DFA}}, A_{\text{NFA}}, A_{\text{REX}}, A_{\text{CFG}} \]
\[ E_{\text{DFA}}, E_{\text{NFA}}, E_{\text{REX}}, E_{\text{CFG}} \]
\[ EQ_{\text{DFA}}, EQ_{\text{NFA}}, EQ_{\text{REX}}, EQ_{\text{CFG}} \]
*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
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 $\mathbb{N}$)</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>
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
D. Sets of regular expressions
E. I don’t know

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.
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 is a DFA and w is in L(B) \}

A_{TM} = \{<M,w> | M is a TM and w 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.
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
$A_{TM}$

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

Decider for this set simulates arbitrary DFA

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

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 $L(M)$. 

$M_{ATM} \neq N$
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."
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."

Which of the following computations halt?
A. Computation of $D$ on $<X>$ where $X$ is TM which accepts string $w$ if first character is 0 and loops otherwise.
B. Computation of $D$ on $<Y>$ where $Y$ is TM with $L(Y) = \Sigma^*$
C. Computation of $D$ on $<D>$
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 $D =$ "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 …
Diagonalization proof: $A_{TM}$ not decidable

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

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

Consider running $D$ on input $<D>$. Because $D$ is a decider:
- either computation halts and accepts …
- or computation halts and rejects …

Self-reference

"Is $<D>$ an element of $L(D)$?"
\( A_{TM} \)

- 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**.
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