Today's learning goals

• Define and explain core examples of computational problems, including $A^{**}$, $E^{**}$, $EQ^{**}$, $HALT_{TM}$ (for ** either DFA or TM)
• Explain what it means for one problem to reduce to another
• Define computable functions, and use them to give mapping reductions between computational problems

HW 6 available - due Tues/Sat
Interim report glitch will be fixed for next coming soon
\[ A_{TM} = \{ <M,w> | M \text{ is a TM and } w \text{ is in } L(M) \} \]

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.
$A_{TM}$

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

Decider for this set simulates arbitrary DFA

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

Simulating arbitrary TM gives us $\not{=} \not{\text{not}}$ a decider!
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 $L(M)$.

$M_{ATM} \neq N$
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  

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

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

i.e. $D \in L(D)$

i.e. $\langle D, \langle D \rangle \rangle \in \text{ATM}$

i.e. $\text{ATM}$ accepts $\langle D, \langle D \rangle \rangle$.

Tracing $D$ on $\langle D \rangle$.

1. Run $\text{ATM}$ $\langle D, \langle D \rangle \rangle$, and by case assumption accepts.
2. Since $\text{ATM}$'s counter accepted, $D$ rejects.

CONTRADICTION!
Case 2: D rejects <D>
so <D> \not\in L(CD)
so <D, <D>> \not\in ATM
so \text{ATM rejects } <D, <D>>.

Tracing D on <D> using def of D.

Step 1: Run ATM on <D, <D>>
By case assumption, ATM rejects.
Step 2: Since ATM rejects, D accepts.

\text{CONTRADICTION!}
Diagonalization proof: $A_{\text{TM}}$ not decidable

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

Define the TM $D = \text{"On input } <M>:\n\begin{enumerate}
\item Run $M_{\text{TM}}$ on $<M, <M>>$.
\item If $M_{\text{TM}}$ accepts, reject; if $M_{\text{TM}}$ rejects, accept.
\end{enumerate}$

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

- either computation halts and accepts
- or computation halts and rejects

Diagonalization???

Self-reference

"Is $<D>$ an element of $L(D)$?"
\[
A_{TM} = \{<M,w> \mid M \text{ is a TM and } w \text{ is in } L(M) \}
\]

Define the TM \( N = \) "On input \( <M,w> \):
\begin{enumerate}
\item Simulate \( M \) on \( w \).
\item If \( M \) accepts, accept. If \( M \) rejects, reject.
\end{enumerate}
\( N \) is a Turing machine that \text{recognizes} \( A_{TM} \).

\textbf{No} Turing machine \text{decides} \( A_{TM} \).
A_{TM}

- Recognizable
- Not decidable

**Fact** (Theorem 4.22): A language is decidable iff it and its complement are both recognizable.

**Corollary 4.23**: The complement of $A_{TM}$ is unrecognizable.
<table>
<thead>
<tr>
<th>Decidable</th>
<th>Recognizable (and not decidable)</th>
<th>Co-recognizable (and not decidable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_{DFA}$</td>
<td>$A_{TM}$</td>
<td>$A_{TM}^c$</td>
</tr>
<tr>
<td>$E_{DFA}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$EQ_{DFA}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Give algorithm!

Diagonalization
Do we have to diagonalize?
Idea

If problem X is no harder than problem Y
...and if Y is easy
...then X must also be easy
Idea

If problem X is no harder than problem Y
…and if X is hard
…then Y must also be hard
Idea

If problem X is no harder than problem Y
...and if Y is **decidable**
...then X must also be **decidable**

If problem X is no harder than problem Y
...and if X is **undecidable**
...then Y must also be **undecidable**
If problem X is no harder than problem Y
...and if Y is **decidable**
...then X must also be **decidable**

If problem X is no harder than problem Y
...and if X is **undecidable**
...then Y must also be **undecidable**

“Problem X is no harder than problem Y” means
“Can convert questions about membership in X to questions about membership in Y”
Problem A is **mapping reducible** to problem B means there is a computable function $f: \Sigma^* \rightarrow \Sigma^*$ such that for all strings $x$ in $\Sigma^*$

$$x \text{ is in } A \iff f(x) \text{ is in } B$$
Mapping reduction

Problem A is **mapping reducible** to problem B means there is a computable function $f: \Sigma^* \rightarrow \Sigma^*$ such that for all strings $x$ in $\Sigma^*$

$$x \text{ is in } A \text{ iff } f(x) \text{ is in } B$$

**Computable function?**

A function $f: \Sigma^* \rightarrow \Sigma^*$ is **computable** iff there is some Turing machine such that, for each $x$, on input $x$ halts with exactly $f(x)$ followed by all blanks on the tape.
Computable functions (aka maps)

Which of the following functions are computable?

A. The string $x$ maps to the string $xx$.

B. The string $<M>$ (where $M$ is a TM) maps to $<M'>$ where $M'$ is the Turing machine that acts like $M$ does, except that if $M$ tries to reject, $M'$ goes into a loop; strings that are not the codes of TMs map to $\varepsilon$.

C. The string $x$ maps to $y$, where $x$ is the binary representation of the number $n$ and $y$ is the binary representation of the number $2^n$.

D. All of the above.

E. None of the above.
Next time

How do reductions help us determine decidability / undecidability / recognizability / unrecognizability?